Luís Felipe da Silva Carlos Pereira

Projetos de Controladores Fracionários e Preditivos com Testabilidade Usando FPGA-*in-the-loop*

Brasil

2022

Luís Felipe da Silva Carlos Pereira

Projetos de Controladores Fracionários e Preditivos com Testabilidade Usando FPGA-*in-the-loop*

Dissertação apresentada à Fundação Universidade do Mato Grosso do Sul (UFMS) para a obtenção do título de Mestre em Engenharia Elétrica na área de Sistemas de Energia.

Fundação Universidade do Mato Grosso do Sul – UFMS

Faculdade de Engenharias, Arquitetura e Urbanismo e Geografia

Programa de Pós-Graduação em Engenharia Elétrica

Orientador: Edson Antonio Batista

Coorientador: Raymundo Cordero Garcia

Brasil

2022

Luís Felipe da Silva Carlos Pereira
Projetos de Controladores Fracionários e Preditivos com Testabilidade Usando FPGA-*in-the-loop*/ Luís Felipe da Silva Carlos Pereira. – Brasil, 2022-
143 p. : il. (algumas color.) ; 30 cm.

Orientador: Edson Antonio Batista

Dissertação (Mestrado) – Fundação Universidade do Mato Grosso do Sul – UFMS
Faculdade de Engenharias, Arquitetura e Urbanismo e Geografia
Programa de Pós-Graduação em Engenharia Elétrica, 2022.

1. Controladores PID de ordem fracionária. 2. Algoritmo de evolução diferencial. 3. MPC. 4. GPC. 5. FPGA-*in-the-loop*. I. Batista, Edson Antonio, *orient*. II. Título.

Luís Felipe da Silva Carlos Pereira

Projetos de Controladores Fracionários e Preditivos com Testabilidade Usando FPGA-*in-the-loop*

Dissertação apresentada à Fundação Universidade do Mato Grosso do Sul (UFMS) para a obtenção do título de Mestre em Engenharia Elétrica na área de Sistemas de Energia.

Trabalho aprovado. Campo Grande - MS, Brasil, 19 de dezembro de 2022:

gov.br Documento assinado digitalmente
EDSON ANTONIO BATISTA
Data: 12/01/2023 12:08:14-0300
Verifique em https://verificador.iti.br

Dr. **Edson Antonio Batista**
Orientador

Dr. **Raymundo Cordero Garcia**
Coorientador

gov.br Documento assinado digitalmente
RUBEN BARROS GODOY
Data: 17/01/2023 21:29:41-0300
Verifique em https://verificador.iti.br

Dr. **Ruben Barros Godoy**
Convidado 1

Dr. **Cristiano Quevedo Andrea**
Convidado 2

Brasil
2022

Este trabalho é dedicado para as pessoas que fizeram parte do meu crescimento científico e aos meus familiares, principalmente aos meus pais.

AGRADECIMENTOS

Ao meu orientador Edson o qual agradeço pela oportunidade, dedicação e disponibilidade para a realização deste trabalho.

Ao Exército Brasileiro, por ter me dado esta oportunidade de realizar o mestrado em uma instituição federal e assim fomentar ainda mais o desenvolvimento do P&D brasileiro.

Às pessoas do BATLAB (UFMS) que deram suporte prático para a criação dos artigos desenvolvidos nesta dissertação.

Aos professores da UFMS que contribuíram para a minha formação.

À CAPES pelo apoio financeiro aos artigos publicados.

Ao projeto de desenvolvimento e pesquisa - P&D ANEEL. PD-06961-0010/2019.

Aos meus familiares que sempre me proporcionaram um ambiente ideal para o meu crescimento intelectual.

Por fim, agradeço aos meus pais por sempre me darem apoio e força para que eu continue em busca dos meus sonhos.

RESUMO

Neste trabalho é apresentado uma nova forma de sintonizar controladores PIDOF robustos com a propriedade de amortecimento iso ou do inglês *iso-damping*, utilizando algoritmos de evolução diferencial e com a redução de ordem de sistemas lineares aplicando norma L2. Além disso, apresenta-se uma forma de projetar controladores preditivos multivariáveis baseados em funções ortonormais de Laguerre e com grau de estabilidade prescrito para a comparação com o PIDOF. Para a validação de ambos os controladores, a ferramenta FPGA-*in-the-loop*, disponível em ambiente MATLAB®/Simulink, foi utilizada em conjunto com uma biblioteca matricial para a elaboração dos códigos em VHDL. Assim, os controladores PIDOF e MPC foram sintonizados para controlar o ciclo de trabalho de operação de um conversor Boost com restrições, e testes experimentais foram realizados em bancada com FPGA para validar ambos os métodos. Os resultados dos controladores supracitados foram satisfatórios, sendo que o MPC-Boost teve, em geral, melhor resposta dinâmica, porém com um custo computacional maior do que o PIDOF-Boost. Por fim, esta dissertação extrapola a aplicação do controle preditivo GPC para o rastreamento de referências polinomiais com funções de Laguerre, restrições na variável de controle e prescrição de estabilidade sobre sistemas multivariáveis. A elaboração do código em VHDL para o GPC tipo m foi comparada com o seu modelo matemático proposto e resultados em FPGA-*in-the-loop* corroboraram o método.

Palavras-chave: Controladores PIDOF. Algoritmo de Evolução Diferencial. Norma L2. MPC. GPC. FPGA-*in-the-loop*.

ABSTRACT

This work presents a new method of tuning robust FOPID controllers that exhibit iso-damping property, using differential evolution algorithm and L2 norm order reduction algorithm for linear systems. Furthermore, it is presented a multivariable predictive controller based on orthonormal Laguerre functions and with a prescribed degree of stability to be compared with the FOPID proposed. For the validation of both controllers in MATLAB®/Simulink environment, the FPGA-*in-the-loop* tool was used along with a matrix library for the elaboration of VHDL codes. Thus, the FOPID and MPC controllers were designed to control the operating duty cycle of a Boost converter with constraints, and experimental simulations were performed using a test bench with FPGA to validate both methods. The results of the controllers aforementioned were satisfactory, and in general, the MPC-Boost presented superior dynamic response, however with a higher computational burden as related to the FOPID controller. Finally, this thesis extrapolates the application of a predictive control GPC for tracking polynomial references with orthogonal Laguerre functions, restrictions on the control variable and stability prescription over multivariable systems. The VHDL code elaboration for the GPC type m was compared with its proposed mathematical model, and the results with FPGA-*in-the-loop* simulation corroborated the method.

Keywords: FOPID controllers. Differential Evolution Algorithm. L2 norm. MPC. GPC. FPGA-*in-the-loop*.

LISTA DE ILUSTRAÇÕES

Figura 1 – Inicialização da máscara do operador integro-diferencial. 41
Figura 2 – Máscara integro-diferencial com filtro Oustaloup refinado. 42
Figura 3 – Diagrama de blocos do PIDOF em paralelo. 42
Figura 4 – Diagrama de blocos do FTMF de um sistema G(s). 44
Figura 5 – Fluxograma do algoritmo ED. 48
Figura 6 – Fluxograma do algoritmo MPC. 74
Figura 7 – Conversor Boost. 76
Figura 8 – Resposta em tensão do conversor Boost em malha aberta. 77
Figura 9 – Resposta em corrente no indutor do conversor Boost em malha aberta. 78
Figura 10 – Esquemático do PIDOF-Boost. 79
Figura 11 – Resposta em tensão do PIDOF-Boost. 81
Figura 12 – Resposta de corrente no indutor do PIDOF-Boost. 82
Figura 13 – Ciclo de trabalho do PIDOF-Boost. 82
Figura 14 – Configuração experimental do teste em bancada com o PIDOF-Boost. . 84
Figura 15 – Inicialização do PIDOF-Boost experimental. 84
Figura 16 – Tensão de saída e corrente de carga durante a perturbação na carga de 16 W a 8W do PIDOF-Boost. 85
Figura 17 – Tensão de saída e corrente de carga durante a perturbação na carga de 8 W a 16 W do PIDOF-Boost. 86
Figura 18 – Bloco “Função MATLAB®” do MPC. 87
Figura 19 – Comparação da tensão entre o MPC-MATLAB® e MPC-FIL do conversor Boost. 88
Figura 20 – Comparação da corrente entre o MPC-MATLAB® e MPC-FIL do conversor Boost. 89
Figura 21 – Comparação do valor numérico do ciclo de trabalho entre o MPC-MATLAB® e MPC-FIL do conversor Boost. 89
Figura 22 – Configuração experimental do teste em bancada com o MPC-Boost. . . 90
Figura 23 – Inicialização do MPC-Boost experimental. 91
Figura 24 – Tensão de saída e corrente de carga durante a perturbação na carga de 16 W a 8 W do MPC-Boost. 92
Figura 25 – Tensão de saída e corrente de carga durante a perturbação na carga de 8 W a 16 W do MPC-Boost. 92
Figura 26 – Tensão de referência a ser rastreada. 93
Figura 27 – Bloco “Função MATLAB®” do GPC tipo 3. 94
Figura 28 – GPC tipo 3 sem restrições (N=1): $e(t)$. 95

Figura 29 – GPC tipo 3 sem restrições (N=1): $u_m(t)$. 96
Figura 30 – GPC tipo 3 sem restrições (N=1): $y(t)$. 96
Figura 31 – GPC tipo 3 sem restrições (N=2): $e(t)$. 97
Figura 32 – GPC tipo 3 sem restrições (N=2): $u_m(t)$. 98
Figura 33 – GPC tipo 3 sem restrições (N=2): $y(t)$. 98
Figura 34 – GPC tipo 3 com restrições (N=2): $e(t)$. 99
Figura 35 – GPC tipo 3 com restrições (N=2): $u_m(t)$. 99
Figura 36 – GPC tipo 3 com restrições (N=2): $\nabla u_m(t)$. 100
Figura 37 – GPC tipo 3 com restrições (N=2): $y(t)$. 100
Figura 38 – Resumo do fluxo de dados para a implementação em FPGA dos controladores PIDOF/MPC-Boost. 102

LISTA DE TABELAS

Tabela 1 – Resumo dos trabalhos relacionados com PI/PIDOF. 23
Tabela 2 – Resumo dos trabalhos relacionados com MPC/GPC. 26
Tabela 3 – Parâmetros do algoritmo ED. 50
Tabela 4 – Especificações do conversor Boost 76
Tabela 5 – Componentes do conversor Boost 77
Tabela 6 – Parâmetros de ganho do PIDOF-Boost. 80
Tabela 7 – Parâmetros eletrônicos para o teste em bancada do conversor Boost. . 83
Tabela 8 – Parâmetros de configuração do MPC-Boost. 87
Tabela 9 – Parâmetros de configuração GPC tipo 3 com N = 1 e sem restrições. . 94
Tabela 10 – Comparativo PIDOF/MPC-Boost. 101

LISTA DE ABREVIATURAS E SIGLAS

ABC	*Artificial Bee Colony*
CF	Cálculo Fracionário
CI	*Cohort Intelligent*
COI	Controlador de Ordem Inteira
COF	Controlador de Ordem Fracionária
CRONE	*Commande Robuste d'Ordre Non-Entier*
CTT	*Compromise Tuning Technique*
DGPC	*Digital Generalized Predictive Control*
DMC	*Dynamic Matrix Control*
DMPC	*Digital Model Predictive Control*
DSP	*Digital Signal Processor*
ED	Evolução Diferencial
EQM	Erro Quadrático Médio
FPGA	*Field-programmable Gate Array*
FTMA	Função de Transferência em Malha Aberta
FTMF	Função de Transferência em Malha Fechada
GPC	*Generalized Predictive Control*
HLS	*High-level Synthesis*
IAE	*Integral of Absolute Error*
IMP	*Internal Mode Principle*
ISE	*Integral of Square Error*
ITAE	*Integral of Time multiplied Absolute Error*
ITSE	*Integral of Time multiplied Square Error*

LQR	*Linear Quadratic Regulator*
LTT	*Lexicographic Tuning Technique*
MCAS	*Maximal Controlled Admissible Set*
MIMO	*Multiple-input Multiple-output*
MPC	*Model Predictive Control*
MPHC	*Model Predictive Heuristic Control*
MPPT	*Maximum Power Point Tracking*
AmpOp	Amplificador Operacional
PI	Proporcional Integral
PID	Proporcional Integral Derivativo
PIDOF	PID de Ordem Fracionária
PIOF	PI de Ordem Fracionária
PMSM	*Permanent Magnet Synchronous Motor*
PSFB	*Phase-shifted Full-bridge*
PWM	*Pulse Width Modulation*
QBGA	*Queen Bee Assisted Genetic Algorithm*
RL	Riemman-Liouville
SCE	Sistema de Condicionamento de Energia
SFR	*Sodium Fast Reactor*
SISO	*Single-input Single-output*
SMPS	*Switched-mode Power Supply*
SPEA	*Strength Pareto Evolutionary Algorithm*
TFC	Teorema Fundamental do Cálculo
VHDL	*VHSIC Hardware Description Language*
XSG	*Xilinx System Generator*

LISTA DE SÍMBOLOS

$\Gamma(.)$	Função Gamma
Ψ	Matrix auxiliar na função custo do MPC/GPC
Ω	Matriz Hessiana do MPC/GPC
$\Phi(.)$	Função de Gel'fand-Shilov
ϕ	Margem de fase de um sistema SISO para uma dada frequência de corte
φ	Matrix auxiliar no espaço de estado aumentado do MPC/GPC
ω	Frequência angular (rad/s)
μ	Ordem fracionária da derivada
λ	Ordem fracionária da integral ou autovalor de um sistema
β	Ordem fracionária
α	Ordem fracionária
η	Vetor dos coeficientes ótimos das funções de Laguerre
ζ	Matrix auxiliar para demonstração do Teorema 3.1.1
δ	Pequena perturbação em um vetor pertencente aos $\mathbb{R}^n$
κ	Número de condição de uma matrix
$\varkappa$	Vetor em (m-1)-ésimas diferenças discretas retrógradas do erro
ρ	Polinômio característico do GPC aumentado.
$\in$	Pertence (conjuntos)
$\mathbb{N}$	Conjunto dos números naturais
$\mathbb{Z}$	Conjunto dos números inteiros
$\mathbb{R}$	Conjunto dos números reais
$\mathbb{C}$	Conjunto dos números complexos
T_s	Período de amostragem do sistema

f_s	Frequência de amostragem do sistema
f_{mosfet}	Frequência de amostragem do Mosfet
N	Número de funções de Laguerre
N_p	Número de predição
N_c	Número de horizonte de controle
$G(s)$	Função de transferência no domínio contínuo $s = \sigma + j\omega$
$G(j\omega)$	Função de transferência no domínio contínuo sobre o eixo complexo $s = j\omega$
$G(z)$	Função de transferência no domínio digital $z = re^{j\omega}$
$L(f(t))$	Transformada de Laplace de uma função f(t)
$\frac{df}{dx}$	Derivada ordinária de f em relação a x
$\frac{\partial f}{\partial x}$	Derivada parcial de f em relação a x
∇	Operador em diferenças discretas
arg	Argumento de uma variável complexa
tan	Função tangente
cos	Função cosseno
sin	Função seno
$\prod_{n=1}^{k} f(n)$	Produtório de uma função f(i), $n = 1, 2, \cdots, k$.
$\sum_{n=1}^{k} f(n)$	Somatório de uma função f(i), $n = 1, 2, \cdots, k$.

SUMÁRIO

1 INTRODUÇÃO 17
1.1 Cálculo Fracionário e Modelo Preditivo 17
1.2 Motivação da Pesquisa 20
1.3 Estado da Arte 21
1.3.1 PIDOF 21
1.3.2 MPC e GPC 24
1.4 Objetivo do Trabalho 28
1.5 Materiais e Métodos 29
1.6 Estrutura da Dissertação 30

2 CONTROLADORES PID EM ORDEM FRACIONÁRIA 31
2.1 Integral de Riemman-Liouville e sua transformada de Laplace 31
2.2 Derivada segundo Caputo e sua transformada de Laplace 36
2.3 Filtro Oustaloup 37
2.4 Modelagem de controladores PID em ordem fracionária 40
2.5 Algoritmo de Evolução Diferencial 47
2.6 Redução da ordem de sistemas fracionários com norma 2 50

3 CONTROLADOR POR MODELO DE PREDIÇÃO 52
3.1 Controlador por modelo de predição com 1 integrador 52
3.1.1 Modelagem do MPC 52
3.1.2 Princípio operacional do MPC com funções de Laguerre 55
3.2 Controlador por modelo de predição generalizado com m integradores 59
3.2.1 Modelagem do GPC tipo m 59
3.2.2 Princípio operacional do GPC tipo m com funções de Laguerre 64
3.3 Minimização da função custo em controladores por modelo de predição 64
3.4 Controlador por modelo de predição com restrições 66
3.5 Controlador por modelo de predição com grau de estabilidade . . . 68

4 APLICAÇÕES DE CONTROLADORES FRACIONÁRIOS E PREDITIVOS COM TESTABILIDADE EM SIMULAÇÕES E FPGA-*IN-THE-LOOP* 75
4.1 Conversor Boost 75
4.1.1 Controlador PIDOF 77
4.1.2 Controlador MPC 85
4.2 Simulação com rastreamento de referência polinomial 91

5 CONCLUSÃO 101
5.1 Principais Contribuições 102
5.2 Publicações 104
5.2.1 Trabalhos relacionados diretamente com a pesquisa 104
5.2.2 Trabalhos relacionados indiretamente com a pesquisa 104
5.3 Proposta de trabalhos futuros 105

REFERÊNCIAS 106

APÊNDICES 113

APÊNDICE A – DEMONSTRAÇÕES MATEMÁTICAS 114
A.1 Funções gama e beta 114
A.2 Operador generalizado discreto em diferenças retrógradas 116
A.3 Funções de Laguerre 119
A.4 Solução do problema LQR por eliminação de blocos 121

B – CÓDIGOS EM VHDL 126
B.1 PIDOF-Boost 126
B.2 MPC-Boost 132
B.3 GPC tipo 3 (N=2) 138

1 INTRODUÇÃO

1.1 CÁLCULO FRACIONÁRIO E MODELO PREDITIVO

O cálculo convencional é fundamentalmente baseado com derivadas e integrais de ordem inteira (df/dx e $\int f(x)dx$ para ordem 1). Os mesmos pesquisadores que lançaram a base para o cálculo diferencial e integral de ordem inteira já se perguntavam, no século XVII, sobre o possível significado matemático de uma derivada de ordem real. A troca de cartas entre L'Hopital e Leibniz sobre a derivada de ordem 1/2 levou ao surgimento do cálculo fracionário (CF) (MILLER; ROSS, 1993). Desde então, grandes matemáticos contribuíram para o desenvolvimento do CF (DAS, Jan. 2011; MILLER; ROSS, 1993).

O cálculo fracionário tem sido amplamente utilizado na área de sistemas de controle, na qual a diferenciação e integração em ordem fracionária podem ser utilizadas para o projeto do controlador (MONJE et al., 2008). Ele generaliza o cálculo convencional de ordem inteira ao utilizar operadores de ordem real ou complexa (DAS, Jan. 2011; OUSTALOUP et al., 2000). Controladores fracionários ganharam popularidade nos últimos anos pela robustez em relação às variações de ganho da planta em malha fechada (amortecimento iso) e incertezas (FLORES et al., 2020; SAHA et al., 2010). Especificações de projeto, como margem de ganho e fase, podem ser ajustadas com mais flexibilidade usando controladores de ordem fracionária (COF) em comparação com os de ordem inteira (COI). O COI se compara ao COF em relação à resposta dinâmica e robusta se projetado com vários polos e zeros, aumentando a quantidade de constantes a serem sintonizadas. Em contrapartida, isso é superado com a adição de apenas 2 ordens fracionárias (integração e diferenciação) no COF (CHEN; PETRAS; XUE, Jun. 2009). Os operadores fracionários possuem memória, ou seja, podem armazenar os primeiros estados e, portanto, melhorar a ação de filtragem. Esta propriedade ajuda a reduzir o esforço de controle. Logo, um sinal de controle mais suave é obtido com um COF se comparado à um COI (MONJE et al., 2010). O COF tem mais parâmetros de sintonia do que o COI (considerando ordem inteira mais próxima) dessa forma, mais especificações de projeto podem ser satisfeitas com o COF. Assim, sistemas de controle mais robustos e precisos podem ser projetados usando COF.

O século XXI viu um crescimento notável na modelagem e controle de sistemas de eletrônica de potência usando cálculo fracionário. Anteriormente, controladores Proporcional Integral (PI) e Proporcional Integral Derivativo (PID) eram os mais usados na indústria. Recentemente, uma considerável quantidade de pesquisas foi realizada com controladores PI/PID de ordem fracionária (PIOF/PIDOF) para vários conversores CC-CC, acionadores elétricos, inversores conectados à rede, inversores fotovoltaicos, conversores

multinível em aplicações de energia eólica, *microgrids*, controle de *Maximum Power Point Tracking* (MPPT) de painéis fotovoltaicos e veículos elétricos/híbridos (YICHEN; HEJIN; DEMING, Dec. 2017; MEHRA; SRIVASTAVA; VARSHNEY, Nov. 2010; PULLAGURAM et al., Mar. 2016; BALASKA et al., 2019). Combinações de sistemas de lógica fuzzy com controle fracionário também foram implementados e demonstraram combinar as vantagens de ambas as formas de controle (PEREIRA et al., 2022a). Simulações e implementações em *hardware* mostram que o COF pode ser efetivamente usado em sistemas eletrônicos de potência para dar melhor resultado do que os controladores convencionais (ZHU; LIU; LIU, Jun. 2014).

No final da década de 1970, (RICHALET et al., 1978) e (CUTLER; RAMAKER, 1980) estabeleceram, independentemente, o fundamentos do controle preditivo baseado em modelo (MPC - do inglês, *Model Predictive Control*). Eles foram capazes de controlar, de forma eficiente, problemas que geram um enorme potencial econômico. A referência (RICHALET et al., 1978) introduziu o controle heurístico preditivo baseado em modelo (MPHC - do inglês, *Model Predictive Heuristic Control*) em 1978, que incluía todas as características de um MPC, tais como:

- um modelo de processo explícito, descrito por funções de resposta ao impulso;
- um horizonte retrocedente;
- restrições de entrada e saída;
- uma determinação iterativa da variável manipulada u.

No entanto, a referência (RICHALET et al., 1978) não reivindicou obter controles ótimos. Em vez disso, os controles futuros foram determinados iterativamente até que atendessem às restrições. Além disso, o termo adicional "heurística" enfatizou a falta da lei de controle explícita. A técnica foi desenvolvida para a indústria de processo com seus sistemas de múltiplas entradas e múltiplas saídas (MIMO - do inglês, *Multiple-input Multiple-output*), atrasos distintos e longos tempos de processamento (RICHALET et al., 1978). Os autores consideraram identificar o modelo de processo *on-line*, embora apenas para mudanças nos pontos de ajuste.

Ao mesmo tempo, a *Shell Oil Company* desenvolveu o controle matricial dinâmico (DMC - do inglês, *Dynamic Matrix Control*) (CUTLER; RAMAKER, 1980). Os autores usaram um modelo linear por partes para prever o comportamento futuro de uma unidade de craqueamento catalítico. Assim, o controlador tomou conhecimento do atraso temporal da planta e do comportamento dinâmico do sistema. Cutler e Ramaker usaram um horizonte de predição retrocedente e atualizaram os coeficientes do modelo com base no erro entre a saída prevista e a saída atualmente medida. Os autores mostraram neste trabalho que o

DMC superou o controle PID clássico em cascata, e alegam que o DMC tem sido aplicado para controlar problemas na *Shell Oil* desde 1974. A principal diferença para o MPHC é que o DMC calcula variáveis de controle ótimas. No entanto, a formulação matricial do problema de controle restringe o DMC a modelos de processos lineares.

Em ambos os trabalhos (RICHALET et al., 1978; CUTLER; RAMAKER, 1980), lançaram-se as bases para uma ampla e rápida disseminação do MPC na indústria de processos petroquímicos. Mesmo com modelos lineares, os tempos de amostragem foram de várias horas (PRETT; GILLETTE, 1980). No início, o foco era simplificar o projeto do controlador e estabelecer uma teoria abrangente para que o método pudesse ser usado na indústria (RICHALET et al., 1978; CUTLER; RAMAKER, 1980; GARCIA; MORARI, 1982). O potencial do MPC não se baseava apenas na previsão, mas também no fato de poder usar modelos não lineares, ambos não suportados pelo controle clássico. Na verdade, a formulação do modelo de processo foi um tópico relevante no início da teoria MPC: (a) formulação de resposta ao impulso (RICHALET et al., 1978), (b) funções de resposta ao degrau linear por partes (CUTLER; RAMAKER, 1980), (c) modelos autorregressivos de médias móveis ARMA (CLARKE; MOHTADI; TUFFS, 1987a; CLARKE; MOHTADI; TUFFS, 1987b) e, (d) formulações de espaço de estado (LEE; MORARI; GARCIA, 1994). Esta flexibilidade na escolha da formulação do modelo foi uma das principais razões para o rápido sucesso do MPC.

As primeiras abordagens simplesmente negligenciavam as incertezas do modelo e as instabilidades do processo, porque a maioria dos processos de engenharia química eram estáveis em malha aberta (GARCIA; PRETT; MORARI, 1989). A partir do final da década de 1980, o foco da pesquisa mudou para a robustez e estabilidade do MPC, que foi especialmente seguido por (BEMPORAD; MORARI, 1999; CAMPO; MORARI, 1986; CAMPO; MORARI, 1987; LAUGHLIN; MORARI, 1987; ZHENG; MORARI, 1995). Com uma janela móvel fixa, o problema de estimação (linear) poderia ser formulado como um problema de programação quadrática (RAO; RAWLINGS; LEE, 2001), que é computacionalmente favorável. Com o advento dos anos 2000 e os computadores com melhor desempenho, foi possível avançar da pesquisa básica para aplicações práticas.

A tendência atual é projetar controladores MPC para problemas complexos, multivariáveis e com restrições para que sejam embarcados em *hardware*. Com o aumento da complexidade, aumenta-se o custo computacional, portanto, aproximações usando funções ortonormais foram lançadas para a diminuição do tamanho do sistema em malha fechada. As principais funções ortonormais utilizadas são as de Laguerre e as de Kautz (WANG, 2009). Tais funções ortonormais são utilizadas para a expansão da variável de controle $u(k)$, que em um sistema estável, pertence ao espaço de Hilbert[1]. Neste trabalho, apenas as funções de Laguerre serão exploradas devido a simplicidade de polo único

[1] Espaço das funções quadrado-somáveis.

(Apêndice A.3). Assim, o projetista diminui o horizonte de controle para apenas algumas funções de Laguerre, diminuindo o tamanho do sistema em malha fechada sem perder sua resposta dinâmica. A ideia inicial de usar as funções de Laguerre no projeto de MPC digital (DMPC) foi publicada em (WANG, 2001; WANG, 2004).

Por fim, este trabalho propõe a implementação do MPC baseado em funções de Laguerre com restrições em VHDL para que, com simulações em FPGA-*in-the-loop* (FIL), o controlador seja embarcado em FPGA e validado em bancada para o conversor Boost. Em paralelo, objetiva-se aplicar o cálculo fracionário em PIDOF, simular o funcionamento sobre um conversor Boost com MATLAB®/Simulink, montar o seu código em VHDL com biblioteca matricial e embarcá-lo em FPGA para um teste em bancada. Comparações são realizadas entre PIDOF-Boost e MPC-Boost em relação à performance, velocidade e custo computacional. Além disso, conceitos teóricos são expandidos com a simulação em FIL para o GPC tipo m, capaz de rastrear funções polinomiais de grau $m-1$, baseado em funções de Laguerre, aumentando a base teórica do MPC.

1.2 MOTIVAÇÃO DA PESQUISA

Na prática, todo sistema de potência precisa de um controlador para corrigir perdas internas e obter um erro nulo assintótico quando uma referência é imposta. Os controladores convencionais estão ultrapassados quanto à demanda por velocidade e robustez das especificações dos sistemas atuais. Dessa forma, é de extrema importância a aplicação dos "recentes" controladores PIDOF quase ótimo e MPC ótimo para atingir tais especificações. As principais motivações para a sintonia do controlador PIDOF são a sua característica de amortecimento iso, maior range operacional em frequência e robustez quando comparado ao PI/PID convencional. Em contrapartida, as principais motivações para a implementação do MPC baseado em funções de Laguerre são versatilidade quanto ao manuseio de restrições e a sintonia ótima. Todo processo está sujeito a restrições operacionais, como por exemplo, no caso do conversor Boost com o ciclo de trabalho do Mosfet restrito à $0 \leq d(t) \leq 1$. Na realidade, evita-se operar na faixa ideal do ciclo de trabalho, sendo muitas vezes recomendado uma faixa de operação menor pelos fabricantes ($0 \leq d(t) \leq 0,9$). Para o GPC tipo m, a motivação foi gerar teoria com a expansão da variável de controle com funções de Laguerre para possíveis aplicações em sistemas com necessidade de rastremanento de referências polinomiais.

Todos os sistemas supracitados são projetados digitalmente, sendo a motivação final do trabalho a demanda por controladores embarcados em FPGA. Este problema se mostra cada vez mais atual através da necessidade de frequências superiores a 100 KHz para implantação de controles em sistemas eletrônicos de potência. Os FPGAs possuem características de paralelismo e ciclos de operação de alta velocidade que favorecem

o bom desempenho dos algoritmos de controle PIDOF e MPC. Além disso, o FPGA possui processamento de alta velocidade e facilidade de codificação em VHSIC *hardware Description Language* (VHDL), características ideais para lidar com sistemas complexos. A extrema complexidade dos sistemas elétricos de potência exige processamento paralelo e de alta velocidade para concluir as operações matemáticas com uma resolução precisa. Em conjunto com a biblioteca matricial, motivou-se o desenvolvimento de códigos em VHDL dos controladores para a simulação em FIL e operação em FPGA.

Além disso, as metodologias para projetar o PIDOF e o GPC tipo m baseado em funções de Laguerre são novas, sendo contribuições para serem exploradas pela comunidade científica e industrial. Portanto, a teoria e os códigos em VHDL apresentados poderão servir de referência para futuros trabalhos.

1.3 ESTADO DA ARTE

Nesta seção, é feita uma síntese dos principais trabalhos que abordam soluções para controladores PIDOF e MPC em sistemas de potência, com o objetivo de estudar as principais técnicas aplicadas e desenvolvidas pela comunidade científica.

1.3.1 PIDOF

A referência (VANITHA; RATHINAKUMAR, 2017) propôs um controlador PIDOF em um conversor CC-CC Buck-Boost alimentado por um sistema fotovoltaico para melhorar a resposta dinâmica do sistema. Este conversor usa dois Mosfets e uma topologia de indutor acoplada. Os resultados da simulação e do *hardware* mostram um tempo de acomodação reduzido e uma resposta estável com o controlador PIDOF.

Em (MERRIKH-BAYAT; JAMSHIDI, 2013), usou-se o algoritmo *Artificial Bee Colony* (ABC) para otimizar um controlador PIDOF para um conversor Boost. Além da não linearidade do conversor Boost, ele possui fase não-mínima quando linearizado. A integral do erro absoluto (IAE - do inglês, *Integral of Absolute Error*) foi escolhido como o índice de desempenho para otimização. O IAE foi definido como $IAE = \int_0^T |V_0 - V_{ref}|dt$. Aqui, V_0, V_{ref} são as tensões de saída e de referência, respectivamente. As funções de transferência fracionárias do PIDOF são aproximadas pelo método de aproximação recursiva do filtro Oustaloup, onde um conjunto de fórmulas recursivas é utilizado na faixa de frequência de interesse escolhida (ω_b, ω_h) para aproximar o grau fracionário em um produtório de funções racionais na ordem inteira. O sistema em malha fechada com o controlador PIDOF é implementado no SIMULINK e simulado repetidamente com os parâmetros do controlador otimizados pelo algoritmo ABC até que o critério de parada seja alcançado. Os resultados mostraram que o controlador PIDOF proporcionou menor tempo de acomodação, com melhor regulação e estabilidade a perturbações de degrau em relação ao controlador

PID. Um resultado importante observado foi que o controlador PIDOF exigiu menor chaveamento liga/desliga em comparação com o controlador PID e, portanto, exigiu menor esforço de controle. Tais resultados evidenciam as vantagens do PIDOF, pois apresentam maior eficiência energética.

Uma técnica de posicionamento de polo dominante para o ajuste do controlador PIOF em conversores Buck e Boost foi usado em (PRAJAPATI; GARG; PRITHVI, 2018). O tempo de acomodação e o *overshoot* foram usados para calcular os polos dominantes do sistema em malha fechada. A referência (AMIRAHMADI et al., 2015) propôs um método para projetar controladores PIDOF ótimos para um conversor Boost por uma abordagem de otimização multiobjetiva. O *Strength Pareto Evolutionary Algorithm* (SPEA), uma abordagem de otimização tridimensional de parâmetros reais, foi usado para obter um controlador PIDOF em um conversor Boost de 5/12 V ($D = 7/12$), 15 KHz. A aproximação *Commande Robuste d'Ordre Non-Entier* (CRONE) de ordem 5 foi usada para o PIDOF. Um conjunto de ganhos quase ótimos, denominado conjunto de Pareto, foi gerado com a otimização dos ganhos e das ordens fracionárias do PIDOF. Tomando as funções custo dependentes do overshoot e o tempo de acomodação, um conjunto de resultados quase ótimos foi gerado em uma tabela de consulta, a partir da qual o projetista poderia escolher os parâmetros quase ótimos. Isso dá ao usuário maior facilidade e flexibilidade no ajuste. Além disso, o overshoot dinâmico foi adicionado como uma função objetivo extra. A otimização da resposta transitória mostrou que, embora os valores das ordens fracionárias fossem pequenos, obteve-se uma melhora significativa na resposta. A verificação experimental para a implementação discreta do PIDOF foi feita usando a placa de simulação em tempo real dSPACE 1104. Os resultados experimentais mostraram uma melhor resposta dinâmica, robustez em relação às variações no ponto de operação e uma rejeição ao comportamento caótico do pulso PWM (o ciclo de trabalho possui média constante, mas há oscilação em alta frequência) quando comparado com o controlador PID.

A abordagem em (SÁNCHEZ et al., 2021b) utilizou uma aproximação biquadrática do operador Laplaciano para sintetizar um controlador PIDOF para um conversor Buck-Boost. O módulo biquadrático é uma técnica que exibe uma resposta de fase plana em uma faixa em torno de uma frequência central. Resultados numéricos e experimentais foram fornecidos para corroborar a robustez e eficácia do método proposto. No que diz respeito ao (SORIANO-SÁNCHEZ et al., 2020), também foi empregada a aproximação biquadrática do operador Laplaciano para projetar um controlador PIDOF para um conversor Buck. É notável que tenha sido projetado o circuito elétrico com circuitos resistor-capacitor e amplificadores operacionais (AmpOp) para o controlador PIDOF. A aproximação de um controlador tipo PID de ordem não inteira para regular a tensão de saída em um conversor Boost é proposta em (SÁNCHEZ et al., 2021a). Também foi utilizada a aproximação biquadrática do operador Laplaciano, visando a característica de amortecimento iso do sistema controlado. Resultados experimentais foram apresentados para confirmar o bom

desempenho e regulação em tensão para o sistema Boost de fase não-mínima. Por outro lado, o efeito do algoritmo *Cohort Intelligent* (CI) na sintonia ótima de um controlador PID de ordem fracionária para um conversor Buck foi estudado em (WARRIER; SHAH, 2021).

Em (DEVARAJ et al., 2021), um controlador PIDOF baseado no algoritmo *Queen Bee Assisted Genetic Algorithm* (QBGA) foi projetado para superar o controlador PID quando são aplicadas variações de parâmetros. Novamente, a superioridade do PIDOF foi notável, no entanto, o trabalho não faz nenhum estudo relacionado com distúrbios de carga e variações de capacitância. Além disso, nenhum intervalo otimizado para a pesquisa de parâmetros foi introduzido no algoritmo. Por fim, projetou-se e implementou-se um controlador digital PIDOF aplicado a um conversor Boost em (SEO; CHOI, 2019). O procedimento para projetar os parâmetros do controlador PIDOF foi dado juntamente com um algoritmo de controle baseado em DSP e testado sob diferentes condições de carga. Apesar de ser uma abordagem interessante, o PIDOF proposto em (SEO; CHOI, 2019) utilizou dois sensores, ou seja, sensores de tensão e corrente, juntamente com uma faixa de população não otimizada, aumentando a carga computacional e tornando a síntese com maior necessidade de processamento em *hardware*, o que pode tornar a solução mais cara. Além disso, também foi recomendado o uso das ordens integro-diferenciais fracionárias no intervalo $]0, 1[$, com a ordem da integral próxima a um, para garantir a estabilidade do sistema em malha fechada, mostrando uma pequena faixa de soluções ótimas para o método proposto.

Na Tabela 1, resumem-se as principais características dos trabalhos apresentados na Seção 1.3.1 conforme a aplicação, método de otimização e implementação em *hardware* dos controladores fracionários PI/PIDOF.

Tabela 1 – Resumo dos trabalhos relacionados com PI/PIDOF.

Referência	**Aplicação**	**Método de otimização**	**Implementação em *Hardware***
(VANITHA; RATHINAKUMAR, 2017)	Controlar um conversor Buck-Boost em um sistema fotovotaico	Nenhum	Componentes analógicos
(MERRIKHBAYAT; JAMSHIDI, 2013)	Controlar um conversor Boost	Algoritmo ABC	Não

(continua)

Referência	**Aplicação**	**Método de otimização**	**Implementação em *Hardware***
(PRAJAPATI; GARG; PRITHVI, 2018)	Controlar conversores Buck e Boost	Alocação de polos dominantes	Não
(AMIRAHMADI et al., 2015)	Controlar um conversor Boost	SPEA	dSPACE 1104
(SÁNCHEZ et al., 2021b)	Controlar um conversor Buck-Boost	Aproximação biquadrática do operador Laplaciano	Componentes analógicos
(SORIANO-SÁNCHEZ et al., 2020)	Controlar um conversor Buck	Aproximação biquadrática do operador Laplaciano	Não
(SÁNCHEZ et al., 2021a)	Controlar um conversor Boost	Aproximação biquadrática do operador Laplaciano	Componentes analógicos
(WARRIER; SHAH, 2021)	Controlar um conversor Buck	Algoritmo CI	Não
(DEVARAJ et al., 2021)	Controlar um conversor Boost	Algoritmo QBGA	Não
(SEO; CHOI, 2019)	Controlar um conversor Boost	Algoritmo recursivo	Texas Instruments TMS320F28335 DSP

1.3.2 MPC E GPC

Para o MPC, o horizonte de controle N_c e o horizonte de predição N_p medem a resolução de controle com a técnica do horizonte retrocedente. Para melhorar a resolução de controle, horizontes de predição e controle devem assumir valores elevados, comprometendo o tamanho do sistema para implementação em *hardware*. Para sistemas estáveis em malha fechada, a variável de controle com MPC converge para zero de tal forma que pode ser expandida por uma base de funções ortonormais em $(0, +\infty)$. As funções ortonormais de Laguerre formam uma base infinita para esse espaço e, com poucos parâmetros, podem ser usadas no lugar da variável de controle com pequeno erro de truncamento (WANG, 2009). A fim de diminuir o tamanho do horizonte de controle, as funções de Laguerre são empregadas juntamente com MPC nesta dissertação. Em (SAEED; WANG; FERNANDO, 2022), argumentou-se que as funções de Laguerre melhoraram a frequência de comutação

do conversor *Phase-shifted Full-bridge* (PSFB) sem comprometer o desempenho dinâmico. Além disso, as funções Laguerre também podem resolver o desafio de diminuir o alto custo do processamento MPC com a escolha de algumas funções sem comprometer o desempenho dinâmico do sistema (ROSSITER; WANG; VALENCIA-PALOMO, 2010). Também vale ressaltar a viabilidade de implementação de restrições lineares em projetos de MPC. Recentemente, diversos trabalhos têm explorado a inserção de restrições operacionais e físicas em sistemas MPC. Métodos de ajuste de MPC com função otimização multiobjetivo sujeitos a restrições nas ações de controle foram considerados por vários autores (CHEN et al., 2019) e (YAMASHITA; ZANIN; ODLOAK, 2016).

O principal ônus para a implementação do MPC em FPGA é o projeto desafiador dos algoritmos contento operações matemáticas. Muitos desenvolvedores usaram ferramentas de síntese de alto nível e geração automática de código para traduzir códigos de alto nível para VHDL em problemas de MPC (PECCIN et al., 2020). A referência (MISHRA et al., 2021) desenvolveu o controlador no ambiente *Xilinx System Generator* (XSG) integrado ao MATLAB®/Simulink, que gera automaticamente o código HDL. O trabalho (LIEGMANN; KARAMANAKOS; KENNEL, 2022) também empregou técnicas de otimização da ferramenta *High-level Synthesis* (HLS) da Xilinx para resolver o problema MPC do conjunto de controle finito de longo horizonte para um inversor fixado em ponto neutro de três níveis acionando uma máquina de indução. O FPGA tem a vantagem do processamento paralelo, (LUCIA et al., 2018) propôs uma estratégia baseada em operações aritméticas que resolvem problemas de otimização para adequar um MPC embarcado em FPGA usando uma ferramenta de síntese de alto nível. No entanto, nenhum desses trabalhos apresentou um desenvolvimento direto do código VHDL com bibliotecas matriciais.

O princípio do modelo interno (IMP - do inglês, *Internal Mode Principle*) afirma que um sistema é capaz de rastrear sinais de entrada com um erro assintótico nulo se um controlador de realimentação for escolhido para fornecer estabilidade em malha fechada quando a malha aberta for escrita com função de transferência de realização mínima e ter pelo menos a parte instável do sinal exógeno embutido no controlador (BENGTSSON, 1977). Ao introduzir um controlador com o IMP em um sistema com função de transferência de realização mínima, (CORDERO et al., 2021) propôs um GPC com erros preditos e operador em diferença retrógrada de segunda ordem aplicado nas variáveis de estado como o estado aumentado de um sistema SISO para rastrear uma referência tipo rampa. Como a planta escolhida não possuia integradores embarcados, o projetista precisou adicionar 2 integradores a fim de que o modo interno de $(T_s z)^2/(z-1)^2$ pudesse ser inserido no loop, resultando em um rastreamento com erro assintótico nulo. Além disso, (CORDERO et al., 2021) estendeu o método de (CORDERO et al., 2021) ao usar operadores em diferenças retrógrados de ordem superior para obter erro assintótico nulo ao rastrear referências polinomiais. Um sistema de realização mínima sem integradores embutidos consegue

rastrear uma referência polinomial de grau $m-1$ se m integradores forem inseridos no sistema de malha aberta. Assim, operadores em diferenças retrógradas de alta ordem foram aplicados sobre o erro e as variáveis de estado do sistema para obter estados aumentados com m integradores incorporados. Novamente, o método foi desenvolvido apenas para sistemas SISO, sem um uso otimizado de parâmetros e estabilidade prescrita.

Em relação ao (CORDERO et al., 2022), foi desenvolvida uma nova abordagem para embarcar um modelo interno de referência senoidal através da aplicação dos operadores em diferença de primeira e segunda ordem em um sistema SISO. Apesar de ser uma abordagem interessante, nenhuma conquista foi feita em relação aos sistemas MIMO ou redução de ordem usando funções ortonormais para extrair com precisão o comportamento da variável de controle. Para este propósito, como sistemas de alta ordem são problemáticos para aplicações práticas, funções ortonormais de Laguerre com estabilidade prescrita foram empregadas para projetar o GPC nesta dissertação.

Na Tabela 2, resumem-se as principais características dos trabalhos apresentados na Seção 1.3.2 conforme a aplicação, método de otimização e implementação em *hardware* dos controladores MPC/GPC.

Tabela 2 – Resumo dos trabalhos relacionados com MPC/GPC.

Referência	**Aplicação**	**Método de otimização**	**Implementação em *Hardware***
(SAEED; WANG; FERNANDO, 2022)	Controlar um conversor PSFB	Algoritmo de Hildreth com restrição na variável de controle e funções de Laguerre	Texas Instrument F28069M InstaSpin DSP
(ROSSITER; WANG; VALENCIA-PALOMO, 2010)	Artigo teórico que trata a estabilidade e facticidade do MPC baseado em funções de Laguerre	Algoritmo *Maximal Controlled Admissible Set* (MCAS)	Não
(CHEN et al., 2019)	Rastreamento de trajetória para robôs móveis	Programação quadrática com rede neural primal-dual (MPC adaptativo)	Sim

(continua)

Referência	**Aplicação**	**Método de otimização**	**Implementação em *Hardware***
(YAMASHITA; ZANIN; ODLOAK, 2016)	Controlar o sistema de referência do fracionador de óleo pesado da companhia *Shell*	*Lexicographic Tuning Technique* (LTT) e *Compromise Tuning Technique* (CTT)	Não
(PECCIN et al., 2020)	Apresentar uma solução baseada no método de multiplicadores de direção alternada com controle preditivo generalizado	Programação quadrática	FPGA
(MISHRA et al., 2021)	Controlar a corrente de um *Permanent Magnet Synchronous Motor* (PMSM) trifásico	Conjunto de controle finito	FPGA
(LIEGMANN; KARAMANAKOS; KENNEL, 2022)	Controlar a frequência de um inversor trifásico	Conjunto de controle finito	FPGA
(LUCIA et al., 2018)	Explorar as operações matemáticas necessárias do controle MPC em *hardware*	Programação quadrática	FPGA
(CORDERO et al., 2021)	Algoritmo com operador em diferenças discretas para o rastreamento de referências em rampa	DMPC	DSP dSPACE DS1104
(CORDERO et al., 2021)	Algoritmo com operador em diferenças discretas para o rastreamento de referências polinomiais de ordem m	DMPC	DSP dSPACE DS1104

(continua)

Referência	**Aplicação**	**Método de otimização**	**Implementação em *Hardware***
(CORDERO et al., 2022)	Algoritmo com operador em diferenças discretas para o rastreamento de referências senoidais	DMPC	DSP dSPACE DS1104

1.4 OBJETIVO DO TRABALHO

- Desenvolver uma nova topologia de PIDOF que resolve o problema de *undershoot* do Boost em malha de tensão garantindo que o sistema em malha fechada tenha derivada inicial nula;
- Deduzir de equações algébricas para os parâmetros de ganho do PIDOF em relação aos graus fracionários ao resolver as equações de especificação de frequência de cruzamento de ganho, especificação de margem de fase e especificação de amortecimento iso;
- Elaborar um algoritmo ED juntamente com o filtro Oustaloup refinado para encontrar soluções quase ótimas das ordens fracionárias integro-diferenciais, ou seja, λ e μ, para um PIDOF;
- Utilizar a norma L2 para reduzir a ordem do PIDOF e assim melhor acomodá-lo em *hardware*, sem perder a resposta dinâmica inicial do sistema. Criar e depurar um código em VHDL do PIDOF reduzido usando bibliotecas matriciais e o *software* QUARTUS®. Testar o PIDOF em VHDL com o FIL do MATLAB®/Simulink e validar o controle embarcado em FPGA com um teste em bancada para o conversor Boost;
- Projetar um DMPC baseado em funções de Laguerre e grau de estabilidade prescrito com restrições nas variáveis de controle para um conversor Boost enquanto controla o ciclo de trabalho de acionamento do seu Mosfet;
- Criar e depurar um código em VHDL do DMPC usando bibliotecas matriciais e o *software* QUARTUS®. Testar o DMPC em VHDL com o FIL do MATLAB®/Simulink e validar o controle embarcado em FPGA com um teste em bancada para o conversor Boost;
- Comparar os resultados de performance, velocidade, robustez e custo computacional entre a implementação do PIDOF-Boost e do MPC-Boost;

- Desenvolver um controle preditivo generalizado (GPC - do inglês, *Generalized Predictive Control*) digital com funções ortonormais de Laguerre, grau de estabilidade prescrito, restrições na variável de entrada e diminuição do número de condição da Hessiana do sistema aumentado, capaz de rastrear referências polinomiais de grau arbitrário $m-1$, quando a planta possuir função de transferência de realização mínima e não possuir zeros em z = 1. Por fim, criar e depurar um código em VHDL do GPC tipo m usando bibliotecas matriciais e o *software* QUARTUS®. Testar o GPC tipo m em VHDL com o FIL do MATLAB®/Simulink.

1.5 MATERIAIS E MÉTODOS

Para o desenvolvimento deste trabalho foram utilizados os materiais listados a seguir:

- Um notebook com as configurações: 16GB 1200MHz DDR4; 6MB L3cache; 2GB GDDR5 NVIDIA GeForce MX130 e; 1.6 GHz Intel® Core i5-8265U;
- Placa FPGA DE2-115 com as especificações: (nome) EP4CE115F29C7; (Tensão do núcleo) 1.2V; (elementos lógicos) 114480; (I/Os) 529; (GPIOs) 529; (Bits de memória) 3981312; (multiplicadores de 9-bits) 532; (PLLs) 4; (*clocks* globais) 20;
- Um conversor Boost desenvolvido pelo BATLAB cujos detalhes são apresentados no Capítulo 4;
- Um conversor AD Dual-channel AD9226 (12 bits);
- Um sistema de condicionamento de energia com um sensor de tensão LV 20-P;
- Um osciloscópio keyight infiniivision DSOX3054T.

A metodologia empregada é constituída pelos passos a seguir:

- Através do cálculo fracionário e a teoria desenvolvida nesta dissertação, utilizou-se o MATLAB®/Simulink para projetar o algoritmo DE para a sintonia dos parâmetros do PIDOF e sua análise de performance. De forma análoga, porém com a teoria do MPC, projetou-se um DMPC e DGPC para as plantas descritas nesta dissertação;
- Utilizou-se o *software* QUARTUS® PRIME LITE EDITION 21.1 com os pacotes de matrizes real e s_fixed, ambos disponibilizados pelo meu orientador Dr. Edson Antonio Batista. Tal *software* foi utilizado para compilar, depurar e embarcar os códigos dos controladores em VHDL no FPGA;

- Empregou-se a ferramenta FIL do MATLAB®/Simulink para validar o funcionamento e performance dos controladores em VHDL. Só depois desta validação que os controladores foram embarcados e testados em bancada.

1.6 ESTRUTURA DA DISSERTAÇÃO

Além do capítulo introdutório, este trabalho está organizado da seguinte forma:

Capítulo 2: É apresentado uma breve descrição da teoria do cálculo fracionário com a utilização dos filtros Oustaloup para a aproximação da ordem fracionária da função de transferência em ordem inteira. Também é desenvolvido uma nova abordagem para a sintonia de um controlador PIDOF com especificações em margem de ganho, margem de fase e amortecimento iso com o algoritmo ED. Por fim, apresenta-se a técnica de redução por norma L2 para sistemas SISO com atraso.

Capítulo 3: Primeiramente, é abordado o estudo de estabilidade, sem a aplicação de restrições, dos controladores MPC e GPC tipo m, multivariáveis, usando funções de Laguerre. Por fim, detalha-se a forma de funcionamento desses controladores quando as restrições na variável de entrada são inseridas e quando há uma necessidade de prescrição de estabilidade no sistema.

Capítulo 4: Neste capítulo, demonstram-se os testes realizados com simulações e *hardware-in-the-loop* com o PIDOF e o MPC propostos para o controle do ciclo de trabalho de um conversor Boost através da ferramenta FIL e, posteriormente, executam-se testes experimentais dos controladores quando embarcados em FPGA para um teste em bancada com o mesmo conversor. Por fim, realizam-se simulações com o controlador GPC tipo 3 com diferentes números de funções de Laguerre, restrições, e prescrição de estabilidade para controlar uma planta quando submetida a uma referência polinomial de ordem menor ou igual a 2.

Capítulo 5: Apresentam-se as conclusões finais, sendo discutidos os resultados, as contribuições e os trabalhos futuros que podem ser realizados a partir dos assuntos e desenvolvimentos desta dissertação.

2 CONTROLADORES PID EM ORDEM FRACIONÁRIA

Neste capítulo, aborda-se toda a teoria de cálculo fracionário utilizada nesta dissertação, o projeto do algoritmo ED para sintonia do PIDOF e a técnica de redução norma 2 para sistemas SISO.

2.1 INTEGRAL DE RIEMMAN-LIOUVILLE E SUA TRANSFORMADA DE LAPLACE

Nesta seção, introduz-se os teoremas de Leibniz para a diferenciação de uma integral e o de Cauchy para integrais consecutivas, além de definir a função Gel'fand-Shilov que é muito importante para o cálculo da transformada de Laplace de operadores fracionários. Por fim, apresenta-se a propriedade de semigrupo da integração fracionária. Primeiramente, é necessário definir algumas notações para o entendimento dessa seção.

Definição 2.1.1. Admitindo-se que f é uma função integrável no intervalo $[a, b]$, com $a \leq t \leq b$, define-se o operador de integração, à esquerda, conforme Eq. (2.1):

$$ {}_aJ_t[f(t)] = \int_a^t f(x)dx. \tag{2.1} $$

Além disso, para funções integráveis, também vale a propriedade de semigrupo (RODRIGUES, 2015), ou seja, Eq. (2.2):

$$ J^m J^n[f(t)] = J^n J^m[f(t)] = J^{n+m}[f(t)]. \tag{2.2} $$

Definição 2.1.2. Admitindo-se que f é uma função diferenciável no domínio definido, define-se o operador diferencial com base na Eq. (2.3):

$$ D[f(t)] = \frac{df(t)}{dt}. \tag{2.3} $$

Através do teorema fundamental do cálculo (TFC), é importante lembrar que o operador diferencial é o inverso à esquerda do operador de integração em Eq. (2.4)

$$ DJ[f(t)] = f(t). \tag{2.4} $$

Por fim, tomando f absolutamente integrável em $[a, b]$, então (RODRIGUES, 2015) garante que $[D^n J^n]f(t) = f(t)$.

Para chegar à integral de Riemman-Liouville, primeiramente deve-se enunciar alguns teoremas do cálculo inteiro para assim extrapolar os resultados para a ordem fracionária. A base para a definição da integral proposta por Riemman-Liouville advém da continuação analítica do Teorema de Cauchy para integrais consecutivas com o auxílio do teorema de Leibniz para a diferenciação de uma integral (MILLER; ROSS, 1993; OLIVEIRA, 2014).

Teorema 2.1.1 (Teorema de Leibniz para Diferenciação de uma Integral)**.** *Seja f(t,x) uma função de duas variáveis, diferenciável em t, e sejam u(t) e v(t) funções diferenciáveis. Então, vale a relação em Eq.* (2.5)

$$\frac{d}{dt}\int_{u(t)}^{v(t)} f(t,x)dx = \int_{u(t)}^{v(t)} \frac{\partial}{\partial t} f(t,x)dx + f(t,v(t))\frac{dv(t)}{dt} - f(t,u(t))\frac{du(t)}{dt} \tag{2.5}$$

Demonstração. Escrevendo a integral $\int_{u(t)}^{v(t)} f(t,x)dx$ como uma função de 3 variáveis, todas com domínio em t, tem-se a Eq. (2.6):

$$P(t,u,v) = \int_u^v f(t,x)dx. \tag{2.6}$$

Com base no TFC, i.e., $\frac{d}{dt}\int_a^t f(x)dx = f(t)$, pode-se calcular a derivada total da função $P(t,u,v)$ em relação a t através da regra da cadeia, obtendo-se a Eq. (2.7):

$$\begin{aligned}\frac{d}{dt}P(t,u,v) =& \frac{\partial P}{\partial t}\frac{\partial t}{\partial t} + \frac{\partial P}{\partial u}\frac{\partial u}{\partial t} + \frac{\partial P}{\partial v}\frac{\partial v}{\partial t} \\ =& (\frac{\partial}{\partial t}\int_u^v f(t,x)dx)\frac{dt}{dt} + (\frac{\partial}{\partial u}\int_u^v f(t,x)dx)\frac{du}{dt} + (\frac{\partial}{\partial v}\int_u^v f(t,x)dx)\frac{dv}{dt} \\ =& \int_u^v \frac{\partial}{\partial t} f(t,x)dx + f(t,v(t))\frac{dv}{dt} - f(t,u(t))\frac{du}{dt}.\end{aligned} \tag{2.7}$$

□

Para a demonstração do teorema de Cauchy para integrais consecutivas (OLIVEIRA, 2014), utilizar-se-á um caso particular da (2.5) com $u(t) = a$ e $v(t) = t$. Dessa forma, (2.5) se transforma na Eq. (2.8):

$$\frac{d}{dt}\int_a^t f(t,x)dx = \int_a^t \frac{\partial}{\partial t} f(t,x)dx + f(t,t). \tag{2.8}$$

Teorema 2.1.2 (Teorema de Cauchy para Integrais Consecutivas)**.** *Seja f(t,x) uma função integrável n vezes no intervalo* $[a,t] \in \mathbb{R}$*, com* $n \in \mathbb{N}$*. Então, é válida a relação na Eq.* (2.9)

$${}_aJ_t^n(f(x)) = \int_a^t dx_n \int_a^{x_n} dx_{n-1} \cdots \int_a^{x_2} f(x_1)dx_1 = \frac{1}{\Gamma(n)}\int_a^t (t-x)^{n-1} f(x)dx, \tag{2.9}$$

onde ${}_aJ_t^n$ *é o operador integração aplicado n vezes.*

Demonstração. Seja $p(t) = \int_a^t (t-x)^{n-1} f(x)dx$ e sabendo que o operador D é inverso de J, à esquerda, pode-se provar que a relação é correta aplicando-se D^n em (2.9) a fim de se encontrar $f(x)$. Derivando $p(t)$ em relação a t e utilizando (2.8), tem-se a Eq. (2.10)

$$\begin{aligned} D[p(t)] =& \frac{d}{dt}\int_a^t (t-x)^{n-1} f(x)dx \\ =& \int_a^t \frac{\partial}{\partial t}((t-x)^{n-1} f(x))dx + (t-x)^{n-1} f(x)|_{x=t} \\ =& (n-1)\int_a^t (t-x)^{n-2} f(x)dx. \end{aligned} \tag{2.10}$$

Observe que o operador diferenciação sempre anulará o segundo fator do resultado como visto em (2.10). Dessa forma, derivando-se $p(t)$ n vezes e pelo TFC, obtém-se a Eq. (2.11):

$$D^n[p(t)] = (n-1)!\frac{d}{dt}\int_a^t f(x))dx = \Gamma(n) f(t). \tag{2.11}$$

Portanto, nota-se que $p(t)/\Gamma(n)$ é a integral de ordem n de $f(t)$ e a igualdade (2.9) é válida para $n \in \mathbb{N}$.

□

Generalizando (2.9) para um número α qualquer, obtém-se a continuação analítica da fórmula de Cauchy para integrais consecutivas, a qual é a definição, dada abaixo, do operador de integração fracionário segundo Riemann-Liouville.

Definição 2.1.3 (Operador Integração Fracionário de Riemann-Liouville)**.** Seja $\alpha \in \mathbb{C}$ com $Re(\alpha) > 0$ e f uma função integrável em qualquer subintervalo de $[a, b]$. Logo, para $t \in [a, b]$ define-se a integral fracionária de Riemann-Liouville de ordem α em $f(t)$, à esquerda, como ${}_aJ_t^\alpha$ na Eq. (2.12) por (KILBAS; SRIVASTAVA; TRUJILLO, 2006):

$${}_aJ_t^\alpha f(t) = \frac{1}{\Gamma(\alpha)}\int_a^t (t-x)^{\alpha-1} f(x)dx, \ t > a. \tag{2.12}$$

Se a = 0, (2.12) resulta na Eq. (2.13):

$${}_0J_t^\alpha f(t) = J^\alpha f(t) = \frac{1}{\Gamma(\alpha)}\int_0^t (t-x)^{\alpha-1} f(x)dx, \ t > 0 \ e \ Re(\alpha) > 0. \tag{2.13}$$

Vale mencionar que também existe tal operador à direita e que as integrais são válidas para $\alpha \in \mathbb{C}$, porém não se utiliza o operador à direita nesse trabalho. Por fim, utilizar-se-á apenas $\alpha \in \mathbb{R}_+$ neste trabalho.

Com posse da definição da integral fracionária de Riemann-Liouville, busca-se o resultado do operador transformada de Laplace sobre a mesma uma vez que o objetivo deste trabalho é projetar controladores PIDOF quase ótimos no domínio da frequência. Para auxiliar, define-se a função de Gel'fand-Shilov (GEL'FAND; SHILOV, 1964).

Definição 2.1.4 (Função de Gel'fand-Shilov)**.** Admitindo-se $Re(\lambda) > 0$, tem-se a Eq. (2.14):

$$\Phi_\lambda(x) = \begin{cases} \dfrac{x^{\lambda-1}}{\Gamma(\lambda)} & , \ x > 0. \\ 0 & , \ x \leq 0 \end{cases} \tag{2.14}$$

Através de (2.13), observa-se que o resultado é equivalente à convolução de $f(t)$ com (2.14). Ou seja, a partir da definição de convolução entre 2 funções no tempo (OPPENHEIM; SCHAFER, 2010), escreve-se a Eq. (2.15):

$$\Phi_\alpha * f(t) = \frac{1}{\Gamma(\alpha)} \int_0^t (t-x)^{\alpha-1} f(x) dx = J^\alpha f(t). \tag{2.15}$$

A partir de (2.15), pode-se provar a propriedade do semigrupo para o operador integral fracionário com o Teorema 2.1.3.

Teorema 2.1.3 (Propriedade de Semigrupo para a Integral de RL)**.** *Seja f absolutamente integrável a Lebesgue em $[a, b]$, $\alpha > 0$ e $\beta > 0$, então as igualdades apresentadas na Eq. (2.16)*

$$({}_aJ_a^\alpha J^\beta f)(t) = ({}_aJ_a^\beta J^\alpha f)(t) = ({}_aJ^{\beta+\alpha} f)(t), \tag{2.16}$$

são válidas em quase todos os pontos de $[a, b]$.

Demonstração. Primeiramente, para o caso particular em que $a = 0$, sabe-se que o operador integrador fracionário pode ser dado pela convolução (2.15), dessa forma, vamos calcular a convolução de duas funções de Gel'fand-Shilov $\Phi_\alpha * \Phi_\beta$ na Eq. (2.17), com $\alpha > 0$ e $\beta > 0$.

$$\begin{aligned} \Phi_\alpha * \Phi_\beta &= \int_0^t \frac{\tau^{\alpha-1}}{\Gamma(\alpha)} \frac{(t-\tau)^{\beta-1}}{\Gamma(\beta)} d\tau \\ &= \frac{t^{\beta-1}}{\Gamma(\alpha)\Gamma(\beta)} \int_0^t \tau^{\alpha-1} (1 - \frac{\tau}{t})^{\beta-1} d\tau. \end{aligned} \tag{2.17}$$

Através da mundança de variável em (2.17), $u = \dfrac{\tau}{t}$, tem-se a Eq. (2.18)

$$\Phi_\alpha * \Phi_\beta = \frac{t^{\alpha+\beta-1}}{\Gamma(\alpha)\Gamma(\beta)} \int_0^1 u^{\alpha-1} (1-u)^{\beta-1} d\tau. \tag{2.18}$$

A integral em (2.18) define a função beta $B(\alpha,\beta) = \int_0^1 u^{\alpha-1}(1-u)^{\beta-1}d\tau$. Adicionalmente, a função B possui a seguinte propriedade (Apêndice A.1): $B(\alpha,\beta) = \dfrac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$. Dessa forma, (2.18) resulta na Eq. (2.19):

$$\begin{aligned}\Phi_\alpha * \Phi_\beta =& \frac{t^{\alpha+\beta-1}}{\Gamma(\alpha)\Gamma(\beta)}B(\alpha,\beta)\\ =& \frac{t^{\alpha+\beta-1}}{\Gamma(\alpha+\beta)}\\ =& \Phi_{\alpha+\beta}.\end{aligned} \tag{2.19}$$

Observe que o resultado seria o mesmo caso os parâmetros fossem comutados, ou seja, a convolução entre dois operadores integradores fracionários é comutativa. Tomando (2.15) e (2.19), tem-se a Eq. (2.20):

$$\begin{aligned}(J^\alpha J^\beta f)(t) =& \Phi_\alpha * J^\beta f(t) = \Phi_\alpha * \Phi_\beta * f(t)\\ =& \Phi_{\alpha+\beta} * f(t) = (J^{\beta+\alpha} f)(t)\end{aligned} \tag{2.20}$$

Para $a \in \mathbb{R}$, verificar (RODRIGUES, 2015). □

A partir de agora, os operadores integradores são tratados a partir de $t > 0$ para facilitar os cálculos. Por fim, observe que, com a propriedade da transformada de Laplace entre convolução de funções, a transformada final é o produto da transformada das funções (HAYKIN, 2001). Assim, a transformada de Laplace da integral fracionária de Riemann-Liouville de uma função é demonstrada pelo Teorema 2.1.4.

Teorema 2.1.4. *Seja $\alpha \in \mathbb{R}_+$, com f integrável em $[a,b]$ e de ordem exponencial, então a transformada de Laplace de $J^\alpha f(t)$ existe e é dada pela Eq. (2.21):*

$$L[J^\alpha f(t)] = \frac{L[f(t)]}{s^\alpha}. \tag{2.21}$$

Demonstração. Partindo de (2.15) e pela propriedade da transformada de convolução entre funções em Eq. (2.22):

$$L[J^\alpha f(t)] = L[\Phi_\alpha * f(t)] = L[\Phi_\alpha]L[f(t)]. \tag{2.22}$$

Porém, com a definição de função Γ generalizada, a transformada de Φ_α é dada pela Eq. (2.23):

$$L[\Phi_\alpha] = \frac{1}{\Gamma(\alpha)}\int_0^\infty t^{\alpha-1}e^{-st}dt. \tag{2.23}$$

Efetuando $u = st$ em (2.23) e com a definição da função Γ generalizada (Apêndice A.1), tem-se a Eq. (2.24):

$$\begin{aligned} L[\Phi_\alpha] =& \frac{1}{\Gamma(\alpha)} \int_0^\infty (\frac{u}{s})^{\alpha-1} e^{-u} \frac{du}{s} \\ =& \frac{1}{\Gamma(\alpha) s^\alpha} \int_0^\infty u^{\alpha-1} e^{-u} du \\ =& \frac{\Gamma(\alpha)}{\Gamma(\alpha) s^\alpha} = \frac{1}{s^\alpha}. \end{aligned} \tag{2.24}$$

Logo, de (2.24) em (2.22), fecha-se a prova.

□

2.2 DERIVADA SEGUNDO CAPUTO E SUA TRANSFORMADA DE LAPLACE

A derivada segundo Riemann-Liouville existe e pode ser encontrada em (TEODORO; OLIVEIRA; OLIVEIRA, 2017), contudo a definição conduz a uma transformada de Laplace que necessita de derivadas fracionárias no estado inicial do problema. De certa forma, a grande maioria dos problemas de engenharia disponibiliza apenas condições iniciais com derivadas inteiras, ficando inviável a interpretação física de uma derivada fracionária. Assim, surgiu-se a derivada fracionária de Caputo e a definição é dada por (ORTIGUEIRA, 2011).

Definição 2.2.1 (DERIVADA DE CAPUTO)**.** Seja f absolutamente integrável em $[a, b]$ e $\alpha \in \mathbb{C}$, com $\alpha > 0$, $n = [\alpha] + 1$ quando $n \notin \mathbb{N}_0$ e $n = \alpha$ quando $n \in \mathbb{N}_0$. Dessa forma, a derivada de caputo de ordem α, à esquerda, de f existe para quase todos os pontos de $[a, b]$ e é representada pela Eq. (2.25):

$${}_a^C D_t^\alpha f(t) = {}_a J_t^{n-\alpha} D^n f(t) = \frac{1}{\Gamma(n-\alpha)} \int_a^t (t-x)^{n-\alpha-1} D^n f(x) dx. \tag{2.25}$$

Neste trabalho, aborda-se apenas a derivada para $a = 0$ e $\alpha \in \mathbb{R}_+$, por conseguinte, para uma melhor notação, utilizar-se-á o operador por ${}^C D^\alpha f(t)$

Com a Definição 2.2.1, pode-se demonstrar a transformada de Laplace da derivada fracionária segundo Caputo (JARAD; ABDELJAWAD, 2020)

Teorema 2.2.1. *Seja $\alpha \in \mathbb{R}_+$ com $n - 1 < \alpha \leq n$ e $n \in \mathbb{N}$. Admitindo-se f integrável em $[a, b]$ e de ordem exponencial, então a transformada de Laplace de ${}^C D^\alpha f(t)$ existe e é dada pela Eq. (2.26):*

$$L[{}^{C}D^{\alpha}f(t)] = s^{\alpha}L[f(t)] - \sum_{k=0}^{n-1} s^{\alpha-1-k}D^{k}f(0). \tag{2.26}$$

Demonstração. Pela Definição 2.2.1 e com o auxílio da função de Gel'fand-Shilov (2.14), tem-se a Eq. (2.27):

$$\begin{aligned} L[{}^{C}D^{\alpha}f(t)] =& L[\frac{1}{\Gamma(n-\alpha)}\int_{a}^{t}(t-x)^{n-\alpha-1}D^{n}f(x)dx] \\ =& L[J^{n-\alpha}D^{n}f(t)] \\ =& L[\Phi_{n-\alpha} * D^{n}f(t)] \\ =& L[\Phi_{n-\alpha}]L[D^{n}f(t)]. \end{aligned} \tag{2.27}$$

Com o Teorema 2.1.4 e sabendo do cálculo inteiro que $L[f(t)] = s^{n}L[f(t)] - \sum_{k=0}^{n-1} s^{n-1-k}D^{k}f(0)$ (JARAD; ABDELJAWAD, 2020), (2.27) resulta na Eq. (2.28):

$$\begin{aligned} L[{}^{C}D^{\alpha}f(t)] =& \frac{1}{s^{n-\alpha}}(s^{n}L[f(t)] - \sum_{k=0}^{n-1} s^{n-1-k}D^{k}f(0)) \\ =& s^{\alpha}L[f(t)] - \sum_{k=0}^{n-1} s^{\alpha-1-k}D^{k}f(0). \end{aligned} \tag{2.28}$$

□

Por fim, vale comentar que a derivada segundo Caputo não possui a propriedade do semigrupo igual a integral de Riemann-Liouville. Entretanto, pela Definição 2.2.1 é trivial verificar que, para $k \in \mathbb{N}$ e $\alpha \in \mathbb{R}$, vale a seguinte propriedade da Eq. (2.29):

$${}^{C}D^{\alpha}D^{k}f(t) = J^{n-\alpha}D^{n}D^{k}f(t) = J^{n-\alpha}D^{n+k}f(t) = {}^{C}D^{k+\alpha}f(t). \tag{2.29}$$

2.3 FILTRO OUSTALOUP

O filtro Oustaloup tem um ajuste dos operadores íntegro-diferenciais de ordem fracionária para a ordem inteira no domínio da transformada de Laplace. Há uma mudança de ordem na função de transferência resultando que s^{α} se transforma em uma função de transferência na ordem inteira. Logo, (XUE; CHEN; ATHERTON, 2008) define o filtro Oustaloup pela Definição 2.3.1:

Definição 2.3.1 (Filtro Oustaloup)**.** Assumindo que o intervalo de frequência de ajuste entre as funções de transferências inteiras e fracionárias é $[\omega_b, \omega_h]$, $2N+1$ é a ordem do filtro e α é a ordem da diferenciação. Logo, o filtro pode ser escrito pela Eq. (2.30):

$$s^{\alpha} = K \prod_{k=-N}^{N} \frac{s + \omega_k^{'}}{s + \omega_k}, \tag{2.30}$$

sendo que os polos, zeros e ganho do filtro (2.30) são calculados pelas Eqs. (2.31a), (2.31b) e (2.31c):

$$\omega_k^{'} = \omega_b (\frac{\omega_h}{\omega_b})^{\frac{k + N + \frac{1}{2}(1-\alpha)}{2N+1}}, \tag{2.31a}$$

$$\omega_k = \omega_b (\frac{\omega_h}{\omega_b})^{\frac{k + N + \frac{1}{2}(1+\alpha)}{2N+1}}, \tag{2.31b}$$

$$K = \omega_h^{\alpha}. \tag{2.31c}$$

Contudo, a qualidade da aproximação do filtro Oustaloup pode não ser muito satisfatório em altas e baixas frequências, mais precisamente, nas frequências máximas e mínimas de ajuste $[\omega_b, \omega_h]$. Com isso, alguns algoritmos surgiram com o efeito de mitigar esse problema e um deles será demonstrado no Teorema 2.3.1 (XUE; ZHAO; CHEN, 2006).

Teorema 2.3.1. *Assumindo que o intervalo de frequência de ajuste entre as funções de transferência inteiras e fracionárias é* $[\omega_b, \omega_h]$, $0 < \alpha < 1$, $b > 0$ *e* $d > 0$. *Então, o operador fracionário* s^{α} *é estável e pode ser aproximado pela Eq.* (2.32)*:*

$$s^{\alpha} = \left(\frac{d\omega_b}{b}\right)^{\alpha} \left(\frac{ds^2 + b\omega_h s}{d(1-\alpha)s^2 + b\omega_h s + d\alpha\omega_h\omega_b}\right) \left(\frac{1 + \frac{sb}{d\omega_b}}{1 + \frac{sd}{b\omega_h}}\right)^{\alpha} \tag{2.32}$$

Demonstração. Dentro do range de frequências, α, b e d especificados, tem-se Eq. (2.33):

$$\begin{aligned} s^{\alpha} = K(s) &= \left(\frac{1 + \frac{sb}{d\omega_b}}{1 + \frac{sd}{b\omega_h}}\right)^{\alpha} \\ K(s) &= \left(\frac{bs}{d\omega_b}\right)^{\alpha} \left(1 + \frac{-ds^2 + d\omega_h\omega_b}{ds^2 + b\omega_h s}.\right)^{\alpha}. \end{aligned} \tag{2.33}$$

O operador fracionário, quando expandido pela série de Taylor, pode ser aproximado pela Eq. (2.34):

$$K(s) = \left(\frac{bs}{d\omega_b}\right)^{\alpha} \left[1 + \alpha p(s) + \frac{\alpha(\alpha+1)}{2} p^2(s) + \cdots\right], \tag{2.34}$$

com

$$p(s) = \frac{-ds^2 + d\omega_h\omega_b}{ds^2 + b\omega_h s}. \tag{2.35}$$

Isolando s^{α} de (2.34), obtém-se a Eq. (2.36):

$$\begin{aligned} s^{\alpha} &= \frac{(d\omega_b)^{\alpha} b^{-\alpha}}{\left[1 + \alpha p(s) + \frac{\alpha(\alpha+1)}{2} p^2(s) + \cdots\right]} K(s) \\ s^{\alpha} &= \frac{(d\omega_b)^{\alpha} b^{-\alpha}}{\left[1 + \alpha p(s) + \frac{\alpha(\alpha+1)}{2} p^2(s) + \cdots\right]} \left(\frac{1 + \frac{sb}{d\omega_b}}{1 + \frac{sd}{b\omega_h}}\right)^{\alpha}. \end{aligned} \tag{2.36}$$

Desprezando os termos acima da primeira ordem da série de Taylor de (2.36), (2.36) resulta na Eq. (2.37):

$$s^{\alpha} \approx \left(\frac{d\omega_b}{b}\right)^{\alpha} \left(\frac{ds^2 + b\omega_h s}{d(1-\alpha)s^2 + b\omega_h s + d\alpha\omega_h\omega_b}\right) \left(\frac{1 + \frac{sb}{d\omega_b}}{1 + \frac{sd}{b\omega_h}}\right)^{\alpha}. \tag{2.37}$$

Para fechar a prova, deve-se verificar a estabilidade de (2.37), i.e., todos os polos no semiplano esquerdo do plano complexo. De imediato, $-b\omega_h/d$ é negativo, restando analisar as raízes da Eq. (2.38)

$$d(1-\alpha)s^2 + b\omega_h s + d\alpha\omega_h\omega_b. \tag{2.38}$$

Observe que, para $0 < \alpha < 1$, o produto e a soma das raízes de (2.38) são positivas e negativas, respectivamente, e as raízes de (2.38) possuem parte real negativa. Logo, (2.37) é estável dentro do range (ω_b, ω_h). □

Vale notar que o Teorema 2.3.1 é válido apenas para $0 < \alpha < 1$. Assim, pode-se utilizar a propriedade 2.29 para calcular uma derivada de ordem 1,7, por exemplo, considerando $D^k f(0) = 0$ para $k = 0, 1$, ou seja (Eq. (2.39)):

$$s^{1,7}L[f(t)] = s^{0,7+1}L[f(t)] = L[{}^{C}D^{0,7}Df(t)] = L[{}^{C}D^{1,7}f(t)]. \tag{2.39}$$

De forma análoga ao filtro Oustaloup, após inúmeros testes de parâmetros b e d, com refinamentos em (2.32) na forma de $\omega_h * \omega_b = 1$, mesmo com tais valores não sendo os utilizados exatamente nas aproximações, o filtro Oustaloup refinado (2.32) também

pode ser aproximado por um modelo racional contínuo através da seguinte definição (XUE; CHEN; ATHERTON, 2008).

Definição 2.3.2 (Filtro Oustaloup Refinado)**.** Assumindo que o intervalo de frequência de ajuste entre as funções de transferências inteiras e fracionárias é $[\omega_b, \omega_h]$, $2N+1$ é a ordem do filtro e $0 < \alpha < 1$ é a ordem da diferenciação. Logo, a modelagem do filtro dado em (2.32) pode ser escrita pela Eq. (2.40):

$$s^\alpha \approx K\left(\frac{ds^2 + b\omega_h s}{d(1-\alpha)s^2 + b\omega_h s + d\alpha}\right)\prod_{k=-N}^{N}\frac{s+\omega_k^{'}}{s+\omega_k}, \tag{2.40}$$

onde $\omega_k^{'}$, ω_k e K de (2.40) são calculados pelas Eqs. (2.41a), (2.41b) e (2.41c):

$$\omega_k^{'} = \omega_b(\frac{\omega_h}{\omega_b})^{\frac{k+N+\frac{1}{2}(1-\alpha)}{2N+1}}, \tag{2.41a}$$

$$\omega_k = \omega_b(\frac{\omega_h}{\omega_b})^{\frac{k+N+\frac{1}{2}(1+\alpha)}{2N+1}}, \tag{2.41b}$$

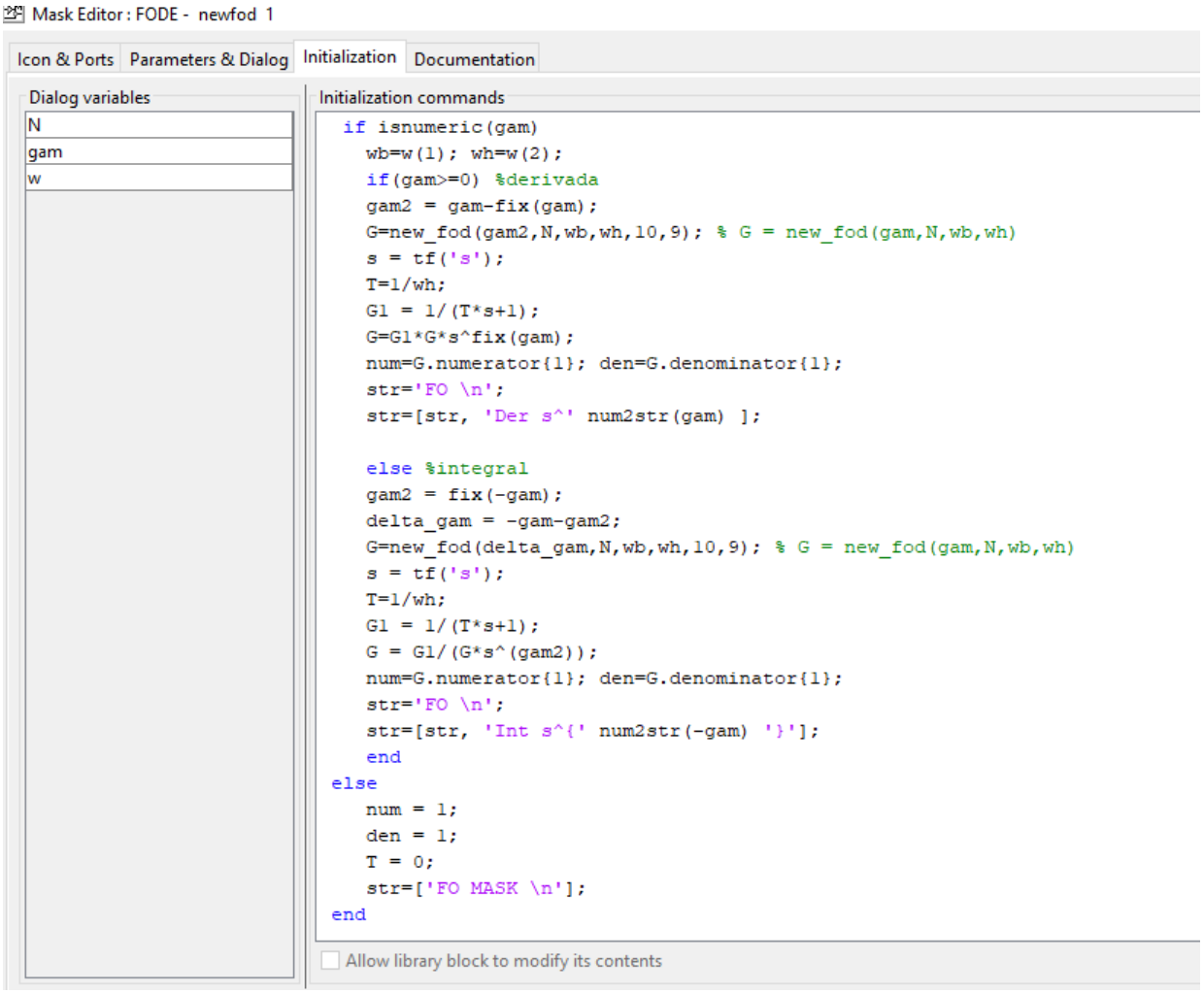
$$K = (\frac{d\omega_h}{b})^\alpha. \tag{2.41c}$$

Vale ressaltar que a aproximação (2.40) possui boa resolução quando $b = 10$ e $d = 9$ e que a Definição 2.3.2 destoa do Teorema 2.3.1 em alguns parâmetros, pois houve otimização do filtro na literatura (XUE; CHEN; ATHERTON, 2008). Para as simulações das derivadas e integrais fracionárias com o filtro Oustaloup refinado, criou-se um bloco máscara no ambiente Simulink com a função "new_fod()" (2.40). Na Figura 1, descreve-se o código de inicialização da máscara com (2.40). O bloco operador integro-diferencial fica disposto na Figura 2.

2.4 MODELAGEM DE CONTROLADORES PID EM ORDEM FRACIONÁRIA

Para um dado sinal de entrada ou ponto de ajuste $r(t)$, o objetivo do sistema de controle é trazer a saída o mais próximo possível do sinal de entrada, onde assume-se que tal sinal permaneça constante na janela de otimização. Esse objetivo é então traduzido a um projeto para encontrar o "melhor" vetor $\mathbf{x}$ de parâmetros de controle tal que uma função de erro entre o ponto de ajuste e a saída prevista seja minimizada. Essa forma de otimização é descrita no tempo e existem inúmeras funções custo que são utilizadas na

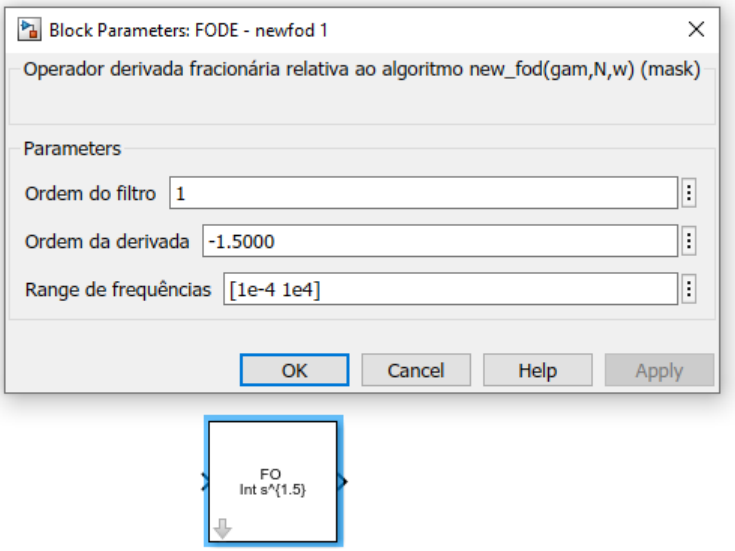
Figura 1 – Inicialização da máscara do operador integro-diferencial.



Fonte: o autor

literatura em conjunto com algoritmos heurísticos para encontrar soluções quase-ótimas de $\mathbf{x}$ como IAE, integral do erro absoluto multiplicado pelo tempo (ITAE - do inglês, *Integral of Time multiplied Absolute Error*), integral do erro quadrático (ISE - do inglês, *Integral of Square Error*) e integral do tempo multiplicada pelo erro ao quadrado (ITSE - do inglês, *Integral of Time multiplied Square Error*). Os algoritmos heurísticos realizam buscas de soluções, os quais não garantem encontrar a solução ótima de um problema, mas são capazes de retornar uma solução de qualidade em um tempo adequado para as necessidades da aplicação. Adicionalmente, através do estudo de caso de cada problema, pode-se inserir restrições ao algoritmo de forma que o mesmo tenha uma carga computacional reduzida. Além disso, tratando-se de problemas de controle no domínio da frequência, as funções custo podem também ser as mais variadas possíveis, contendo margem de fase, margem de ganho, frequência de sensibilidade e de corte. Dessa forma, técnicas são abordadas nesta

Figura 2 – Máscara integro-diferencial com filtro Oustaloup refinado.

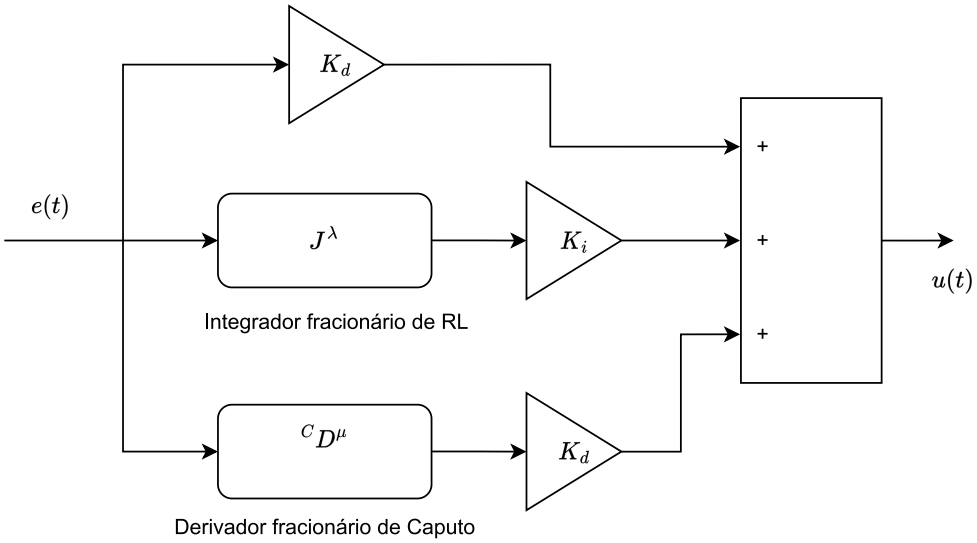


Fonte: o autor

seção com o intuito de encontrar parâmetros quase ótimos para a solução do controlador PID de ordem fracionária (PIDOF).

Inicialmente, na Figura 3, mostra-se o diagrama de blocos de um PIDOF em paralelo. Dessa forma, verifica-se que a equação do controlador fracionário no domínio da transformada de Laplace "s", considerando $D^k f(0) = 0$ para todos os $0 \leq k \leq n-1$ existentes, é dada pela Eq. (2.42):

Figura 3 – Diagrama de blocos do PIDOF em paralelo.



Fonte: o autor

$$C(s) = \frac{U(s)}{E(s)} = K_p + K_i s^{-\lambda} + K_d s^{\mu}. \tag{2.42}$$

Observe que o PIDOF possui 5 variáveis denominadas por K_p, K_i, K_d, λ e μ. Em relação ao controlador PID canônico, o PIDOF oferece 2 graus de liberdade sintonizáveis a mais com a adição dos graus fracionários λ e μ. Isso é muito importante para a busca de controladores mais robustos quanto à frequências elevadas e baixas através das funções de transferência de sensibilidade complementar e de supressão ao distúrbio de carga, respectivamente. (MONJE et al., 2008) propôs um método de otimização para ajuste do PIDOF com cinco equações, sendo uma delas o critério de robustez para variações de ganho em malha fechada, mais conhecido como "amortecimento iso" e frequentemente citado como o mais atrativo recurso de controle em ordem fracionária. Para ajustar o controlador, é necessário especificar o ângulo da margem de fase (ϕ_m), a frequência angular para que a magnitude do ganho seja unitária (ω_{gc}), a magnitude A (em dB) da atenuação do ruído para frequências maiores que ω_t e a magnitude B (em dB) da supressão ao distúrbio de carga para frequências menores que ω_s. Definindo arg(.) como o argumento da função de transferência e $G(j\omega_{gc})$ a função de transferência do sistema a ser controlado, as condições apresentadas em (MONJE et al., 2008) são dadas pelas Eqs. (2.43a), (2.43b), (2.43c), (2.43d) e (2.43e):

- Frequência de corte ω_{gc} para a FTMA:

$$|C(j\omega_{gc})G(j\omega_{gc})| = 1; \tag{2.43a}$$

- Ângulo da margem de fase ϕ_m para a FTMA:

$$arg(C(j\omega_{gc})G(j\omega_{gc})) = -\pi + \phi_m; \tag{2.43b}$$

- Variação da fase da FTMA nula em ω_{gc} (amortecimento iso):

$$\frac{d}{d\omega}(arg(C(j\omega)G(j\omega)))_{\omega=\omega_{gc}} = 0; \tag{2.43c}$$

- Função de transferência de sensibilidade complementar:

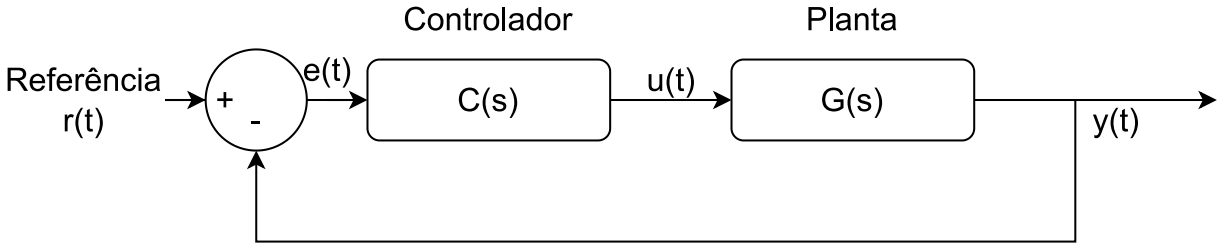
$$|T(j\omega)| = \left|\frac{C(j\omega)G(j\omega)}{1 + C(j\omega)G(j\omega)}\right|_{dB} \leq A\ dB,\ \forall\ \omega \geq \omega_t; \tag{2.43d}$$

- Função de transferência de supressão ao distúrbio de carga:

$$|S(j\omega)| = \left|\frac{1}{1 + C(j\omega)G(j\omega)}\right|_{dB} \leq B\ dB\ ,\forall\ \omega \leq \omega_s. \tag{2.43e}$$

Sem perda de generalidade, pode-se escrever a função de transferência do sistema a ser controlado $G(s)$ em função do seu módulo e fase, ou seja, $G(s) = |G(s)|e^{j\phi_{G(s)}}$. Dessa forma, através da Figura 4, verifica-se que as equações (2.43a) - (2.43b) são obtidas pelo denominador da Eq. (2.44) e estão relacionadas quanto a margem de estabilidade tanto em rotação e ganho que o diagrama de Nyquist possui para não atingir o ponto (-1,0) do plano s. Além disso, a equação (2.43c) mostra que a fase do sistema é plana no ponto de sintonia, implicando em um sistema mais robusto para variações de ganho em malha fechada. Para sistemas que exibem propriedade de amortecimento iso, os *overshoots* das respostas ao degrau em malha fechada permanecerão quase constantes para diferentes valores do ganho do controlador. Isso garantirá que o sistema de malha fechada seja robusto para variações de ganho (FLORES et al., 2020).

Figura 4 – Diagrama de blocos do FTMF de um sistema G(s).



Fonte: o autor

$$FTMF(s) = \frac{G(s)C(s)}{1 + G(s)C(s)}. \tag{2.44}$$

Manipulando a equação (2.43a) com (2.42), obtém-se a Eq. (2.45):

$$[K_p + (K_i\omega_{gc}^{-\lambda}\cos(\frac{\pi\lambda}{2}) + K_d\omega_{gc}^{\mu}\cos(\frac{\pi\mu}{2}))]^2 + [K_d\omega_{gc}^{\mu}\sin(\frac{\pi\mu}{2}) - K_i\omega_{gc}^{-\lambda}\sin(\frac{\pi\lambda}{2})]^2 = \frac{1}{G^2}. \tag{2.45}$$

Sendo, $G = |G(j\omega_{gc})|$ e $\phi_g = \phi_G(j\omega_{gc})$. Através da propriedade dos números complexos $arg(zw) = arg(z) + arg(w)$, busca-se K_p na Eq. (2.46) ao aplicar essa propriedade em (2.43b).

$$
\begin{aligned}
arg(C(j\omega_{gc})G(j\omega_{gc})) &= -\pi + \phi_m \\
\therefore arg(C(j\omega_{gc})) + arg(G(j\omega_{gc})) &= -\pi + \phi_m \\
\therefore \phi_g + \arctan\left(\frac{K_d\omega_{gc}^{\mu}\sin(\frac{\pi\mu}{2}) - K_i\omega_{gc}^{-\lambda}\sin(\frac{\pi\lambda}{2})}{K_p + (K_i\omega_{gc}^{-\lambda}\cos(\frac{\pi\lambda}{2}) + K_d\omega_{gc}^{\mu}\cos(\frac{\pi\mu}{2}))}\right) &= -\pi + \phi_m \\
\therefore \frac{K_d\omega_{gc}^{\mu}\sin(\frac{\pi\mu}{2}) - K_i\omega_{gc}^{-\lambda}\sin(\frac{\pi\lambda}{2})}{K_p + (K_i\omega_{gc}^{-\lambda}\cos(\frac{\pi\lambda}{2}) + K_d\omega_{gc}^{\mu}\cos(\frac{\pi\mu}{2}))} &= \tan\underbrace{(\phi_m - \phi_g - \pi)}_{\phi_x} \\
\therefore K_p = \frac{1}{\tan\phi_x}[K_d\omega_{gc}^{\mu}(\sin(\frac{\pi\mu}{2}) - \cos(\frac{\pi\mu}{2})\tan\phi_x) - K_i\omega_{gc}^{-\lambda}(\sin(\frac{\pi\lambda}{2}) - \cos(\frac{\pi\lambda}{2})\tan\phi_x)] \\
\therefore K_p = \frac{1}{\sin\phi_x}[K_d\omega_{gc}^{\mu}\sin(\frac{\pi\mu}{2} - \phi_x) - K_i\omega_{gc}^{-\lambda}\sin(\frac{\pi\lambda}{2} + \phi_x)].
\end{aligned}
\tag{2.46}
$$

Substituindo (2.46), K_p, em (2.45), tem-se a Eq. (2.47):

$$
\begin{aligned}
&[K_d\omega_{gc}^{\mu}\sin(\frac{\pi\mu}{2} - \phi_x) - K_i\omega_{gc}^{-\lambda}\sin(\frac{\pi\lambda}{2} + \phi_x) + (K_i\omega_{gc}^{-\lambda}\cos(\frac{\pi\lambda}{2}) + K_d\omega_{gc}^{\mu}\cos(\frac{\pi\mu}{2}))\sin\phi_x]^2 \\
&\quad +(\sin\phi_x)^2[K_d\omega_{gc}^{\mu}\sin(\frac{\pi\mu}{2}) - K_i\omega_{gc}^{-\lambda}\sin(\frac{\pi\lambda}{2})]^2 = (\frac{\sin\phi_x}{G})^2 \\
\therefore\ &[K_d\omega_{gc}^{\mu}(\sin(\frac{\pi\mu}{2} - \phi_x) + \cos(\frac{\pi\mu}{2})\sin\phi_x) + K_i\omega_{gc}^{-\lambda}(\cos(\frac{\pi\lambda}{2})\sin\phi_x - \sin(\frac{\pi\lambda}{2} + \phi_x))]^2 \\
&\quad +(\sin\phi_x)^2[K_d\omega_{gc}^{\mu}\sin(\frac{\pi\mu}{2}) - K_i\omega_{gc}^{-\lambda}\sin(\frac{\pi\lambda}{2})]^2 = (\frac{\sin\phi_x}{G})^2 \\
\therefore\ &[K_d\omega_{gc}^{\mu}\sin(\frac{\pi\mu}{2}) - K_i\omega_{gc}^{-\lambda}\sin(\frac{\pi\lambda}{2})]^2(\cos\phi_x)^2 + (\sin\phi_x)^2[K_d\omega_{gc}^{\mu}\sin(\frac{\pi\mu}{2}) - K_i\omega_{gc}^{-\lambda}\sin(\frac{\pi\lambda}{2})]^2 \\
&\quad = (\frac{\sin\phi_x}{G})^2 \\
&\therefore [K_d\omega_{gc}^{\mu}\sin(\frac{\pi\mu}{2}) - K_i\omega_{gc}^{-\lambda}\sin(\frac{\pi\lambda}{2})]^2 = (\frac{\sin\phi_x}{G})^2 \\
&\therefore K_d = \frac{K_i\omega_{gc}^{-\lambda}\sin(\frac{\pi\lambda}{2}) \pm |\sin\phi_x|/G}{\omega_{gc}^{\mu}\sin(\frac{\pi\mu}{2})}.
\end{aligned}
\tag{2.47}
$$

Agora, resta manipular (2.43c). Utilizando da mesma propriedade da fase do produto de dois números complexos na Eq. (2.48):

$$\frac{d}{d\omega}(arg(C(j\omega)G(j\omega)))_{\omega=\omega_{gc}} = 0$$

$$\therefore \frac{d}{d\omega}(\phi_G(j\omega) + \arctan\left(\frac{K_d\omega_{gc}^{\mu}\sin(\frac{\pi\mu}{2}) - K_i\omega_{gc}^{-\lambda}\sin(\frac{\pi\lambda}{2})}{K_p + K_i\omega_{gc}^{-\lambda}\cos(\frac{\pi\lambda}{2}) + K_d\omega_{gc}^{\mu}\cos(\frac{\pi\mu}{2})}\right))_{\omega=\omega_{gc}} = 0$$

$$\therefore \frac{d\phi_g}{d\omega} + \frac{d}{d\omega}\arctan\underbrace{\left(\frac{K_d\omega_{gc}^{\mu}\sin(\frac{\pi\mu}{2}) - K_i\omega_{gc}^{-\lambda}\sin(\frac{\pi\lambda}{2})}{K_p + K_i\omega_{gc}^{-\lambda}\cos(\frac{\pi\lambda}{2}) + K_d\omega_{gc}^{\mu}\cos(\frac{\pi\mu}{2})}\right)_{\omega=\omega_{gc}}}_{A} = 0 \tag{2.48}$$

$$\therefore \left.\frac{d\arctan(A)}{d\omega}\right|_{\omega=\omega_{gc}} = -\frac{d\phi_g}{d\omega},$$

com $\frac{d}{d\omega}(\phi_G(j\omega_{gc})) = \frac{d\phi_g}{d\omega}$. Efetuando a derivada de $\arctan(A)$ em (2.48), obtém-se a Eq. (2.49):

$$\frac{1}{1+A(j\omega_{gc})^2}\left.\frac{dA}{d\omega}\right|_{\omega=\omega_{gc}} = -\frac{d\phi_g}{d\omega} \rightarrow \frac{(denA(j\omega_{gc}))^2}{(numA(j\omega_{gc}))^2 + (denA(j\omega_{gc}))^2}\left.\frac{dA}{d\omega}\right|_{\omega=\omega_{gc}} = -\frac{d\phi_g}{d\omega}. \tag{2.49}$$

Porém, note que a Eq. (2.45) pode ser representada pela Eq. (2.50) :

$$(numA(j\omega_{gc}))^2 + (denA(j\omega_{gc}))^2 = 1/G^2. \tag{2.50}$$

Por outro lado, desenvolvendo a derivada de A em função de ω (Eq. (2.50)), resulta-se na Eq. (2.51):

$$\left.\frac{dA}{d\omega}\right|_{\omega=\omega_{gc}} = \frac{[K_d\mu\omega_{gc}^{\mu-1}\sin(\frac{\pi\mu}{2}) + K_i\lambda\omega_{gc}^{-\lambda-1}\sin(\frac{\pi\lambda}{2})]denA - numA[K_d\mu\omega_{gc}^{\mu-1}\cos(\frac{\pi\mu}{2}) - K_i\lambda\omega_{gc}^{-\lambda-1}\cos(\frac{\pi\lambda}{2})]}{(denA)^2}. \tag{2.51}$$

Substituindo (2.50) e (2.51) em (2.49), tem-se a Eq. (2.52):

$$[K_d\mu\omega_{gc}^{\mu-1}\sin(\frac{\pi\mu}{2}) + K_i\lambda\omega_{gc}^{-\lambda-1}\sin(\frac{\pi\lambda}{2})]denA - numA[K_d\mu\omega_{gc}^{\mu-1}\cos(\frac{\pi\mu}{2}) - K_i\lambda\omega_{gc}^{-\lambda-1}\cos(\frac{\pi\lambda}{2})] = -\frac{1}{G^2}\frac{d\phi_g}{d\omega}. \tag{2.52}$$

Observe que $denA = numA \tan\phi_x$. Aplicando isso em (2.52), tem-se a Eq. (2.53):

$$\begin{aligned}
&[K_d\mu\omega_{gc}^{\mu-1}\sin(\frac{\pi\mu}{2}) + K_i\lambda\omega_{gc}^{-\lambda-1}\sin(\frac{\pi\lambda}{2})]\cos\phi_x - \sin\phi_x[K_d\mu\omega_{gc}^{\mu-1}\cos(\frac{\pi\mu}{2}) - K_i\lambda\omega_{gc}^{-\lambda-1}\cos(\frac{\pi\lambda}{2})] \\
&\qquad = -\frac{1}{G^2}\frac{d\phi_g}{d\omega}\frac{\sin\phi_x}{numA} \\
&\therefore K_d\mu\omega_{gc}^{\mu-1}\sin(\frac{\pi\mu}{2} - \phi_x) + K_i\lambda\omega_{gc}^{-\lambda-1}\sin(\frac{\pi\lambda}{2} + \phi_x) = -\frac{1}{G^2}\frac{d\phi_g}{d\omega}\frac{\sin\phi_x}{numA}.
\end{aligned} \tag{2.53}$$

Porém, de (2.47), $numA = \pm\dfrac{|\sin\phi_x|}{G}$. Portanto, (2.53) se reduz a Eq. (2.54):

$$K_d\mu\omega_{gc}^{\mu}\sin(\frac{\pi\mu}{2} - \phi_x) + K_i\lambda\omega_{gc}^{-\lambda}\sin(\frac{\pi\lambda}{2} + \phi_x) = \mp\frac{\omega_{gc}}{G}\frac{d\phi_g}{d\omega}\frac{\sin\phi_x}{|\sin\phi_x|}. \tag{2.54}$$

Substituindo (2.47) em (2.54) resulta na Eq. (2.55):

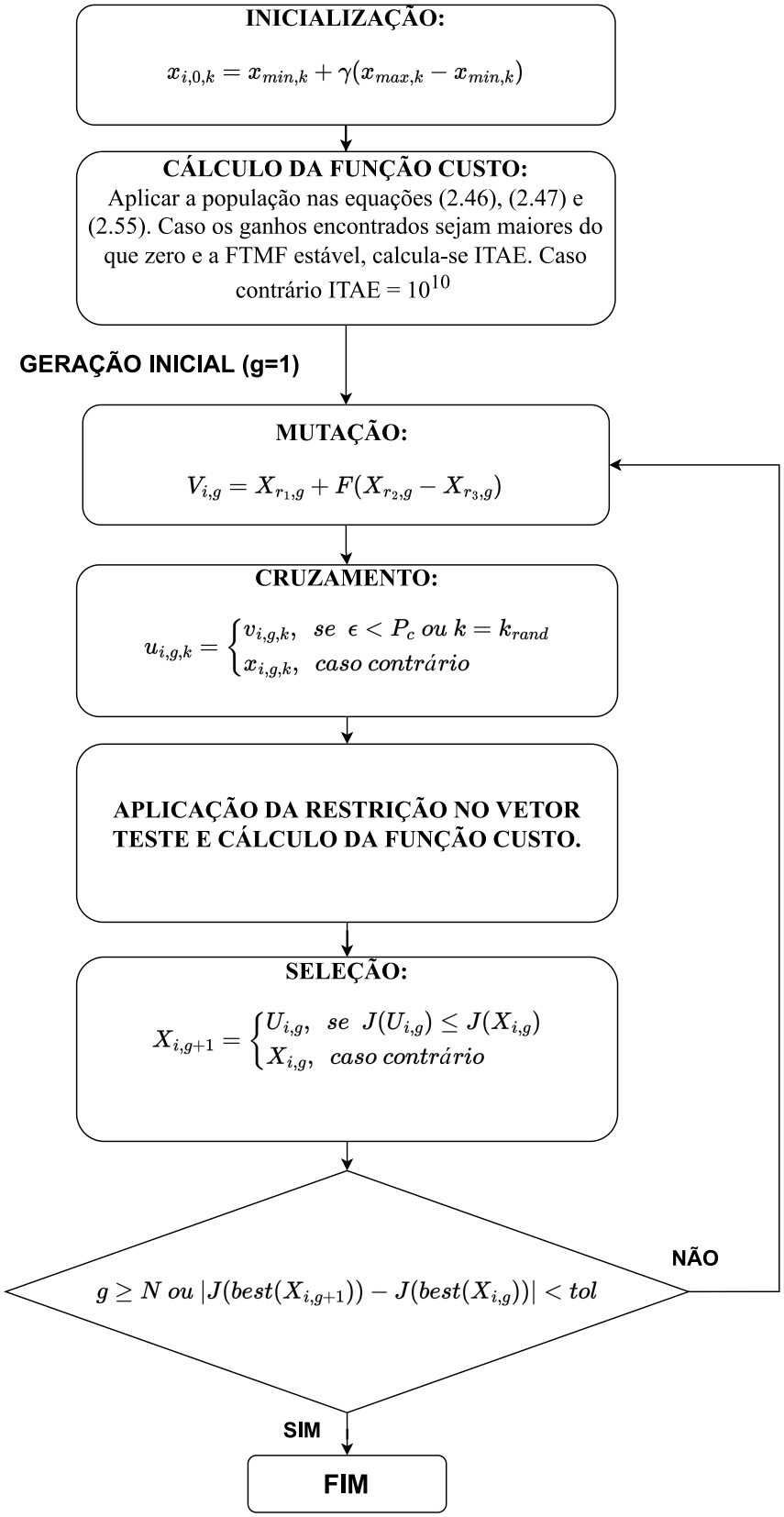
$$\begin{aligned}
&\left[K_i\omega_{gc}^{-\lambda}\sin(\frac{\pi\lambda}{2}) \pm \frac{|\sin\phi_x|}{G}\right]\mu\sin(\frac{\pi\mu}{2} - \phi_x) + K_i\lambda\omega_{gc}^{-\lambda}\sin(\frac{\pi\lambda}{2} + \phi_x)\sin(\frac{\pi\mu}{2}) \\
&\qquad = \mp\frac{\omega_{gc}}{G}\frac{d\phi_g}{d\omega}\frac{\sin\phi_x}{|\sin\phi_x|}\sin(\frac{\pi\mu}{2}) \\
&\therefore K_i = \mp\frac{\omega_{gc}^{\lambda}}{G}\left[\frac{\omega_{gc}\dfrac{d\phi_g}{d\omega}\dfrac{\sin\phi_x}{|\sin\phi_x|}\sin(\dfrac{\pi\mu}{2}) + \mu\sin(\dfrac{\pi\mu}{2} - \phi_x)|\sin\phi_x|}{\mu\sin(\dfrac{\pi\lambda}{2})\sin(\dfrac{\pi\mu}{2} - \phi_x) + \lambda\sin(\dfrac{\pi\lambda}{2} + \phi_x)\sin(\dfrac{\pi\mu}{2})}\right].
\end{aligned} \tag{2.55}$$

Através de (2.46), (2.47) e (2.55), encontra-se um PIDOF robusto com as características de velocidade ω_{gc} e ϕ_m pré-definidas em conjunto com o amortecimento iso. Contudo, tais equações são dependentes de λ e μ, gerando um problema multivariado. Com as equações previamente desenvolvidas, lançaremos a base de um algoritmo heurístico de Evolução Diferencial (ED) que foi utilizado na dissertação para a busca de uma solução quase ótima para λ e μ.

2.5 ALGORITMO DE EVOLUÇÃO DIFERENCIAL

A Evolução Diferencial (ED) é um algoritmo meta-heurístico que usa uma estratégia de busca global baseada em população para resolver problemas de otimização com função mono objetiva e possui operador de mutação diferencial simples com competição de seleção um-a-um (STANOVOV; AKHMEDOVA; SEMENKIN, 2022; STANOVOV et al., 2022; ZHENG et al., 2021). A ED é composta por procedimentos de mutação, cruzamento e seleção. O fluxograma do algoritmo de evolução diferencial é mostrado na Figura 5. Em um processo iterativo, a população de cada geração g contém N_p indivíduos com

Figura 5 – Fluxograma do algoritmo ED.



Fonte: o autor

valores decimais. Suponha que o i-ésimo indivíduo da geração g, com $i = 1, 2, \cdots, N_p$, seja representado como um vetor d-dimensional $X_{i,g}$ na Eq. (2.56):

$$X_{i,g} = \begin{bmatrix} x_{i,g,1} & x_{i,g,2} & \cdots & x_{i,g,d} \end{bmatrix}, \; i = 1, 2, \cdots, N_p. \tag{2.56}$$

O primeiro processo do algoritmo ED é a inicialização. A população é gerada por uma função aleatória limitada pelos valores das constantes máximas e mínimas. Seja $x_{min,k} \leq x_k \leq x_{max,k}$, para $k = 1, 2, \cdots, d$ e $\gamma \in [0, 1]$ um número aleatório uniformemente distribuído em $g = 0$. Assim, o k-ésimo elemento do i-ésimo indivíduo pode ser obtido na Eq. (2.57):

$$x_{i,0,k} = x_{min,k} + \gamma(x_{max,k} - x_{min,k}). \tag{2.57}$$

A ideia principal do algoritmo ED é a mutação vetorial baseada em diferenças, de modo que cada vetor doador possa ser gerado pela Eq. (2.58) quando é usado a configuração DE/rand/1/bin. Sendo: "*rand*" significa que os vetores populacionais são escolhidos aleatoriamente, "1" implica que apenas uma operação em diferença vetorial é empregada em (2.58) para gerar a população mutacionada e "bin" indica que o cruzamento binário é usado durante a formação da população experimental (STORN; PRICE; LAMPINEN, 2005). Observe que o operador mutacional utiliza três elementos diferentes entre si $(X_{r_1,g}, X_{r_2,g}, X_{r_3,g})$ para a geração do vetor doador $V_{i,g}$, todos diferentes de $X_{i,g}$.

$$V_{i,g} = X_{r_1,g} + F(X_{r_2,g} - X_{r_3,g}). \tag{2.58}$$

Para aumentar a diversidade da população, uma operação de cruzamento entra em ação após a geração do vetor doador por meio de mutação. Vale ressaltar que a população possui codificação numérica decimal. O vetor doador $V_{i,g}$ troca os componentes com o vetor alvo $X_{i,g}$ sob a operação de cruzamento binário para formar o vetor teste $U_{i,g} = \begin{bmatrix} u_{i,g,1} & u_{i,g,2} & \cdots & u_{i,g,d} \end{bmatrix}$ (Eq. (2.59)) a partir de uma probabilidade de cruzamento predefinida (GARCÍA et al., 2021).

$$u_{i,g,k} = \begin{cases} v_{i,g,k}, & se \;\; \epsilon < P_c \; ou \; k = k_{rand} \\ x_{i,g,k}, & c.c. \end{cases}, \tag{2.59}$$

sendo: $\epsilon \in [0, 1]$ é um número aleatório uniformemente distribuído, P_c é a probabilidade de cruzamento definida pelo projetista e k_{rand} é um número inteiro também gerado por uma distribuição uniforme dentro do intervalo $[1, d]$. Note que a probabilidade de diversidade para a geração do vetor teste através do vetor doador é de P_c/d. Após o operador de cruzamento, o vetor teste deve ser limitado em relação aos valores máximos e mínimos permitidos da população. Além disso, a função custo é executada com o novo vetor teste

limitado. Finalmente, a seleção gananciosa é realizada em relação à competição um-a-um entre a função custo dos vetores teste e alvo conforme a Eq. (2.60):

$$X_{i,g+1} = \begin{cases} U_{i,g}, & se \quad J(U_{i,g}) \leq J(X_{i,g}) \\ X_{i,g}, & c.c. \end{cases} . \tag{2.60}$$

O processo iterativo mencionado na Figura 5 é repetido até que um critério de parada seja alcançado, por exemplo, se o número de iterações atinge um valor específico (N) ou a diferença entre os melhores alvos de iterações sucessivas é menor que um valor aceitável (10^{-6} neste trabalho). Os parâmetros da ED empregados nesta dissertação são mostrados na Tabela 3. Restrições foram introduzidas no algoritmo ED para se obter resposta rápida em paralelo com robustez. Vale ressaltar que o algoritmo só calcula ITAE (Eq. (2.61)) se o sistema em malha fechada for estável com K_p, K_i e K_d positivos.

Tabela 3 – Parâmetros do algoritmo ED.

Especificação	**Valor**
Tamanho da população, N_p	20
Número de iterações, N	200
Probabilidade de cruzamento, P_c	0,7
Fator de escala, F	0,8

$$ITAE(t) = \int_0^\infty t|e(t)|\, dt. \tag{2.61}$$

Dessa forma, para plantas mais complexas, pode-se definir $x = (\lambda, \mu) \in [0, 2]$ como a população de busca do ED e, em conjunto com (2.46), (2.47) e (2.55) positivos somados a ω_{gc} e ϕ_m predefinidos, serem utilizados para a formação do sistema em malha fechada. Se o sistema de malha fechada for estável, o ITAE será calculado e dessa forma o algoritmo progride até uma solução quase-ótima de $x = (\lambda, \mu)$. Neste trabalho, o algoritmo ED rodou 100 vezes e o que resultou em menor ITAE foi escolhido como a solução ótima para o problema.

2.6 REDUÇÃO DA ORDEM DE SISTEMAS FRACIONÁRIOS COM NORMA 2

Considerando um sistema linear, invariante no tempo, causal ($f(t) = 0$ para $t < 0$) e de dimensão finita, a distância norma 2 de uma função no domínio "s" é dada pela Eq. (2.62):

Definição 2.6.1 (Norma 2)**.**

$$\|G(s)\|_2 = \left(\frac{1}{2\pi}\int_{-\infty}^{\infty}|G(j\omega)|^2 d\omega\right)^{1/2}. \tag{2.62}$$

Uma função de transferência racional é estritamente própria se, e somente se, $G(j\infty) = 0$, ou seja, o grau do numerador é menor que o do denominador. Um sistema estável e estritamente próprio possui uma norma 2 finita (BOYD et al., 1994). Através das Seções 2.4 e 2.5, apresentou-se como encontrar os parâmetros de um PIDOF quase ótimo, porém os filtros Oustaloup e Oustaloup refinado geram sistemas grandes quando se executa a aproximação da ordem fracionária para inteira, mesmo quando se é utilizado tamanho $N = 1$. Logo, busca-se uma aproximação utilizando um método de redução quase ótimo para melhor condicionar o sistema para ser embarcado em FPGA. Uma planta reduzida genérica, com tempo morto definido por τ, é dada pela Eq. (2.63):

$$G_{r/m,\tau}(s) = \frac{\sum_{k=0}^{r}\beta_{r+1-k}s^k}{s^m + \sum_{k=0}^{m-1}\alpha_{m-k}s^k}e^{-\tau s}. \tag{2.63}$$

Uma função custo para minimizar a norma 2 do sinal de erro gerado entre a função original e a reduzida pode ser definido por (XUE; CHEN; ATHERTON, 2008) (Eq. (2.64)):

$$J = \min_{\theta}\|G(s) - G_{r/m,\tau}(s)\|_2, \tag{2.64}$$

sendo θ o conjunto de parâmetros a serem otimizados (Eq. (2.65)), tais que:

$$\theta = [\beta_1, \cdots, \beta_r, \alpha_1, \cdots, \alpha_m, \tau]. \tag{2.65}$$

Para um cálculo computacional mais simples no MATLAB® de (2.64), o tempo de atraso τ pode ser aproximado por uma função racional pura através da técnica de aproximação de Padé (XUE; ATHERTON, 1994), resultando em uma função de transferência sem termo exponencial. Com isso, $G_{r/m,\tau}(s)$ se torna $G_{(r+p)/(m+p)}(s)$ com p sendo a ordem da aproximação de Padé. Logo, (2.64) pode ser descrita pela Eq. (2.66):

$$J = \min_{\theta}\|G(s) - G_{(r+p)/(m+p)}(s)\|_2 \tag{2.66}$$

e a norma 2 pode ser calculada recursivamente utilizando o algoritmo em (ÅSTRÖM, 1970).

3 CONTROLADOR POR MODELO DE PREDIÇÃO

Nesta seção é abordado o estudo de estabilidade, sem a aplicação de restrições, dos controladores MPC e GPC tipo m, multivariáveis, usando funções de Laguerre. Por fim, detalha-se a forma de funcionamento desses controladores quando as restrições na variável de entrada são inseridas e quando há uma necessidade de prescrição de estabilidade no sistema.

3.1 CONTROLADOR POR MODELO DE PREDIÇÃO COM 1 INTEGRADOR

3.1.1 MODELAGEM DO MPC

Suponha um sistema em espaço de estados discreto descrito pela Eq. (3.1):

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u_m(k), \\ y(k) &= C_m x_m(k), \end{aligned} \tag{3.1}$$

sendo $x_m \in \mathbb{R}^{n_1 \times 1}$, $A_m \in \mathbb{R}^{n_1 \times n_1}$, $B_m \in \mathbb{R}^{n_1 \times v}$, $u_m \in \mathbb{R}^{v \times 1}$, $C_m \in \mathbb{R}^{q \times n_1}$ e $y \in \mathbb{R}^{q \times 1}$. Em outras palavras, o sistema dinâmico possui v entradas, q saídas e n_1 estados. Pela a definição de diferenças retrógradas (Apêndice A.2), $\nabla x_m(k) = x_m(k) - x_m(k-1)$ e $\nabla u(k) = u_m(k) - u_m(k-1)$. Dessa forma, é direto que (3.1) pode ser expressa pela Eq. (3.2):

$$\nabla x_m(k+1) = A_m \nabla x_m(k) + B_m \nabla u(k). \tag{3.2}$$

Nota-se que $\nabla u(k)$ ainda é uma variável de entrada (a variação da entrada). Um novo vetor é escolhido por $x(k) = [\nabla x_m(k)^T y(k)^T]^T$. Aplicando o operador ∇ na variável de saída de (3.1), verifica-se a Eq. (3.3):

$$\begin{aligned} y(k+1) - y(k) &= C_m(x_m(k+1) - x_m(k)) \\ y(k+1) - y(k) &= C_m(\nabla x_m(k+1)). \end{aligned} \tag{3.3}$$

Substituindo (3.2) em (3.3), tem-se a Eq. (3.4):

$$y(k+1) = y(k) + C_m A_m \nabla x_m(k) + C_m B_m \nabla u(k). \tag{3.4}$$

Através de (3.2) e (3.4), o sistema em espaço de estados discreto (3.1) torna-se a Eq. (3.5):

$$\begin{aligned} \overbrace{\begin{bmatrix} \nabla x_m(k+1) \\ y(k+1) \end{bmatrix}}^{x(k+1)} &= \overbrace{\begin{bmatrix} A_m & o_m^T \\ C_m A_m & I_q \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} \nabla x_m(k) \\ y(k) \end{bmatrix}}^{x(k)} + \overbrace{\begin{bmatrix} B_m \\ C_m B_m \end{bmatrix}}^{B} \nabla u(k), \\ y(k) &= \overbrace{\begin{bmatrix} o_m & I_q \end{bmatrix}}^{C} \begin{bmatrix} \nabla x_m(k) \\ y(k) \end{bmatrix}, \end{aligned} \tag{3.5}$$

sendo

$$o_m = \overbrace{\begin{bmatrix} 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}}^{n_1} \Bigg\} q, \tag{3.6}$$

e I_q é a matriz identidade de ordem q, $x \in \mathbb{R}^{(n_1+q)\times 1}$, $A \in \mathbb{R}^{(n_1+q)\times(n_1+q)}$, $B \in \mathbb{R}^{(n_1+q)\times v}$ e $C \in \mathbb{R}^{q\times(n_1+q)}$. Vale notar que esta modelagem vale para sistemas MIMO. O trio (A,B,C) apresentado em (3.5) é chamado de conjunto de matrizes de espaço aumentadas de (A_m, B_m, C_m) com relação a transformação $x(k) = [\nabla x_m(k)^T y(k)^T]^T$. Tal transformação garante a inserção de q integradores em (3.5), pois a Eq. (3.7):

$$det \begin{bmatrix} (zI_{n_1} - A_m) & o_m^T \\ C_m A_m & (z-1)I_q \end{bmatrix} = (z-1)^q det(zI_{n_1} - A_m). \tag{3.7}$$

A equação (3.7) não mostra que cada saída possui um integrador, mas que o sistema completo possui q integradores. Assim, mostrar-se-á no próximo teorema uma condição de que o sistema com matrizes aumentadas é controlável e observável com algumas informações do sistema (3.1).

Teorema 3.1.1. *Assuma que o sistema (3.1) é controlável e observável com função de transferência mínima dada pela Eq. (3.8):*

$$G_m(z) = C_m(zI_{n_1} - A_m)^{-1} B_m. \tag{3.8}$$

Assim, a função de transferência do modelo (3.5) possui a representação dada pela Eq. (3.9):

$$G(z) = \frac{z}{z-1} G_m(z), \tag{3.9}$$

sendo também controlável e observável se, e somente se, $G_m(z)$ não possuir zeros em $z = 1$ (WANG, 2009).

Demonstração. Para provar que o modelo aumentado é controlável e observável, precisa-se mostrar que (3.9) é verdadeira. Depois disso, os resultados seguem da estrutura mínima do modelo aumentado sem cancelamento de polo-zero. Observe que a função de transferência do modelo aumentado é dada pela Eq. (3.10):

$$G(z) = C(zI_{n_1+q} - A)^{-1}B. \tag{3.10}$$

Note que, com (3.5), $M = (zI_{n_1+q} - A)$ é dado pela Eq. (3.11):

$$M = \begin{bmatrix} zI_{n_1} - A_m & o_m^T \\ -C_m A_m & (z-1)I_q \end{bmatrix} \tag{3.11}$$

Agora, é perceptível que a matriz M possui a forma $M = \begin{bmatrix} M_{11} & 0 \\ M_{21} & M_{22} \end{bmatrix}$. Como M é inversível, define-se N a sua inversa, ou seja, $N = M^{-1}$. Assim, $N = \begin{bmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \end{bmatrix}$ e efetuando $NM = I$, tem-se a Eq. (3.12):

$$\begin{gathered} N_{11}M_{11} = I_{n_1} \rightarrow N_{11} = M_{11}^{-1} \\ N_{12} = 0 \\ N_{22}M_{22} = I_q \rightarrow N_{22} = M_{22}^{-1} \\ N_{21}M_{11} + N_{22}M_{21} = 0 \rightarrow N_{21} = -M_{22}^{-1}M_{21}M_{11}^{-1}. \end{gathered} \tag{3.12}$$

De (3.12) em (3.11), tem-se a Eq. (3.13):

$$N = M^{-1} = (zI_{n_1+q} - A)^{-1} = \begin{bmatrix} (zI_{n_1} - A_m)^{-1} & o_m^T \\ (z-1)^{-1}C_m A_m (zI_{n_1} - A_m)^{-1} & (z-1)^{-1}I_q \end{bmatrix}. \tag{3.13}$$

Pela definição de C e B, (3.10) se reduz a Eq. (3.14):

$$G(z) = \frac{1}{z-1}C_m A_m (zI_{n_1} - A_m)^{-1}B_m + \frac{1}{z-1}C_m B_m. \tag{3.14}$$

Em (3.14), note que se $L = zI_{n_1} - A_m \rightarrow A_m = zI_{n_1} - L$. Logo, $A_m L^{-1} = (zI_{n_1} - L)L^{-1} = zL^{-1} - I_{n_1}$. Por conseguinte, pela Eq. (3.15):

$$\begin{aligned} G(z) =& \frac{1}{z-1}C_m(zL^{-1} - I_{n_1})B_m + \frac{1}{z-1}C_m B_m \\ G(z) =& \frac{z}{z-1}C_m L^{-1}B_m \\ G(z) =& \frac{z}{z-1}C_m(zI_{n_1} - A_m)^{-1}B_m = \frac{z}{z-1}G_m(z). \end{aligned} \tag{3.15}$$

Sob a suposição de que o modelo de planta $G_m(z)$ não tem zero em $z = 1$ e possui uma realização mínima, a função de transferência do modelo aumentado tem uma estrutura mínima de (3.9), portanto, também é controlável e observável.

□

Baseado no novo sistema (3.5), iniciado em k_i, as futuras variáveis de estado são calculadas sequencialmente usando um conjunto de parâmetros de controle futuros até a amostra $k_i + m$ (Eq. (3.16)).

$$\begin{aligned} x(k_i+1) &= Ax(k_i) + B\nabla u(k_i), \\ x(k_i+2) &= A^2x(k_i) + AB\nabla u(k_i) + B\nabla u(k_i+1), \\ &\vdots \\ x(k_i+m) &= A^m x(k_i) + \sum_{i=0}^{m-1} A^{m-i-1}B\nabla u(k_i+i). \end{aligned} \tag{3.16}$$

Observe que, de (3.5), a variável de saída predita de k_i até $k_i + m$ é encontrada ao multiplicar a variável de estado futura por C, ou seja (Eq. (3.17)):

$$y(k_i+m) = CA^m x(k_i) + \sum_{i=0}^{m-1} CA^{m-i-1}B\nabla u(k_i+i). \tag{3.17}$$

3.1.2 PRINCÍPIO OPERACIONAL DO MPC COM FUNÇÕES DE LAGUERRE

A ideia central é expressar a trajetória de controle incremental futura $\nabla u(k_i + m)$, $m = 0, 1, 2, \cdots$, usando uma expansão ortonormal. Ao fazer isso, o problema de encontrar a trajetória de controle incremental futura é convertido em encontrar o conjunto de coeficientes ótimos que geram a expansão de $\nabla u(k_i + m)$ na base da função ortonormal. Um conjunto de funções de Laguerre é usado no projeto (Apêndice A.3). O conjunto de funções de Laguerre forma um grupo de funções candidatas com ordens apropriadas e, com o aumento do número de termos, a expansão ortonormal irá convergir para a trajetória de controle ótimo subjacente. Assim, a precisão é garantida para melhorar à medida que o número de termos, N, aumenta.

Como consequência, aproximações satisfatórias do sinal de controle ∇u podem exigir um número muito grande de parâmetros no caso de amostragem rápida (frequência elevada), processos dinâmicos complicados, altas demandas de desempenho em malha fechada, levando a soluções mal condicionadas numericamente e carga computacional alta para implementação em *hardware*. Em vez disso, uma abordagem mais apropriada é propor redes de Laguerre no projeto de controladores de modelo preditivo (WANG, 2009).

Assim, através da escolha dos polos e a quantidade de funções de Laguerre, pode-se exprimir a expansão de $\nabla u(k_i + m)$, de um sistema dinâmico estável SISO, conforme a Eq. (3.18) (WANG, 2009):

$$\nabla u(k_i + m) = \sum_{j=1}^{N} c_j(m) l_j(m) = L(m)^T \eta, \tag{3.18}$$

sendo que $\eta^T = \begin{bmatrix} c_1 & c_2 & \cdots & c_N \end{bmatrix}$ compreende N coeficientes de Laguerre e $L(m)^T$ é o vetor função de Laguerre transposto em um tempo amostral m. Aqui, o horizonte controle N_c é retirado devido à aproximação da expansão de Laguerre. Entretanto, o número de termos N é utilizado em conjunto com a para aproximar melhor o comportamento de ∇u (WANG, 2009).

Baseado nas definições das funções de Laguerre em sistemas SISO (Apêndice A.3), pode-se estender para sistemas MIMO com a livre flexibilidade de escolha para a e N de cada entrada. Seja $\nabla u(k_i) = \begin{bmatrix} \nabla u_1(k_i) & \nabla u_2(k_i) & \cdots & \nabla u_v(k_i) \end{bmatrix}^T$ e a matriz de entrada B particionada em $B = \begin{bmatrix} B_1 & B_2 & \cdots & B_v \end{bmatrix}$, (3.18) pode ser descrita como o i-ésimo ($1 \leq i \leq v$) sinal de controle em conjunto com a escolha do polo a_i e ordem N_i conforme a Eq. (3.19):

$$\nabla u_i(k_i + m) = L_i(m)^T \eta_i, \tag{3.19}$$

no qual η_i e $L_i(m)$ são as descrições da rede de Laguerre do i-ésimo controle. Logo, aplicando (3.19) em (3.16) e (3.17), a predição da variável de estado futura $x(k_i + m)$ e a predição da saída do sistema $y(k_i + m)$, em um instante de amostragem m, transformam-se nas Eqs. (3.20a) e (3.20b):

$$\begin{aligned} x(k_i + m) &= A^m x(k_i) + \sum_{i=0}^{m-1} A^{m-i-1} B L(i)^T \eta \\ x(k_i + m) &= A^m x(k_i) + \sum_{i=0}^{m-1} A^{m-i-1} [B_1 L_1(i)^T \cdots B_v L_v(i)^T] \eta \\ x(k_i + m) &= A^m x(k_i) + \varphi(m)^T \eta, \end{aligned} \tag{3.20a}$$

$$\begin{aligned} y(k_i + m) &= CA^m x(k_i) + \sum_{i=0}^{m-1} CA^{m-i-1} B L(i)^T \eta \\ y(k_i + m) &= CA^m x(k_i) + \sum_{i=0}^{m-1} CA^{m-i-1} [B_1 L_1(i)^T \cdots B_v L_v(i)^T] \eta \\ y(k_i + m) &= CA^m x(k_i) + C\varphi(m)^T \eta, \end{aligned} \tag{3.20b}$$

sendo que os vetores parâmetros η^T e a matriz auxiliar $\varphi(m)^T$ (Eq. (3.21)) são formados pelos seus respectivos coeficientes individuais dados por $\eta^T = \begin{bmatrix} \eta_1^T & \eta_2^T & \cdots & \eta_v^T \end{bmatrix} \in \mathbb{R}^{1 \times \sum_{i=1}^{v} N_i}$

e

$$\varphi(m)^T = \sum_{i=0}^{m-1} A^{m-i-1} \begin{bmatrix} B_1 L_1(i)^T & \cdots & B_v L_v(i)^T \end{bmatrix}. \tag{3.21}$$

A teoria do MPC tipo 1 garante que tal sistema, quando otimizado por uma função custo, consegue rastrear uma referência em degrau com erro em estado estacionário nulo (WANG, 2009). Dessa forma, definindo os vetores Y (vetor de saídas do modelo previsto), U (vetor das ações de controle futuro) e N_p (horizonte de predição) pelas Eqs. (3.22a) e (3.22b):

$$Y = \begin{bmatrix} y(k_i+1) & y(k_i+2) & \cdots & y(k_i+N_p) \end{bmatrix}^T, \tag{3.22a}$$

$$U = \begin{bmatrix} \nabla u(k_i) & \nabla u(k_i+1) & \cdots & \nabla u(k_i+N_p-1) \end{bmatrix}^T, \tag{3.22b}$$

com $Y \in \mathbb{R}^{N_p \times q}$ e $U \in \mathbb{R}^{N_p \times v}$. O objetivo do MPC tipo 1 é encontrar um U ótimo que minimiza uma função custo J. Seja $R_s \in \mathbb{R}^{N_p \times q}$ o vetor da referência futura de Y em (3.22a), assume-se que R_s é constante sobre o horizonte de predição com $r(k_i) \in \mathbb{R}^{q \times 1}$, ou seja (Eq. (3.23)):

$$R_s^T = r(k_i) \overbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}}^{N_p}. \tag{3.23}$$

Em (WANG, 2009), a função custo J é definida pela Eq. (3.24):

$$J = (R_s - Y)^T (R_s - Y) + U^T R^* U, \;\; R^* = r_w I_{N_p}, \tag{3.24}$$

com $r_w > 0$. A equação (3.24) baseia-se na minimização do erro entre o sinal de referência e o sinal de saída, com a dependência da variável de entrada ponderada com r_w. As razões para esta escolha incluem simplicidade, praticidade do custo, a relevância para as aplicações e a semelhança com os sistemas de controle preditivo clássicos. Vale mencionar que, para r_w pequeno, a atuação do controle dever ser alta para estabilizar o sistema, deixando o sistema em malha fechada mais rápido, entretanto, isso também gera perda na robustez do mesmo (WANG, 2009). Primeiramente, trabalha-se a porção $U^T R^* U$ (Eq. (3.25) e (3.26)).

$$\begin{aligned} U^T R^* U &= r_w \sum_{m=0}^{N_p} \nabla^T u(k_i+m) \nabla u(k_i+m) \\ U^T R^* U &= r_w \sum_{m=0}^{N_p} \eta^T L(m) L^T(m) \eta \end{aligned} \tag{3.25}$$

$$
\begin{aligned}
U^T R^* U &= r_w \sum_{m=0}^{N_p} \eta^T \begin{bmatrix} L_1(m) & 0_1 & \cdots & 0_1 \\ 0_2 & L_2(m) & \cdots & 0_2 \\ \vdots & \vdots & \ddots & \vdots \\ 0_v & 0_v & \cdots & L_v(m) \end{bmatrix} \begin{bmatrix} L_1(m)^T & 0_2^T & \cdots & 0_v^T \\ 0_1^T & L_2(m)^T & \cdots & 0_v^T \\ \vdots & \vdots & \ddots & \vdots \\ 0_1^T & 0_2^T & \cdots & L_v(m)^T \end{bmatrix} \eta \\
U^T R^* U &= r_w \eta^T \begin{bmatrix} \sum_{m=0}^{N_p} L_1(m) L_1(m)^T & 0 & \cdots & 0 \\ 0 & \sum_{m=0}^{N_p} L_2(m) L_2(m)^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sum_{m=0}^{N_p} L_v(m) L_v(m)^T \end{bmatrix} \eta \\
U^T R^* U &= r_w \eta^T \begin{bmatrix} I_{N_1} & 0 & \cdots & 0 \\ 0 & I_{N_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I_{N_v} \end{bmatrix} \eta \\
U^T R^* U &= \eta^T (r_w I_{\sum N_i}) \eta = \eta^T R_L \eta.
\end{aligned} \tag{3.26}
$$

Observe que foi utilizado a propriedade da ortonormalidade das funções de Laguerre, N_p é grande o suficiente para aproximar $N_p \to \infty$ e $R_L \in \mathbb{R}^{\sum N_i \times \sum N_i}$. Em seguida, manipula-se o termo $(R_s - Y)^T (R_s - Y)$ de (3.24), resultando na Eq. (3.27).

$$
\begin{aligned}
(R_s - Y)^T (R_s - Y) &= \sum_{m=1}^{N_p} (r(k_i) - y(k_i + m))^T (r(k_i) - y(k_i + m)) \\
(R_s - Y)^T (R_s - Y) &= \sum_{m=1}^{N_p} (r(k_i) - Cx(k_i + m))^T (r(k_i) - Cx(k_i + m)).
\end{aligned} \tag{3.27}
$$

Criando um vetor estado auxiliar da forma que $x_r(k_i) = \begin{bmatrix} 0 & \cdots & r(k_i)^T \end{bmatrix}^T$. Dessa forma, pode-se escrever $r(k_i) = C x_r(k_i)$, de modo que (3.27) se transforma na Eq. (3.28):

$$
\begin{aligned}
(R_s - Y)^T (R_s - Y) &= \sum_{m=1}^{N_p} (x_r(k_i) - x(k_i + m))^T C^T C (x_r(k_i) - x(k_i + m)) \\
(R_s - Y)^T (R_s - Y) &= \sum_{m=1}^{N_p} \underline{x}(k_i + m)^T Q \underline{x}(k_i + m),
\end{aligned} \tag{3.28}
$$

com $Q = C^T C$ uma matrix positiva semi-definida e a nova variável de estado do sistema aumentado dada pela Eq. (3.29):

$$
\underline{x}(k_i + m) = \begin{bmatrix} \nabla x_m(k_i + m)^T & y(k_i + m)^T - r(k_i)^T \end{bmatrix}^T. \tag{3.29}
$$

Por fim, a função custo (3.24) se transformou na Eq. (3.30):

$$J = \sum_{m=1}^{N_p} \underline{x}(k_i+m)^T Q \underline{x}(k_i+m) + \eta^T R_L \eta. \tag{3.30}$$

Em relação a (3.20a) e (3.29), com uma referência tipo degrau, é direto pela Eq. (3.31):

$$\begin{aligned} \underline{x}(k_i+m) = A^m \underline{x}(k_i) + (A^m - I_{(n_1+q)}) \begin{bmatrix} o_1 & r(k_i)^T \end{bmatrix}^T \\ + \sum_{i=0}^{m-1} A^{m-i-1} B L(i)^T \eta, \end{aligned} \tag{3.31}$$

com $o_1 = \overbrace{\begin{bmatrix} 0 & \cdots & 0 \end{bmatrix}}^{n_1}$. Mas, de (3.5), perceba que

$$\begin{aligned} (A^k - I_{(n_1+q)}) \begin{bmatrix} o_1 & r(k_i)^T \end{bmatrix}^T \\ \begin{bmatrix} A_m^k - I_{n_1} & o_{n_1 \times q} \\ C_m \sum_{i=0}^{k-1} A_m^{i+1} & o_{q \times q} \end{bmatrix} \begin{bmatrix} o_1 & r(k_i)^T \end{bmatrix}^T = 0. \end{aligned} \tag{3.32}$$

Finalmente, pode se dizer que (3.20a) não se altera com a modificação da variável de estado (3.29), ou seja (Eq. (3.33)):

$$\underline{x}(k_i+m) = A^m \underline{x}(k_i) + \varphi(m)^T \eta, \tag{3.33}$$

3.2 CONTROLADOR POR MODELO DE PREDIÇÃO GENERALIZADO COM M INTEGRADORES

O controle por modelo de predição generalizado ou *Generalized Predictive Control* (GPC) é um controlador no qual o sistema de matrizes aumentadas (A,B,C) possui $m \in \mathbb{N}^*$ integradores, conduzindo à erros estacionários nulos quando uma referência polinômial de ordem $m-1$ é inserida. Ou seja, é uma generalização do MPC tipo 1 para ordens mais elevadas. Nas Seções 3.2.1 e 3.2.2, demonstra-se a modelagem de um GPC tipo m e o princípio operacional utilizando as funções de Laguerre para aproximar a variável de controle aumentada u.

3.2.1 MODELAGEM DO GPC TIPO M

Considere o seguinte sistema discreto MIMO de ordem n_1 dado pela Eq. (3.34):

$$\begin{aligned} x_m(i+1) &= A_m x_m(i) + B_m u_m(i), \\ y_m(i) &= C_m x_m(i), \end{aligned} \tag{3.34}$$

sendo $x_m \in \mathbb{R}^{n_1 \times 1}$, $A_m \in \mathbb{R}^{n_1 \times n_1}$, $B_m \in \mathbb{R}^{n_1 \times n_{in}}$, $u_m \in \mathbb{R}^{n_{in} \times 1}$, $C_m \in \mathbb{R}^{m_1 \times n_1}$ e $y_m \in \mathbb{R}^{m_1 \times 1}$. Em outras palavras, o sistema possui n_{in} entradas, m_1 saídas e n_1 estados. Definindo o erro de rastreamento a uma referência $r(i)$ pela Eq. (3.35):

$$e(i) = r(i) - y_m(i), \tag{3.35}$$

e utilizando o operador generalizado discreto em diferenças retrógradas (Apêndice A.2) ($\nabla_1^m = \nabla^m$) em (3.34) e em (3.35) deslocado de uma amostra, tem-se as Eqs. (3.36a) e (3.36b):

$$\nabla^m x_m(i+1) = A_m \nabla^m x_m(i) + B_m \nabla^m u_m(i), \tag{3.36a}$$

$$\begin{aligned} \nabla^m e(i+1) &= \nabla^m r(i+1) - C_m \nabla^m x_m(i+1), \\ \nabla^m e(i+1) &= \nabla^m r(i+1) - C_m A_m \nabla^m x_m(i) - C_m B_m \nabla^m u_m(i). \end{aligned} \tag{3.36b}$$

Novamente em relação ao Apêndice A.2, através do rastreamento de uma referência polinômial $r(i) = i^{m-1}$, $\nabla^m r(i+1) = 0$, logo (3.36b) resulta na Eq. (3.37):

$$\nabla^m e(i+1) = -C_m A_m \nabla^m x_m(i) - C_m B_m \nabla^m u_m(i). \tag{3.37}$$

Seja $a_j = \nabla^{m-j} e(i+1)$. De acordo com a propriedade da soma telescópica, $\sum_{j=1}^{g}(a_j - a_{j-1}) = a_g - a_0$ (PRICE, 1973). Além disso, $\nabla^{m-(j-1)} e(i+1) = \nabla^{m-j} e(i+1) - \nabla^{m-j} e(i)$. Por conseguinte, $a_j - a_{j-1} = \nabla^{m-j} e(i+1) - \nabla^{m-(j-1)} e(i+1) = \nabla^{m-j} e(i)$. Através desses resultados, $\nabla^{m-g} e(i+1)$ pode ser encontrado na Eq. (3.38):

$$\underbrace{\nabla^{m-g} e(i+1)}_{a_g} = \underbrace{\nabla^m e(i+1)}_{a_0} + \sum_{j=1}^{g} \underbrace{\nabla^{m-j} e(i)}_{a_j - a_{j-1}}. \tag{3.38}$$

Substituindo (3.37) em (3.38), resulta na Eq. (3.39):

$$\nabla^{m-g} e(i+1) = -C_m A_m \nabla^m x_m(i) - C_m B_m \nabla^m u_m(i) + \sum_{j=1}^{g} \nabla^{m-j} e(i). \tag{3.39}$$

Tomando $\varkappa(i) = [\underbrace{e_1(i), \cdots, e_{m_1}(i)}_{e(i)}, \cdots, \underbrace{\nabla^{m-1} e_1(i), \cdots, \nabla^{m-1} e_{m_1}(i)}_{\nabla^{m-1} e(i)}]^T \in \mathbb{R}^{(m_1 * m) \times 1}$.
Manipulando (3.39), para $g = 1, \cdots, m$, $\varkappa(i+1)$ pode ser calculada em (3.40):

$$\begin{aligned}\underbrace{\begin{bmatrix} e(i+1) \\ \nabla e(i+1) \\ \vdots \\ \nabla^{m-1}e(i+1) \end{bmatrix}}_{\varkappa(i+1)} &= \underbrace{\begin{bmatrix} I_{m_1} & I_{m_1} & \cdots & I_{m_1} \\ 0_{m_1} & I_{m_1} & \cdots & I_{m_1} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{m_1} & 0_{m_1} & \cdots & I_{m_1} \end{bmatrix}}_{E} \underbrace{\begin{bmatrix} e(i) \\ \nabla e(i) \\ \vdots \\ \nabla^{m-1}e(i) \end{bmatrix}}_{\varkappa(i)} \\ &- \underbrace{\begin{bmatrix} I_{m_1} \\ I_{m_1} \\ \vdots \\ I_{m_1} \end{bmatrix}}_{\gamma} \times (C_m A_m \nabla^m x_m(i) + C_m B_m \nabla^m u_m(i)),\end{aligned} \tag{3.40}$$

sendo que $E \in \mathbb{R}^{(m_1*m)\times(m_1*m)}$ é uma matriz triangular superior e $\gamma \in \mathbb{R}^{(m_1*m)\times(m_1)}$. Aqui, I_a denota uma matriz identidade de ordem a, $0_{m\times n}$ uma matriz nula com m linhas e n colunas e 0_a uma matriz quadrática nula de ordem a com $a \in \mathbb{N}_+$. Juntando (3.36a) com (3.40), obtém-se o modelo aumentado do sistema MIMO (Eq. (3.41)) sendo $A \in \mathbb{R}^{(m*m_1+n_1)\times(m*m_1+n_1)}$, $B \in \mathbb{R}^{(m*m_1+n_1)\times n_{in}}$, $C \in \mathbb{R}^{m_1\times(m*m_1+n_1)}$, $x \in \mathbb{R}^{(m*m_1+n_1)\times 1}$, com x o novo vetor de estados aumentado enquanto $u(i) = \nabla^m u_m(i)$ e $y(i)$ são os novos modelos de entrada e saída do sistema, respectivamente.

$$\begin{aligned}\underbrace{\begin{bmatrix} \varkappa(i+1) \\ \nabla^m x_m(i+1) \end{bmatrix}}_{x(i+1)} &= \underbrace{\begin{bmatrix} E & -\gamma C_m A_m \\ 0_{n_1\times(m*m_1)} & A_m \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \varkappa(i) \\ \nabla^m x_m(i) \end{bmatrix}}_{x(i)} + \underbrace{\begin{bmatrix} -\gamma C_m B_m \\ B_m \end{bmatrix}}_{B} \underbrace{\nabla^m u_m(i)}_{u(i)}, \\ \underbrace{e(i)}_{y(i)} &= \underbrace{\begin{bmatrix} I_{m_1} & 0_{m_1\times(m_1*(m-1)+n_1)} \end{bmatrix}}_{C} x(i).\end{aligned} \tag{3.41}$$

Note que a equação do polinômio característico do modelo aumetado A é:

$$\rho(z) = det \begin{bmatrix} (zI_{m_1*m} - E) & +\gamma C_m A_m \\ 0_{n_1\times(m*m_1)} & (zI_{n_1} - A_m) \end{bmatrix} = (z-1)^{m_1*m} det(zI - A_m), \tag{3.42}$$

Na Eq. (3.42), utilizou-se a propriedade do determinante de uma matriz triangular superior ser igual ao produto dos elementos da diagonal principal. Portanto, os autovalores do modelo aumentado são a união dos autovalores do modelo da planta original com $m_1 * m$ autovalores em $z = 1$. Isso significa que existem $m_1 * m$ integradores no modelo aumentado proposto. Agora, prova-se no Teorema 3.2.1 que m integradores estão embutidos em cada saída do sistema aumentado.

Teorema 3.2.1. *Assuma que o sistema (3.34) é controlável e observável com função de transferência mínima dada pela Eq. (3.43):*

$$G_m(z) = C_m(zI_{n_1} - A_m)^{-1}B_m. \tag{3.43}$$

Assim, a função de transferência do modelo (3.41) possui a representação dada pela Eq. (3.44):

$$G(z) = -(\frac{z}{z-1})^m G_m(z), \tag{3.44}$$

sendo também controlável e observável se, e somente se, $G_m(z)$ não possuir zeros em $z = 1$.

Demonstração. De forma análoga à demonstração do Teorema 3.1.1, a prova deste Teorema se reduz a provar (3.44) e, por continuidade, como $G_m(z)$ é controlável, observável e de ordem mínima sem zeros em $z = 1$, $G(z)$ também é controlável e observável. Seja $L = zI_{n_1} - A_m$, $\Gamma = zI_{m_1*m} - E$, $\Gamma^{-1} = \begin{bmatrix} Z \\ \Theta \end{bmatrix}$ e $I = \begin{bmatrix} I_{m_1*m} & 0_{(m*m_1)\times n_1} \\ 0_{n_1\times(m*m_1)} & I_{n_1} \end{bmatrix}$, sendo que $Z = [\zeta_1, \zeta_2, \cdots, \zeta_m] \in \mathbb{R}^{m_1\times(m*m_1)}$ é as m_1 primeiras linhas de Γ^{-1}, com $\zeta \in \mathbb{R}^{m_1\times m_1}$ e $\Theta \in \mathbb{R}^{m_1*(m-1)\times(m*m_1)}$. Em relação a (3.34), (3.41) e as definições supracitadas, as funções de transferência do sistema original e do modelo aumentado são respectivamente $G_m(z) = C_m(zI_{n_1} - A_m)^{-1}B_m = C_mL^{-1}B_m$ e $G(z) = C(zI - A)^{-1}B$. Note que $A_mL^{-1} = (zI_{n_1} - L)L^{-1} = zL^{-1} - I_{n_1}$. Dessa forma $G(z)$ pode ser calculado da seguinte forma (Eq. (3.45)):

$$\begin{aligned}
G(z) &= C(zI - \begin{bmatrix} E & -\gamma C_m A_m \\ 0_{n_1\times(m*m_1)} & A_m \end{bmatrix})^{-1}B \\
G(z) &= \begin{bmatrix} I_{m_1} & 0_{m_1\times(m_1*(m-1))} & 0_{m_1\times n_1} \end{bmatrix} \times \begin{bmatrix} \Gamma & \gamma C_m A_m \\ 0_{n_1\times(m*m_1)} & L \end{bmatrix}^{-1} \begin{bmatrix} -\gamma C_m B_m \\ B_m \end{bmatrix} \\
G(z) &= -\begin{bmatrix} I_{m_1} & 0_{m_1\times(m_1*(m-1))} & 0_{m_1\times n_1} \end{bmatrix} \times \begin{bmatrix} \Gamma^{-1} & -\Gamma^{-1}\gamma C_m A_m L^{-1} \\ 0_{n_1\times(m*m_1)} & L^{-1} \end{bmatrix} \begin{bmatrix} \gamma C_m B_m \\ -B_m \end{bmatrix} \\
G(z) &= -\begin{bmatrix} I_{m_1} & 0_{m_1\times(m_1*(m-1))} \end{bmatrix} \begin{bmatrix} \Gamma^{-1}\gamma C_m B_m + \Gamma^{-1}\gamma C_m(A_mL^{-1})B_m \end{bmatrix} \\
G(z) &= -\begin{bmatrix} I_{m_1} & 0_{m_1\times(m_1*(m-1))} \end{bmatrix} \Gamma^{-1}\gamma \begin{bmatrix} C_m B_m + C_m(zL^{-1} - I_{n_1})B_m \end{bmatrix} \\
G(z) &= -\begin{bmatrix} I_{m_1} & 0_{m_1\times(m_1*(m-1))} \end{bmatrix} \begin{bmatrix} Z \\ \Theta \end{bmatrix} z\gamma \underbrace{C_mL^{-1}B_m}_{G_m(z)} \\
G(z) &= -Z\gamma z G_m(z)
\end{aligned} \tag{3.45}$$

Observe que $(\Gamma^{-1}\Gamma)^T = I_{m_1*m} = \Gamma^T(\Gamma^{-1})^T = \Gamma^T \begin{bmatrix} Z^T & \Theta^T \end{bmatrix} = \begin{bmatrix} \Gamma^T Z^T & \Gamma^T \Theta^T \end{bmatrix} = I_{m_1*m}$ e $\Gamma^T Z^T \in \mathbb{R}^{(m_1*m)\times m_1}$, por isso $\Gamma^T Z^T = \begin{bmatrix} I_{m_1} & 0_{m_1\times(m_1*(m-1))} \end{bmatrix}^T$ (Eq. (3.46)). Logo:

$$\underbrace{\begin{bmatrix} (z-1)I_{m_1} & 0_{m_1} & \cdots & 0_{m_1} \\ -I_{m_1} & (z-1)I_{m_1} & \cdots & 0_{m_1} \\ \vdots & \vdots & \ddots & \vdots \\ -I_{m_1} & -I_{m_1} & \cdots & (z-1)I_{m_1} \end{bmatrix}}_{\Gamma^T} \underbrace{\begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_m \end{bmatrix}}_{Z^T} = \begin{bmatrix} I_{m_1} \\ 0_{m_1} \\ \vdots \\ 0_{m_1} \end{bmatrix}. \tag{3.46}$$

Por (BOYD; VANDENBERGHE, 2018), encontra-se $Z\gamma$ quando se é aplicado os seguintes passos: Passo 1) é direto que $\zeta_1 = \dfrac{I_{m_1}}{z-1}$ e $\zeta_k = \dfrac{\sum_{j=1}^{k-1}\zeta_j}{z-1}$, $k = 2, \ldots, m$. Fazendo $b_1 = \zeta_1$ e $b_k = \sum_{j=1}^{k}\zeta_j$, logo $\zeta_k = \dfrac{b_{k-1}}{z-1}$ e $b_k = \zeta_k + \sum_{j=1}^{k-1}\zeta_j = \zeta_k + b_{k-1}$. Assim (Eq. (3.47)):

$$b_k = \frac{b_{k-1}}{z-1} + b_{k-1} = \frac{z}{z-1}b_{k-1},\ b_k = \sum_{j=1}^{k}\zeta_j. \tag{3.47}$$

Passo 2) é notável que (3.47) é uma progressão geométrica com $b_m = b_1(\dfrac{z}{z-1})^{m-1} = \dfrac{z^{m-1}}{(z-1)^m}I_{m_1}$ (PRICE, 1973). Passo 3) também se nota que o fator $Z\gamma$ (Eq. (3.48)) é necessário, por isso, de acordo com (3.40):

$$Z\gamma = \begin{bmatrix} \zeta_1 & \cdots & \zeta_m \end{bmatrix} \begin{bmatrix} I_{m_1} & \cdots & I_{m_1} \end{bmatrix}^T = \sum_{j=1}^{m}\zeta_j = \ b_m = \frac{z^{m-1}}{(z-1)^m}I_{m_1}. \tag{3.48}$$

Substituindo (3.48) em (3.45), resulta-se na Eq. (3.49):

$$G(z) = -\frac{z^{m-1}}{(z-1)^m}I_{m_1}zG_m(z) = -(\frac{z}{z-1})^m I_{m_1}G_m(z). \tag{3.49}$$

O termo $\dfrac{z}{z-1}$ de (3.49) denota um integrador discreto. É observável que $G(z) \in \mathbb{R}^{m_1\times n_{in}}$ tal que cada função de transferência de saída, $G_k(z)$, pode ser escrita em relação às entradas $U_1, U_2, ..., U_{n_{in}}$ em (3.50):

$$G_k(z) = -(\frac{z}{z-1})^m \sum_{j=1}^{n_{in}} G_{m_{k,j}}(z)U_j(z), \tag{3.50}$$

sendo $k = 1, 2, ..., m_1$. Logo, (3.50) garante que cada saída possui m integradores como desejado. Para fechar a prova, é necessário que $G_m(z)$ não possua zeros em $z = 1$ e tenha uma estrutura mínima. O número de integradores embutidos está relacionado com o grau, $m-1$, da referência. □

3.2.2 PRINCÍPIO OPERACIONAL DO GPC TIPO M COM FUNÇÕES DE LAGUERRE

Com a rede de Laguerre escolhida para cada entrada do sistema MIMO, i.e., N_i e a_i, e seguindo todas as equações da Seção 3.1.2, verifica-se que as equações (3.18) -(3.24) são as mesmas para o GPC tipo m, exceto as definições das variáveis y, u e x. A partir de (3.24), a modificação está relacionada ao vetor de saídas preditas do modelo Y ser composto pelos erros de predição. Como o valor do erro desejado é nulo, o vetor de referência R_s é um conjunto nulo, i.e., $R_s = 0_{N_p \times m_1}$. Dessa forma, é direto que (3.24) em conjunto com (3.25) e (3.41) resulta na Eq. (3.51):

$$\begin{aligned} J &= \sum_{n=1}^{N_p} y(i+n)^T y(i+n) + \eta^T R_L \eta \\ J &= \sum_{n=1}^{N_p} x(i+n)^T Q x(i+n) + \eta^T R_L \eta, \end{aligned} \tag{3.51}$$

com $Q = C^T C$ e $R_L = r_w I_{\sum N_i}$. Perceba que, como não houve mudança na variável de estado $x(i)$, pois a referência predita deve ser nula, $x(i+n)$ também respeita (3.20a), ou seja (Eq. (3.52)):

$$x(i+n) = A^n x(i) + \varphi(n)^T \eta. \tag{3.52}$$

Dessa forma, pode-se dizer que, para ambos os controladores analisados, o princípio operacional é o mesmo. Ambos possuem as mesmas fórmulas de função custo, com matrizes auxiliares semelhantes, porém com definições de x, u, y e matrizes aumentadas diferentes. Dessa forma, as próximas fórmulas a serem desenvolvidas são equivalentes para ambos os controladores, i.e., MPC e GPC tipo m.

3.3 MINIMIZAÇÃO DA FUNÇÃO CUSTO EM CONTROLADORES POR MODELO DE PREDIÇÃO

O objetivo desta seção é calcular a otimização da função custo J dos controladores preditivos em relação ao vetor de coeficientes de Laguerre η. Substituindo (3.52) em (3.51), para um tempo k_i, resulta-se na Eq. (3.53):

$$\begin{aligned} J &= \sum_{n=1}^{N_p} (A^n x(k_i) + \varphi(n)^T \eta)^T Q (A^n x(k_i) + \varphi(n)^T \eta) + \eta^T R_L \eta \\ &= \sum_{n=1}^{N_p} [x(k_i)^T (A^T)^n Q A^n x(k_i) + x(k_i)^T (A^T)^n Q \varphi(n)^T \eta \\ &\quad + \eta^T \varphi(n) Q A^n x(k_i) + \eta^T \varphi(n) Q \varphi(n)^T \eta] + \eta^T R_L \eta. \end{aligned} \tag{3.53}$$

Notando que $\eta^T\varphi(n)QA^nx(k_i)$ é um número, ou seja, $(\eta^T\varphi(n)QA^nx(k_i))^T = x(k_i)^T(A^T)^nQ\varphi(n)^T\eta$, (3.53) pode ser escrita pela Eq. (3.54):

$$J = \eta^T(\sum_{n=1}^{N_p}\varphi(n)Q\varphi(n)^T + R_L)\eta + 2\eta^T(\sum_{n=1}^{N_p}\varphi(n)QA^n)x(k_i) + \sum_{n=1}^{N_p}x(k_i)^T(A^T)^nQA^nx(k_i). \tag{3.54}$$

Por J ser uma função convexa com concavidade positiva, executa-se a derivada parcial da função custo $\frac{\partial J}{\partial \eta}$ para encontrar o mínimo conforme a Eq. (3.55).

$$\frac{\partial J}{\partial \eta} = 2(\sum_{m=1}^{Np}\varphi(m)Q\varphi(m)^T + R_L)\eta + 2(\sum_{m=1}^{Np}\varphi(m)QA^m)x(k_i). \tag{3.55}$$

Por simplicidade, define-se as Eq. (3.56a) e (3.56b):

$$\Omega = \sum_{m=1}^{Np}\varphi(m)Q\varphi(m)^T + R_L, \tag{3.56a}$$

$$\Psi = \sum_{m=1}^{Np}\varphi(m)QA^m. \tag{3.56b}$$

Logo, assumindo que Ω^{-1} existe quando $\frac{\partial J}{\partial \eta} = 0$, a solução ótima para o vetor parâmetros η^* é dada pela Eq. (3.57):

$$\eta^* = -\Omega^{-1}\Psi x(k_i). \tag{3.57}$$

Obtendo o vetor ótimo η^*, a variável de controle atuante no sistema de malha fechada é obtido a partir da lei do controle horizontal retrocedente (WANG, 2009). Para sistemas MIMO, a atuação ótima $u^*(k_i)$ é obtida pela Eq. (3.58):

$$u^*(k_i) = L(0)^T\eta^* = \begin{bmatrix} L_1(0)^T & 0_2^T & \cdots & 0_{n_{in}}^T \\ 0_1^T & L_2(0)^T & \cdots & 0_{n_{in}}^T \\ \vdots & \vdots & \ddots & \vdots \\ 0_1^T & 0_2^T & \cdots & L_{n_{in}}(0)^T \end{bmatrix}\eta^*, \tag{3.58}$$

sendo que 0_k^T, $k = 1, 2, \cdots n_{in}$, representa um vetor linha nulo com dimensão idêntica a $L_k(0)^T$, i.e., N_k. Por fim, observe que se utiliza $\varphi(m)$ nos cálculos anteriores. Assim, desenvolve-se a obtenção através da forma recursiva a partir de (3.21) pela Eq. (3.59), lembrando que $L(k_i + 1) = A_lL(k_i)$.

$$\begin{aligned}\varphi(1)^T &= BL(0)^T \\ \varphi(2)^T &= ABL(0)^T + BL(1)^T = ABL(0)^T + BL(0)^T A_l^T = A\varphi(1)^T + \varphi(1)^T A_l^T \\ \varphi(3)^T &= A^2BL(0)^T + ABL(1)^T + BL(2)^T = A\varphi(2)^T + \varphi(1)^T (A_l^2)^T.\end{aligned} \tag{3.59}$$

Observe que, por (3.59), a equação de recorrência de $\varphi(m)^T$ é dada pela Eq. (3.60):

$$\varphi(m)^T = A\varphi(m-1)^T + \varphi(1)^T (A_l^{m-1})^T, \tag{3.60}$$

sendo $\varphi(1)^T = BL(0)^T$ e $m = 2, 3, \cdots, N_p$. Assim, o problema do controlador preditivo sem restrições, quando dado sua formulação em espaço de estados por (3.5) ou (3.41), respeitado os teoremas anteriores, é resolvido pelas fórmulas até então supracitadas.

3.4 CONTROLADOR POR MODELO DE PREDIÇÃO COM RESTRIÇÕES

Os controladores por modelo de predição com funções de Laguerre possuem a característica de não precisar de muitas predições da variável de controle para controlar o sistema. Isso reduz em muito a quantidade necessária de restrições e, por conseguinte, o tempo de otimização com a programação quadrática. Dessa forma, reduz-se a quantidade de *clocks* internos para o ciclo de controle, abrangendo sistemas com maiores frequências de operação (SAEED; WANG; FERNANDO, 2022). Neste trabalho, aborda-se as restrições no sinal de controle $\nabla u_m(k_i)$ e $u_m(k_i)$, pois elas não sofrem interferência quanto aos distúrbios e variações físicas do sistema (variações de valores dos componentes e ruídos inerentes ao sistema). Por fim, as restrições utilizadas são as medidas no tempo atual $k = k_i$, pois se busca simplificar ao máximo o algoritmo para a síntese do *hardware*.

Assim, para uma amostra k_i, com limites superior e inferior de $u_m(k_i)$ e $\nabla u_m(k_i)$ sendo u_{max}, u_{min}, ∇u_{max} e ∇u_{min}, o procedimento de otimização é minimizar a função custo J (Eq. (3.61)):

$$\begin{aligned} J &= \eta^T \Omega \eta + 2\eta^T \Psi x(k_i) \\ s.a: \ \nabla u_{min} &\leq \nabla u_m(k_i) \leq \nabla u_{max} \\ u_{min} &\leq u_m(k_i) \leq u_{max}. \end{aligned} \tag{3.61}$$

Através do Teorema A.2.1, para $l = 1$, é direto que para o GPC tipo m, tem-se a

Eq. (3.62):

$$\begin{aligned} \nabla^m u_m(k_i) &= \sum_{i=0}^{m} C_m^i (-1)^i u_m(k_i - i) \\ u_m(k_i) &= \nabla^m u_m(k_i) - \sum_{i=1}^{m} C_m^i (-1)^i u_m(k_i - i) \\ \nabla u_m(k_i) &= \nabla^m u_m(k_i) - \sum_{i=1}^{m} C_m^i (-1)^i u_m(k_i - i) - u_m(k_i - 1). \end{aligned} \tag{3.62}$$

Assim, pelo horizonte retrocedente, (3.61) se reduz a Eq. (3.63):

$$\begin{aligned} J = \ & \eta^T \Omega \eta + 2\eta^T \Psi x(k_i) \\ s.a: \ & \nabla u_{min} \leq L(0)^T \eta - \sum_{i=1}^{m} C_m^i (-1)^i u_m(k_i - i) - u_m(k_i - 1) \leq \nabla u_{max} \\ & u_{min} + \sum_{i=1}^{m} C_m^i (-1)^i u_m(k_i - i) \leq L(0)^T \eta \leq u_{max} + \sum_{i=1}^{m} C_m^i (-1)^i u_m(k_i - i). \end{aligned} \tag{3.63}$$

Observe que, em (3.63), ambas as restrições possuem a mesma $L(0)^T$ acompanhando a variável a ser otimizada (η). Isso mostra que as duas inequações de restrições são linearmente dependentes, resultando em apenas uma ser ativada por vez com a programação quadrática de Hildreth's (WANG, 2009). Logo, não há necessidade de executar o algoritmo de Hildreth no sistema para verificar qual restrição será ativada, salvando ainda mais memória na síntese do *hardware*. Por fim, a solução do problema é buscar saber qual restrição é a mais importante para uma solução factível e deixá-la como a decisória para o valor ótimo de η. A restrição mais importante para este trabalho é a associada ao sinal de controle, uma vez que ele é o atuador direto ao sistema, evitando a queima de equipamentos. Por conseguinte, a restrição na variação da variável de controle é menos importante. Ao assumir esta hierarquia, a cada iteração do horizonte retrocedente, após o cálculo da matriz de parâmetros η, o algoritmo calcula $L(0)^T\eta - \sum_{i=1}^{m} C_m^i(-1)^i u_m(k_i - i) - u_m(k_i - 1)$. Se o valor calculado for maior que a restrição máxima, o valor passa a ser este valor máximo, da mesma forma, se a diferença da variável de controle for menor que a restrição mínima, o valor passa a ser esse valor mínimo. Ao fazer isso, o algoritmo atualiza a variável de controle e se o valor for maior que a restrição máxima, a diferença da variável de controle se torna u_{max} - $u_m(k_i - 1)$ e a variável de controle se torna u_{max}, da mesma forma, se o valor da variável de controle for menor que a restrição mínima, a diferença da variável de controle se torna u_{min} - $u_m(k_i - 1)$ e a variável de controle se torna u_{min}, fechando a solução do problema (3.63).

3.5 CONTROLADOR POR MODELO DE PREDIÇÃO COM GRAU DE ESTABILIDADE

Observando J em (3.63), verifica-se que o valor de N_p interfere no tempo de solução da otimização quadrática. Porém, deve-se notar também que a solução ótima (3.57) é um sistema linear onde uma pequena perturbação em $x(k_i)$ pode alterar ou até desestabilizar a solução do sistema, i.e., $\Omega\eta^* = -\Psi x(k_i) \leftrightarrow Ax = b$, com $A = \Omega$, $x = \eta^*$ e $b = -\Psi x(k_i)$. Vale lembrar que Ω é uma matriz simétrica (positiva definida) e que a construção é dada por (3.56a). Com o aumento de N_p, o termo $\sum_{m=1}^{Np} \varphi(m)Q\varphi(m)^T + R_L$ fica com norma-2 cada vez maior, uma vez que $\varphi(m)Q\varphi(m)^T$ é positiva semi-definida (matriz simétrica) e R_L positiva definida (por definição). Para continuar o raciocínio, define-se o número de condição de uma matriz simétrica.

Definição 3.5.1 (Número de condição de uma matriz simétrica)**.** Para uma matriz A simétrica não-singular em $\mathbb{R}^{nxn}$, com autovalores $\lambda_1, \cdots, \lambda_n$. Então, o número de condição desta matriz, através da definição de norma-2, é dada pela Eq. (3.64):

$$\kappa(A) = \frac{\lambda_{max}}{\lambda_{min}}. \tag{3.64}$$

Voltando ao problema linear $Ax = b$, cuja a solução é $x^* = A^{-1}b$, a Seção 3.4 tratou do problema de restrições no GPC tipo m. Desta forma, o problema com restrições segue a mesma solução do problema linear perturbado, ou seja, uma nova solução $\hat{x} = x + \delta x$ para que a Eq. (3.65)

$$A(x + \delta x) = b + \delta b, \tag{3.65}$$

sendo $\delta b \in \mathbb{R}^n$ a perturbação. A equação (3.65) precisa ter uma solução bem condicionada, e isso é garantido com um bom condicionamento de A uma vez que existe um majorante para o erro relativo dado por (GANDER, 2005; SCHWARZ, 1997) na Eq. (3.66):

$$\frac{||\delta x^*||}{||x^*||} \leq \kappa(A)\frac{||\delta b||}{||b||}. \tag{3.66}$$

Assim, para um problema linear bem condicionado, $\kappa(A)$ deve ser suficientemente pequeno para garantir que grandes δb não alterem muito a solução do problema. Como o problema do GPC tipo m possui um tamanho considerável com a escolha das matrizes Q, R_L e do número de predição N_p, buscar-se-á então uma alternativa para diminuir $\kappa(\Omega)$ e, assim, deixar o sistema suficientemente condicionado mesmo com a utilização de um N_p grande.

A técnica a ser utilizada neste trabalho é ponderar exponencialmente, com o termo a^{-1}, a função custo J definida com os estados e entradas aumentadas $x(k_i)$ e $u(k_i)$. Desta forma, J fica dada pela Eq. (3.67):

$$J = \sum_{j=1}^{N_p} a^{-2j} x(k_i+j)^T Q x(k_i+j) + \sum_{j=0}^{N_p} a^{-2j} u(k_i+j)^T R u(k_i+j), \tag{3.67}$$

sujeita às inequações de restrições expressas pela Eq. (3.68):

$$MU \leq \gamma, \tag{3.68}$$

sendo M e γ as matrizes dos dados das restrições e o vetor de parâmetros U dado pela Eq. (3.69):

$$U^T = \begin{bmatrix} u(k_i)^T & u(k_i+1)^T & \cdots & u(k_i+N_p)^T \end{bmatrix}. \tag{3.69}$$

Vale lembrar que a equação de espaço de estados aumentadas é dada pela Eq. (3.70):

$$x(k_i+j+1) = Ax(k_i+j) + Bu(k_i+j). \tag{3.70}$$

Note que para $a = 1$ em (3.67), tem-se o caso estudado até o presente momento. Para $a > 1$, verifica-se que os termos mais atuais são os que dão mais ênfase à predição. Essa é a configuração a ser utilizada neste trabalho, pois é a qual diminui $\kappa(\Omega)$. Definindo a sequência de controle incremental ponderada exponencialmente pela Eq. (3.71):

$$\hat{U}^T = \begin{bmatrix} a^{-0}u(k_i)^T & a^{-1}u(k_i+1)^T & \cdots & a^{-N_p}u(k_i+N_p)^T \end{bmatrix}, \tag{3.71}$$

e as variáveis de estado ponderadas pela Eq. (3.72):

$$\hat{X}^T = \begin{bmatrix} a^{-1}x(k_i+1)^T & a^{-2}x(k_i+2)^T & \cdots & a^{-N_p}x(k_i+N_p)^T \end{bmatrix}, \tag{3.72}$$

pode-se enunciar o Teorema 3.5.1 que transforma a função custo de acordo com a escolha de a. A demonstração não será descrita, pois a mesma pode ser encontrada em (WANG, 2009).

Teorema 3.5.1. *A otimização mínima de (3.67) sujeita às restrições (3.68) e equação de estado (3.70) pode ser encontrada se minimizado a Eq. (3.73):*

$$\begin{aligned} \hat{J} = \sum_{j=1}^{N_p} \hat{x}(k_i+j)^T Q \hat{x}(k_i+j) + \sum_{j=0}^{N_p} \hat{u}(k_i+j)^T R \hat{u}(k_i+j), \\ s.a\ M_a \hat{U} \leq \gamma, \end{aligned} \tag{3.73}$$

onde a equação de estados referente a $\hat{J}$ é dada pela Eq. (3.74):

$$\hat{x}(k_i + j + 1) = \frac{A}{a}\hat{x}(k_i + j) + \frac{B}{a}\hat{u}(k_i + j), \tag{3.74}$$

e M_a definida pela Eq. (3.75):

$$M_a = M \begin{bmatrix} I & 0 & \cdots & 0 \\ 0 & aI & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & a^{N_p} I \end{bmatrix}. \tag{3.75}$$

A escolha de valores de $a > 1$, em (3.74), em conjunto com valores pequenos de R geralmente geram sistemas estáveis, porém só o fato de $a > 1$ não resulta em condição suficiente para estabilidade. No entanto, com a diminuição de $\kappa(\Omega)$, pode-se analisar o GPC tipo m com $N_p \to \infty$, aproximando o problema (3.73) para o problema LQR (A.4). Dessa forma, assumindo o par $(a^{-1}A, a^{-1}B)$ controlável e o par $(a^{-1}A, C)$ observável, então a matriz ganho K (A.64) é dada pela Eq. (3.76):

$$K = (R + a^{-2}B^T P_\infty B)^{-1} a^{-2} B^T P_\infty A, \tag{3.76}$$

com P_∞ encontrado por (A.63), resultando na Eq. (3.77):

$$P_\infty = Q + a^{-2}A^T P_\infty A - a^{-2}A^T P_\infty B(R + a^{-2}B^T P_\infty B)^{-1} a^{-2} B^T P_\infty A. \tag{3.77}$$

O sistema em malha fechada deste sistema, por retroalimentação, reduz-se a Eq. (3.78):

$$\hat{x}(k_i + j + 1) = a^{-1}(A - BK)\hat{x}(k_i + j), \tag{3.78}$$

é estável se todos os autovalores $a^{-1}(A - BK)$ estiverem dentro do círculo unitário, ou seja (Eq. (3.79)):

$$\begin{aligned} a^{-1}|\lambda_{max}(A - BK)| &< 1 \\ |\lambda_{max}(A - BK)| &< a. \end{aligned} \tag{3.79}$$

Ou seja, a escolha de um $a > 1$ não garante a estabilidade do sistema em malha fechada original. O objetivo agora é encontrar uma maneira de manter a diminuição do número de condição de Ω e deixar os autovalores do sistema ponderado iguais ao do

sistema original. A ideia para resolver este problema é modificar as matrizes Q e R de tal forma que os autovalores do sistema ponderado sejam divididos por a. Isso é verificado no Teorema 3.5.2, com $N_p \to \infty$.

Teorema 3.5.2. *Assumindo a equação de espaço de estados aumentadas pela Eq. (3.80)*

$$x(k_i + j + 1) = Ax(k_i + j) + Bu(k_i + j), \tag{3.80}$$

a solução ótima de $u(k_i + j)$ por otimização da função custo J_a definida pela Eq. (3.81)

$$J_a = \sum_{j=1}^{\infty} a^{-2j} x(k_i + j)^T Q_a x(k_i + j) + \sum_{j=0}^{\infty} a^{-2j} u(k_i + j)^T R_a u(k_i + j) \tag{3.81}$$

é idêntica à solução encontrada ao minimizar a função custo original dada pela Eq. (3.82)

$$J = \sum_{j=1}^{\infty} x(k_i + j)^T Q x(k_i + j) + \sum_{j=0}^{\infty} u(k_i + j)^T R u(k_i + j), \tag{3.82}$$

sendo Q_a e R_a encontrados pela Eq. (3.83):

$$\begin{aligned} \gamma &= \frac{1}{a}, \\ Q_a &= \gamma^2 Q + (1 - \gamma^2) P_\infty, \\ R_a &= \gamma^2 R, \end{aligned} \tag{3.83}$$

e P_∞ encontrado pela solução da equação de Riccati invariante no tempo (Eq. (3.84))

$$A^T [P_\infty - P_\infty B (R + B^T P_\infty B)^{-1} B^T P_\infty] A + Q - P_\infty = 0. \tag{3.84}$$

Demonstração. Para uma condição inicial $x(k_i)$, a solução de (3.82), quando não há restrições, é dado pela solução de Riccati invariante no tempo (3.84), com o ganho K dado por (A.64). Porém, o ganho $\hat{K}$ da solução do problema ponderado (3.81) é dado pela Eq. (3.85)

$$\begin{aligned} \hat{K} &= (R_a + a^{-2} B^T \hat{P}_\infty B)^{-1} a^{-2} B^T \hat{P}_\infty A \\ \hat{A}^T [\hat{P}_\infty - \hat{P}_\infty \hat{B} (R_a + \hat{B}^T \hat{P}_\infty \hat{B})^{-1} \hat{B}^T \hat{P}_\infty] \hat{A} + Q_a - \hat{P}_\infty &= 0. \end{aligned} \tag{3.85}$$

Multiplicando cada A e B por a/a em (3.84), com $\hat{A} = A/a$ e $\hat{B} = B/a$ e $\gamma = 1/a$, tem-se a Eq. (3.86):

$$\frac{\hat{A}^T}{\gamma} [P_\infty - P_\infty \frac{\hat{B}}{\gamma} (R + \frac{\hat{B}^T}{\gamma^2} P_\infty \hat{B})^{-1} \frac{\hat{B}^T}{\gamma} P_\infty] \frac{\hat{A}}{\gamma} + Q - P_\infty = 0. \tag{3.86}$$

Multiplicando (3.86) por γ^2 e depois somando e subtraindo por P_∞, resulta-se na Eq. (3.87):

$$\begin{aligned}\hat{A}^T[P_\infty - P_\infty\hat{B}(R\gamma^2 + \hat{B}^T P_\infty \hat{B})^{-1}\hat{B}^T P_\infty]\hat{A} + \gamma^2 Q - \gamma^2 P_\infty + P_\infty - P_\infty = 0,\\ \hat{A}^T[P_\infty - P_\infty\hat{B}(R_a + \hat{B}^T P_\infty \hat{B})^{-1}\hat{B}^T P_\infty]\hat{A} + Q_a - P_\infty = 0.\end{aligned} \tag{3.87}$$

Com (3.87), prova-se (3.83). Observe que as equações (3.87) e (3.85) tornam-se idênticas, fechando a prova. □

Dado o Teorema 3.5.2, o problema do mal condicionamento de sistemas MIMO pode ser resolvido com a escolha de um parâmetro a. Contudo, até o presente momento, o projetista possui a liberdade de acelerar o controle com a escolha da diagonal principal de R, ou seja, pequenos valores proporcionam $u(k_i)$ maiores (WANG, 2009). Porém, o projetista ainda não possui a capacidade de manipular a região dos autovalores do sistema. O Teorema 3.5.3 enuncia o modo de como elaborar um controlador por modelo de predição com um grau de estabilidade λ, i.e., autovalores do sistema em malha fechada dentro do círculo de módulo λ.

Teorema 3.5.3. *Assumindo a equação de espaço de estados aumentadas pela Eq.* (3.88)

$$x(k_i + j + 1) = Ax(k_i + j) + Bu(k_i + j), \tag{3.88}$$

a solução ótima de $u(k_i + j)$ *por otimização da função custo* J_a *definida pela Eq.* (3.89)

$$J_a = \sum_{j=1}^{\infty} a^{-2j} x(k_i + j)^T Q_a x(k_i + j) + \sum_{j=0}^{\infty} a^{-2j} u(k_i + j)^T R_a u(k_i + j) \tag{3.89}$$

é idêntica à solução encontrada ao minimizar a função custo original da Eq. (3.90)

$$J = \sum_{j=1}^{\infty} \lambda^{-2j} x(k_i + j)^T Q x(k_i + j) + \sum_{j=0}^{\infty} \lambda^{-2j} u(k_i + j)^T R u(k_i + j), \tag{3.90}$$

sendo $a \geq 1$, $0 < \lambda < 1$*; com* Q_a *e* R_a *encontrados pela Eq.* (3.91)*:*

$$\begin{aligned}\gamma &= \frac{\lambda}{a}\\ Q_a &= \gamma^2 Q + (1 - \gamma^2)P_\infty\\ R_a &= \gamma^2 R,\end{aligned} \tag{3.91}$$

e P_∞ *encontrado pela solução da equação de Riccati invariante no tempo (Eq.* (3.92)*)*

$$\frac{A^T}{\lambda}[P_\infty - P_\infty\frac{B}{\lambda}(R + \frac{B^T}{\lambda^2}P_\infty B)^{-1}\frac{B^T}{\lambda}P_\infty]\frac{A}{\lambda} + Q - P_\infty = 0. \tag{3.92}$$

Demonstração. Como o GPC tipo m está sendo analisado em um horizonte de predição muito grande, utiliza-se a analogia com o controlador LQR. Desta forma, com a equação de Riccati, resolvendo o sistema em malha fechada com a otimização de (3.90), encontra-se algo análogo à (3.79), porém com um limite de majoração em λ dado pela Eq. (3.93)

$$\begin{aligned} \lambda^{-1}|\lambda_{max}(A-BK)| &< 1 \\ |\lambda_{max}(A-BK)| &< \lambda. \end{aligned} \tag{3.93}$$

Ou seja, garante-se que todos os autovalores do sistema em malha fechada estão dentro do círculo de raio $0 < \lambda < 1$, garantindo a estabilidade do sistema original. Observe também que (3.90) é resolvido pela equação (3.92). Multiplicando todas as matrizes A e B por a/a e substituindo $\hat{A} = A/a$ e $\hat{B} = B/a$, com a definição de $\gamma = \lambda/a$, tem-se a Eq. (3.94):

$$\frac{\hat{A}^T}{\gamma}[P_\infty - P_\infty\frac{\hat{B}}{\gamma}(R + \frac{\hat{B}^T}{\gamma^2}P_\infty\hat{B})^{-1}\frac{\hat{B}^T}{\gamma}P_\infty]\frac{\hat{A}}{\gamma} + Q - P_\infty = 0. \tag{3.94}$$

Multiplicando (3.94) por γ^2 e depois somando e subtraindo por P_∞, resulta-se na Eq. (3.95):

$$\begin{aligned} \hat{A}^T[P_\infty - P_\infty\hat{B}(R\gamma^2 + \hat{B}^T P_\infty\hat{B})^{-1}\hat{B}^T P_\infty]\hat{A} + \gamma^2 Q - \gamma^2 P_\infty + P_\infty - P_\infty = 0, \\ \hat{A}^T[P_\infty - P_\infty\hat{B}(R_a + \hat{B}^T P_\infty\hat{B})^{-1}\hat{B}^T P_\infty]\hat{A} + Q_a - P_\infty = 0. \end{aligned} \tag{3.95}$$

Com (3.95), prova-se (3.91). Observe que a equação (3.95) resolve o problema com a otimização de (3.89), fechando a prova. □

Vale lembrar que os Teoremas 3.5.2 e 3.5.3 são válidos para $N_p \to \infty$. Porém, tomando N_p grande o suficiente, a literatura (WANG, 2009) demonstra, com vários exemplos, o sucesso do método, sendo também comprovado por este trabalho. Com um grau de estabilidade assegurado, pode-se agora implementar o GPC tipo m com restrições. Pela Seção 3.4, a restrição implementada está no tempo atual, i.e., $k = k_i$, e nesta amostra, conforme (3.69) e (3.71), $\hat{u}(k_i) = a^{-0}u(k_i) = u(k_i)$. Assim, as restrições de (3.63) ficam inalteradas e o problema do GPC tipo m se reduz a Eq. (3.96):

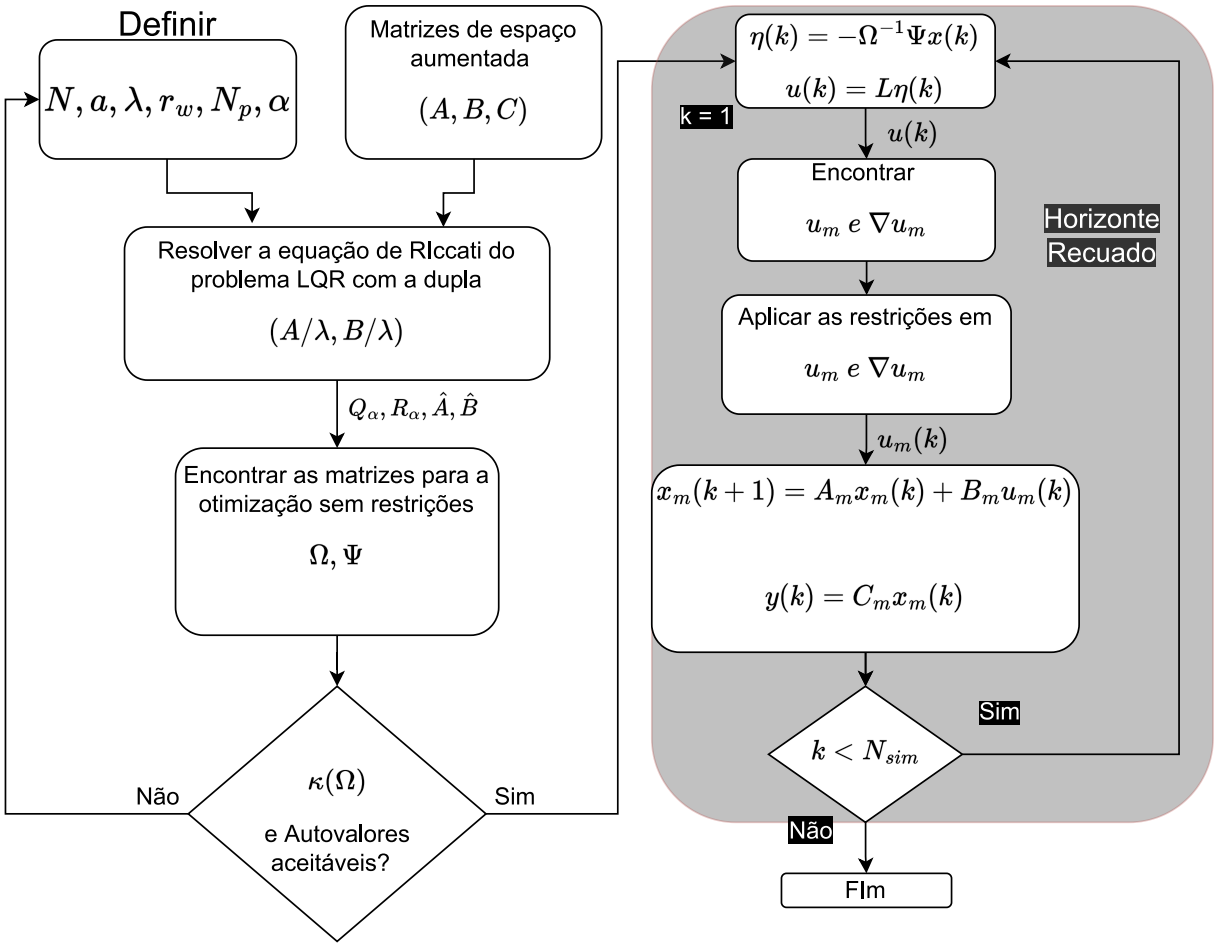
$$\begin{aligned} J &= \eta^T\Omega\eta + 2\eta^T\Psi x(k_i) \\ s.a: \ \nabla u_{min} \leq L(0)^T\eta - \sum_{i=1}^{m} C_m^i(-1)^i u_m(k_i - i) - u_m(k_i - 1) &\leq \nabla u_{max} \\ u_{min} + \sum_{i=1}^{m} C_m^i(-1)^i u_m(k_i - i) \leq L(0)^T\eta &\leq u_{max} + \sum_{i=1}^{m} C_m^i(-1)^i u_m(k_i - i), \end{aligned} \tag{3.96}$$

com matrizes Ω e Ψ dadas pela Eq. (3.97)

$$\Omega = \sum_{m=1}^{Np} \varphi(m) Q_a \varphi(m)^T + R_a,$$
$$\Psi = \sum_{m=1}^{Np} \varphi(m) Q_a \hat{A}^m, \tag{3.97}$$

e matrizes aumentadas modificadas $\hat{A} = A/a$ e $\hat{B} = B/a$. Ressalta-se que as demonstrações desenvolvidas do comportamento matemático dos controladores MPC e GPC tipo m baseados em funções de Laguerre com restrições e prescrição de estabilidade são de apoio para a aplicação das plantas nesta dissertação. O fluxograma do algoritmo MPC baseado em funções de Laguerre e com grau de estabilidade prescrito é dado na Figura 6. Os testes destes tipos de controladores nas plantas Boost e (CORDERO et al., 2021) são apresentados no Capítulo 4.

Figura 6 – Fluxograma do algoritmo MPC.



Fonte: o autor

4 APLICAÇÕES DE CONTROLADORES FRACIONÁRIOS E PREDITIVOS COM TESTABILIDADE EM SIMULAÇÕES E FPGA-*IN-THE-LOOP*

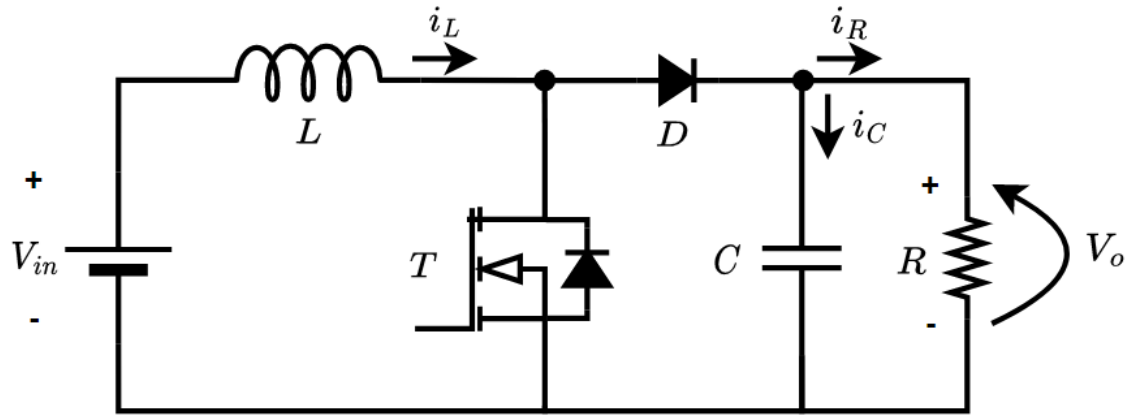
Neste capítulo, enunciar-se-á brevemente os sistemas a serem controlados. Em seguida, a sintonização de cada controlador será realizada com a simulação em Simulink. Logo depois, o FIL de cada controlador é sintetizado e comparado com a simulação matemática. Nos Apêndices B.1, B.2 e B.3, encontram-se os códigos-fonte em VHDL de cada controlador. Os códigos em VHDL foram construídos e depurados usando o *software* QUARTUS® PRIME LITE EDITION 21.1 juntamente com pacotes de matrizes estáticos *fixed_matrix_pkg.vhdl* e *real_matrix_pkg.vhdl*. A placa utilizada foi a CYCLONE IV E EP4CE115F29C7 com 114480 elementos lógicos e a interface de programação entre placa e computador foi vinculada pelo Joint *Test Action Group* (JTAG).

4.1 CONVERSOR BOOST

O conversor Boost é um dispositivo eletrônico de potência CC-CC que diminui a corrente, enquanto aumenta a tensão, da entrada (alimentação) até a saída (carga) (PEREIRA et al., 2022a). Este conversor está dentro da classe de fontes chaveadas (SMPS - do inglês, *Switched-mode Power Supply*) que contém, normalmente, dois semicondutores (um diodo e um transistor) e armazenadores de energia: um capacitor e um indutor, controlados por modulação por largura de pulso (PWM - do inglês, *Pulse Width Modulation*). Seu funcionamento baseia-se no carregamento do indutor pela fonte de entrada ao fechar o interruptor principal, com a carga sendo suprida pelo capacitor e na seguinte etapa, ao abrir o interruptor, inverte-se a polaridade do indutor pela lei de Lenz, impondo sobre o capacitor e a carga 2 fontes em série, que resulta em um aumento da tensão (ERICKSON, 2007). Nesta dissertação, a topologia do circuito do conversor Boost é mostrada na Figura 7.

A especificação do conversor e seus componentes são fornecidos nas Tabelas 4 e 5. Como a dinâmica do sistema, assim como qualquer SMPS, é não-linear, pela modelagem média de espaço de estados (ERICKSON, 2007), a função de transferência (4.1) para um conversor Boost linearizado em modo de tensão e em modo contínuo de condução (MCC) pode ser obtida pela Eq. (4.1):

Figura 7 – Conversor Boost.



Fonte: o autor

$$\frac{\tilde{v_o}(s)}{\tilde{d}(s)} = G_{Bo}(s) = \frac{V_o(1-D) - sLI_L}{LCs^2 + \frac{L}{R}s + (1-D)^2}, \tag{4.1}$$

sendo que $\tilde{v_o}(s)$ e $\tilde{d}(s)$ são as perturbações de pequenos sinais sobre os valores médios de estado estacionário de V_o e D (razão cíclica). Vale mencionar que o D na Figura 7 caracteriza o diodo e não a razão cíclica do conversor.

Tabela 4 – Especificações do conversor Boost

Especificação	**Valor**
Tensão de entrada, V_{in}	10 V
Tensão de saída, V_o	40 V
Ciclo de trabalho, D	0,75
Frequência de chaveamento do Mosfet, f_{sw}	20 KHz
Potência nominal de saída, P_o	16 W
Corrente nominal no indutor, I_L	1,6 A
Ripple máximo de corrente em %,	11,72%
Ripple máximo de tensão em %,	0,06%

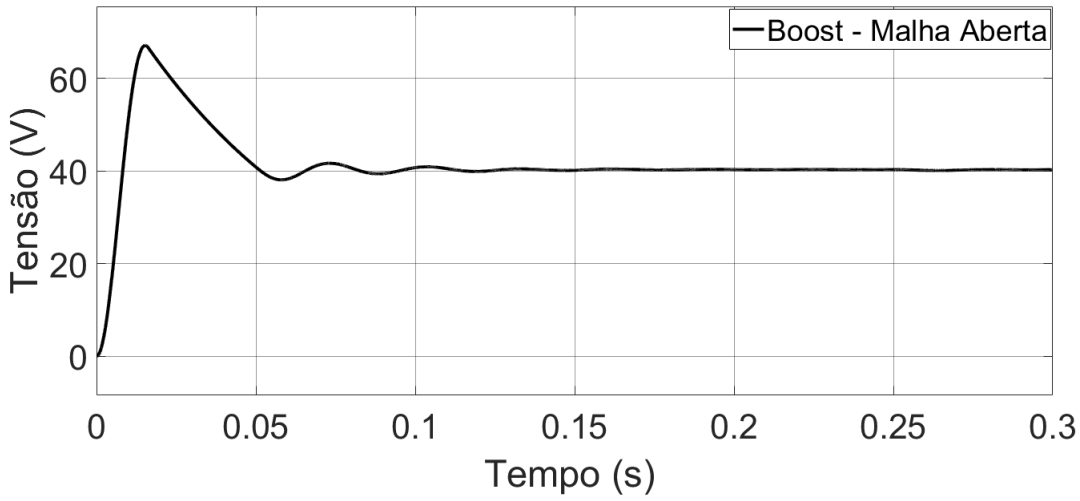
Através dos parâmetros disponíveis nas Tabelas 4 e 5, simula-se o comportamento do conversor Boost em malha aberta com um degrau de razão cíclica em 0,75 nas Figuras 8 e 9. Ressalta-se que a simulação é feita em ambiente MATLAB®/Simulink com a biblioteca *Simscape* com tensão direta de 0,8 V no diodo. Note que há grande *overshoot* na corrente de partida sobre o indutor, sendo impraticável a implementação do conversor Boost com a fonte de corrente existente em laboratório (5 A). Logo, tanto o PIDOF-Boost quanto o

Tabela 5 – Componentes do conversor Boost

Especificação	**Valor**
Carga resistiva, R	100 Ω
Indutância, L	2 mH
Capacitância de saída, C	680 μF

MPC-Boost devem atenuar a corrente de partida e, ao mesmo tempo, controlar o sistema com um certo grau de robustez ao se variar a carga. Por fim, verifica-se que o tempo de assentamento do conversor Boost em malha aberta é de 107 ms.

Figura 8 – Resposta em tensão do conversor Boost em malha aberta.

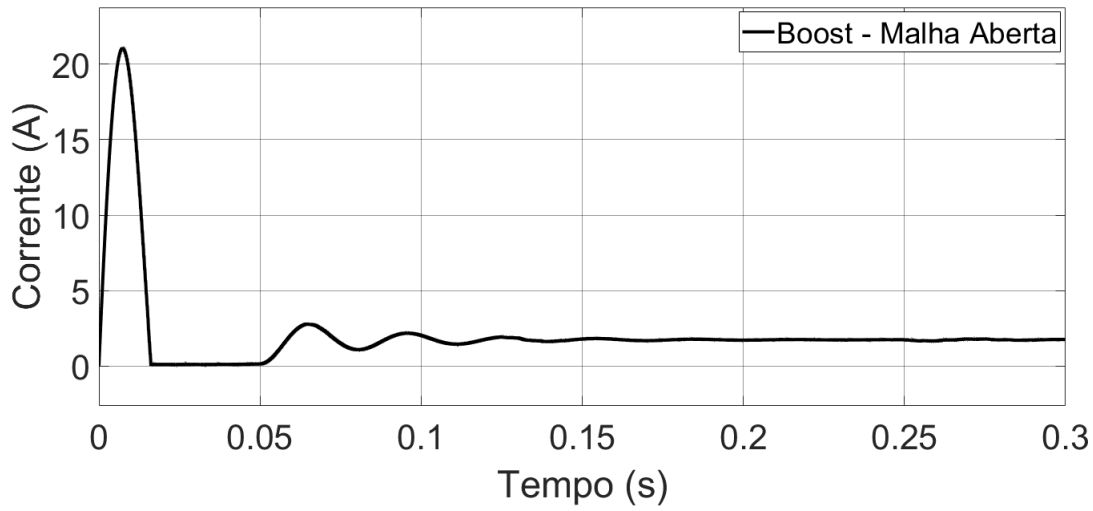


Fonte: o autor

4.1.1 CONTROLADOR PIDOF

Para as simulações com o controlador PIDOF, realizou-se o algoritmo DE/rand/1/bin juntamente com o limite superior do índice ITAE definido como 0,5 segundos e passo temporal de 1 ms. Nesta dissertação, o ITAE foi utilizado, pois há a necessidade de se penalizar o *overshoot* de tensão para que a drenagem de corrente no indutor seja mais suave. Com o tempo penalizando o aumento do índice, também se objetiva a regulagem rápida em tensão para que o controlador não fique muito mais lento que a sua resposta em malha aberta. Os parâmetros do algoritmo ED são fornecidos na Tabela 3. O range numérico de busca para λ e μ escolhido foi $[0, 5, 1, 5]$. Todas as simulações foram realizadas em ambiente MATLAB®/Simulink, e um bloco de saturação foi inserido antes do conversor Boost para limitar o seu ciclo de trabalho entre 0 e 0,9. Isso é necessário para dar segurança à operação dinâmica do Mosfet. Posteriormente, a abordagem de redução em norma L2

Figura 9 – Resposta em corrente no indutor do conversor Boost em malha aberta.



Fonte: o autor

será aplicada ao controlador. A ideia chave para reduzir o número de condição do PIDOF e o tamanho foi separá-lo em duas partes: PI de ordem fracionária (PIOF) (Eq. (4.2)) e; derivada de ordem fracionária (DOF) (Eq. (4.3)).

$$PIOF(s) = K_p + \frac{K_i}{s^\lambda}, \tag{4.2}$$

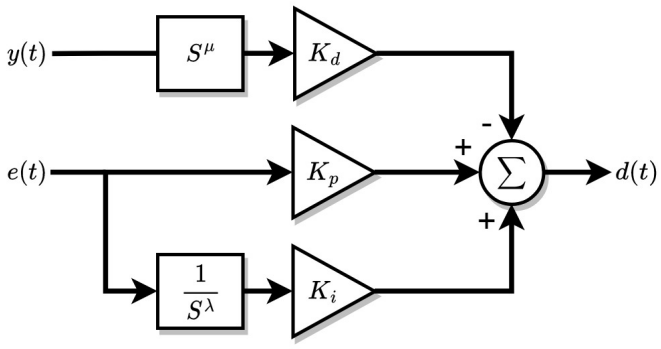
$$DOF(s) = K_d s^\mu. \tag{4.3}$$

Vale ressaltar que, para cada operador integrador ou derivador, existe um filtro passa-baixa de primeira ordem embutido com frequência de corte em ω_h rad/s para evitar funções de transferência impróprias. Encontrado o PIDOF de ordem reduzida, gera-se a sua representação em espaço de estados pela função tf2ss do MATLAB®. Subsequentemente, a representação contínua em espaço de estados é discretizada pela função c2dm do MATLAB®. Muitas simulações foram feitas com diferentes tempos de amostragem, e para esta dissertação, a resolução de frequência do filtro Oustaloup refinado foi o fator chave para a escolha do tempo de amostragem. Um intervalo de $T_s = [1/(10\omega_h), 2/(\omega_h)]$ foi testado. Um bom equilíbrio entre uso de memória e resolução em frequência do sistema foi alcançado com $T_s = 2/\omega_h$ para o conversor Boost. O número de bits na parte inteira das variáveis do FPGA foi escolhido como 16, enquanto a parte fracionária foi de 40 para o PIDOF-Boost. O controlador foi projetado para a necessidade de apenas um sensor de leitura da tensão na saída do conversor Boost. A quantidade de bits para a tensão de entrada no FPGA (sensor de tensão) foi escolhida por 12. Para sincronização, o *clock* do FPGA deve ser suficiente para realizar todos os cálculos em $T_s = 200$ μs, então uma modulação por largura de pulso (PWM) foi projetada para acionar o tempo de amostragem

com ciclo de trabalho de um *clock* do FPGA. Além disso, de acordo com a frequência de comutação do Mosfet de $f_{mosfet} = 20$ KHz (Tabela 4), outro PWM deve ser inserido no final do código para transformar o ciclo de trabalho numérico de f_s para sua representação de pulso de f_{mosfet}. Pode-se controlar a resolução do ciclo de trabalho escolhendo o *clock* do FPGA tal que um *clock* de $f_{clk} = 2$ MHz foi projetado, resultando em um fator de escala de $f_{clk}/f_{mosfet} = 100$ sobre o valor numérico do ciclo de trabalho. Por fim, o sinal final comuta o Mosfet e controla o funcionamento do conversor.

Para evitar um *undershoot* indesejável de tensão quando o Boost é perturbado por degrau de tensão na inicialização, uma diferente topologia de PIDOF é proposta em (PEREIRA et al., 2022a). Logo, o esquema do controlador PIDOF-Boost é mostrado na Figura 10.

Figura 10 – Esquemático do PIDOF-Boost.



Fonte: o autor

Observe que a entrada do conversor é o ciclo de trabalho $d(t)$ aplicado ao Mosfet T. Assim, a nova FTMF, para o conversor Boost linearizado e controlado pelo PIDOF da Figura 10, é dado pela Eq. (4.4):

$$FTMF_{Bo}(s) = \frac{G_{Bo}(s)(K_p + K_i s^{-\lambda})}{1 + G_{Bo}(s)(K_p + K_i s^{-\lambda} + K_d s^{\mu})}. \tag{4.4}$$

Note que o denominador de (4.4) é idêntico ao denominador de (2.44) portanto, todas as equações desenvolvidas na Seção 2.4 também satisfazem esta nova topologia de controle. Assim, para o conversor Boost linearizado (4.1), a frequência de cruzamento de ganho e os valores de margem de fase escolhidos foram $\omega_{gc} = 2\pi 200$ rad/s, ou seja, 200 Hz e $\phi_m = 60^o$.

De acordo com o algoritmo ED da Figura 5, após 100 simulações com $N = 1$ e $[\omega_b, \omega_h] = [10^{-4}, 10^4]$ rad/s, os melhores valores de λ e μ encontrados foram $\lambda = 1,5$ e $\mu = 1,3692$ para o ITAE mínimo. Além disso, os parâmetros de ganho do PIDOF para o conversor Boost são introduzidos na Tabela 6 junto com o índice de desempenho ITAE.

Tabela 6 – Parâmetros de ganho do PIDOF-Boost.

K_i	K_p	K_d	J_{ITAE}
1,3447	$1,5502 \cdot 10^{-3}$	$9,2047 \cdot 10^{-8}$	0.0047

Neste cenário, nenhum controlador PID é estável com as mesmas especificações de ω_{gc} e ϕ_m. Portanto, destaca-se a superioridade do PIDOF em relação ao PID quanto ao amortecimento iso e estabilidade. De acordo com os parâmetros de ajuste da Tabela 6, as expressões s^{μ} e $s^{-\lambda}$ para o controlador PIDOF-Boost são dadas pelas Eqs. (4.5) e (4.6):

$$s^{\mu} = \frac{4,571 \cdot 10^{5} s^{6} + 5,147 \cdot 10^{9} s^{5} + 7,605 \cdot 10^{11} s^{4} + 2,448 \cdot 10^{11} s^{3} + 1,694 \cdot 10^{8} s^{2}}{s^{6} + 2,906 \cdot 10^{4} s^{5} + 2,161 \cdot 10^{8} s^{4} + 2,547 \cdot 10^{11} s^{3} + 7,907 \cdot 10^{11} s^{2} + 5,307 \cdot 10^{9} s + 1,755 \cdot 10^{5}}, \tag{4.5}$$

$$s^{-\lambda} = \frac{52,7 s^{5} + 1,285 \cdot 10^{6} s^{4} + 2,529 \cdot 10^{9} s^{3} + 1,174 \cdot 10^{10} s^{2} + 1,176 \cdot 10^{8} s + 5270}{s^{7} + 2,121 \cdot 10^{4} s^{6} + 1,132 \cdot 10^{8} s^{5} + 1,114 \cdot 10^{10} s^{4} + 2,399 \cdot 10^{3} s^{3} + 1,111 \cdot 10^{6} s^{2}}. \tag{4.6}$$

Para garantir o requisito de tensão máxima em 3,3 V para a placa FPGA, um sensor de tensão deve ser inserido com ganho $H_{boost} = 3/V_o = 3/40 = 0,075$ para condicionamento do sinal. Portanto, o fator de escala para as variáveis de entrada na linha 140 do Apêndice B.1 para o PIDOF-Boost é $1/H_{boost} = 13,33$. A redução em norma L2 mais precisa para (4.5) e (4.6), com o menor tamanho possível, foi com a ordem (numerador, denominador) = (1,1). Para encontrar a expressão reduzida de $s^{-\lambda}$, foi necessário multiplicar (4.6) por s^2 e depois aplicar a redução em norma L2 e, assim, dividir o resultado por $1/s^2$. Portanto, as expressões DOF e PIOF reduzidas para o controlador PIDOF-Boost são dadas pelas Eqs. (4.7) e (4.8):

$$DOF_{boost\_red}(s) = \frac{0,04393 s}{s + 2,03 \cdot 10^{4}}, \tag{4.7}$$

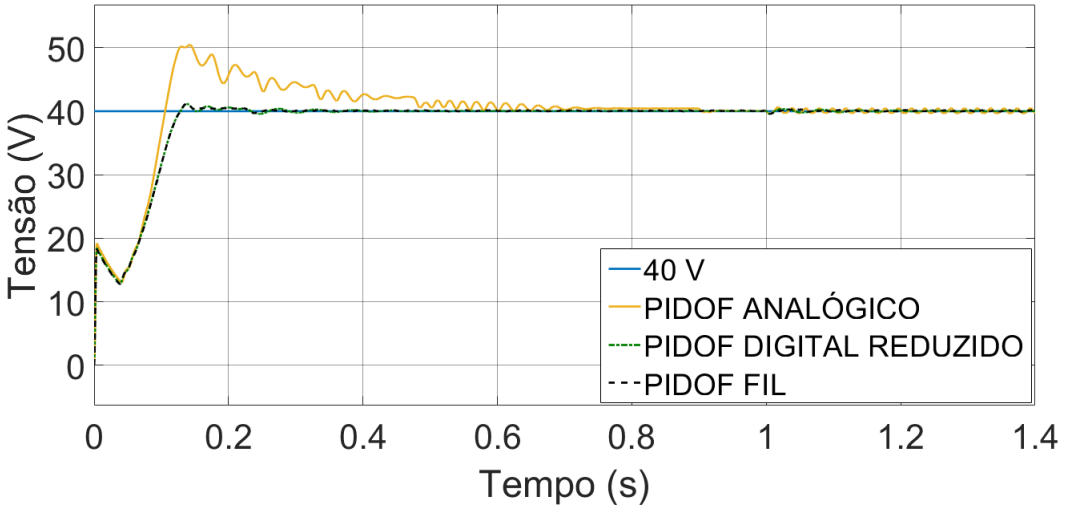
$$PIOF_{boost\_red}(s) = \frac{0,00155 s^{3} + 0,145 s^{2} + 31,82 s + 0,5966}{s^{3} + 93,54 s^{2}}. \tag{4.8}$$

Aplicando as funções tf2ss() e c2dm() do MATLAB® com $T_s = 200$ μs em (4.7) e (4.8), encontram-se as matrizes discretas em espaço de estado do controlador PIDOF-Boost reduzido.

Para realizar uma simulação numérica mais confiável, o conversor Boost foi projetado no Simulink com a biblioteca SIMSCAPE e com tensão direta no diodo de 0,8 V. Assim, o controlador PIDOF foi simulado ao longo do tempo até 1,4 segundos com a seleção do solver em passo fixo de $5 \cdot 10^{-7}$ segundos. Vale lembrar que o bloco integro-diferencial

(Figura 2) foi também utilizado para a simulação. Para comparar o FIL de ordem reduzida com os PIDOF-Boost analógicos e discretos, iniciou-se a simulação com uma potência de operação em 75%, ou seja, $R = 400/3\ \Omega$, até 1s, onde foi adicionada uma carga $R = 400\ \Omega$ em paralelo para atingir 100% da potência de operação. Nas Figuras 11, 12 e 13, mostra-se a resposta em tensão, corrente do indutor e o ciclo de trabalho desta simulação. Nota-se uma redução na corrente de partida do indutor de 20 A para 5 A (em relação a malha aberta - Figura 9), atingindo a necessidade imposta pela fonte de corrente existente para a implementação do teste em bancada com o PIDOF-Boost. Adicionalmente, a inclusão do controlador PIDOF resultou em um pequeno atraso no sistema quando comparado com a malha aberta. O tempo de acomodação para o PIDOF FIL é de 120 ms enquanto o da malha aberta é de 107 ms. O controlador PIDOF analógico atinge um tempo de estabilização de 0,6 s com uma resposta robusta contra variações de carga. Além disso, a variação de tensão foi inferior a 2% em relação ao estado estacionário com perturbação de carga para uma transição de potência de 12 W para 16 W. Contudo, a melhoria de *overshoot* e do tempo de estabilização foi bastante perceptível com a redução em norma L2 e sem perder o tempo de subida. Nota-se que o PIDOF digital reduzido aumentou a robustez à variação de carga com oscilação de tensão inferior a 1%. Finalmente, as reduções de tamanho de seis para um em DOF(s) e sete para três em PIOF(s) destacam a eficácia do método para a implementação do PIDOF em *hardware*.

Figura 11 – Resposta em tensão do PIDOF-Boost.

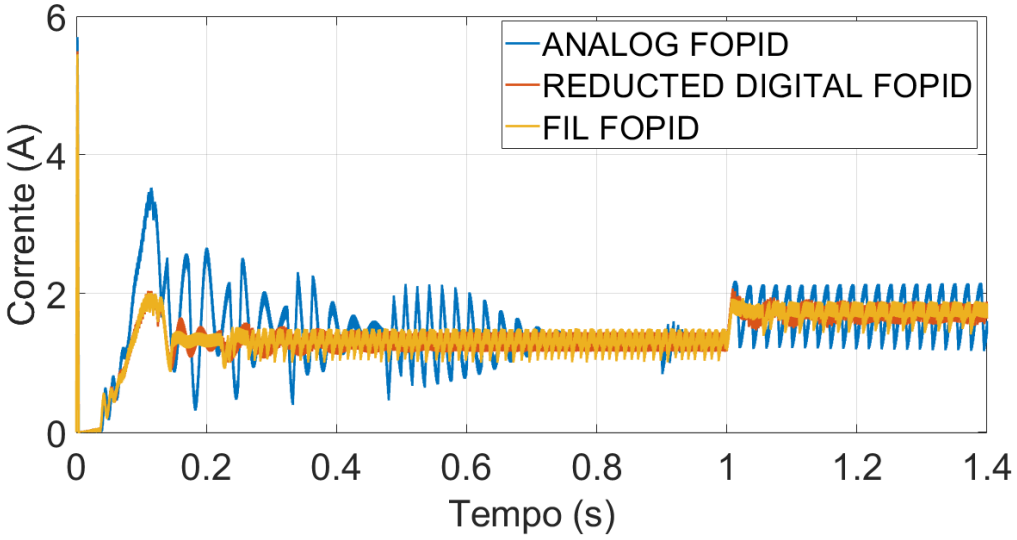


Fonte: o autor

Para um estudo experimental, realizou-se um teste em bancada com os parâmetros especificados nas Tabelas 4, 5 e 7. A fim de verificar a viabilidade prática e a eficácia do método proposto, consideram-se dois casos:

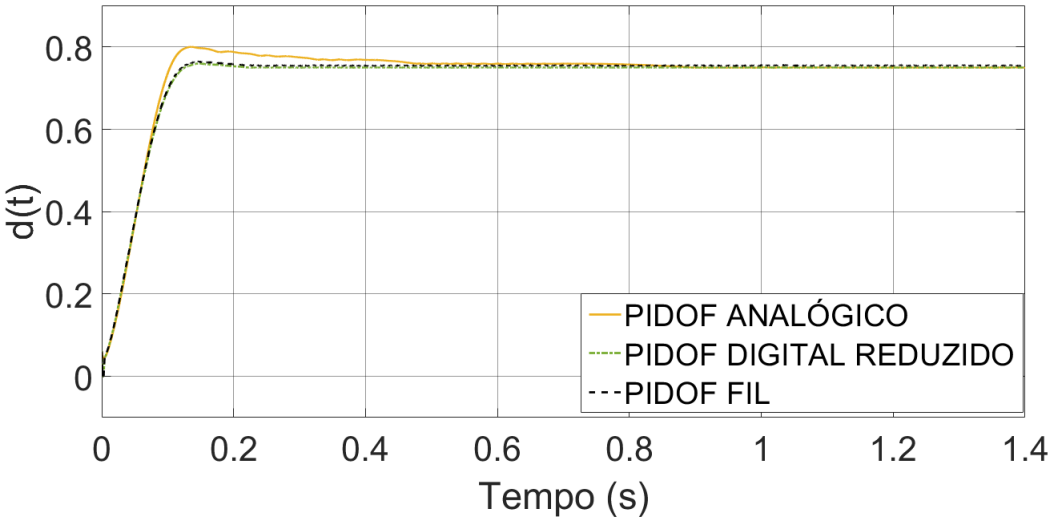
Caso 1 Cenário de inicialização para analisar a resposta transiente do conversor.

Figura 12 – Resposta de corrente no indutor do PIDOF-Boost.



Fonte: o autor

Figura 13 – Ciclo de trabalho do PIDOF-Boost.



Fonte: o autor

Caso 2 A carga resistiva R muda de 100 $\Omega \rightarrow$ 200 $\Omega \rightarrow$ 100 Ω enquanto a tensão da fonte de entrada $V_{in} = 10\ V$ e a tensão de referência $V_o = 40\ V$ são mantidas constantes.

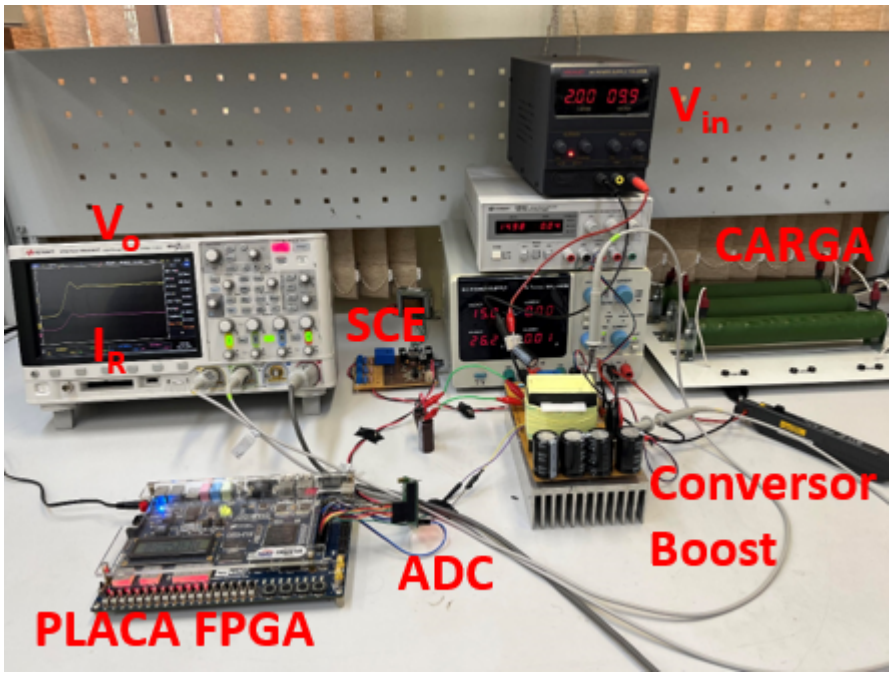
Tabela 7 – Parâmetros eletrônicos para o teste em bancada do conversor Boost.

Parâmetro	**Especificação**
Sensor de tensão	LV 20-P
Diodo de saída, D	RHRP860 FAIRCHILD
Mosfet principal, T	IRG4PH50UD INFINEON
Conversor AD	Dual-channel AD9226 (12 bits)

Uma das principais vantagens do método proposto em relação à implementação em *hardware* é a exigência de apenas um sensor. O sensor de tensão reside em um sistema de condicionamento de energia (SCE) fora da placa do conversor Boost. Além do SCE atenuar a tensão, ele insere um offset de 1,49 V para evitar que o valor medido fique abaixo de 0 V. Além disso, a saída do conversor AD funciona com a forma de complemento dois e o ganho do sensor é de 177 (ao invés dos 13,333). Um osciloscópio keyight infiniivision DSOX3054T é empregado para medir e plotar a tensão de saída V_o e a corrente de carga I_R. Na Figura 14, apresenta-se a configuração experimental do teste em bancada com o PIDOF-Boost. Sob o *clock* do FPGA de 50 MHz, o clk_{T_s} foi projetado com ciclo de trabalho de 5000 clks, ou seja, 50%, para evitar possíveis erros da leitura de entrada no conversor AD. Logo, vale ressaltar que o *clock* do FPGA de 50 MHz leva a um aumento do fator de escala numérico (linha 373 do Apêndice B.1) do ciclo de trabalho para 2500. Este aumento na frequência eleva a resolução do sistema, resultando em uma resposta mais suave do que a gerada pela simulação com FIL. Vale mencionar ainda que a restrição de $0 \leq d(t) \leq 0,9$ também foi introduzida no FPGA.

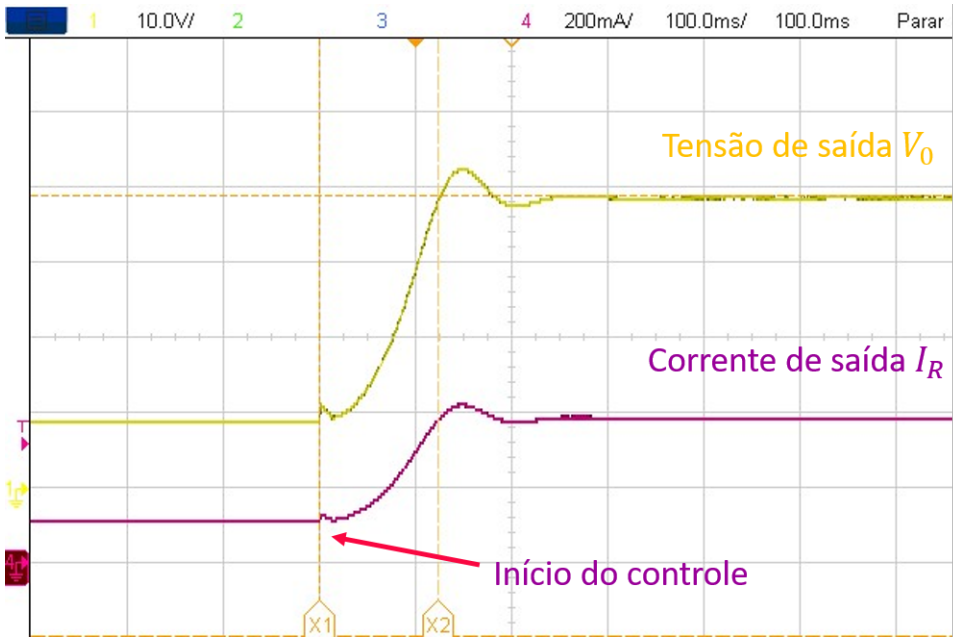
O caso 1 está representado na Figura 15. Perceba que há uma introdução de 10% no *overshoot* no que se diz respeito à simulação FIL. Isso é explicado pela limitação da fonte de alimentação em relação à corrente. A fonte fornece uma corrente de até 5 A no indutor, portanto a simulação experimental não pode se mover mais rápido do que a simulação FIL. Porém, o tempo de acomodação entre elas permanece quase o mesmo. Além disso, o tempo de subida para atingir o regime estacionário de tensão foi de 124 ms ($0\ V \rightarrow 40\ V$).

Figura 14 – Configuração experimental do teste em bancada com o PIDOF-Boost.



Fonte: o autor

Figura 15 – Inicialização do PIDOF-Boost experimental.

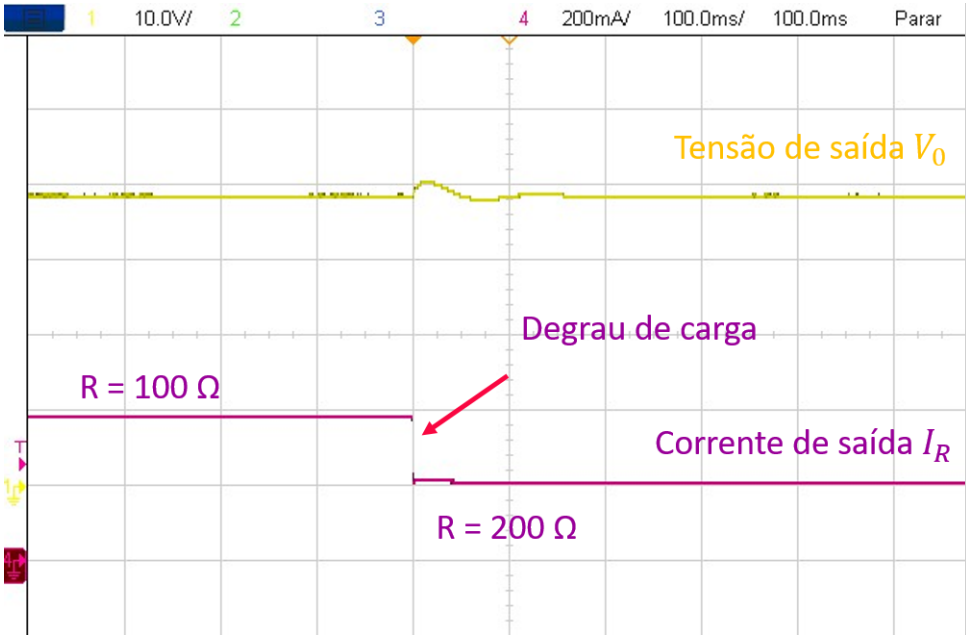


Fonte: o autor

Nas Figuras 16 e 17, ilustra-se o caso 2. Este caso demonstra a mudança de potência de 16 W para 8 W e vice-versa. Observe um pico de oscilação de 1% na tensão em relação

ao seu valor estacionário com um tempo de estabilização de 76 ms. Isto destaca a robustez do controlador PIDOF contra variações de carga e está de acordo com a simulação FIL. Finalmente, o controlador proposto alcança convergência rápida e uma ondulação de tensão em estado estacionário adequada em torno de 300 $mV \approx 0,75\%$ da tensão nominal de saída.

Figura 16 – Tensão de saída e corrente de carga durante a perturbação na carga de 16 W a 8W do PIDOF-Boost.



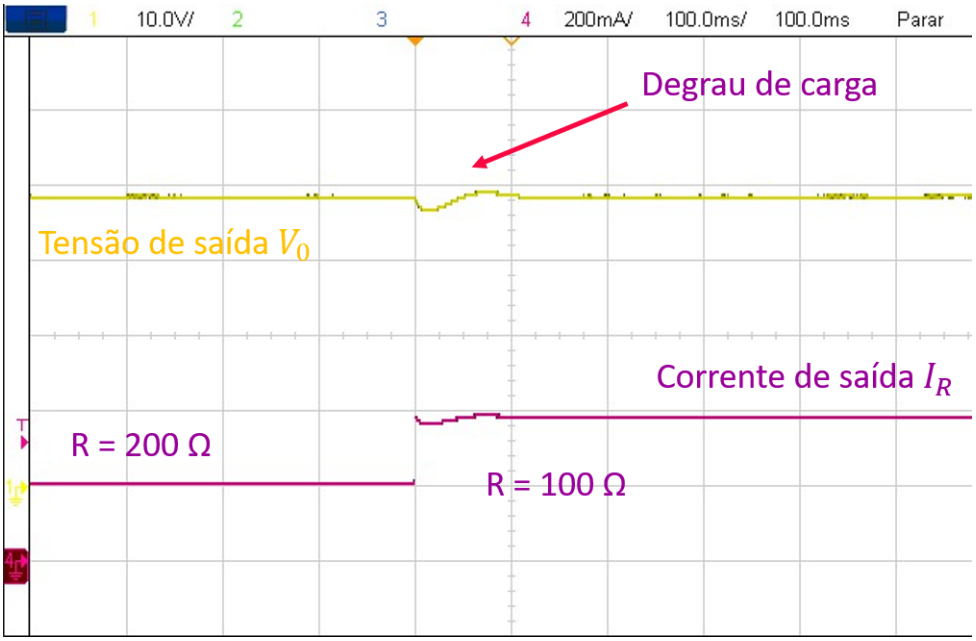
Fonte: o autor

4.1.2 CONTROLADOR MPC

Para as simulações com controladores MPC tipo 1, o mesmo conversor Boost foi utilizado. Todas as configurações relacionadas ao *clock* de simulação do FIL e à frequência de amostragem do conversor do projeto PIDOF-Boost foram utilizadas para a implementação do código MPC tipo 1, ou seja, $f_{clk} = 2$ MHz e $T_s = 200$ μs. O número de bits na parte inteira das variáveis do FPGA para o MPC-Boost foi escolhido como 21 enquanto a parte fracionária foi de 28. O controlador também foi projetado para a necessidade de apenas um sensor de tensão cuja quantidade de bits foi escolhida por 12.

Um grande problema encontrado para a sintonia do MPC tipo 1 com funções de Laguerre para o conversor Boost foi ajustar sua velocidade com a demanda inicial de corrente necessária para a inicialização do mesmo. Ao seguir a literatura (WANG, 2009) e (SAEED; WANG; FERNANDO, 2022), a otimização quadrática por função convexa da teoria do MPC conduziu a um controlador muito rápido e com drenagem inicial da corrente do indutor muito elevada. Como o laboratório da UFMS tem apenas uma fonte de

Figura 17 – Tensão de saída e corrente de carga durante a perturbação na carga de 8 W a 16 W do PIDOF-Boost.



Fonte: o autor

alimentação com limitação de corrente em 5 A, foi necessário desacelerar o controlador para que a corrente de partida diminuísse. Dessa forma, o controlador pôde ser implementado em bancada.

Conforme a teoria disposta Seção 3.1, um modo de diminuir a velocidade do controlador é aumentar o valor de r_w em (3.24). Dessa forma, o valor de η fica restrito a uma pequena variação e, por conseguinte, $u(k)$ também, desacelerando o sistema. Contudo, como o N_p deve ser grande, um r_w elevado aumenta o valor do número de condição $\kappa(\Omega)$ da Hessiana (3.64). Ao definir r_w alto, é necessário impor um certo grau de estabilidade, i.e., diminuir λ_{max} do sistema em malha fechada. Observou-se que, ao diminuir λ_{max} do sistema em malha fechada, aumentava-se o $\kappa(\Omega)$. Neste cenário, os polos das funções da Laguerre são definidos por α e a quantidade de funções por N. Dessa forma, o projeto final ficou com a sintonia do valor (a) para ponderar exponencialmente a função custo e reduzir $\kappa(\Omega)$. Na Tabela 8, apresentam-se parâmetros de configuração do MPC-Boost.

Pela restrição do ciclo de trabalho do conversor Boost, uma restrição $0 \leq u(k) \leq 0,9$ foi configurada. Isso se fez necessário para não sobrecarregar o Mosfet no teste em bancada. Como o controle foi sintonizado com a cautela de limite máximo de corrente, a taxa de variação numérica do ciclo de trabalho por T_s não afeta muito o sistema e, dessa forma, impôs-se que $-1 \leq \nabla u(k) \leq 1$.

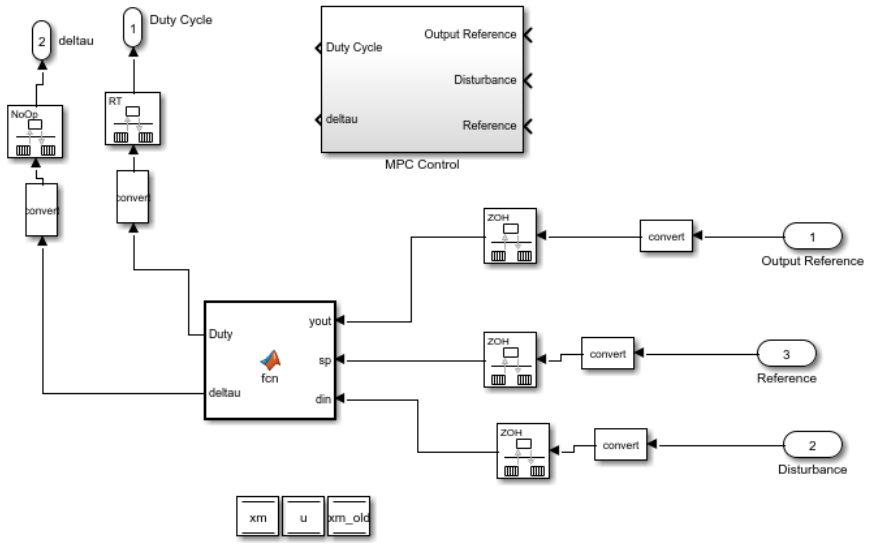
Com um bloco "Função MATLAB®", escreveu-se uma função do MPC-Boost

Tabela 8 – Parâmetros de configuração do MPC-Boost.

Parâmetro	**Valor**
Constante multiplicativa da matriz R_L, r_w	$5 \cdot 10^8$
Polo das funções de Laguerre, α	0,4
Número de funções de Laguerre, N	5
Número de predição, N_p	300
Limite superior do autovalor, λ_{max}	0,97
Fator de ponderação exponencial, a	1,001

para simulação no Simulink. A função MATLAB® que é inserida no bloco é executada simultaneamente com o sistema em análise, simulando um sistema digital próximo a um *hardware*. Verifica-se o bloco MPC gerado pela função do MATLAB® na Figura 18. Note que três variáveis globais foram criadas para a atualização do MPC: (xm) estado atual; (u) valor numérico do ciclo de trabalho e; (xm_old): estado anterior.

Figura 18 – Bloco "Função MATLAB®" do MPC.

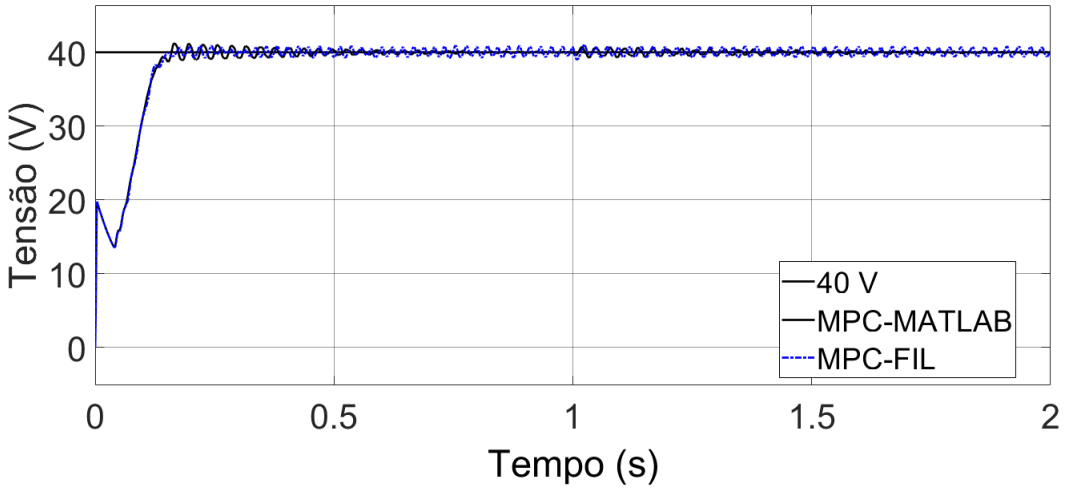


Fonte: o autor

Através de simulações numéricas, o controlador MPC foi simulado ao longo do tempo até dois segundos com a seleção do solver em passo fixo de $5 \cdot 10^{-7}$ segundos. Comparou-se a implementação do controlador MPC-MATLAB® discreto e MPC-FIL nesta primeira análise. A simulação começou com 75% da potência nominal de operação do conversor, ou seja, $R = 400/3\ \Omega$, até 1s, onde foi adicionada uma carga $R = 400\ \Omega$, em paralelo, para atingir a potência de 100% de operação. Nas Figuras 19, 20 e 21, mostram-se a tensão, o ciclo de trabalho e a corrente desta simulação.

A primeira discussão a ser feita é sobre a maior oscilação em tensão em estado estacionário do FIL na Figura 19. Isso se deu pelo erro de truncamento para valores menores que 2^{-28} nas operações do FPGA e a escolha de 12 bits para as variáveis de entrada e saída do FPGA. Dessa forma, para a potência de 12 W, o MPC-MATLAB® possui um *ripple* em estado estacionário menor que 300 mV enquanto o MPC-FIL ultrapassa 1,45 V. Na potência de 16 W, o distúrbio de carga gera uma oscilação muito pequena para ambos os controles, porém, a diferença percentual de *ripple* se torna mais pronunciável entre o MPC-MATLAB® e o MPC-FIL. O MPC-MATLAB® fica com um *ripple* menor de 250 mV e o MPC-FIL com 1,35 V, um aumento percentual relativo na diferença de 14,79% em relação à operação de 12 W. Esta oscilação é mais evidente ao verificar as correntes na Figura 20. Note também uma alta corrente inicial no indutor, atingindo o valor de 6 A. Por fim, o valor numérico do ciclo de trabalho é disposto na Figura 21. Na Figura 21, observa-se a não necessidade de se aplicar restrições quanto $\nabla d(k)$, pois o controle precisa de range variável livre para manipular a corrente e assim controlar a tensão. Por fim, vale comentar que o *overshoot* do MPC-Boost é menor do que o do PIDOF-Boost e que os tempos de subida são semelhantes.

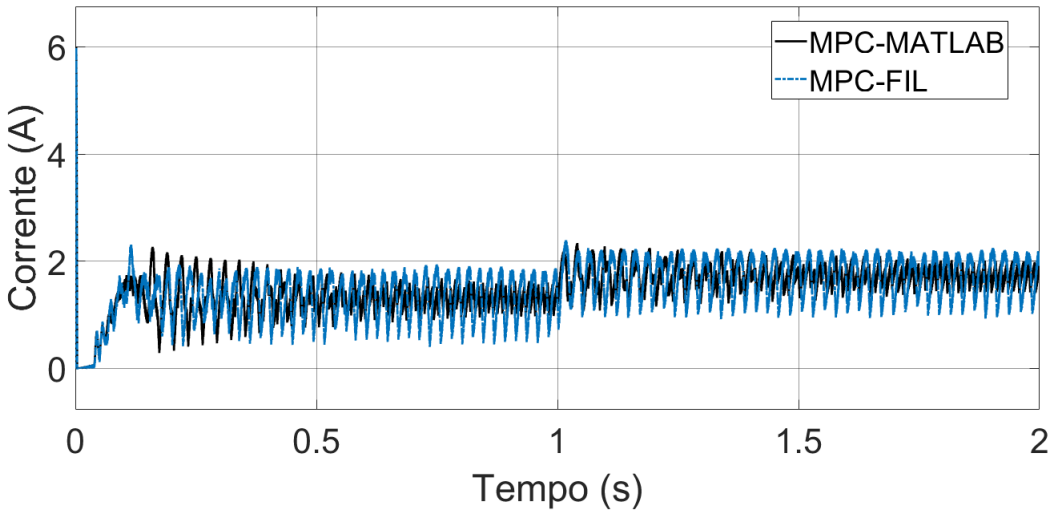
Figura 19 – Comparação da tensão entre o MPC-MATLAB® e MPC-FIL do conversor Boost.



Fonte: o autor

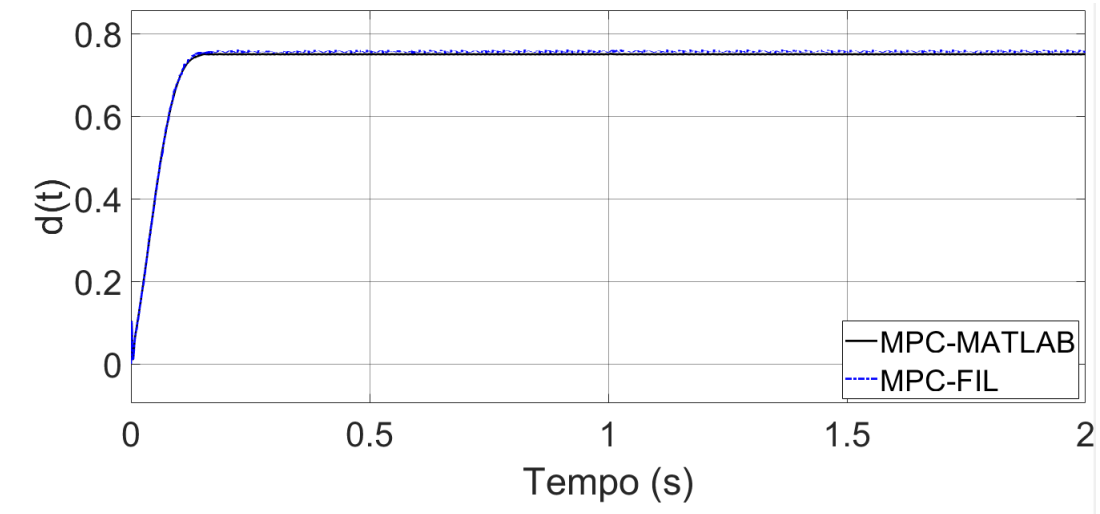
Para o estudo experimental, realizou-se um teste em bancada com os parâmetros especificados nas Tabelas 4, 5 e 7. Ou seja, o experimento foi realizado com todos os elementos utilizados no PIDOF-Boost, porém com o algoritmo do Apêndice B.2 ajustado para o MPC-Boost. Na Figura 22, demonstra-se a configuração experimental do teste em bancada com o MPC-Boost. Como o tempo de amostragem do conversor Boost e a frequência do FPGA permanecem os mesmos, as dinâmicas dos pulsos diferem apenas de alguns bits de sinalização entre o cálculo sequencial dos controladores. Dessa forma, o

Figura 20 – Comparação da corrente entre o MPC-MATLAB® e MPC-FIL do conversor Boost.



Fonte: o autor

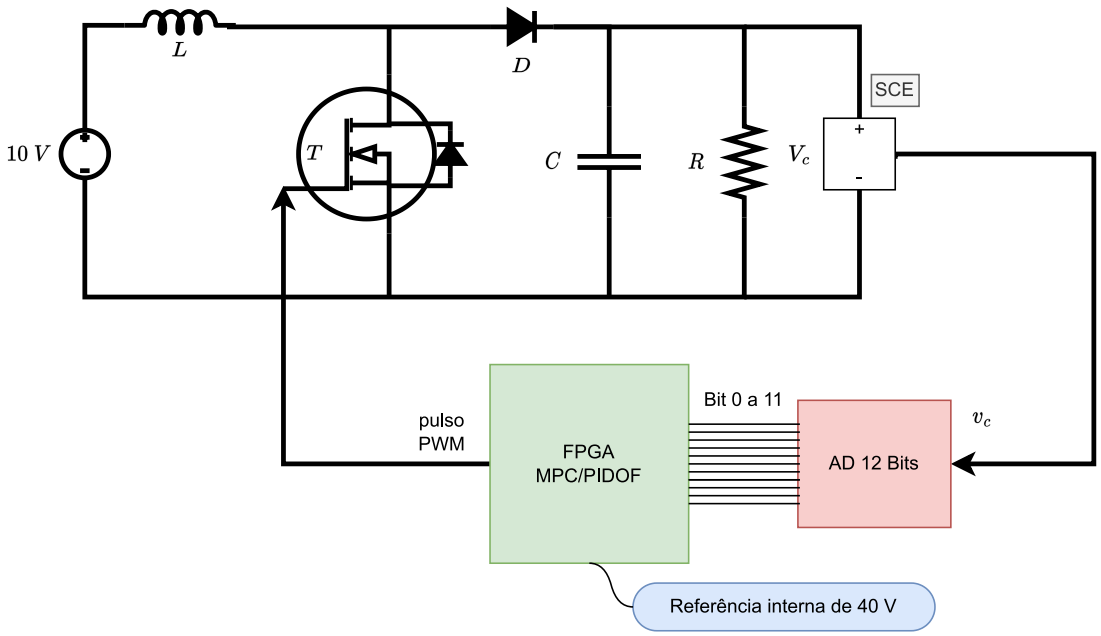
Figura 21 – Comparação do valor numérico do ciclo de trabalho entre o MPC-MATLAB® e MPC-FIL do conversor Boost.



Fonte: o autor

MPC-Boost precisa de 30 *clocks* internos para efetuar um passo de filtragem enquanto o PIDOF-Boost precisa apenas de 17 *clocks*.

Figura 22 – Configuração experimental do teste em bancada com o MPC-Boost.



Fonte: o autor

A fim de verificar a viabilidade prática e a eficácia do método proposto, consideram-se dois casos:

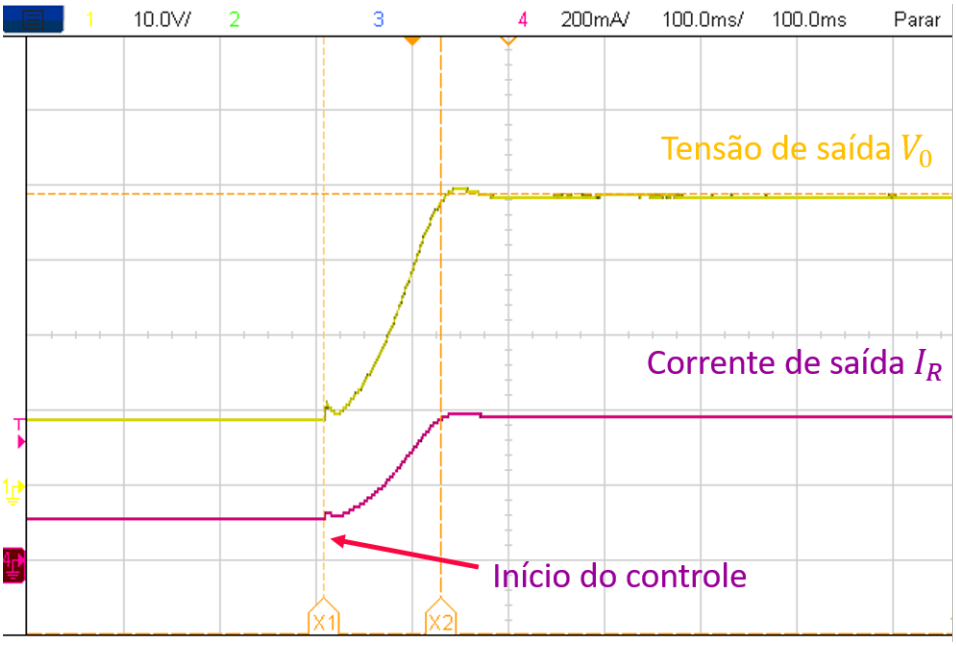
Caso 1 Cenário de inicialização para analisar a resposta transiente do conversor.

Caso 2 A carga resistiva R muda de 100 Ω → 200 Ω → 100 Ω enquanto a tensão da fonte de entrada $V_{in} = 10\ V$ e a tensão de referência $V_o = 40\ V$ são mantidas constantes.

O Caso 1 é visto na Figura 23. É importante notar o *overshoot* do MPC-Boost de apenas 2,56%, muito inferior ao do PIDOF-Boost. Mesmo com a limitação da fonte de corrente em 5 A sobre o indutor, o tempo de subida do MPC-Boost foi de 122 ms ($0\ V \rightarrow 40\ V$), 2 ms mais rápido que o PIDOF-Boost. Entretanto, em relação à simulação FIL da figura 19, a oscilação da tensão em regime estacionário do MPC-Boost diminui para um valor de 200 mV enquanto o tempo de subida deste também cai de 150 ms para 122 ms. A justificativa desta melhora está relacionada com o aumento da resolução em *clocks* internos do sistema em bancada de 400 para 10000.

Nas Figuras 24 e 25, pode-se observar os resultados em bancada para o Caso 2. Observa-se um pico de oscilação de 5,13% na tensão em relação ao seu valor estacionário com um tempo de estabilização de 42 ms. Isto destaca a robustez do controlador MPC-Boost contra variações de carga, superando a simulação MPC-FIL e MPC-MATLAB® das

Figura 23 – Inicialização do MPC-Boost experimental.



Fonte: o autor

Figuras 19 e 20. Finalmente, comparando o MPC-Boost com o PIFOF-Boost, o MPC-Boost estabiliza mais rápido quanto aos distúrbios de carga, porém tem um desvio de tensão maior.

Por fim, uma simulação é realizada com uma planta de segunda ordem para corroborar o método do GPC tipo m ao rastrear uma função polinomial de ordem 2.

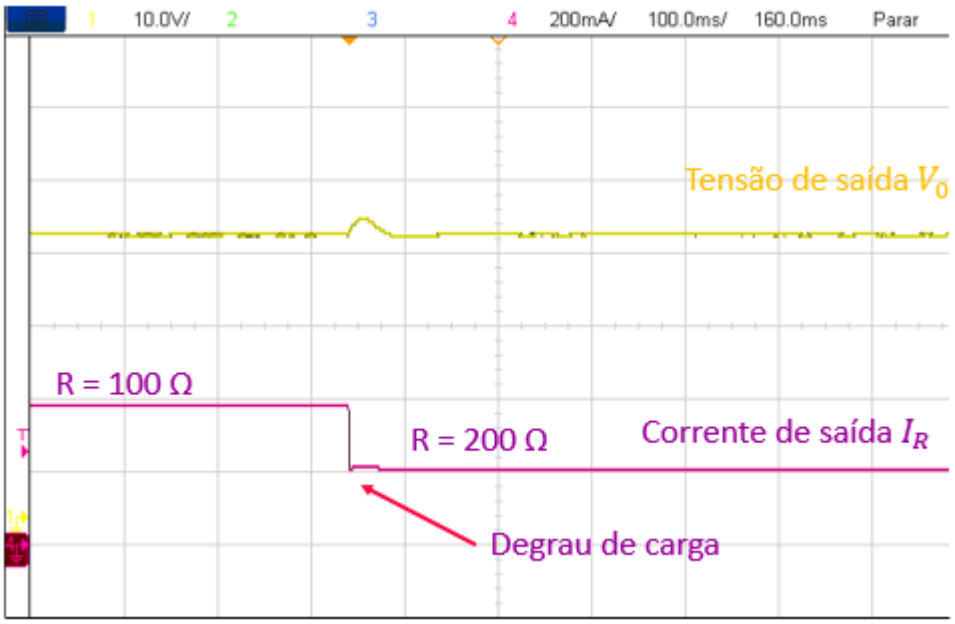
4.2 SIMULAÇÃO COM RASTREAMENTO DE REFERÊNCIA POLINOMIAL

A planta utilizada para essa seção foi retirada de (CORDERO et al., 2021) e está descrita na Eq. (4.9):

$$G(s) = \frac{53,48s + 5347,59}{s^2 + 253,5s + 10695,18}. \tag{4.9}$$

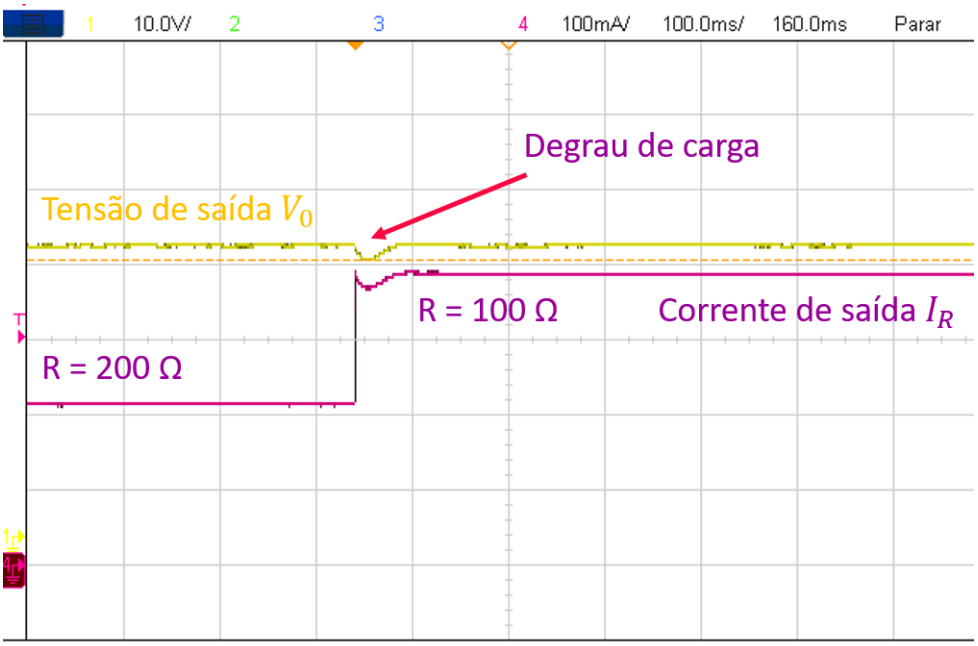
Note que o sistema descrito em (4.9) é um sistema *Single-Input Single-Output* (SISO). Assim, discretizando (4.9) com $T_s = 200\ \mu$s através da função c2dm() do MATLAB®, obtêm-se as matrizes de espaço de estado (A_m, B_m, C_m, D_m) do sistema a ser controlado.

Figura 24 – Tensão de saída e corrente de carga durante a perturbação na carga de 16 W a 8 W do MPC-Boost.



Fonte: o autor

Figura 25 – Tensão de saída e corrente de carga durante a perturbação na carga de 8 W a 16 W do MPC-Boost.

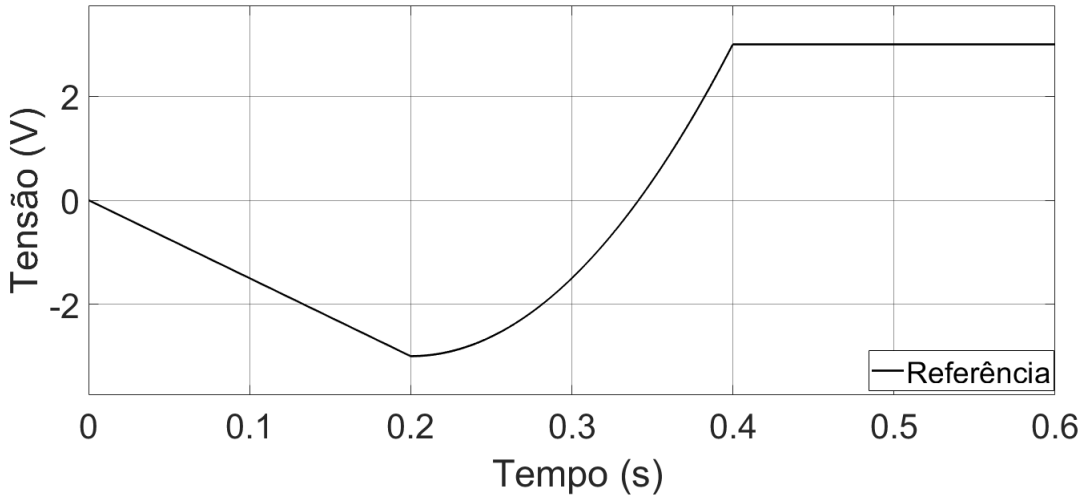


Fonte: o autor

$$A_m = \begin{bmatrix} 0,9504 & -2,0856 \\ 1,95 \cdot 10^{-4} & 0,9998 \end{bmatrix}, \; B_m = \begin{bmatrix} 1,95 \cdot 10^{-4} \\ 1,9567 \cdot 10^{-8} \end{bmatrix}, \; C_m = \begin{bmatrix} 53,48 \\ 5,3476 \cdot 10^{3} \end{bmatrix}, \; D_m = 0. \tag{4.10}$$

O polinômio de maior ordem a ser rastreado é uma parábola conforme a Figura 26. Como (4.10) é controlável, observável e possui estrutura mínima, aplica-se o Teorema 3.2.1 para se obter erro assintótico nulo ao sistema aumentado de (4.10) com m = 3 integradores embutidos, i.e., GPC tipo 3.

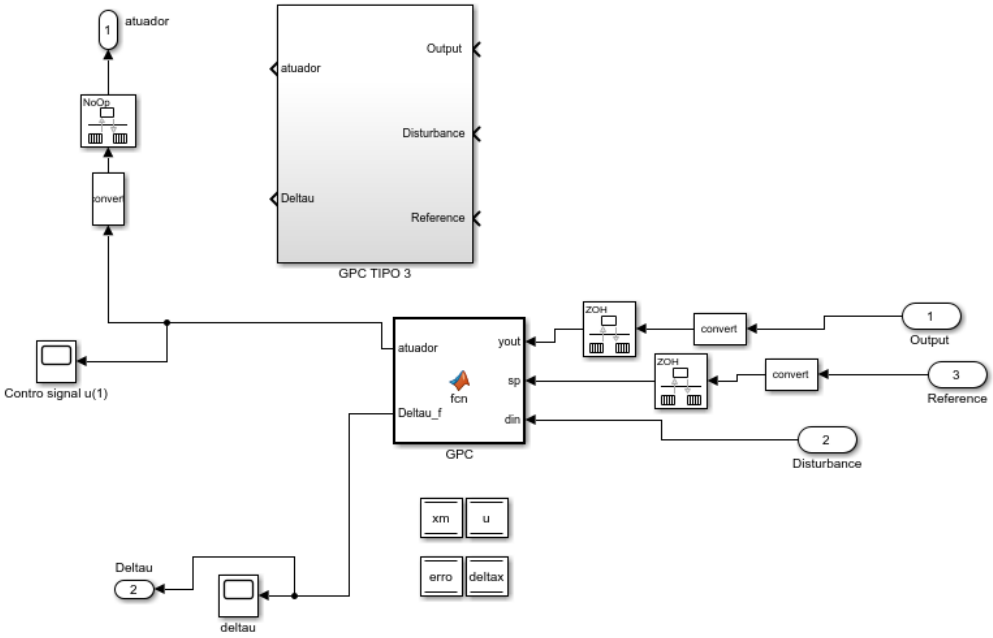
Figura 26 – Tensão de referência a ser rastreada.



Fonte: o autor

Através da teoria na Seção 3.2, objetivou-se, inicialmente, projetar um controlador com o menor tamanho possível e sem restrições. Isso se dá com $N = 1$, ou seja, apenas uma função de Laguerre para identificar a variável manipulada aumentada $u(t) = \nabla^3 u_m(t)$. Com um bloco "Função MATLAB®", escreveu-se uma função do GPC tipo 3 para o uso no Simulink. A função MATLAB® que é inserida no bloco é executada simultaneamente com o sistema em análise, simulando um sistema digital próximo a um *hardware*. Na Figura 27, verifica-se o bloco GPC tipo 3 gerado pela função do MATLAB®. Note que quatro variáveis globais foram criadas para a atualização do GPC tipo 3: (xm = $x_m(k+1) \in \mathbb{R}^{2\times 1}$) estado futuro; ($u = \begin{bmatrix} u(k) & u(k-1) & u(k-2) & u(k-3) \end{bmatrix}^T$) vetor da variável manipulada; (erro = $\begin{bmatrix} e(k) & e(k-1) & e(k-2) \end{bmatrix}$) vetor erro e; (deltax = $\begin{bmatrix} x(k) & x(k-1) & x(k-2) & x(k-3) \end{bmatrix}$) vetor estados. As entradas desse bloco são: (*Output*) saída a ser controlada; (*Disturbance*) distúrbio conhecido e; (*Reference*) referência. Por fim, as saídas são: (atuador) variável manipulada ou $u_m(t)$ e; (Deltau) $\nabla u_m(t)$.

Figura 27 – Bloco "Função MATLAB®" do GPC tipo 3.



Fonte: o autor

Na Tabela 9, apresenta-se os parâmetros de configuração do GPC tipo 3 com N = 1 (uma função de Laguerre) e sem restrições.

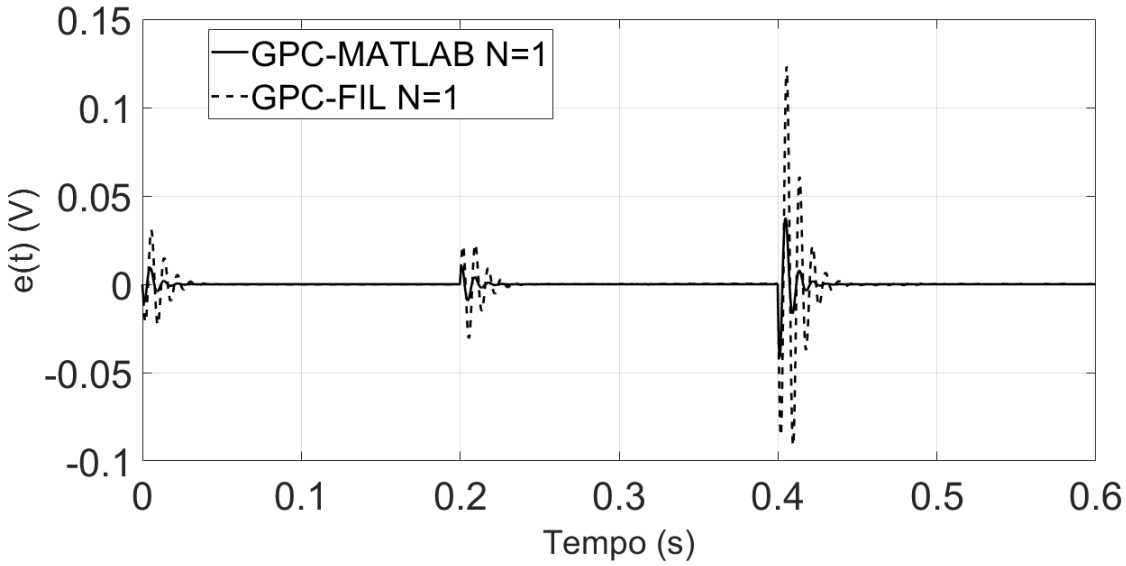
Tabela 9 – Parâmetros de configuração GPC tipo 3 com N = 1 e sem restrições.

Parâmetro	**Valor**
Constante multiplicativa da matriz R_L, r_w	10^{-5}
Polo da função de Laguerre, α	0,4
Número de funções de Laguerre, N	1
Número de predição, N_p	300
Limite superior do autovalor, λ_{max}	1
Fator de ponderação exponencial, a	1,2

Para o projeto FIL, os números de bits das variáveis utilizadas foram escolhidos por: 20 bits na parte inteira e; 50 bits na parte fracionária, totalizando 71 bits (1 de sinal). Isto se fez necessário, pois o maior valor declarado no código VHDL foi $338241 < 2^{20} - 1$. Para o valor fracionário, simulações com o bloco da Figura 27 foram realizadas e 50 bits foram suficientes. Por fim, os sinais de entrada e saída foram declarados com 16 bits, com a exceção do sinal "distúrbio" o qual foi descartado dos testes. O código em VHDL para a implementação do FIL foi projetado para efetuar todos os cálculos em 30 clks. Dessa forma, um *clock* interno de $T_{clk} = T_s/40 = 5$ μs foi escolhido para a simulação. Nas Figuras 28, 29, 30, mostra-se o erro ($e(t)$), variável manipulada ($u_m(t)$) e a saída controlada ($y(t)$)

do GPC tipo 3 FIL e MATLAB® com uma função de Laguerre e sem restrições.

Figura 28 – GPC tipo 3 sem restrições (N=1): $e(t)$.

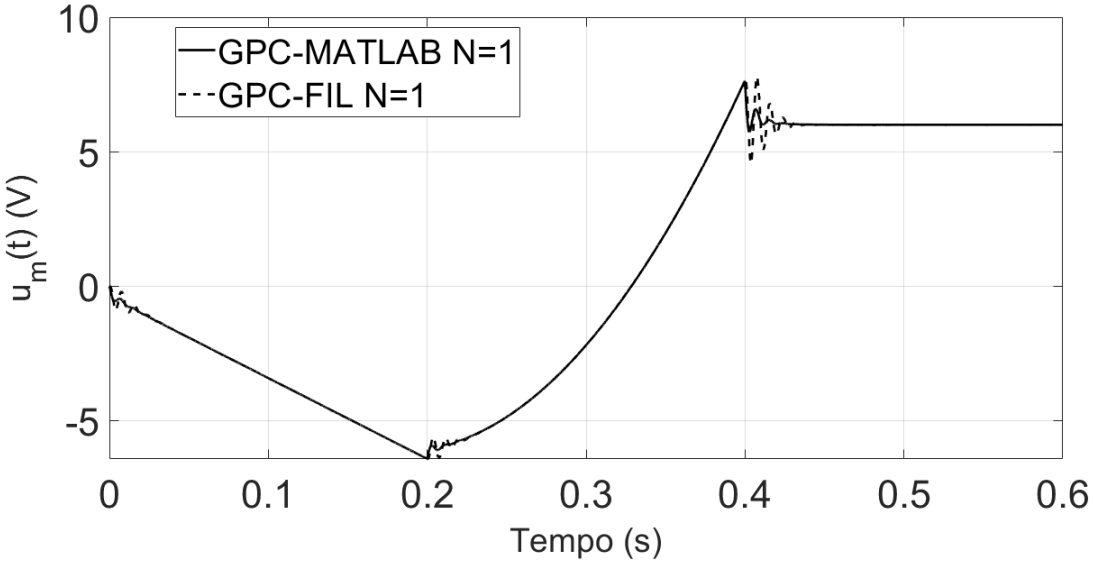


Fonte: o autor

Note que o erro assintótico é nulo na Figura 28 para o rastreamento da reta (0 s<t<0,2 s), parábola (0,2 s<t<0,4 s) e 3 V (0,4 s<t<0,6 s) com apenas uma função de Laguerre. O erro quadrático médio (EQM) encontrado de e(t) para o GPC-MATLAB® (N=1) foi de $1,2 \cdot 10^{-5}$. Para o GPC-FIL (N=1), o EQM foi de $1,36 \cdot 10^{-4}$. Esta diferença se dá pelo truncamento de bits dos sinais de entrada e saída. O projetista poderia escolher um número maior de bits, porém o custo para a construção do circuito aumentaria com a aquisição de conversores AD com mais de 16 bits.

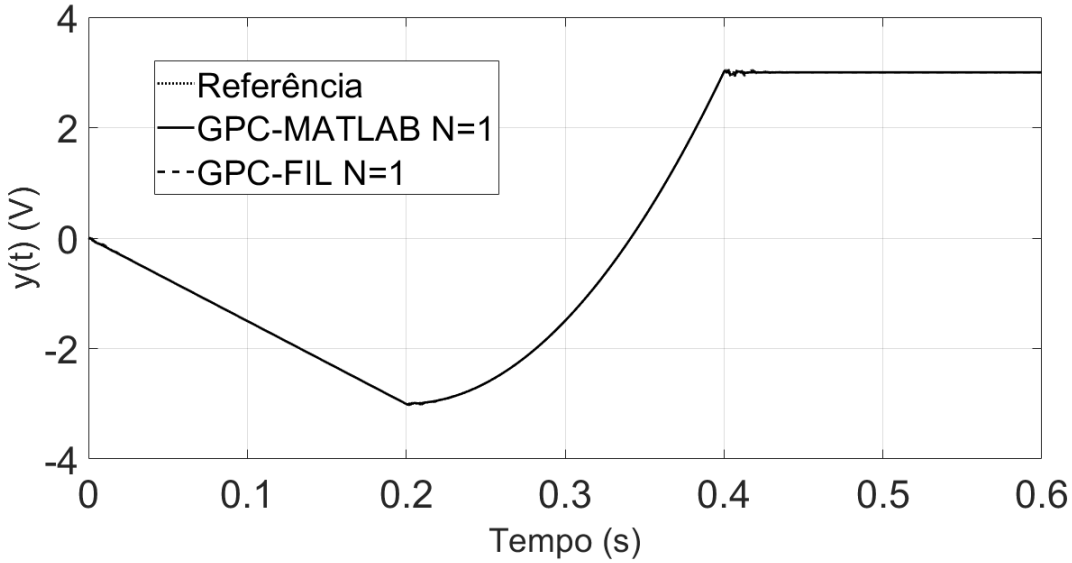
Subsequentemente, realiza-se um teste com 2 funções de Laguerre, sem restrições, com o objetivo de buscar transições mais suaves. O único parâmetro divergente desta simulação com a anterior foi o número de funções de Laguerre mudar para N=2. O fator de ponderação ($a = 1,2$) conduziu a um número de condição ($\kappa(\Omega) = 107,71$) aceitável para o GPC tipo 3 (N=2). Caso o fator fosse escolhido como $a = 1$, então $\kappa(\Omega) = 1,51 \cdot 10^5$, o que deixaria o problema mal condicionado e a resposta em malha fechada comprometida para implementação em *hardware*. Nas Figuras 31, 32, 33, apresenta-se o erro ($e(t)$), variável manipulada ($u_m(t)$) e a saída controlada ($y(t)$) do GPC tipo 3 FIL e MATLAB® (N=2) sem restrições. Observe que o maior pico de erro reduziu de 0,124 V na Figura 28 para 0,042 V na Figura 31 quando se comparado o GPC-FIL (N=2) com o GPC-FIL (N=1). Isto fica mais evidente quando se é calculado o EQM de e(t). O EQM foi de $4,74 \cdot 10^{-7}$ para o GPC-MATLAB® (N=2) enquanto $4,4 \cdot 10^{-6}$ para o GPC-FIL (N=2). Simulações com mais funções de Laguerre tendem a diminuir mais o erro de rastreamento, entretanto

Figura 29 – GPC tipo 3 sem restrições (N=1): $u_m(t)$.



Fonte: o autor

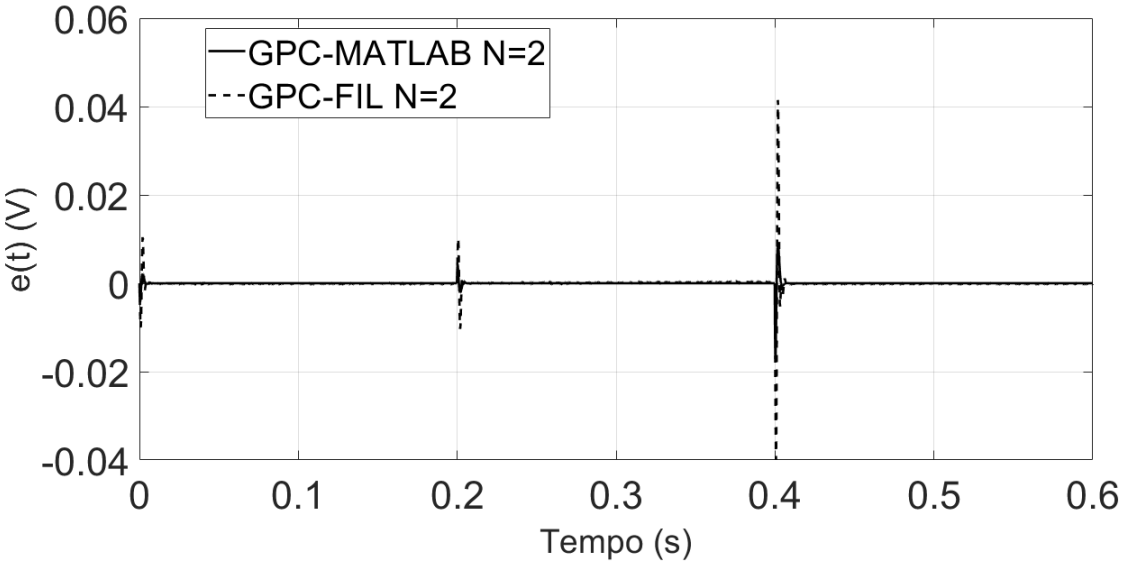
Figura 30 – GPC tipo 3 sem restrições (N=1): $y(t)$.



Fonte: o autor

elas aumentam o tamanho do sistema, causando um aumento de custo computacional. Dessa forma, a utilização de duas funções de Laguerre foi estabelecida neste trabalho como um bom equilíbrio entre performance e custo computacional.

Figura 31 – GPC tipo 3 sem restrições (N=2): $e(t)$.

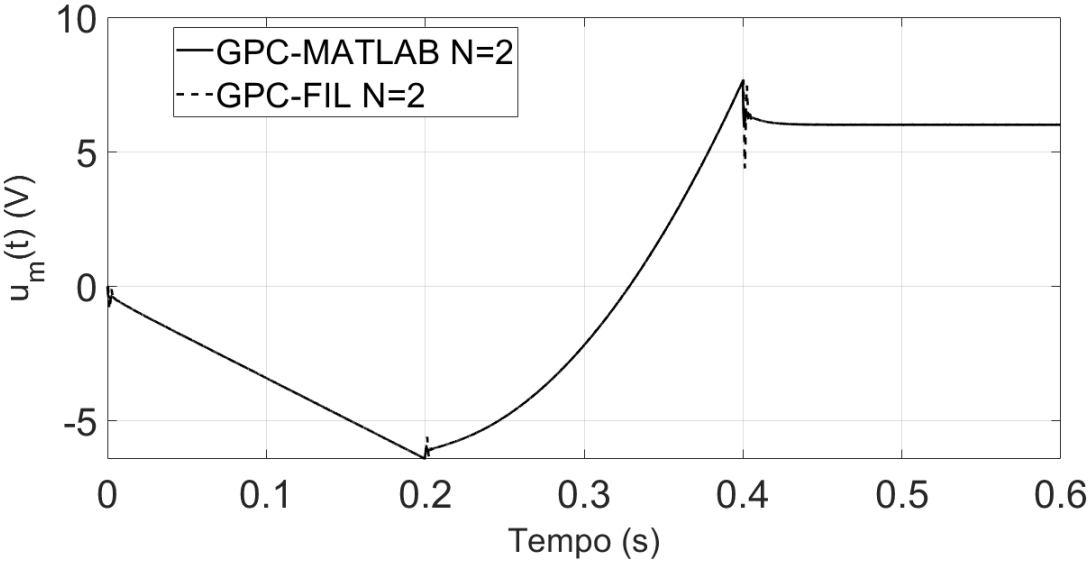


Fonte: o autor

A próxima simulação apresenta a aplicação de restrições na variável manipulada e sua taxa ($u_m(t)$ e $\nabla u_m(t)$) conforme o algoritmo do Apêndice B.3. Para implementações práticas, o conversor AD é um elemento importante para o projeto e, por conseguinte, a quantidade de bits dos sinais de entrada e saída interfere diretamente no custo do *hardware*. Com o truncamento de bits, um erro em regime estacionário surge e isso pode ser controlado a partir de restrições no atuador. Assim, supondo que o circuito de condicionamento tenha uma boa operação com $-6\ V \leq u_m(t) \leq 6\ V$ e com resolução de velocidade aceitável em $-0,04\ V/T_s \leq \nabla u_m(t) \leq 0,04\ V/T_s$, pode-se apresentar o GPC tipo 3 (N=2) com restrições a partir das Figuras 34 ($e(t)$), 35 ($u_m(t)$), 36 ($\nabla u_m(t)$) e 37 ($y(t)$).

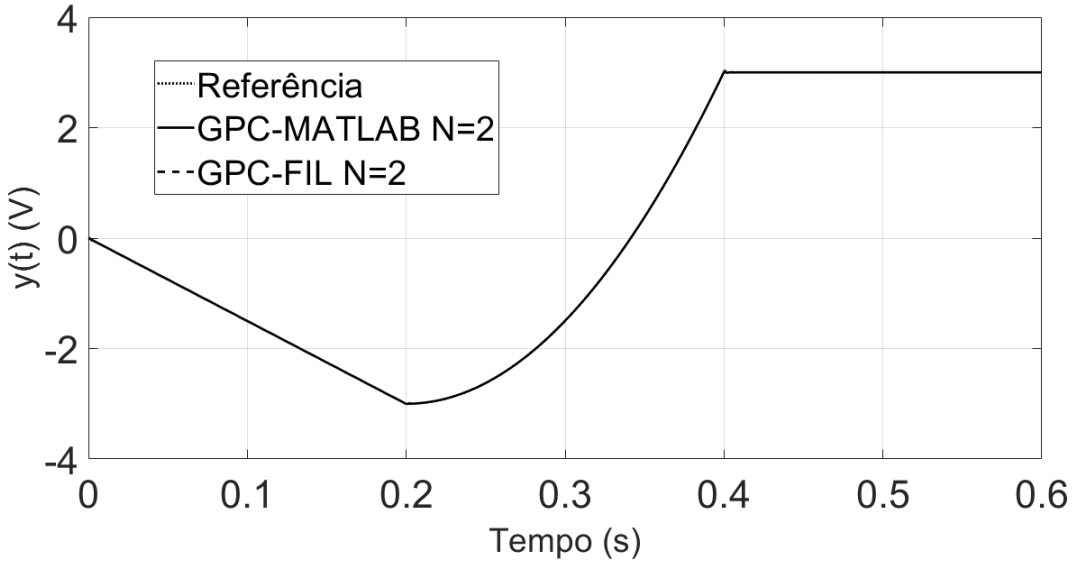
Observe que a introdução de restrições fez com que os EQM's de e(t) das simulações GPC-MATLAB® e GPC-FIL (N=2) ficassem muito próximos. O EQM do GPC-MATLAB® (N=2) foi de 0,0023 enquanto o EQM do GPC-FIL (N=2) foi de 0,0024. Isso corrobora o fato de que a introdução de restrições compensa quando o projetista não possui muita resolução em bits para a implementação do *hardware*. Por fim, vale comentar que ambas as restrições foram ativadas em momentos diferentes e, no geral, o sistema rastreou a referência com atraso nas mudanças de comportamento, pois a restrição de $u_m(t)$ foi ativada.

Figura 32 – GPC tipo 3 sem restrições (N=2): $u_m(t)$.



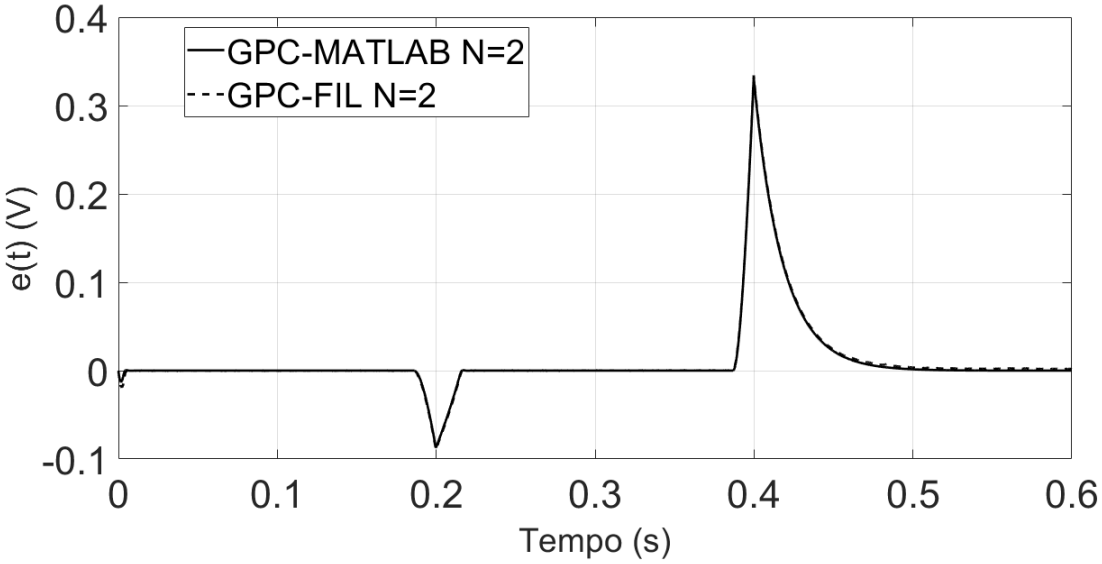
Fonte: o autor

Figura 33 – GPC tipo 3 sem restrições (N=2): $y(t)$.



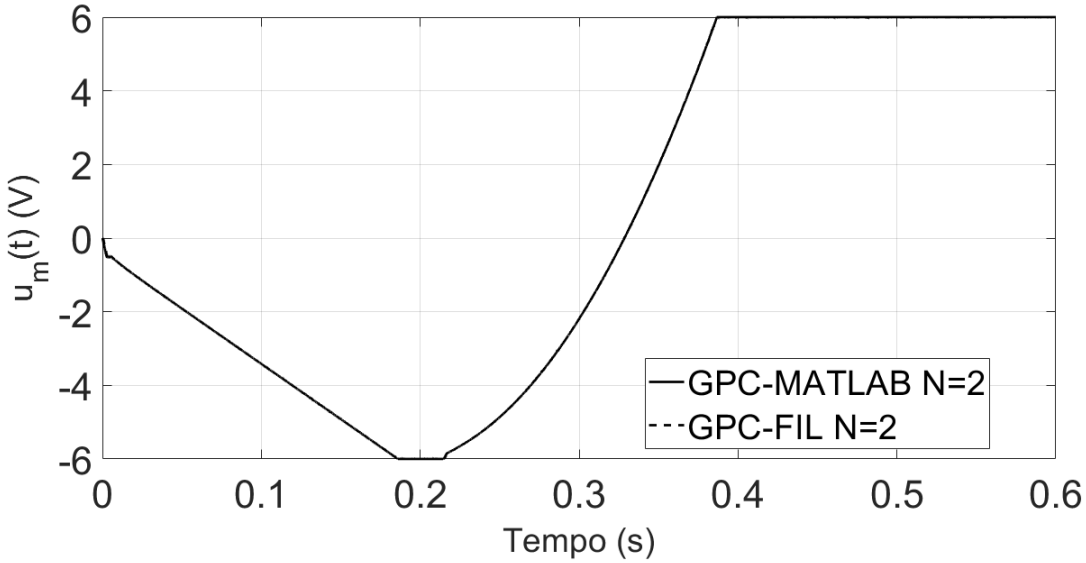
Fonte: o autor

Figura 34 – GPC tipo 3 com restrições (N=2): $e(t)$.



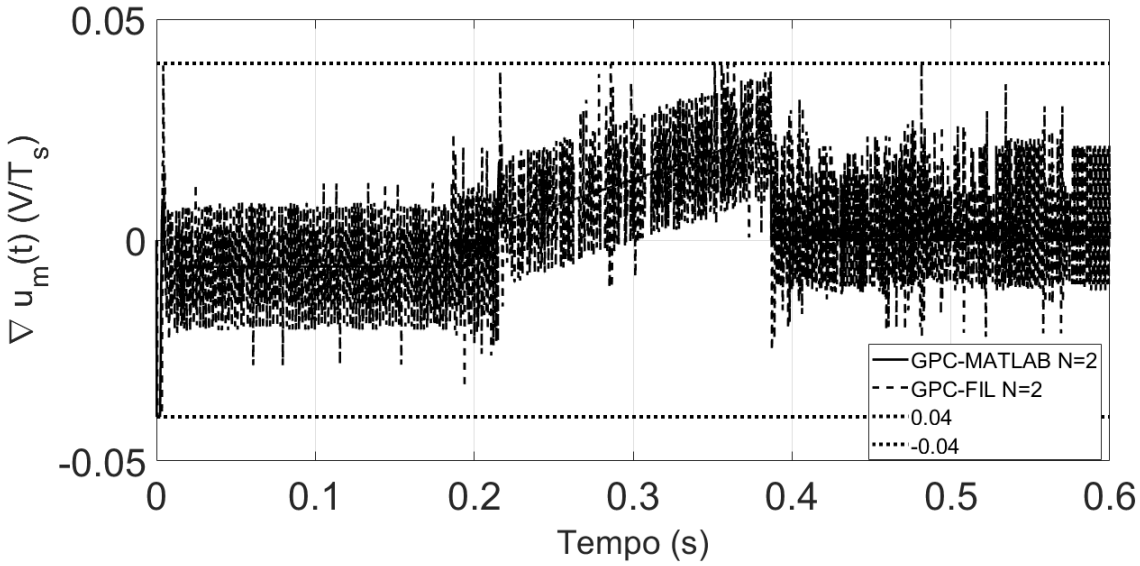
Fonte: o autor

Figura 35 – GPC tipo 3 com restrições (N=2): $u_m(t)$.



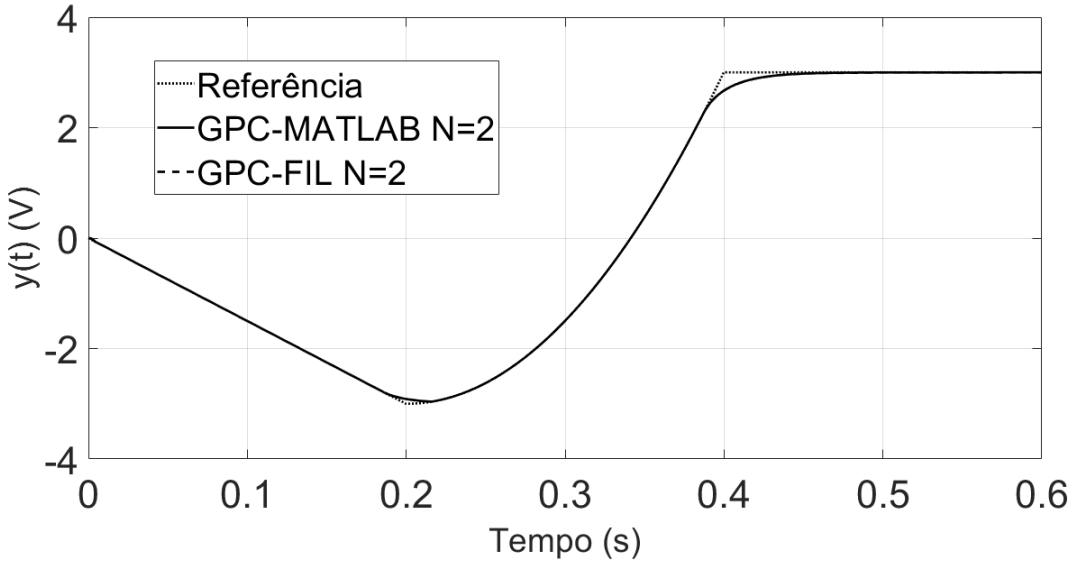
Fonte: o autor

Figura 36 – GPC tipo 3 com restrições (N=2): $\nabla u_m(t)$.



Fonte: o autor

Figura 37 – GPC tipo 3 com restrições (N=2): $y(t)$.



Fonte: o autor

5 CONCLUSÃO

Esta dissertação centrou-se em projetar controladores PIDOF e MPC desde a fase de simulação matemática, passando pela implementação dos códigos em VHDL do FIL com a biblioteca matricial até a validação experimental em bancada para um conversor Boost. O objetivo dos controladores sintonizados nas seções 4.1.1 e 4.1.2 foi controlar o ciclo de trabalho do Mosfet que realiza o chaveamento de funcionamento do conversor Boost com restrições. Para fins de comparação, na Tabela 10 se expõe o tempo de subida e o *overshoot* do regime transitório, *ripple* em tensão do regime estacionário, tempo de assentamento (TA) e o desvio de tensão após o distúrbio de carga no conversor em relação à 40 V, número de bits das variáveis no código em VHDL e a frequência de *clock* mínima ($f_{clk_{min}}$) para a implementação dos controladores PIDOF/MPC-Boost.

Tabela 10 – Comparativo PIDOF/MPC-Boost.

Indicador/Parâmetro	**PIDOF**	**MPC**	**Vencedor**
Tempo de subida	124 ms	122 ms	MPC
overshoot	10%	2,56%	MPC
ripple	300 mV	200 mV	MPC
TA	76 ms	42 ms	MPC
Desvio de tensão	1%	5,13%	PIDOF
Número de bits	57	50	MPC
$f_{clk_{min}}$	85 KHz	150 KHz	PIDOF

Na Figura 38, mostra-se o resumo dos dados de implementação de cada controlador ao compilar com a plataforma QUARTUS®. Através da Tabela 10, demonstra-se que o MPC foi superior ao PIDOF, porém o PIDOF utilizou menor capacidade computacional para a compilação do código em VHDL. Por fim, os controladores apresentaram respostas muito semelhantes e que o PIDOF, mesmo não apresentando uma sintonia ótima [1], consegue competir com o MPC em relação à resposta dinâmica.

Para o GPC tipo 3, conclui-se que a implementação de restrições é essencial quando o projetista não possui um conversor ADC com muitos bits de resolução para os sinais de entrada/saída. No mais, um compromisso para a escolha do número de funções de Laguerre entre a resposta dinâmica do sistema e o tamanho em memória é necessária ao implementar o GPC tipo 3 em FPGA. Por fim, pequenos EQM's foram encontrados nas

[1] Os algoritmos meta-heurísticos, mesmo que simulados várias vezes, podem encontrar, no máximo, a solução ótima para o intervalo de busca imposto. O MPC, por se tratar de um problema com otimização de função convexa, possui sempre um mínimo global.

Figura 38 – Resumo do fluxo de dados para a implementação em FPGA dos controladores PIDOF/MPC-Boost.

MPC-Boost

Flow Summary	
<<Filter>>	
Flow Status	Successful - Thu Oct 13 21:00:32 2022
Quartus Prime Version	21.1.0 Build 842 10/21/2021 SJ Lite Edition
Revision Name	boost_mpc_bancada
Top-level Entity Name	boost_mpc_bancada
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	14,052 / 114,480 (12 %)
Total registers	839
Total pins	29 / 529 (5 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	372 / 532 (70 %)
Total PLLs	0 / 4 (0 %)

PIDOF-Boost

Flow Summary	
<<Filter>>	
Flow Status	Successful - Thu Oct 13 21:22:39 2022
Quartus Prime Version	21.1.0 Build 842 10/21/2021 SJ Lite Edition
Revision Name	boost_fopid_bancada
Top-level Entity Name	boost_fopid_bancada
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	9,054 / 114,480 (8 %)
Total registers	1142
Total pins	29 / 529 (5 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	327 / 532 (61 %)
Total PLLs	0 / 4 (0 %)

Fonte: o autor

simulações, mostrando que o GPC tipo 3 possui erro assintótico nulo com a aplicação ou não de restrições na variável de controle, conforme visto nas Figuras 28, 31 e 34.

5.1 PRINCIPAIS CONTRIBUIÇÕES

As principais contribuições para o desenvolvimento desta dissertação são destacadas a seguir:

- Uma nova topologia de controlador PIDOF que supera o problema de undershoot em regime transitório para plantas de fase não-mínima. Garantindo uma derivada inicial nula conforme em (PEREIRA et al., 2022a), implica-se um undershoot inicial nulo. Logo, bons resultados foram obtidos para a regulagem em tensão do conversor Boost operando em modo tensão através do controlador apresentado na Figura 10;
- Através dos estudos desenvolvidos em (PEREIRA et al., 2022b; PEREIRA et al., 2021) ao resolver as equações de especificação de frequência de cruzamento de ganho (2.43a), de margem de fase (2.43b) e de amortecimento iso (2.43c), equações algébricas para os ganhos do PIDOF e PIOF foram desenvolvidas em função dos

graus fracionários μ, λ ((2.46), (2.47), (2.55)). Observe que essas equações algébricas podem ser usadas por qualquer sistema linear invariante no tempo (LTI) de entrada única e saída única (SISO). Por fim, o artigo (PEREIRA et al., 2021) explora (2.43a), (2.43b), (2.43d) e (2.43e) com a utilização de um algoritmo genético e filtro Oustaloup para a obtenção da solução quase ótima de λ;

- Um algoritmo DE, com o filtro Oustaloup refinado, foi utilizado para encontrar soluções quase ótimas para as ordens fracionárias integro-diferenciais, ou seja, λ e μ, para controladores PIDOF;

- As ordens das funções de transferência de PIOF e DOF ((4.2)-(4.3)) foram reduzidas através da norma L2 para otimizar o tamanho de memória necessária no FPGA. O código em VHDL do PIDOF de tamanho reduzido foi implementado usando bibliotecas matriciais e incorporado em FPGA. Finalmente, o controlador PIDOF foi testado com a ferramenta FIL do MATLAB®/Simulink e validado em bancada para o conversor Boost de acordo com a Seção 4.1.1. Nenhum dos trabalhos relacionados na literatura exploraram tal abordagem para a codificação e teste de um PIDOF-Boost;

- Através da teoria disposta na Seção 3.1 e (WANG, 2009), projetou-se um MPC-MIMO digital baseado em funções de Laguerre e grau de estabilidade prescrito com restrições nas variáveis de controle de um *Sodium Fast Reactor* (SFR). Tal método foi explorado para reduzir o número de condição da matriz Hessiana do MPC-MIMO enquanto são controladas a potência fracionária, temperaturas do tanque e núcleo do reator em sua potência de operação em 100%. Por fim, elaborou-se um código em VHDL do MIMO MPC baseado em funções de Laguerre com biblioteca matricial e o embarcou em FPGA para a validação com FIL conforme (PEREIRA et al., 2022c). Nenhum dos trabalhos anteriores explorou tal abordagem para codificar e testar um MPC em SFR. Adicionamente, o MPC-Boost foi projetado com um refinamento do código em (PEREIRA et al., 2022c) (Apêndice B.2) e os resultados foram validados em 4.1.2.

- Desenvolveu-se um GPC com funções de Laguerre ortonormais capaz de rastrear referências polinomiais de grau m-1, visando rastrear as temperaturas do núcleo e do tanque de sódio com 100% de produção de energia de um SFR, mesmo com inserção de distúrbios na reatividade e na temperatura de sódio secundária durante uma mudança de temperatura linear do núcleo e do tanque de sódio. Para projetar este controlador, escreveu-se (PEREIRA et al., 2022d). Este artigo generalizou a teoria de controle apresentada em (CORDERO et al., 2021) através de sistemas MIMO com a redução do número de condição de sistemas com enormes matrizes de espaço de estado e a redução de custo computacional com a utilização das funções

de Laguerre nas variáveis de controle. Por fim, implementou-se o código B.3 do GPC tipo 3 e os resultados foram validados na Seção 4.2.

5.2 PUBLICAÇÕES

Nesta seção são apresentados os trabalhos publicados ao longo do curso de mestrado, em ordem cronológica, que tiveram relação direta e indireta com o conteúdo apresentado nesta dissertação.

5.2.1 TRABALHOS RELACIONADOS DIRETAMENTE COM A PESQUISA

1. **L. F. S. C. Pereira**, A. A. de Carvalho, J. L. Perez, E. A. Batista, R. B. Godoy and M. A. G. de Brito, "Design of FOPI Controller for The Frequency Fixed Orthogonal Signal Generator (FFOSG) Based PLL through Genetic Algorithm," 2021 14th IEEE International Conference on Industry Applications (INDUSCON), 2021, pp. 1385-1391, doi: 10.1109/INDUSCON51756.2021.9529408.

2. **Pereira, Luís F.d.S.C.**, Edson Batista, Moacyr A.G. de Brito, and Ruben B. Godoy. 2022. "A Robustness Analysis of a Fuzzy Fractional Order PID Controller Based on Genetic Algorithm for a DC-DC Boost Converter", Electronics 11, no. 12: 1894. https://doi.org/10.3390/electronics11121894. (**QUALIS A2**)

3. **L. F. S. C. Pereira**, E. A. Batista, J. O. P. Pinto, B. R. Upadhyaya, J. W. Hines and J. B. Coble, Model Predictive Control for Sodium Fast Reactors Based on Laguerre Functions and FPGA-*in-the-loop* Environment, Nuclear Engineering and Design, vol. 400, 2022. doi: 10.1016/j.nucengdes.2022.112041. (**QUALIS A2**)

4. **L. F. S. C. Pereira**, E. A. Batista, J. O. P. Pinto, B. R. Upadhyaya and J. W. Hines, "A New Proposal Generalized Predictive Control Algorithm with Polynomial Reference Tracking Applied for Sodium Fast Reactors," in IEEE Access, vol. X, pp. X-X, 2022, (submetido). (**QUALIS A1**)

5. **Pereira, L.F.d.S.C.**;Volpato, A.S.; Batista, E.A.; Garcia, R.C.; Godoy, R.B.; de Brito, M.A.G. Experimentally Validated FPGA-based Fractional Order PID Controllers tuned by Differential Evolution Algorithm. Sensors 2022, 1, 0. (submetido). (**QUALIS A1**)

5.2.2 TRABALHOS RELACIONADOS INDIRETAMENTE COM A PESQUISA

1. M. A. G. de Brito, A. A. de Carvalho, R. B. Godoy, A. S. Volpato, **L. F. S. C. Pereira** and E. A. Batista, "A New Positive-Sequence Detector Phase-Locked Loop

Algorithm for DC Offset Rejection," 2021 Brazilian Power Electronics Conference (COBEP), 2021, pp. 1-5, doi: 10.1109/COBEP53665.2021.9684140.

5.3 PROPOSTA DE TRABALHOS FUTUROS

Todos os códigos em VHDL foram sintetizados a partir de matrizes ou parâmetros projetados por algoritmos em compiladores de alto nível (MATLAB®). Dessa forma, os parâmetros de funcionamento dos controladores PIDOF e MPC já foram predefinidos (*offline*) e só assim o algoritmo em VHDL é implementado. Isso demonstra uma estaticidade para o projeto, não o deixando apto para possíveis aplicações adaptativas. Para o PIDOF, pode-se montar uma sequência de rotinas para a criação de um PIDOF adaptativo:

- Uma rotina em VHDL para o cálculo da derivada/integral fracionária por Grünwald-Letnikov (GL) conforme (MONIR et al., 2022). Tal método possui uma atratividade muito grande para sistemas embarcados por conta da forma digital. Para um trabalho futuro, pode-se desenvolver um algoritmo recursivo ótimo truncado para a operação;
- Um algoritmo iterativo de DE para a sintonia do controlador através da minimização de uma função custo;
- Por fim, gerar a função de transferência final do PIDOF para a operação.

O mesmo pode ser proposto para o MPC e GPC. Definindo os termos de configuração de entrada como o número de funções de Laguerre (N), constante multiplicativa da matriz R_L (r_w), polo da função de Laguerre, número de predição (N_p), limite superior do autovalor (λ_{max}) e fator de ponderação exponencial (a); é possível criar um código em VHDL que, com apenas a introdução da planta digitalizada, encontre o MPC desejado. Outra proposta é pesquisar mais sobre a correlação entre os parâmetros de configuração do MPC para desenvolver algoritmos mais precisos e robustos. Por fim, outra proposta de trabalho é aprimorar a biblioteca matricial utilizada nesta dissertação, objetivando a introdução de matrizes dinâmicas e correções de problemas quanto às operações algébricas mais complexas.

REFERÊNCIAS

AMIRAHMADI, A. et al. Optimum design of integer and fractional-order pid controllers for boost converter using spea look-up tables. *J. Power Electron.*, v. 15, n. 1, p. 160–176, 2015. Https://doi.org/10.6113/JPE.2015.15.1.160. Citado 2 vezes nas páginas 22 e 24.

APOSTOL, T. M. *Calculus, Vol. 2: Multi-Variable Calculus and Linear Algebra with Applications to Differential Equations and Probability*. New York: J. Wiley, 1967. Citado na página 115.

BALASKA, H. et al. Adaptive cruise control system for an electric vehicle using a fractional order model reference adaptive strategy. *IFAC-PapersOnLine*, v. 52, n. 13, p. 194–199, 2019. Https://doi.org/10.1016/j.ifacol.2019.11.096. Citado na página 18.

BEMPORAD, A.; MORARI, M. *Robust model predictive control: a survey. In: Garulli A, Tesi A (eds). Robustness in identification and control*. London: Springer London, 1999. 207–226 p. ISBN https://doi.org/10.1007/BFb0109870. Citado na página 19.

BENGTSSON, G. Output regulation and internal models - a frequency domain approach. *Automatica*, v. 13, n. 4, p. 333–345, 1977. Https://doi.org/10.1016/0005-1098(77)90016-4. Citado na página 25.

BOYD, S.; VANDENBERGHE, L. Introduction to applied linear algebra- vectors, matrices, and least squares. *University Press*, Cambrigde, UK, 2018. Citado na página 63.

BOYD, S. P. et al. *Linear Matrix Inequalities in System and Control Theory*. Philadelphia: SIAM, 1994. Citado na página 51.

CAMPO, P.; MORARI, M. ∞-norm formulation of model predictive control problems. *1986 American Control Conference*, p. 339–343, 1986. Doi: 10.23919/ACC.1986.4788961. Citado na página 19.

CAMPO, P.; MORARI, M. Robust model predictive control. *1987 American Control Conference*, p. 1021–1026, 1987. Doi: 10.23919/ACC.1987.4789462. Citado na página 19.

CHEN, Y. et al. Model predictive tracking control of nonholonomic mobile robots with coupled input constraints and unknown dynamics. *in IEEE Transactions on Industrial Informatics*, v. 15, n. 6, p. 3196–3205, 2019. Doi: 10.1109/TII.2018.2874182. Citado 2 vezes nas páginas 25 e 26.

CHEN, Y.; PETRAS, I.; XUE, D. Fractional order control - a tutorial. *in Proc. Amer. Control Conf.*, p. 1397–1411, Jun. 2009. Doi: 10.1109/ACC.2009.5160719. Citado na página 17.

CLARKE, D.; MOHTADI, C.; TUFFS, P. Generalized predictive control—part i. the basic algorithm. *Automatica*, v. 23, n. 2, p. 137–148, 1987. Https://doi.org/10.1016/0005-1098(87)90087-2. Citado na página 19.

CLARKE, D.; MOHTADI, C.; TUFFS, P. Generalized predictive control—part ii extensions and interpretations. *Automatica*, v. 23, n. 2, p. 149–160, 1987. Https://doi.org/10.1016/0005-1098(87)90088-4. Citado na página 19.

CORDERO, R. et al. Ramp-tracking generalized predictive control system-based on second order difference. *IEEE Trans. Circuits Syst. II, Exp. Briefs*, v. 68, n. 4, p. 1283–1287, 2021. 10.1109/TCSII.2020.3019028. Citado 2 vezes nas páginas 25 e 27.

CORDERO, R. et al. Development of a resonant generalized predictive controller for sinusoidal reference tracking. *in IEEE Transactions on Circuits and Systems II: Express Briefs*, v. 69, n. 3, p. 1218–1222, 2022. Doi: 10.1109/TCSII.2021.3102535. Citado 2 vezes nas páginas 26 e 28.

CORDERO, R. et al. Development of a generalized predictive control system for polynomial reference tracking. *in IEEE Transactions on Circuits and Systems II: Express Briefs*, v. 68, n. 8, p. 2875–2879, 2021. Doi: 10.1109/TCSII.2021.3058625. Citado 5 vezes nas páginas 25, 27, 74, 91 e 103.

CUTLER, C.; RAMAKER, B. Dynamic matrix control - a computer control algorithm. *Proceedings of the Joint Automatic Control Conference*, 1980. Citado 2 vezes nas páginas 18 e 19.

DAS, S. *Functional Fractional Calculus*. Berlin, Germany: Springer-Verlag, Jan. 2011. 612 p. ISBN 978-3642205446. Citado na página 17.

DEVARAJ, S. et al. Robust queen bee assisted genetic algorithm (qbga) optimized fractional order pid (fopid) controller for not necessarily minimum phase power converters. *IEEE Access*, v. 9, p. 93331–93337, 2021. 10.1109/ACCESS.2021.3092215. Citado 2 vezes nas páginas 23 e 24.

ERICKSON, R. *DC–DC Power Converters*. Stuttgart: In Wiley Encyclopedia of Electrical and Electronics Engineering, J.G. Webster 412 (Ed.), 2007. Doi.org/10.1002/047134608X.W5808.pub2. Citado na página 75.

FLORES, C. et al. Iso-damping fractional-order control for robust automated car-following. *Journal of Advanced Research*, v. 25, 2020. Doi.org/10.1016/j.jare.2020.05.013. Citado 2 vezes nas páginas 17 e 44.

GANDER, W. *Lineare Algebra: Endliche Arithmetik*. Zurich: ETH Zurich, 2005. Citado na página 68.

GARCIA, C.; MORARI, M. Internal model control. 1. a unifying review and some new results. *Industrial & Engineering Chemistry Process Design and Development*, v. 21, n. 2, p. 308–323, 1982. Https://doi.org/10.1021/i200017a016. Citado na página 19.

GARCIA, C.; PRETT, D.; MORARI, M. Model predictive control: theory and practice - a survey. *Automatica*, v. 25, n. 3, p. 335–348, 1989. Https://doi.org/10.1016/0005-1098(89)90002-2. Citado na página 19.

GARCÍA, R. C. et al. Improved acquisition system for sensored vector control through frequency-domain multiplexing, synchronous sampling, and differential evolution. *IEEE Sensors Journal*, v. 21, n. 6, p. 7784–7792, 2021. Doi:10.1109/JSEN.2020.3045372. Citado na página 49.

GEL'FAND, I. M.; SHILOV, G. E. *Generalized Functions*. vol 1: Academic Press, London, 1964. Citado na página 34.

HAYKIN, S. *Communication Systems*. 4th Edition: John Wiley & Sons, Inc, 2001. ISBN 0471178691. Citado na página 35.

JAMSHIDI, M.; TAROKH, M.; SHAFAI, B. *Computer-Aided Analysis and Design of Linear Control Systems*. New Jersey: Prentice–Hall, 1992. Citado na página 121.

JARAD, F.; ABDELJAWAD, T. Generalized fractional derivatives and laplace transform. *Discrete and Continuous Dynamical Systems - S*, US, v. 13, n. 3, 2020. Doi:10.3934/dcdss.2020039. Citado 2 vezes nas páginas 36 e 37.

KILBAS, A.; SRIVASTAVA, H.; TRUJILLO, J. *Theory and Applications of Fractional Differential Equations*. Amsterdam: North-Holland Mathematics Studies, 2006. 541 p. Citado na página 33.

LAUGHLIN, D.; MORARI, M. Smith predictor design for robust performance. *1987 American Control Conference*, p. 637–642, 1987. Https://doi.org/10.1080/00207178708933912. Citado na página 19.

LEE, J.; MORARI, M.; GARCIA, C. State-space interpretation of model predictive control. *Automatica*, v. 30, n. 4, p. 707–717, 1994. Https://doi.org/10.1016/0005-1098(94)90159-7. Citado na página 19.

LIEGMANN, E.; KARAMANAKOS, P.; KENNEL, R. Real-time implementation of long-horizon direct model predictive control on an embedded system. *in IEEE Open Journal of Industry Applications*, v. 3, p. 1–12, 2022. Doi: 10.1109/OJIA.2021.3133477. Citado 2 vezes nas páginas 25 e 27.

LUCIA, S. et al. Optimized fpga implementation of model predictive control for embedded systems using high-level synthesis tool. *in IEEE Transactions on Industrial Informatics*, v. 14, n. 1, p. 137–145, 2018. Doi: 10.1109/TII.2017.2719940. Citado 2 vezes nas páginas 25 e 27.

MEHRA, V.; SRIVASTAVA, S.; VARSHNEY, P. Fractional-order pid controller design for speed control of dc motor. *in Proc. 3rd Int. Conf. Emerg. Trends Eng. Technol.*, p. 422–425, Nov. 2010. Doi: 10.1109/ICETET.2010.123. Citado na página 18.

MERRIKH-BAYAT, F.; JAMSHIDI, A. Comparing the performance of optimal pid and optimal fractional-order pid controllers applied to the nonlinear boost converter. *in The Fifth IFAC Symposium on Fractional Differentiation and Its Applications (FDA12)*, p. 1–6, 2013. Https://arxiv.org/abs/1312.7517. Citado 2 vezes nas páginas 21 e 23.

MILLER, K. S.; ROSS, B. *An Introduction to the Fractional Calculus and Fractional Differential Equations*. New York/Singapore: John Wiley & Sons, Inc., 1993. 384 p. ISBN 978-0471588849. Citado 2 vezes nas páginas 17 e 32.

MISHRA, I. et al. Step-by-step development and implementation of fs-mpc for a fpga-based pmsm drive system. *Electronics*, v. 10, n. 395, 2021. Https://doi.org/10.3390/electronics10040395. Citado 2 vezes nas páginas 25 e 27.

MONIR, M. et al. A unified fpga realization for fractional-order integrator and differentiator. *Electronics*, v. 11, n. 2052, 2022. Doi.org/10.3390/ electronics11132052. Citado na página 105.

MONJE, C. et al. *Fractional Order Systems and Control — Fundamentals and Applications, Advances in Industrial Control*. London, U.K.: Springer-Verlag, 2010. 415 p. ISBN doi: 10.1007/978-1-84996-335-0. Citado na página 17.

MONJE, C. A. et al. Tuning and auto-tuning of fractional order controllers for industry applications. *Control Engineering Practice*, v. 16, n. 7, 2008. Doi.org/10.1016/j.conengprac.2007.08.006. Citado 2 vezes nas páginas 17 e 43.

OLIVEIRA, D. S. *Derivada Fracionária e as Funções de Mittag-Leffler*. Dissertação (Mestrado) — Universidade Estadual de Campinas, Campinas, 2014. Citado na página 32.

OPPENHEIM, A. V.; SCHAFER, R. W. *Discrete-time Signal Processing*. 3rd Edition: Pearson Education Inc, 2010. ISBN 9789332505742. Citado na página 34.

ORTIGUEIRA, M. D. *Fractional Calculus for Scientists and Engineers*. London/New York: Springer Dordrecht Heidelberg, 2011. 159 p. Doi:10.1007/978-94-007-0747-4. ISBN 0471178691. Citado na página 36.

OUSTALOUP, A. et al. Frequency-band complex noninteger differentiator: Characterization and synthesis. *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, v. 47, n. 1, p. 25–39, 2000. Doi: 10.1109/81.817385. Citado na página 17.

PECCIN, V. et al. Fast constrained generalized predictive control with admm embedded in an fpga. *in IEEE Latin America Transactions*, v. 18, n. 2, p. 422–429, 2020. Doi: 10.1109/TLA.2020.9085299. Citado 2 vezes nas páginas 25 e 27.

PEREIRA, L. et al. A robustness analysis of a fuzzy fractional order pid controller based on genetic algorithm for a dc-dc boost converter. *Electronics*, v. 11, n. 1894, 2022. Doi.org/10.3390/electronics11121894. Citado 4 vezes nas páginas 18, 75, 79 e 102.

PEREIRA, L. et al. Experimentally validated fpga-based fractional order pid controllers tuned by differential evolution algorithm. *Sensors*, x, n. x, p. xx–xx, 2022. Xx. Citado na página 102.

PEREIRA, L. F. S. C. et al. Model predictive control for sodium fast reactors based on laguerre functions and fpga-in-the-loop environment, nuclear engineering and design. *Nuclear Engineering and Design*, v. 400, p. 1–14, 2022. Https://doi.org/10.1016/j.nucengdes.2022.112041. Citado na página 103.

PEREIRA, L. F. S. C. et al. A new proposal generalized predictive control algorithm with polynomial reference tracking applied for sodium fast reactors. *in IEEE Access*, XX, n. XX, p. x–x, 2022. XX. Citado na página 103.

PEREIRA, L. F. S. C. et al. Design of fopi controller for the frequency fixed orthogonal signal generator (ffosg) based pll through genetic algorithm. *2021 14th IEEE International Conference on Industry Applications (INDUSCON)*, p. 1385–1391, 2021. Doi: 10.1109/INDUSCON51756.2021.9529408. Citado 2 vezes nas páginas 102 e 103.

PORAT, B. *Digital Processing of Random Signals : Theory and Methods*. New Jersey: Prentice–Hall, 1994. Citado na página 120.

PRAJAPATI, S.; GARG, M. M.; PRITHVI, B. Design of fractional-order pi controller for dc–dc power converters. *in Proc. 8th IEEE India Int. Conf. Power Electron. (IICPE)*, p. 1–6, 2018. 10.1109/IICPE.2018.8709430. Citado 2 vezes nas páginas 22 e 24.

PRETT, D.; GILLETTE, R. Optimization and constrained multivariable control of a catalytic cracking unit. *Jt Autom Control Conf*, p. 73–78, 1980. Https://doi.org/10.1109/JACC.1980.4232010. Citado na página 19.

PRICE, G. B. Telescoping sums and the summation of sequences. *The Two-Year College Mathematics J., Spring*, v. 4, 1973. Citado 2 vezes nas páginas 60 e 63.

PULLAGURAM, D. et al. Non-linear fractional order controllers for autonomous microgrid system. *in Proc. IEEE 6th Int. Conf. Power Syst. (ICPS)*, p. 1–6, Mar. 2016. Doi: 10.1109/ICPES.2016.7584247. Citado na página 18.

RAO, C.; RAWLINGS, J.; LEE, J. Constrained linear state estimation - a moving horizon approach. *Automatica*, v. 37, n. 10, p. 1619–1628, 2001. Https://doi.org/10.1016/S0005-1098(01)00115-7. Citado na página 19.

RICHALET, J. et al. Model predictive heuristic control. *Automatica*, v. 14, n. 5, p. 413–428, 1978. Https://doi.org/10.1016/0005-1098(78)90001-8. Citado 2 vezes nas páginas 18 e 19.

RODRIGUES, F. G. *Sobre Cálculo Fracionário e Soluções da Equação de Bessel*. Tese (Doutorado) — Universidade Estadual de Campinas, Campinas, 2015. Citado 3 vezes nas páginas 31, 35 e 114.

ROSSITER, J. A.; WANG, L.; VALENCIA-PALOMO, G. Efficient algorithms for trading off feasibility and performance in predictive control. *Int. J. Control*, v. 83, n. 4, p. 789–797, 2010. Https://doi.org/10.1080/00207170903437129. Citado 2 vezes nas páginas 25 e 26.

SAEED, J.; WANG, L.; FERNANDO, N. Model predictive control of phase shift full-bridge dc–dc converter using laguerre functions. *IEEE Transactions on Control Systems Technology*, v. 30, p. 819–826, 2022. Doi:10.1109/TCST.2021.3069148. Citado 4 vezes nas páginas 24, 26, 66 e 85.

SAHA, S. et al. Design of a fractional order phase shaper for iso-damped control of a phwr under step-back condition. *IEEE Transactions on Nuclear Science*, v. 57, n. 3, p. 1602–1612, 2010. 10.1109/TNS.2010.2047405. Citado na página 17.

SÁNCHEZ, A. et al. Non-integer order approximation of a pid-type controller for boost converters. *Energies*, v. 14, n. 3153, 2021. Https://doi.org/10.3390/en14113153. Citado 2 vezes nas páginas 22 e 24.

SÁNCHEZ, A. S. et al. Fractional-order approximation of pid controller for buck–boost converters. *Micromachines*, v. 12, n. 591, 2021. Https://doi.org/10.3390/mi12060591. Citado 2 vezes nas páginas 22 e 24.

SCHWARZ, H. *Numerische Mathematik. Lehrbuch Mathematik, 4th edition*. Stuttgart: Teubner Stuttgart, 1997. Citado na página 68.

SEO, S.; CHOI, H. Digital implementation of fractional order pid-type controller for boost dc–dc converter. *IEEE Access*, v. 7, p. 142652–142662, 2019. 10.1109/ACCESS.2019.2945065. Citado 2 vezes nas páginas 23 e 24.

SILVA, T. O. e. Laguerre filters: An introduction. *Revista do DETUA*, v. 1, p. 237–248, 1995. Citado na página 119.

SKAF, J. Solving the lqr problem by block elimination. *Stanford University - EE363 (Notes)*, 2008. Citado na página 123.

SORIANO-SÁNCHEZ, A. et al. Fractional-order approximation and synthesis of a pid controller for a buck converter. *Energies*, v. 13, n. 629, 2020. Https://doi.org/10.3390/en13030629. Citado 2 vezes nas páginas 22 e 24.

STANOVOV, V.; AKHMEDOVA, S.; SEMENKIN, E. Neuroevolution for parameter adaptation in differential evolution. *Algorithms*, v. 15, n. 122, 2022. Doi.org/10.3390/a15040122. Citado na página 47.

STANOVOV, V. et al. Improving the quantum multi-swarm optimization with adaptive differential evolution for dynamic environments. *Algorithms*, v. 15, n. 154, 2022. Doi.org/10.3390/a15050154. Citado na página 47.

STORN, R.; PRICE, K.; LAMPINEN, J. *Differential Evolution – a practical approach to global optimization*. Berlin: Springer, 2005. Citado na página 49.

SZEGÖ, G. *Orthogonal Polynomials, vol. XXIII*. US: Americal Mathematical Society Colloquium Publications, fourth edition, 1975. Citado na página 114.

TEODORO, G. S.; OLIVEIRA, D. S.; OLIVEIRA, E. C. On fractional derivatives. *Sociedade Brasileira de Física*, Brazil, v. 20, n. 2, 2017. Doi.org/10.1590/1806-9126-rbef-2017-0213. Citado na página 36.

VANITHA, D.; RATHINAKUMAR, M. Fractional order pid controlled pv buck-boost converter with coupled inductor. *Int. J. Power Electron. Drive Syst. (IJPEDS)*, v. 8, n. 3, p. 1401–1407, 2017. DOI: 10.11591/ijpeds.v8i3.pp1401-1407. Citado 2 vezes nas páginas 21 e 23.

WANG, L. Discrete time model predictive control design using laguerre functions. *Proceedings of the 2001 American Control Conference*, p. 2430–2435, 2001. Doi:10.1109/ACC.2001.946117. Citado na página 20.

WANG, L. Discrete model predictive controller design using laguerre functions. *Journal of Process Control*, v. 14, n. 2, p. 131–142, 2004. Https://doi.org/10.1016/S0959-1524(03)00028-3. Citado na página 20.

WANG, L. *Model Predictive Control System Design and Implementation Using MATLAB (1st. ed.)*. London: Springer Publishing Company, 2009. Citado 14 vezes nas páginas 19, 24, 53, 55, 56, 57, 65, 67, 69, 72, 73, 85, 103 e 119.

WARRIER, P.; SHAH, P. Optimal fractional pid controller for buck converter using cohort intelligent algorithm. *Appl. Syst. Innov.*, v. 4, n. 50, 2021. Https://doi.org/10.3390/asi4030050. Citado 2 vezes nas páginas 23 e 24.

XUE, D.; ATHERTON, D. P. A suboptimal reduction algorithm for linear systems with a time delay. *International Journal of Control*, v. 60, 1994. Doi: 10.1080/00207179408921460. Citado na página 51.

XUE, D.; CHEN, Y.; ATHERTON, D. P. *Linear Feedback Control: Analysis and Design with MATLAB (Advances in Design and Control). (1st. ed.)*. USA: Society for Industrial and Applied Mathematics, 2008. Citado 3 vezes nas páginas 37, 40 e 51.

XUE, D.; ZHAO, C.; CHEN, Y. A modified approximation method of fractional order system. *International Conference on Mechatronics and Automation*, China, 2006. Doi: 10.1109/ICMA.2006.257769. Citado na página 38.

YAMASHITA, A.; ZANIN, A.; ODLOAK, D. Tuning of model predictive control with multi-objective optimization. *Brazilian Journal of Chemical Engineering [online]*, v. 33, n. 2, p. 333–346, 2016. Doi.org/10.1590/0104-6632.20160332s20140212. Citado 2 vezes nas páginas 25 e 27.

YICHEN, Z.; HEJIN, X.; DEMING, L. Feedback control of fractional $pi^{\lambda}d^{\mu}$ for dc/dc buck converters. *in Proc. Int. Conf. Ind. Inform.- Comput. Technol., Intell. Technol., Ind. Inf. Integr. (ICIICII)*, p. 219–222, Dec. 2017. Citado na página 18.

YOUNG, N. *An Introduction to Hilbert Space*. Cambridge, UK: Cambridge University Press, 1988. Citado na página 120.

ZHENG, A.; MORARI, M. Stability of model predictive control with mixed constraints. *IEEE Trans Autom Control*, v. 40, n. 10, p. 1818–1823, 1995. Https://doi.org/10.1109/9.467664. Citado na página 19.

ZHENG, X. et al. Self-adaptive multi-task differential evolution optimization: With case studies in weapon–target assignment problem. *Electronics*, v. 10, n. 2945, 2021. Doi.org/10.3390/electronics10232945. Citado na página 47.

ZHU, D.; LIU, L.; LIU, C. Optimal fractional-order pid control of chaos in the fractional-order buck converter. *in Proc. 9th IEEE Conf. Ind. Electron. Appl.*, p. 787–791, Jun. 2014. Doi: 10.1109/ICIEA.2014.6931269. Citado na página 18.

ÅSTRÖM, K. J. *Introduction to Stochastic Control Theory*. London: Academic Press, 1970. Citado na página 51.

Apêndices

APÊNDICE A – DEMONSTRAÇÕES MATEMÁTICAS

A.1 FUNÇÕES GAMA E BETA

As funções gama e beta são importantes para o desenvolvimento dos operadores fracionários integrais de RL e diferencial de Caputo, dessa forma, a função gama é definida em termos da integral de Euler de segunda espécie (SZEGÖ, 1975):

Definição A.1.1. A função gama converge absolutamente para $z \in \mathbb{C}$, $Re(z) > 0$ e é definida por:

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx. \tag{A.1}$$

Com a definição A.1.1 pode-se provar o seguinte lema:

Lema A.1.1 (Fórmula da Redução)**.** $\Gamma(z+1) = z\Gamma(z)$.

Demonstração. Aplicando a integral por partes em (A.1):

$$\begin{aligned} \Gamma(z+1) &= \int_0^\infty x^z e^{-x} dx \\ \Gamma(z+1) &= \cancel{[-e^x x^z]_0^\infty} + z\int_0^\infty x^{z-1} e^{-x} dx = z\Gamma(z). \end{aligned} \tag{A.2}$$

□

Observe que a função gama, quando no domínio dos naturais, reduz-se à função fatorial, ou seja, $\Gamma(n+1) = n!$. É trivial verificar para n = 0, pois $\Gamma(1) = \int_0^\infty e^{-x} dx = [-e^x]_0^\infty = 1$. Por indução em n, tomando válida a expressão (A.2), devemos prová-la tambem para $n+1$.

$$\begin{aligned} \Gamma(n+2) &= \int_0^\infty x^{n+1} e^{-x} dx \\ \Gamma(n+2) &= \cancel{[-e^x x^{n+1}]_0^\infty} + (n+1)\int_0^\infty x^n e^{-x} dx = (n+1)\Gamma(n+1) \\ \Gamma(n+2) &= (n+1)!. \end{aligned} \tag{A.3}$$

Introduz-se, agora, o conceito da função beta, bem como a relação que esta função possui com a função gama (RODRIGUES, 2015).

Definição A.1.2 (Função Beta)**.** A função beta B(p,q) é definida em termos da integral de Euler de primeira ordem.

$$B(p,q) = \int_0^1 t^{p-1}(1-t)^{q-1}dt,\ p,q\ \notin \mathbb{Z}^-. \tag{A.4}$$

Lema A.1.2. *A função beta tem a propriedade da simetria, ou seja:*

$$B(p,q) = B(q,p). \tag{A.5}$$

Demonstração. Introduzindo uma mudança de variável $x = 1-t$ na definição A.1.2 é imediato que

$$B(p,q) = \int_1^0 (1-t)^{p-1}t^{q-1}(-dx) = \int_0^1 t^{q-1}(1-t)^{p-1}dx = B(q,p). \tag{A.6}$$

□

Introduzindo a mudança de variável $t = \cos^2\theta \rightarrow dt = -2\cos\theta\sin\theta d\theta$ na definição A.1.2, resulta-se em:

$$\begin{aligned} B(p,q) &= -2\int_{\pi/2}^0 (1-\cos^2\theta)^{p-1}(\cos^2\theta)^{q-1}\cos\theta\sin\theta d\theta \\ B(p,q) &= 2\int_0^{\pi/2} \sin^{2p-1}\theta\cos^{2q-1}\theta d\theta. \end{aligned} \tag{A.7}$$

Lema A.1.3.

$$B(p,q) = \frac{\Gamma(p)\Gamma(q)}{\Gamma(p+q)}. \tag{A.8}$$

Demonstração. Tomando o produto:

$$\Gamma(p)\Gamma(q) = \int_0^\infty x^{p-1}e^{-x}dx\int_0^\infty y^{q-1}e^{-y}dy. \tag{A.9}$$

Introduzindo as mudanças de variáveis $x = u^2$ e $y = v^2$ em (A.9) e aplicando o teorema de Fubini (APOSTOL, 1967) :

$$\Gamma(p)\Gamma(q) = 4\int_0^\infty\int_0^\infty e^{-(u^2+v^2)}u^{2p-1}v^{2q-1}dudv. \tag{A.10}$$

Realizando a transformação de variáveis $(u,v) \rightarrow (\rho,\theta)$, ou seja, $u = \rho\sin\theta$ e $v = \rho\cos\theta$ em (A.10), sabendo que o jacobiano desta transformação é ρ, tem-se:

$$\begin{aligned}
\Gamma(p)\Gamma(q) &= 4\int_0^{\pi/2}\int_0^{\infty} e^{-\rho^2}(\rho\cos\theta)^{2p-1}(\rho\sin\theta)^{2q-1}\rho d\rho d\theta \\
\Gamma(p)\Gamma(q) &= 4\int_0^{\pi/2}\int_0^{\infty} e^{-\rho^2}\rho^{2p+2q-1}\cos^{2p-1}\theta\sin^{2q-1}\theta d\rho d\theta \\
\Gamma(p)\Gamma(q) &= 2\int_0^{\infty} e^{-\rho^2}\rho^{2p+2q-1}d\rho \underbrace{2\int_0^{\pi/2}\cos^{2p-1}\theta\sin^{2q-1}\theta d\theta}_{B(p,q)} \\
\Gamma(p)\Gamma(q) &= 2B(p,q)\int_0^{\infty} e^{-\rho^2}\rho^{2p+2q-1}d\rho.
\end{aligned} \tag{A.11}$$

Note que, para $r = \rho^2$ em (A.11), resulta-se:

$$\begin{aligned}
\Gamma(p)\Gamma(q) &= B(p,q)\underbrace{\int_0^{\infty} e^{-r}r^{p+q-1}dr}_{\Gamma(p+q)} \\
B(p,q) &= \frac{\Gamma(p)\Gamma(q)}{\Gamma(p+q)}.
\end{aligned} \tag{A.12}$$

□

A.2 OPERADOR GENERALIZADO DISCRETO EM DIFERENÇAS RETRÓGRADAS

Definição A.2.1. Seja $k \in \mathbb{N}$ e u uma função qualquer, o operador em diferenças retrógradas discreto ∇ é definido por:

$$\nabla u(k) = u(k) - u(k-1). \tag{A.13}$$

Definição A.2.2. Seja $k \in \mathbb{N}$ e $u(k) \in \mathbb{R}$, o operador em diferenças retrógradas generalizado discreto ∇_l é definido por:

$$\nabla_l u(k) = u(k) - u(k-l),\ l \in \mathbb{N}. \tag{A.14}$$

Observe que com a definição A.2.2, o operador ∇_l é linear, pois para a e b escalares não nulos:

$$\begin{aligned}
\nabla_l(au(k) + bv(k)) &= (au(k) + bv(k)) - (au(k-l) + bv(k-l)) \\
\nabla_l(au(k) + bv(k)) &= a(u(k) - u(k-l)) + b(v(k) - v(k-l)) \\
\nabla_l(au(k) + bv(k)) &= a\nabla_l u(k) + b\nabla_l v(k).
\end{aligned} \tag{A.15}$$

Com o conhecimento de ser um operador linear, pode-se enunciar alguns lemas que são de extrema necessidade para o desenvolvimento do GPC para rastrear referências polinomiais.

Lema A.2.1. *A relação entre ∇_l e ∇ é dada por:*

$$1 - \nabla_l = (1 - \nabla)^l, \tag{A.16}$$

e

$$1 - \nabla_l = \sum_{i=0}^{l} C_l^i (-1)^i \nabla^i. \tag{A.17}$$

Demonstração. Para $l = 1$ a igualdade é trivial. Dessa forma, por indução em $m \in \mathbb{N}$, se a afirmação é verdadeira para $n = m$, então é preciso provar que é também verdadeira para $n = m + 1$. Multiplicando ambos os lados de (A.16), válida em m, por $(1 - \nabla)$:

$$\begin{aligned}
(1-\nabla)^{m+1}u(k) &= (1-\nabla_m)(1-\nabla)u(k) \\
(1-\nabla)^{m+1}u(k) &= (1-\nabla-\nabla_m+\nabla_m\nabla)u(k) \\
(1-\nabla)^{m+1}u(k) &= \cancel{u(k)} - \cancel{u(k)} + u(k-1) - u(k) + u(k-m) + \nabla_m(u(k)-u(k-1)) \\
(1-\nabla)^{m+1}u(k) &= -\cancel{u(k)} + \cancel{u(k-1)} + \cancel{u(k-m)} + \cancel{u(k)} - \cancel{u(k-m)} - \cancel{u(k-1)} + u(k-1-m) \\
(1-\nabla)^{m+1}u(k) &= u(k) - (u(k) - u(k-1-m)) = (1-\nabla_{m+1})u(k).
\end{aligned} \tag{A.18}$$

Provado (A.16), aplica-se a fórmula do binômio de Newton em $(1 - \nabla)^l$ e a prova de (A.17) fica imediata. □

Com o conhecimento da definição A.2.2, pode-se extrapolar o operador para ordem superiores, resultando assim no operador generalizado de n-ésima ordem conforme definição abaixo:

Definição A.2.3. O operador em diferenças retrógradas generalizado discreto de n-ésima ordem ∇_l^n é definido por:

$$\nabla_l^n = \nabla_l(\nabla_l^{n-1}),\ n \in \mathbb{N}. \tag{A.19}$$

Vale a pena notar que o operador ∇_l^n também é linear, ou seja, $\nabla_l^n(au(k)+bv(k)) = a\nabla_l^n u(k) + b\nabla_l^n v(k)$, para a, b não nulos. Tal afirmação é facilmente provada por indução, pois ela é válida para $n = 1$ por (A.15). Assim, tomando verdadeira para $n = m$, resta-se provar que é válida para $n = m + 1$ e, de fato:

$$\nabla_l\nabla_l^m(au(k) + bv(k)) = \nabla_l(a\nabla_l^m u(k) + b\nabla_l^m v(k)) = a\nabla_l^{m+1}u(k) + b\nabla_l^{m+1}v(k). \tag{A.20}$$

O próximo teorema demonstra como calcular o operador em diferenças generalizado de n-ésima ordem para qualquer função discreta real.

Teorema A.2.1. *Se $u(k) \in \mathbb{R}$ e n, $l \in \mathbb{N}$, então:*

$$\nabla_l^n u(k) = \sum_{i=0}^{n} C_n^i (-1)^i u(k - il). \tag{A.21}$$

Demonstração. Novamente por indução em $n \in \mathbb{N}$ é direto que (A.21) vale para n = 1. Dessa forma, tomando verdadeira para $n = m$, resta-se provar que é válida para $n = m+1$.

$$\begin{aligned}
\nabla_l^{n+1} u(k) &= \nabla_l \nabla_l^n u(k) = \nabla_l (\sum_{i=0}^{n} C_n^i (-1)^i u(k - il)) \\
\nabla_l^{n+1} u(k) &= \sum_{i=0}^{n} C_n^i (-1)^i [u(k - il) - u(k - l(i+1))]) \\
\nabla_l^{n+1} u(k) &= \sum_{i=0}^{n} C_n^i (-1)^i u(k - il) - \underbrace{\sum_{i=0}^{n} C_n^i (-1)^i u(k - l(i+1))}_{I}.
\end{aligned} \tag{A.22}$$

Efetuando uma mudança de variável em I, com $j = i + 1$, tem-se:

$$\begin{aligned}
I &= \sum_{j=1}^{n+1} C_n^{j-1} (-1)^{j-1} u(k - lj) = -\sum_{j=1}^{n+1} C_n^{j-1} (-1)^j u(k - lj) \\
I &= -[\sum_{j=0}^{n} C_n^{j-1} (-1)^j u(k - lj) + (-1)^{n+1} u(k - l(n+1))].
\end{aligned} \tag{A.23}$$

Substituindo (A.23) em (A.22), tem-se:

$$\begin{aligned}
\nabla_l^{n+1} u(k) &= \sum_{j=0}^{n} \underbrace{[C_n^{j-1} + C_n^j]}_{C_{n+1}^j} (-1)^j u(k - lj) + (-1)^{n+1} u(k - l(n+1)) \\
\nabla_l^{n+1} u(k) &= \sum_{i=0}^{n+1} C_{n+1}^i (-1)^i u(k - il).
\end{aligned} \tag{A.24}$$

□

Lema A.2.2. *Se l, m e n $\mathbb{N}$, então*

$$\nabla_l^n k^n = n! l^n \text{ e } \nabla_l^m k^n = 0, \text{ se } m > n. \tag{A.25}$$

Demonstração. Observe que para n = 1, $\nabla_l k = k - (k - l) = 1! l^1$. Nota-se também que, para $n = 1$ e $m = 2$, $\nabla_l^2 k = \nabla_l l = 0$, assim, para qualquer $m > n$, $\nabla_l^m k^n = 0$. Por indução em $n \in \mathbb{N}$, como a expressão é verdadeira para $n = 1$, se (A.25) é verdadeira para $n = m$, então também precisa ser válida para $n = m + 1$.

$$\nabla_l^{n+1}k^{n+1} = \nabla_l^n[k^{n+1} - (k-l)^{n+1}] = \nabla_l^n[k^{n+1} - \sum_{i=0}^{n+1} C_{n+1}^i k^{n+1-i}(-1)^i]$$
$$\nabla_l^{n+1}k^{n+1} = \nabla_l^n[C_{n+1}^1 k^n l - C_{n+1}^2 k^{n-1} + \cdots] = \nabla_l^n[C_{n+1}^1 k^n l] - \cancel{\nabla_l^n[C_{n+1}^2 k^{n-1} + \cdots]}$$
$$\nabla_l^{n+1}k^{n+1} = (n+1)l(n!l^n) = (n+1)!l^{n+1}. \tag{A.26}$$

□

Finalmente, para uma função discreta polinomial $u(k) = k^n$, utilizando (A.21) e (A.25), o operador generalizado de ordem m é:

$$\nabla_l^m k^n = \begin{cases} \sum_{i=0}^n C_n^i(-1)^i(k-il)^n & , m < n; \\ n!l^n & , m = n; \\ 0 & , m > n. \end{cases} \tag{A.27}$$

A.3 FUNÇÕES DE LAGUERRE

A transformada Z da função de Laguerre $\phi_n(k)$ no tempo discreto é usualmente definida por (SILVA, 1995):

$$\Phi_n(z) = \frac{\sqrt{1-a^2}}{1-az^{-1}}(\frac{z^{-1}-a}{1-az^{-1}})^{n-1}, \tag{A.28}$$

sendo que a é o polo com $0 \leq a < 1$, para assegurar a estabilidade das autofunções (A.28). Um modo mais direto de se calcular as transformadas Z das n-ésimas funções de Laguerre é verificar que elas satisfazem uma equação de recorrência, ou seja:

$$\Phi_n(z) = \Phi_{n-1}(z)(\frac{z^{-1}-a}{1-az^{-1}}), \tag{A.29}$$

com $\Phi_1(z) = \dfrac{\sqrt{1-a^2}}{1-az^{-1}}$. Seja $l_n(k)$ a transformada inversa de Z de $\Phi_n(z)$, o conjunto discreto das funções de Laguerre pode ser expressado na seguinte forma vetorial:

$$L(k) = [l_1(k), l_2(k), \cdots, l_n(k)]^T. \tag{A.30}$$

A forma no tempo discreta (A.30) também possui um comportamento recursivo dentre seus períodos de cálculo k da seguinte forma (WANG, 2009):

$$L(k+1) = A_l L(k), \tag{A.31}$$

sendo que a matriz A_l é $n \times n$ e calculada em termos de a e $\beta = \sqrt{1-a^2}$, com condição inicial de $L(0)^T$ sendo:

$$L(0)^T = \sqrt{\beta}[1, -a, a^2 \cdots, (-1)^{n-1}a^{n-1}]^T. \tag{A.32}$$

Como um exemplo, para $n = 4$:

$$A_l = \begin{bmatrix} a & 0 & 0 & 0 \\ \beta & a & 0 & 0 \\ -a\beta & \beta & a & 0 \\ a^2\beta & -a\beta & \beta & a \end{bmatrix} \; e \; L(0) = \sqrt{\beta} \begin{bmatrix} 1 \\ -a \\ a^2 \\ -a^3 \end{bmatrix}. \tag{A.33}$$

Vale ressaltar que as funções de Laguerre $l_n(k)$ formam uma base no espaço de Hilbert l^2 que é o espaço de todas as funções $g(k)$ satisfazendo (YOUNG, 1988):

$$\sum_{k=0}^{\infty} g^2(k) < \infty. \tag{A.34}$$

Observe que respostas ao impulso de todos os sistemas estáveis, causais e lineares pertencem a l^2. Um conjunto de sequências de l^2 é dito completo se qualquer sequência desse espaço de Hilbert puder ser aproximada, arbitrariamente (na norma induzida pelo produto interno), por uma combinação linear das sequências desse conjunto. Se essas sequências são ortonormais, então o conjunto é chamado de base ortonormal de l^2. Através da definição de produto interno entre duas funções discretas (PORAT, 1994):

$$< f(k), g(k) >= \sum_{k=0}^{\infty} f(k)g(k), \tag{A.35}$$

e com o conhecimento de que as funções de Laguerre são ortonormais, ou seja:

$$\sum_{k=0}^{\infty} l_i(k)l_j(k) = \begin{cases} 1, & if \; i = j \\ 0, & if \; i \neq j \end{cases}, \tag{A.36}$$

então as funções de Laguerre no tempo discreto formam uma base que pode expandir qualquer função $f(k) \in l^2$ da forma:

$$f(k) = \sum_{i=0}^{\infty} \eta_i l_i(k) = HL(k), \tag{A.37}$$

com

$$\eta_i =< f(k), l_i(k) >= \sum_{k=0}^{\infty} f(k)l_i(k). \tag{A.38}$$

A.4 SOLUÇÃO DO PROBLEMA LQR POR ELIMINAÇÃO DE BLOCOS

Primeiramente é necessário enunciar o teorema que garante o mínimo da função custo para o problema LQR. A condição necessária e suficiente para que a solução ótima dentre um conjunto dos controles admissíveis U, limitado em um certo range de valores, seja factível até nas extremidades de tal conjunto é enunciado no teorema de Pontryagin. Ele se faz necessário, pois garante a solução ótima em pontos situados nas bordas dos conjuntos de $u \in U$, onde há possíveis descontinuidades (JAMSHIDI; TAROKH; SHAFAI, 1992). Para tal, considere o sistema:

$$\frac{dx}{dt} = f(x, u), \tag{A.39}$$

com estado inicial

$$x(t_0) = x_0, \tag{A.40}$$

com uma classe de controles admissíveis

$$u(t) \in U, \tag{A.41}$$

e função custo

$$J = F[x(t_f)] + \int_{t_0}^{t_f} L[x(t), u(t)]dt, \tag{A.42}$$

sendo L(.) continuamente diferenciável em $\mathbb{R}^n \times U$. Para a prova do teorema, precisa-se definir a função Hamiltoniana.

Definição A.4.1. Define-se $H(x, u, \lambda)$ uma função escalar real de $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ e $\lambda \in \mathbb{R}^n$ e dada por

$$H(x, u, \lambda) = L(x, u) + \lambda^T f(x, u), \tag{A.43}$$

sendo λ chamada de vetor "costate". H(.) é conhecida como a função Hamiltoniana.

Agora, o princípio mínimo pode ser declarado com o Teorema A.4.1 (JAMSHIDI; TAROKH; SHAFAI, 1992).

Teorema A.4.1. *[Teorema de Pontryagin] Suponha que u^* é um controle ótimo para o problema com as equações e restrições (A.39)-(A.42), com x^* a correspondente trajetória de estado ótima. Logo, existe um vetor não-nulo λ tal que:*

$$\frac{dx}{dt} = \frac{\partial H}{\partial \lambda} = f(x, u), \tag{A.44a}$$

$$\frac{d\lambda}{dt} = -\frac{\partial H}{\partial x} = -\frac{\partial L}{\partial x} - (\frac{\partial f}{\partial x})^T \lambda, \tag{A.44b}$$

$$H(x^*, u^*, \lambda^*) = \min_{u \in U} H(x^*, u, \lambda^*). \tag{A.44c}$$

Além disso, $H(x^*, u^*, \lambda^*)$ *é constante para* $0 \leq t \leq t_f$ *e* (A.44b) *precisa ser acomodada com uma condição de transversalidade que depende ainda de* $x(t_f)$ *e de* t_f*, porém tal condição não será abordada aqui por fugir do escopo deste trabalho.*

Agora, vamos ao problema de um sistema linear discreto. Considere o sistema com estados $x(k) \in \mathbb{R}^n$, entradas $u(k) \in \mathbb{R}^m$ e sistema inicial dado por $x(0) = x_0$.

$$x(k+1) = Ax(k) + Bu(k), \ k = 0, 1, \cdots, N-1. \tag{A.45}$$

Em problemas LQR, o objetivo é encontrar o conjunto de entradas $U = [u(0), \cdots, u(N-1)]$ que minimiza a função custo

$$J(U) = \frac{1}{2} \sum_{k=0}^{N-1} [x(k)^T Q x(k) + u(k)^T R u(k)] + \frac{1}{2} x(N)^T Q_f x(N), \tag{A.46}$$

onde Q e Q_f são matrizes positivas semi-definidas e R é uma matriz positiva definida. Define-se a função Hamiltoniana por:

$$H(k) = \frac{1}{2} \sum_{k=0}^{N-1} [x(k)^T Q x(k) + u(k)^T R u(k)] + \lambda(k+1)^T [Ax(k) + Bu(k)]. \tag{A.47}$$

Através do Teorema A.4.1, as condições necessárias e suficientes para a solução do problema são dadas por:

$$\lambda(k) = \frac{\partial H(k)}{\partial x(k)}, \tag{A.48a}$$

$$\lambda(N) = \frac{\partial}{\partial x(N)} (1/2 x(N) Q_f x(N)), \tag{A.48b}$$

$$x(k+1) = \frac{\partial H(k)}{\partial \lambda(k+1)}, \tag{A.48c}$$

$$x(k_0) = x_0, \ (conhecida) \tag{A.48d}$$

$$\frac{\partial H(k)}{\partial u(k)} = 0. \tag{A.48e}$$

Resolvendo as (A.48a) - (A.48e) em conjunto com (A.47), tem-se:

$$\begin{gathered}\lambda(k) = Qx(k) + A^T\lambda(k+1),\ k = 1, \cdots, N-1 \\ x(k+1) = Ax(k) + Bu(k),\ k = 0, \cdots, N-1 \\ Ru(k) + B^T\lambda(k+1) = 0,\ k = 0, \cdots, N-1 \\ x(0) = x_0 \\ \lambda(N) = Q_f x(N).\end{gathered} \tag{A.49}$$

Note que (A.49) é um conjunto de $N(m+2n)$ equações lineares nas variáveis u, x e λ. Dessa forma, pode-se escrever (A.49) em forma matricial (SKAF, 2008), ou seja:

$$\begin{bmatrix} R & B^T & 0 & \cdots & \cdots & \cdots & \cdots \\ B & 0 & -I & 0 & \cdots & \cdots & \cdots \\ 0 & -I & Q & 0 & A^T & \cdots & \cdots \\ \vdots & \vdots & 0 & R & B^T & \cdots & \cdots \\ \vdots & \vdots & A & B & 0 & -I & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & -I \\ 0 & \cdots & \cdots & \cdots & \cdots & -I & Q_f \end{bmatrix} \begin{bmatrix} u(0) \\ \lambda(1) \\ x(1) \\ u(1) \\ \lambda(2) \\ x(2) \\ \vdots \\ x(N) \end{bmatrix} = \begin{bmatrix} 0 \\ -Ax_0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{A.50}$$

Observe que (A.50) exibe uma estrutura matricial em faixas, no qual elementos não nulos aparecem apenas em uma faixa ao redor da diagonal principal. Por eliminação de variáveis no sentido reverso de (A.50), i.e., de N até 1, pode-se encontrar a equação de recursividade de Riccati como a solução de (A.50). Logo, eliminando as variáveis $\lambda(N)$, $x(N)$, $u(N-1)$, $\lambda(N-1)$ e $x(N-1)$, tem-se as seguintes equações:

$$\begin{bmatrix} -I & Q & 0 & A^T & 0 \\ 0 & 0 & R & B^T & 0 \\ 0 & A & B & 0 & -I \\ 0 & 0 & 0 & -I & P_N \end{bmatrix} \begin{bmatrix} \lambda(N-1) \\ x(N-1) \\ u(N-1) \\ \lambda(N) \\ x(N) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{A.51}$$

com $P_N = Q_f$. Observe que as variáveis $\lambda(N)$ e $x(N)$ são facilmente determinadas de (A.51) por:

$$\begin{bmatrix}\lambda(N)\\ x(N)\end{bmatrix} = \begin{bmatrix}0 & -I\\ -I & P_N\end{bmatrix}^{-1}\left(\begin{bmatrix}0\\0\end{bmatrix} - \begin{bmatrix}0 & A & B\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\lambda(N-1)\\ x(N-1)\\ u(N-1)\end{bmatrix}\right)$$
$$\begin{bmatrix}\lambda(N)\\ x(N)\end{bmatrix} = \begin{bmatrix}P_N & I\\ I & 0\end{bmatrix}\begin{bmatrix}0 & A & B\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\lambda(N-1)\\ x(N-1)\\ u(N-1)\end{bmatrix}. \tag{A.52}$$

Agora, resolvendo (A.51) para $\lambda(N-1)$, $x(N-1)$ e $u(N-1)$, tem-se:

$$\begin{bmatrix}-I & Q & 0\\ 0 & 0 & R\end{bmatrix}\begin{bmatrix}\lambda(N-1)\\ x(N-1)\\ u(N-1)\end{bmatrix} + \begin{bmatrix}A^T & 0\\ B^T & 0\end{bmatrix}\begin{bmatrix}\lambda(N)\\ x(N)\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}. \tag{A.53}$$

De (A.52) em (A.53), resulta em:

$$\left(\begin{bmatrix}-I & Q & 0\\ 0 & 0 & R\end{bmatrix} + \begin{bmatrix}A^T & 0\\ B^T & 0\end{bmatrix}\begin{bmatrix}P_N & I\\ I & 0\end{bmatrix}\begin{bmatrix}0 & A & B\\ 0 & 0 & 0\end{bmatrix}\right)\begin{bmatrix}\lambda(N-1)\\ x(N-1)\\ u(N-1)\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$$
$$\begin{bmatrix}-I & (Q+A^TP_NA) & A^TP_NB\\ 0 & B^TP_NA & (R+B^TP_NB)\end{bmatrix}\begin{bmatrix}\lambda(N-1)\\ x(N-1)\\ u(N-1)\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}. \tag{A.54}$$

É direto que de (A.54):

$$u(N-1) = -(R+B^TP_NB)^{-1}B^TP_NAx(N-1). \tag{A.55}$$

Assim, resolvendo (A.54) para $\lambda(N-1)$, com (A.55), tem-se:

$$-\lambda(N-1) + \underbrace{[Q+A^TP_NA - A^TP_NB(R+B^TP_NB)^{-1}B^TP_NA]}_{P_{N-1}}x(N-1)$$
$$\lambda(N-1) = P_{N-1}x(N-1). \tag{A.56}$$

Verifica-se que o sistema (A.50) se reduziu a

$$\begin{bmatrix} R & B^T & 0 & \cdots & \cdots & \cdots & \cdots \\ B & 0 & -I & 0 & \cdots & \cdots & \cdots \\ 0 & -I & Q & 0 & A^T & \cdots & \cdots \\ \vdots & \vdots & 0 & R & B^T & \cdots & \cdots \\ \vdots & \vdots & A & B & 0 & -I & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & -I \\ 0 & \cdots & \cdots & \cdots & \cdots & -I & P_{N-1} \end{bmatrix} \begin{bmatrix} u(0) \\ \lambda(1) \\ x(1) \\ u(1) \\ \lambda(2) \\ x(2) \\ \vdots \\ x(N-1) \end{bmatrix} = \begin{bmatrix} 0 \\ -Ax_0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{A.57}$$

Realizando tal processo reversamente até $u(0)$, $\lambda(1)$ e $x(1)$, encontrar-se-á as mesmas fórmulas de recorrência que (A.55) e (A.56), exceto com diferentes índices de k. Finalmente, prova-se que a equação recursiva de Riccati gera a solução dos P_k para a solução do problema LQR em que as entradas ótimas são calculadas por:

$$u(k) = -(R + B^T P_{k+1} B)^{-1} B^T P_{k+1} A x(k), \tag{A.58}$$

para $k = 0, 1, \cdots, N-1$ e $P_N = Q_f$, no qual P_k é calculado por:

$$P_k = Q + A^T P_{k+1} A - A^T P_{k+1} B (R + B^T P_{k+1} B)^{-1} B^T P_{k+1} A, \tag{A.59}$$

e o ganho $K(k)$:

$$K(k) = (R + B^T P_{k+1} B)^{-1} B^T P_{k+1} A, \tag{A.60}$$

da forma que o sistema (A.45), com a realimentação $u(k) = -K(k)x(k)$, fica:

$$x(k+1) = (A - BK(k))x(k). \tag{A.61}$$

Note que para o caso invariante no tempo, i.e., $N \to \infty$ e $Q_f = 0$, pode-se tomar a solução de P_k quando ele atinge o seu estado estacionário, ou seja:

$$P_\infty = \lim_{k \to \infty} P_k, \tag{A.62}$$

e a solução de Riccati (A.59) se reduz a:

$$P_\infty = Q + A^T P_\infty A - A^T P_\infty B (R + B^T P_\infty B)^{-1} B^T P_\infty A, \tag{A.63}$$

com

$$K = (R + B^T P_\infty B)^{-1} B^T P_\infty A. \tag{A.64}$$

B CÓDIGOS EM VHDL

B.1 PIDOF-BOOST


```
1   -- FOPID BOOST 12 bits with order reduction
2   -- SEPTEMBER/2022
3   -- Autor: Luís Felipe da Silva Carlos Pereira
4   -- email: luis.pereira@ufms.br
5   -- Update 13/09/2022
6   -- [wb,wh] = [1e-4,1e+4]
7   -- Ts = 1/(wh/2) % Sampling time 2*10^-4 - 5Khz
8   -- N = 1
9   -- wgc = 2*pi*200 rad/s
10  -- mg = 60*pi/180 rad (60°)
11  -- filter new_fod with norm2 reduction
12  -- reduction to (1,1)
13  -- G1 = 1/(s/(wh)+1);
14
15  ------- Boost plant
16
17     -- V0 = 40; % Voltage reference
18     -- D = 0.75; % Ratio voltage
19     -- L = 2e-3;% Indutance
20     -- IL = 1.6;% Dc inductor current
21     -- C = 680e-6;% Capacitance
22     -- R = 100;% Resistance
23
24  library ieee;
25  library ieee_proposed;
26  use ieee.std_logic_1164.all;
27  use ieee.numeric_std.all;
28  use ieee.fixed_float_types.all;
29  use ieee.fixed_pkg.all;
30  use ieee_proposed.real_matrix_pkg.all;
31  use ieee_proposed.fixed_matrix_pkg.all;
32
33  entity boost_red_200hz is
34        generic (
35              N1 : integer := 16; -- integer # of bits
36              N2 : integer := 40; -- fractional # of bits
37              freq_sample : integer := 400; --  freq_sample clks of 2 Mhz = 5 Khz =
    (Ts/freq_sample)
38              duty_cycle : integer := 1 -- 1 clk of 2 Mhz = 2 Mhz
39              );
40        Port (
41              clk : in std_logic; -- clk: external clock
42              n_reset : in  std_logic; -- external reset
43              sample_trig_out   : out  std_logic; -- Ts
44              V : in  std_logic_vector (11 downto 0); -- input voltage DAC
45              filter_done : out std_logic; -- flag
46              Mv : out  std_logic_vector (11 downto 0); -- duty cycle
47              PWM_mosfet: out  std_logic -- PWM to mosfet
48              );
49  end boost_red_200hz;
50
51  architecture arch of boost_red_200hz is
52
53
54     -- (PIDOF-BOOST)
55
56     ------------------------------------------------------------
57     ------------------------------------------------------------
58
59     -- Initializing all vectors and matrices
60
61     -- Plant Matrices
62     signal Ad1  :  sfixed_matrix (0 to 2, 0 to 2) := zeros(3,3);
63     signal Ad2  :  sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
64     signal Bd1  :  sfixed_matrix (0 to 2, 0 to 0) := zeros(3,1);
65     signal Bd2  :  sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
66     signal Cd1  :  sfixed_matrix (0 to 0, 0 to 2) := zeros(1,3);
67     signal Cd2  :  sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
68     signal Dd1, Dd2   :  sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
69
70     -- define each product factor truncated sample (resize)
71     signal pf_AX1 : sfixed_matrix (0 to 2, 0 to 0) := zeros(3,1);
72     signal pf_AX2 : sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
```

```
   signal pf_BU1 : sfixed_matrix (0 to 2, 0 to 0) := zeros(3,1);
   signal pf_BU2 : sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
   signal pf_CX1, pf_CX2 : sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
   signal pf_DU1, pf_DU2, Hin: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);

   -- define auxiliary variable to store input signal

   signal xmMaisHum1 : sfixed_matrix (0 to 2, 0 to 0) := zeros(3,1);
   signal xmMaisHum2 : sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
   signal xm1 : sfixed_matrix (0 to 2, 0 to 0) := zeros(3,1);
   signal xm2 : sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
   signal u1, u2: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1); -- inputed u
   signal y1, y2: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1); -- y = Cx+Du
   signal d, d_max, d_min: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1); -- duty ↵
constraints
   shared variable pwm_aux: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1); -- ↵
freq_sample2*freq_sample

   -- state machine signals
   type state_type is (idle, run);
   signal state_reg, state_next : state_type;

   -- counter signals
   signal q_reg, q_next: integer := 0;

   -- signal q_reg, q_next: unsigned(4 downto 0);
   signal q_reset, q_add : std_logic;

   -- data path flags
   signal mul_coefs, sum_val, trunc_val, constraint, output_flag: std_logic;

   -- PWM signal statement
   signal sample_trig : std_logic := '0'; -- PWM in
   signal sample_trig2 : std_logic := '0'; -- PWM out

begin

-- process to read matrices and update input values

      -- MATRIX Ad

      Ad1(0,0) <= to_sfixed(0.981466294108261, N1,-N2);
      Ad1(1,0) <= to_sfixed(0.000198140850756413, N1,-N2);
      Ad1(1,1) <= to_sfixed(1.0, N1,-N2);
      Ad1(2,0) <= to_sfixed(1.98758637349320e-08, N1,-N2);
      Ad1(2,1) <= to_sfixed(0.0002, N1,-N2);
      Ad1(2,2) <= to_sfixed(1.0, N1,-N2);
      Ad2(0,0) <= to_sfixed(0.0172412592488506, N1,-N2);

      ---- MATRIX Bd

      Bd1(0,0)<=to_sfixed(0.000198140850756413 ,N1,-N2);
      Bd1(1,0)<=to_sfixed(1.98758637349320e-08 ,N1,-N2);
      Bd1(2,0)<=to_sfixed(1.32712072339847e-12 ,N1,-N2);
      Bd2(0,0)<=to_sfixed(4.84063956955875e-05 ,N1,-N2);

      ---- MATRIX Cd

      Cd1(0,0)<=to_sfixed(-2.77555756156289e-17 ,N1,-N2);
      Cd1(0,1)<=to_sfixed(31.8184638206395 ,N1,-N2);
      Cd1(0,2)<=to_sfixed(0.596630553925904 ,N1,-N2);
      Cd2(0,0)<=to_sfixed(-891.829823233546 ,N1,-N2);

      ---- MATRIX Dd

      Dd1(0,0)<=to_sfixed(0.00155029212119842 ,N1,-N2);
      Dd2(0,0)<=to_sfixed(0.0439276350608418 ,N1,-N2);

      ---- Voltage and error gain to real values
      Hin(0,0) <= to_sfixed(13.3333, N1,-N2);
      d_max(0,0) <= to_sfixed(0.9, N1,-N2);
      d_min(0,0) <= to_sfixed(0.0, N1,-N2);
```

```
144         -- PWM Process
145      process (clk,n_reset)
146
147      variable count: integer := 0;
148
149      begin
150
151      if(n_reset = '0') then
152         sample_trig <= '0';
153         count := 0;
154
155      elsif(clk'event and clk = '1') then
156
157      ---- Duty cycle control
158         if(count < duty_cycle) then
159            sample_trig <= '1';
160         else
161            sample_trig <= '0';
162         end if;
163
164      ---- Counter control
165         if(count = (freq_sample-1)) then
166            count := 0;
167         else
168            count := count + 1;
169         end if;
170
171      end if;
172
173      sample_trig_out <= sample_trig;
174      end process;
175
176      process(clk, n_reset, sample_trig, output_flag)
177
178      variable uaux2: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
179
180      begin
181         if(n_reset = '0') then
182
183         --- Reseting input values
184         u1(0,0) <= to_sfixed(0.0, N1, -N2);
185         u2(0,0) <= to_sfixed(0.0, N1, -N2);
186         uaux2(0,0) := to_sfixed(0.0, N1, -N2);
187
188         ---Initializing the state variables
189         xm1 <= zeros(3,1);
190         xm2 <= zeros(1,1);
191
192
193         elsif(clk'event and clk = '1') then
194
195            -- update input values
196            if(sample_trig = '1') then
197               uaux2(0,0) := resize(to_sfixed(V,2,-9),N1,-N2,fixed_saturate,fixed_round);
198               u1(0,0) <= ↵
      resize((to_sfixed(40.0,6,-2)-uaux2(0,0)*Hin(0,0)),N1,-N2,fixed_saturate,fixed_round); ↵
      -- e(t)
199               u2(0,0) <= ↵
      resize(uaux2(0,0)*Hin(0,0),N1,-N2,fixed_saturate,fixed_round); -- y(t)
200            end if;
201
202            -- update state and manipulated variables
203            if(output_flag = '1') then
204               xm1 <= resize(xmMaisHum1,N1,-N2,fixed_saturate,fixed_round);
205               xm2 <= resize(xmMaisHum2,N1,-N2,fixed_saturate,fixed_round);
206            end if;
207
208          end if;
209
210      end process;
211
212      -- STATE UPDATE AND TIMING
213      process(clk, n_reset)
```

```
214      begin
215         if(n_reset = '0') then
216            state_reg <= idle;
217            q_reg <= 0;
218            -- q_reg <= (others => '0');
219         elsif (clk'event and clk = '1')  then
220            state_reg <= state_next;
221            q_reg <= q_next;
222         end if;
223      end process;
224
225      -- COUNTER FOR TIMING: integers
226       q_next <= 0 when q_reset = '1' else
227                     q_reg + 1 when q_add = '1' else
228                     q_reg;
229
230
231      -- process for control of data path flags
232      process( q_reg, state_reg, sample_trig)
233      begin
234
235         q_reset <= '0';
236         q_add <= '0';
237         mul_coefs <= '0';
238         sum_val <= '0';
239         trunc_val <= '0';
240         constraint <= '0';
241         output_flag <= '0';
242         filter_done <= '0';
243
244         case state_reg is
245
246         when idle =>
247
248            if(sample_trig = '1') then
249               state_next <= run;
250               -- flag_sample <= '0';
251            else
252               state_next <= idle;
253            end if;
254
255         when run =>
256
257            if( q_reg < 1) then
258               q_add <= '1';
259               state_next <= run;
260            elsif( q_reg < 6) then
261               mul_coefs <= '1';
262               q_add <= '1';
263               state_next <= run;
264            elsif( q_reg < 11) then
265               sum_val <= '1';
266               q_add <= '1';
267               state_next <= run;
268            elsif( q_reg < 14) then
269               trunc_val <= '1';
270               q_add <= '1';
271               state_next <= run;
272            elsif( q_reg < 16) then
273               constraint <= '1';
274               q_add <= '1';
275               state_next <= run;
276            elsif( q_reg < 17) then
277               output_flag <= '1';
278               q_add <= '1';
279               state_next <= run;
280            else
281               q_reset <= '1';
282               filter_done <= '1';
283               state_next <= idle;
284            end if;
285         end case;
286      end process;
```

```
287
288      -- product among factors
289      process(clk, mul_coefs)
290      begin
291         if(n_reset = '0') then
292            pf_BU1 <= zeros(3,1);
293            pf_AX1 <= zeros(3,1);
294            pf_CX1 <= zeros(1,1);
295            pf_DU1 <= zeros(1,1);
296            pf_BU2 <= zeros(1,1);
297            pf_AX2 <= zeros(1,1);
298            pf_CX2 <= zeros(1,1);
299            pf_DU2 <= zeros(1,1);
300
301         elsif(clk'event and clk = '1') then
302            if(mul_coefs = '1') then
303               pf_AX1 <= resize(Ad1*xm1,N1,-N2,fixed_saturate,fixed_round);
304               pf_BU1 <= resize(Bd1*u1,N1,-N2,fixed_saturate,fixed_round);
305               pf_CX1 <= resize(Cd1*xm1,N1,-N2,fixed_saturate,fixed_round);
306               pf_DU1 <= resize(Dd1*u1,N1,-N2,fixed_saturate,fixed_round);
307
308               pf_AX2 <= resize(Ad2*xm2,N1,-N2,fixed_saturate,fixed_round);
309               pf_BU2 <= resize(Bd2*u2,N1,-N2,fixed_saturate,fixed_round);
310               pf_CX2 <= resize(Cd2*xm2,N1,-N2,fixed_saturate,fixed_round);
311               pf_DU2 <= resize(Dd2*u2,N1,-N2,fixed_saturate,fixed_round);
312            end if;
313         end if;
314      end process;
315
316      process(clk, sum_val)
317      begin
318         if(n_reset = '0') then
319            xmMaisHum1 <= zeros(3,1);
320            xmMaisHum2 <= zeros(1,1);
321            y1 <= zeros(1,1);
322            y2 <= zeros(1,1);
323         elsif(clk'event and clk = '1') then
324            if(sum_val = '1') then
325               xmMaisHum1 <= resize(pf_AX1 + pf_BU1,N1,-N2,fixed_saturate,fixed_round); ↵
    --x(k+1) = A*x(k)+B*u(k)
326               y1 <= resize(pf_CX1 + pf_DU1,N1,-N2,fixed_saturate,fixed_round); -- y(k) ↵
    = C*x(k)+D*u(k)
327
328               xmMaisHum2 <= resize(pf_AX2 + pf_BU2,N1,-N2,fixed_saturate,fixed_round);
329               y2 <= resize(pf_CX2 + pf_DU2,N1,-N2,fixed_saturate,fixed_round);
330            end if;
331         end if;
332      end process;
333
334      process(clk, trunc_val, constraint)
335      begin
336         if(n_reset = '0') then
337            d <= zeros(1,1);
338         elsif(clk'event and clk = '1') then
339
340            if (trunc_val = '1') then
341               d(0,0) <= resize(y1(0,0)-y2(0,0),N1,-N2,fixed_saturate,fixed_round);
342            end if;
343
344            if (constraint = '1') then
345               if (d(0,0) > d_max(0,0)) then
346                  d(0,0) <= d_max(0,0);
347               elsif (d(0,0) < d_min(0,0)) then
348                  d(0,0) <= d_min(0,0);
349               end if;
350            end if;
351         end if;
352      end process;
353
354      process(clk, output_flag)
355
356      variable duty_cycle2: integer := 0;
357      variable count_flag: integer := 0;
```

```
358      variable count2: integer := 0;
359
360      begin
361
362      if(n_reset = '0') then
363         sample_trig2 <= '0';
364         count2 := 0;
365         count_flag := 0;
366         pwm_aux(0,0) := to_sfixed(0.0, N1, -N2);
367         Mv <= (others => '0');
368
369      elsif(clk'event and clk = '1') then
370
371      if(output_flag = '1') then
372
373      pwm_aux(0,0) := ↵
     resize(to_sfixed(0.25,N1,-N2)*to_sfixed(freq_sample,N1,-N2)*d(0,0),N1,-N2,fixed_satura↵
     te,fixed_round);
374      Mv <= to_slv(resize(d(0,0),1,-10,fixed_saturate,fixed_round));
375      count_flag := 1; -- synchronize with the first output
376      duty_cycle2 := to_integer(pwm_aux(0,0));
377
378      end if;
379
380      ---- Counter control Output PWM
381
382      if(count_flag = 1) then -- Ts = 5Khz and f_mosfet = 20Khz -> freq_sample/4
383         if(count2 = freq_sample/4-1 ) then
384         count2 := 0;
385         else
386         count2 := count2+1;
387         end if;
388
389      ---- Duty cycle control
390         if(count2 < duty_cycle2) then -- because cont starts at 0
391            sample_trig2 <= '1';
392         else
393            sample_trig2 <= '0';
394         end if;
395      end if;
396      end if;
397
398      PWM_mosfet <= sample_trig2;
399      end process;
400
401   end arch;
402
```

B.2 MPC-BOOST


```
1   -- MPC-Boost 12 bits
2   -- september/2022
3   -- Update 05/09/2022
4   -- Autor: Luís Felipe da Silva Carlos Pereira
5   -- email: luis.pereira@ufms.br
6   ----------------------------------------------------------------
7   ----------------------------------------------------------------
8   -- Ts = 2e-4
9   -- N = 5
10  -- a = 0.4
11  -- alpha = 1.001
12  -- lambda= 0.97;
13
14     -- V0 = 40; % Voltage reference
15     -- D = 0.75; % Ratio voltage
16     -- L = 2e-3;% Indutance
17     -- IL = 1.6;% Dc inductor current
18     -- C = 680e-6;% Capacitance
19     -- R = 100;% Resistance
20
21  library ieee;
22  library ieee_proposed;
23  use ieee.std_logic_1164.all;
24  use ieee.std_logic_unsigned.all;
25  use ieee.fixed_float_types.all;
26  use ieee.fixed_pkg.all;
27  use ieee_proposed.real_matrix_pkg.all;
28  use ieee_proposed.fixed_matrix_pkg.all;
29
30  entity boost_MPC_1 is
31        generic (
32              N1 : integer := 21; -- integer # of bits
33              N2 : integer := 28; -- fractional # of bits
34              freq_sample : integer := 400; --  freq_sample clks of 2 Mhz = 5 Khz ↵
    (Ts/freq_sample)
35              duty_cycle : integer := 200
36                );
37        Port (
38              clk : in std_logic; -- clk: external clock
39              n_reset : in  std_logic; -- external reset
40              V : in std_logic_vector (11 downto 0) ; -- Manipulated Output
41              PWM_mosfet: out std_logic; -- PWM d(t)
42              sample_trig_out : out  std_logic; -- Ts
43              Mv : out std_logic_vector (11 downto 0); -- Manipulated variable d(t)
44              filter_done : out std_logic -- EOF
45              );
46  end boost_MPC_1;
47
48  architecture arch of boost_MPC_1 is
49
50
51     --  (MPC-Boost)
52     ----------------------------------------------------------------
53
54     -- Plant Matrices
55     signal Ad   :  sfixed_matrix (0 to 1, 0 to 1) := zeros(2,2);
56     signal Bd   :  sfixed_matrix (0 to 1, 0 to 0) := zeros(2,1);
57
58     -- MPC optimizated matrices
59     signal gain :  sfixed_matrix (0 to 4, 0 to 2) := zeros(5,3);
60     signal L_m1 : sfixed_matrix (0 to 0, 0 to 4) := zeros(1,5);
61
62     -- MPC constraints
63     signal u_max: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
64     signal u_min: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
65     signal deltau_max: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
66     signal deltau_min: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
67     signal Hin: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1); -- Voltage gain
68
69     -- define previous, current, next state spaces variables. Define previous and ↵
    current u variables
70     signal xmMaisHum, xm, xm_old: sfixed_matrix (0 to 1, 0 to 0) := zeros(2,1);
71     signal u1, u2: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
```

```
 72
 73      -- define each sum or subtraction factor inside the process of buildmatrices
 74      shared variable delta_x: sfixed_matrix (0 to 1, 0 to 0) := zeros(2,1);
 75      shared variable delta_tracking: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
 76      shared variable Xf: sfixed_matrix (0 to 2, 0 to 0) := zeros(3,1);
 77
 78      -- define each multiplication inside the process
 79      signal eta: sfixed_matrix (0 to 4, 0 to 0) := zeros(5,1);
 80      signal deltau1: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
 81
 82      -- define auxiliary variables to store input signals
 83      signal Yout_signal: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
 84      signal rki_signal: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
 85      shared variable Vaux1: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1); -- inputed V
 86
 87      -- state machine signals
 88      type state_type is (idle, run);
 89      signal state_reg, state_next : state_type;
 90
 91      -- counter signals
 92      signal q_reg, q_next: integer := 0;
 93
 94      -- signal q_reg, q_next: unsigned(4 downto 0);
 95      signal q_reset, q_add : std_logic;
 96
 97      -- data path flags
 98      signal sub_states, build_matrix, mul_coefs1, mul_coefs2, constraints1, ↵
     constraints2, trunc_out, output_flag: std_logic;
 99
100      -- PWM statement
101      signal sample_trig : std_logic := '1'; -- PWM in
102      signal sample_trig2 : std_logic := '1'; -- PWM out
103
104      begin
105
106         -- MATRIX Ad
107         Ad(0,0) <= to_sfixed(0.996145967649221, N1,-N2);
108         Ad(0,1) <= to_sfixed(-9.17486174794658, N1,-N2);
109         Ad(1,0) <= to_sfixed(0.000199644991635318, N1,-N2);
110         Ad(1,1) <= to_sfixed(0.999081923408564, N1,-N2);
111
112         ---- MATRIX Bd
113         Bd(0,0)<=to_sfixed(0.000199644991635318 ,N1,-N2);
114         Bd(1,0)<=to_sfixed(1.99773466296473e-08 ,N1,-N2);
115
116         --------- gain = -inv(omega)*psi
117         gain(0,0)<=to_sfixed(1621.09764846255,N1,-N2);
118         gain(0,1)<=to_sfixed(753517.470408691 ,N1,-N2);
119         gain(0,2)<=to_sfixed(0.00113505984276760 ,N1,-N2);
120         gain(1,0)<=to_sfixed(1076.95688726709 ,N1,-N2);
121         gain(1,1)<=to_sfixed(343932.477358918 ,N1,-N2);
122         gain(1,2)<=to_sfixed(0.000475668505481433 ,N1,-N2);
123         gain(2,0)<=to_sfixed(600.955368841878 ,N1,-N2);
124         gain(2,1)<=to_sfixed(-21410.6474474702 ,N1,-N2);
125         gain(2,2)<=to_sfixed(0.000164234071847281 ,N1,-N2);
126         gain(3,0)<=to_sfixed(169.966750200016 ,N1,-N2);
127         gain(3,1)<=to_sfixed(-363188.588596935 ,N1,-N2);
128         gain(3,2)<=to_sfixed(0.000178870710935343 ,N1,-N2);
129         gain(4,0)<=to_sfixed(-238.040156818766 ,N1,-N2);
130         gain(4,1)<=to_sfixed(-700986.420473546 ,N1,-N2);
131         gain(4,2)<=to_sfixed(0.000513934459908591 ,N1,-N2);
132
133         ---------L_m1 (a = 0.4)
134         L_m1(0,0) <= to_sfixed(0.916515138991168,N1,-N2);
135         L_m1(0,1) <= to_sfixed(-0.366606055596467,N1,-N2);
136         L_m1(0,2) <= to_sfixed(0.146642422238587,N1,-N2);
137         L_m1(0,3) <= to_sfixed(-0.0586569688954348,N1,-N2);
138         L_m1(0,4) <= to_sfixed(0.0234627875581739,N1,-N2);
139
140         ---deltau_max and deltau_min
141         deltau_max(0,0) <= to_sfixed(1.0,N1,-N2);
142         deltau_min(0,0) <= to_sfixed(-1.0,N1,-N2);
143
```

```
144         ---  u_max and u_min
145         u_max(0,0) <= to_sfixed(0.9,N1,-N2);
146         u_min(0,0) <= to_sfixed(0.0,N1,-N2);
147
148         ---- Voltage gain to real values
149         Hin(0,0) <= to_sfixed(13.3333, N1,-N2);
150
151         -- PWM Process
152      process (clk,n_reset)
153
154      variable count: integer := 0;
155
156      begin
157
158      if(n_reset = '0') then
159         sample_trig <= '1'; -- on reset (restart the process)
160         count := 0;
161
162      elsif(clk'event and clk = '1') then
163      ---- Duty cycle control
164         if(count < duty_cycle) then -- because cont starts at 0
165            sample_trig <= '1';
166         else
167            sample_trig <= '0';
168         end if;
169
170      ---- Counter control
171         if(count = (freq_sample-1)) then -- because cont starts at 0
172            count := 0;
173         else
174            count := count + 1;
175         end if;
176      end if;
177      sample_trig_out <= sample_trig;
178      end process;
179
180      process(clk, n_reset, sample_trig, output_flag)
181      begin
182         if(n_reset = '0') then
183         Yout_signal(0,0) <= to_sfixed(0.0, N1, -N2);
184         rki_signal(0,0)  <= to_sfixed(0.0, N1, -N2);
185         xm <= zeros(2,1);
186         xm_old <= zeros(2,1);
187         u1 <= zeros(1,1);
188         Vaux1(0,0) := to_sfixed(0.0, N1, -N2);
189
190         elsif(clk'event and clk = '1') then
191
192            -- update input values
193            if(sample_trig = '1') then
194               Vaux1(0,0) := resize(to_sfixed(V,2,-9),N1,-N2,fixed_saturate,fixed_round);
195               Yout_signal(0,0) <=  ↵
    resize(Vaux1(0,0)*Hin(0,0),N1,-N2,fixed_saturate,fixed_round);
196               rki_signal(0,0) <= to_sfixed(40.0,N1,-N2);
197            end if;
198
199            -- update state and manipulated variables
200            if(output_flag = '1') then
201            xm_old <= resize(xm,N1,-N2,fixed_saturate,fixed_round); -- Refreshing xm1_old
202            u1 <= resize(u2,N1,-N2,fixed_saturate,fixed_round); -- Refreshing u1
203            xm <= resize(xmMaisHum,N1,-N2,fixed_saturate,fixed_round); -- refreshing xm1
204            end if;
205          end if;
206      end process;
207
208      -- STATE UPDATE AND TIMING
209      process(clk, n_reset)
210      begin
211         if(n_reset = '0') then
212            state_reg <= idle;
213            q_reg <= 0;
214         elsif (clk'event and clk = '1')  then
215            state_reg <= state_next;
```

```
216            q_reg <= q_next;
217         end if;
218      end process;
219
220      -- COUNTER FOR TIMING: integers
221       q_next <= 0 when q_reset = '1' else
222                     q_reg + 1 when q_add = '1' else
223                     q_reg;
224
225      -- process for control of data path flags
226      process(q_reg, state_reg, sample_trig)
227      begin
228
229         q_reset <= '0';
230         q_add <= '0';
231         sub_states <= '0'; -- deltax and delta_tracking
232         build_matrix <= '0'; -- build Xf
233         mul_coefs1 <= '0'; -- build eta
234         mul_coefs2 <= '0'; -- build deltau
235         constraints1 <= '0'; -- deltau constraints
236         constraints2   <= '0'; -- u constraints
237         trunc_out <= '0'; -- build xmMaisHum
238         output_flag <= '0'; -- read the output value
239         filter_done <= '0'; -- MV output and update u values
240
241         case state_reg is
242         when idle =>
243
244            if(sample_trig = '1') then
245               state_next <= run;
246            else
247               state_next <= idle;
248            end if;
249
250         when run =>
251            if( q_reg < duty_cycle+1) then      -- 1
252               q_add <= '1';
253               state_next <= run;
254            elsif( q_reg < duty_cycle+3) then   -- 3
255               sub_states <= '1';
256               q_add <= '1';
257               state_next <= run;
258            elsif( q_reg < duty_cycle+6) then   -- 6
259               build_matrix <= '1';
260               q_add <= '1';
261               state_next <= run;
262            elsif( q_reg < duty_cycle+11) then   -- 11
263               mul_coefs1 <= '1';
264               q_add <= '1';
265               state_next <= run;
266            elsif( q_reg < duty_cycle+16) then   -- 16
267               mul_coefs2 <= '1';
268               q_add <= '1';
269               state_next <= run;
270            elsif( q_reg < duty_cycle+18) then   -- 18
271               constraints1 <= '1';
272               q_add <= '1';
273               state_next <= run;
274            elsif( q_reg < duty_cycle+20) then   -- 21
275               constraints2 <= '1';
276               q_add <= '1';
277               state_next <= run;
278            elsif( q_reg < duty_cycle+29) then   -- 29
279               trunc_out <= '1';
280               q_add <= '1';
281               state_next <= run;
282            elsif( q_reg < duty_cycle+30) then   -- 30
283               output_flag <= '1';
284               q_add <= '1';
285               state_next <= run;
286            else
287               q_reset <= '1';
288               filter_done <= '1';
```

```
289               state_next <= idle;
290            end if;
291         end case;
292      end process;
293
294      -- building deltax and delta_tracking matrices
295      process(clk, sub_states)
296      begin
297         if(n_reset = '0') then
298            delta_x := zeros(2,1);
299            delta_tracking := zeros(1,1);
300         elsif(clk'event and clk = '1') then
301            if(sub_states = '1') then
302            delta_x := resize(xm-xm_old,N1,-N2,fixed_saturate,fixed_round);
303            delta_tracking := ↵
       resize(Yout_signal-rki_signal,N1,-N2,fixed_saturate,fixed_round);
304            end if;
305         end if;
306      end process;
307
308      -- building Xf matrix
309      process(clk, build_matrix)
310      begin
311         if(n_reset = '0') then
312            Xf := zeros(3,1);
313         elsif(clk'event and clk = '1') then
314            if(build_matrix = '1') then
315            BuildMatrix(delta_x,Xf,0,0);
316            BuildMatrix(delta_tracking,Xf,2,0);
317            end if;
318         end if;
319      end process;
320
321      -- building eta matrix
322      process(clk, mul_coefs1)
323      begin
324         if(n_reset = '0') then
325            eta <= zeros(5,1);
326         elsif(clk'event and clk = '1') then
327            if(mul_coefs1 = '1') then
328            eta <= resize(-gain*Xf,N1,-N2,fixed_saturate,fixed_round);
329            end if;
330         end if;
331      end process;
332
333      -- building deltau1 matrix
334      process(clk, mul_coefs2, constraints2)
335      begin
336         if(n_reset = '0') then
337            deltau1 <= zeros(1,1);
338            u2 <= zeros(1,1);
339         elsif(clk'event and clk = '1') then
340
341            if(mul_coefs2 = '1') then
342
343            deltau1 <= resize(L_m1*eta,N1,-N2,fixed_saturate,fixed_round);
344
345            end if;
346
347            if (constraints1 = '1') then
348               if (deltau1(0,0) > deltau_max(0,0)) then
349                  deltau1(0,0) <= deltau_max(0,0);
350               elsif (deltau1(0,0) < deltau_min(0,0)) then
351                  deltau1(0,0) <= deltau_min(0,0);
352               end if;
353            u2 <= resize(u1+deltau1,N1,-N2,fixed_saturate,fixed_round);
354            end if;
355
356            if (constraints2 = '1') then
357               if u2(0,0) > u_max(0,0) then
358                  deltau1(0,0) <= ↵
       resize(u_max(0,0)-u1(0,0),N1,-N2,fixed_saturate,fixed_round);
359                  u2(0,0) <= u_max(0,0);
```

```
360               elsif u2(0,0) < u_min(0,0) then
361                  deltau1(0,0) <= ↵
      resize(u_min(0,0)-u1(0,0),N1,-N2,fixed_saturate,fixed_round);
362                  u2(0,0) <= u_min(0,0);
363               end if;
364            end if;
365
366         end if;
367      end process;
368
369      -- Computing xmMaisHum
370      process(clk, trunc_out)
371      begin
372         if(n_reset = '0') then
373         xmMaisHum <= zeros(2,1);
374         elsif(clk'event and clk = '1') then
375            if (trunc_out = '1') then
376               xmMaisHum <= resize(Ad*xm+Bd*u2,N1,-N2,fixed_saturate,fixed_round); -- ↵
      x(k+1) = Am*x(k)+Bm*u(k);
377            end if;
378         end if;
379      end process;
380
381      -- PWM OUT
382      process(clk, output_flag)
383      variable duty_cycle2: integer := 0;
384      variable count_flag: integer := 0;
385      variable count2: integer := 0;
386      variable pwm_aux: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
387
388      begin
389      if(n_reset = '0') then
390         sample_trig2 <= '1'; -- on reset (restart the process)
391         count2 := 0;
392         duty_cycle2 := 0;
393         count_flag := 0;
394         pwm_aux(0,0) := to_sfixed(0.0, N1, -N2);
395         Mv <= (others => '0');
396
397      elsif(clk'event and clk = '1') then
398
399      if(output_flag = '1') then
400
401      pwm_aux(0,0) := ↵
      resize(to_sfixed(0.25,N1,-N2)*to_sfixed(freq_sample,N1,-N2)*u2(0,0),N1,-N2,fixed_satur↵
      ate,fixed_round);
402      Mv <= to_slv(resize(u2(0,0),1,-10,fixed_saturate,fixed_round));
403      count_flag := 1; -- synchronize with the first output
404      duty_cycle2 := to_integer(pwm_aux(0,0));
405
406      end if;
407
408      ---- Counter control Output PWM
409      if(count_flag = 1) then -- Ts = 5Khz and f_mosfet = 20Khz -> freq_sample/4
410         if(count2 = freq_sample/4-1 ) then
411         count2 := 0;
412         else
413         count2 := count2+1;
414         end if;
415
416      ---- Duty cycle control
417         if(count2 < duty_cycle2) then -- because cont starts at 0
418            sample_trig2 <= '1';
419         else
420            sample_trig2 <= '0';
421         end if;
422      end if;
423      end if;
424
425      PWM_mosfet <= sample_trig2;
426      end process;
427   end arch;
```

B.3 GPC TIPO 3 (N=2)


```
-- GPC TIPO 3 (N=2) MIMO
-- October/2022
-- Autor: Luís Felipe da Silva Carlos Pereira
-- email: luis.pereira@ufms.br

------------------------------------------------------------
---------------------INÍCIO---------------------------------
------------------------------------------------------------

library ieee;
library ieee_proposed;

use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

use ieee.fixed_float_types.all;
use ieee.fixed_pkg.all;

use ieee_proposed.real_matrix_pkg.all;
use ieee_proposed.fixed_matrix_pkg.all;


entity GPC_mestrado is
            generic (   N1 : integer := 20;
                        N2 : integer := 50;
                        freq_sample : integer := 40;
                        duty_cycle : integer := 10
                     );
            Port ( clk : in std_logic;
                   n_reset : in  std_logic;
                   Yout : in std_logic_vector (15 downto 0);
                   rki : in std_logic_vector (15 downto 0);
                   sample_trig_out : out  std_logic;
                   delta_MV: out std_logic_vector (15 downto 0);
                   Mv : out std_logic_vector (15 downto 0);
                   filter_done : out std_logic
                 );
end GPC_mestrado;

architecture arch of GPC_mestrado is


   --  GPC TYPE 3 CONTROLLER WITH LAGUERRE FUNCTIONS AND CONSTRAINTS

   ------------------------------------------------------------
   ------------------------------------------------------------


   -- Define all constant matrices

   -- Plant Matrices
   signal Ad   :  sfixed_matrix (0 to 1, 0 to 1) := zeros(2,2);
   signal Bd   :  sfixed_matrix (0 to 1, 0 to 0) := zeros(2,1);

   -- MPC optimizated matrices

   signal gain :  sfixed_matrix (0 to 1, 0 to 4) := zeros(2,5);
   signal L_m1 : sfixed_matrix (0 to 0, 0 to 1) := zeros(1,2);

   -- MPC constraints

   signal u_max: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
   signal u_min: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
   signal deltau_max: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
   signal deltau_min: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);

   -- define previous, current, next state spaces variables
   -- define previous and current u variables
   shared variable x: sfixed_matrix (0 to 1, 0 to 3) := zeros(2,4);
   signal xm: sfixed_matrix (0 to 1, 0 to 0) := zeros(2,1);
   shared variable Deltax: sfixed_matrix (0 to 1, 0 to 0) := zeros(2,1);
   shared variable xmMaisHum: sfixed_matrix (0 to 1, 0 to 0) := zeros(2,1);
   shared variable u: sfixed_matrix (0 to 3, 0 to 0) := zeros(4,1);
```

```
74      signal deltau1: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
75      signal deltau2: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
76
77      -- define each sum or subtraction factor inside the process of buildmatrices
78      shared variable erro: sfixed_matrix (0 to 2, 0 to 0) := zeros(3,1);
79      shared variable Derro: sfixed_matrix (0 to 2, 0 to 0) := zeros(3,1);
80      shared variable Xf: sfixed_matrix (0 to 4, 0 to 0) := zeros(5,1);
81
82      -- define each multiplication inside the process
83      signal eta: sfixed_matrix (0 to 1, 0 to 0) := zeros(2,1);
84
85      -- define auxiliary variables to store input signals
86      shared variable Yout_signal: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
87      shared variable rki_signal: sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1);
88      signal a: sfixed_matrix (0 to 3, 0 to 0) := zeros(4,1);
89
90      -- define state machine signals
91      type state_type is (idle, run);
92      signal state_reg, state_next : state_type;
93
94      -- define counter signals
95      signal q_reg, q_next: integer := 0;
96
97      -- define signal q_reg, q_next: unsigned(4 downto 0);
98      signal q_reset, q_add : std_logic;
99
100     -- define data path flags
101     signal sub_states, build_matrix, mul_coefs1, mul_coefs2, constraints1: std_logic;
102     signal constraints2, trunc_out, output_flag, update_val: std_logic;
103
104     -- PWM statement
105     signal sample_trig : std_logic := '1';
106
107 begin
108
109         -- MATRIX Ad
110         Ad(0,0) <= to_sfixed(0.950356995019389, N1,-N2);
111         Ad(0,1) <= to_sfixed(-2.08556762946519, N1,-N2);
112         Ad(1,0) <= to_sfixed(0.000195000704005467, N1,-N2);
113         Ad(1,1) <= to_sfixed(0.999789673484775, N1,-N2);
114
115         ---- MATRIX Bd
116         Bd(0,0)<=to_sfixed(0.000195000704005467 ,N1,-N2);
117         Bd(1,0)<=to_sfixed(1.96655423494657e-08 ,N1,-N2);
118
119         --------- gain = -inv(omega)*Psi
120         gain(0,0)<=to_sfixed(1.58839330451132 ,N1,-N2);
121         gain(0,1)<=to_sfixed(9.75752169136440 ,N1,-N2);
122         gain(0,2)<=to_sfixed(36.8359064659624 ,N1,-N2);
123         gain(0,3)<=to_sfixed(-4148.39436438991 ,N1,-N2);
124         gain(0,4)<=to_sfixed(-410917.877549659 ,N1,-N2);
125         gain(1,0)<=to_sfixed(-1.45518646610023 ,N1,-N2);
126         gain(1,1)<=to_sfixed(-8.24311408478952 ,N1,-N2);
127         gain(1,2)<=to_sfixed(-25.4530152891940 ,N1,-N2);
128         gain(1,3)<=to_sfixed(933.369042602644 ,N1,-N2);
129         gain(1,4)<=to_sfixed(74038.5155087244,N1,-N2);
130
131         ---------L_m1
132         L_m1(0,0) <= to_sfixed(0.916515138991168 ,N1,-N2);
133         L_m1(0,1) <= to_sfixed(-0.366606055596467 ,N1,-N2);
134
135         ---deltau_max and deltau_min
136         deltau_max(0,0) <= to_sfixed(0.04,N1,-N2);
137         deltau_min(0,0) <= to_sfixed(-0.04,N1,-N2);
138
139         ---  u_max and u_min
140         u_max(0,0) <= to_sfixed(6.0,N1,-N2);
141         u_min(0,0) <= to_sfixed(-6.0,N1,-N2);
142
143         --- binomial coefficients "a"
144         a(0,0) <= to_sfixed(1.0,N1,-N2);
145         a(1,0) <= to_sfixed(-3.0,N1,-N2);
146         a(2,0) <= to_sfixed(3.0,N1,-N2);
```

```
147         a(3,0) <= to_sfixed(-1.0,N1,-N2);
148
149      process (clk,n_reset)
150
151      variable count: integer := 0;
152
153      begin
154
155      if(n_reset = '0') then
156         sample_trig <= '1';
157         count := 0;
158      elsif(clk'event and clk = '1') then
159         if(count < duty_cycle) then
160            sample_trig <= '1';
161         else
162            sample_trig <= '0';
163         end if;
164
165         if(count = (freq_sample-1)) then
166            count := 0;
167         else
168            count := count + 1;
169         end if;
170      end if;
171
172      sample_trig_out <= sample_trig;
173      end process;
174
175      process(clk, n_reset, sample_trig, update_val)
176      begin
177         if(n_reset = '0') then
178
179         Yout_signal(0,0) := to_sfixed(0.0, N1, -N2);
180         rki_signal(0,0)  := to_sfixed(0.0, N1, -N2);
181         x := zeros(2,4);
182         u(1,0) := to_sfixed(0.0, N1, -N2);
183         u(2,0) := to_sfixed(0.0, N1, -N2);
184         u(3,0) := to_sfixed(0.0, N1, -N2);
185         erro := zeros(3,1);
186
187         elsif(clk'event and clk = '1') then
188
189            -- update input values
190            if(sample_trig = '1') then
191               Yout_signal(0,0) := ₪
    resize(to_sfixed(Yout,3,-12),N1,-N2,fixed_saturate,fixed_round);
192               rki_signal(0,0) := ₪
    resize(to_sfixed(rki,3,-12),N1,-N2,fixed_saturate,fixed_round);
193               erro(0,0) := ₪
    resize(rki_signal(0,0)-Yout_signal(0,0),N1,-N2,fixed_saturate,fixed_round);
194
195            end if;
196
197            -- update state and manipulated variables
198            if(update_val = '1') then
199
200            ----- Updating xm and delta_x
201
202            x(0,3) := x(0,2);
203            x(1,3) := x(1,2);
204            x(0,2) := x(0,1);
205            x(1,2) := x(1,1);
206            x(0,1) := x(0,0);
207            x(1,1) := x(1,0);
208            x(0,0) := xmMaisHum(0,0);
209            x(1,0) := xmMaisHum(1,0);
210
211            ----- Updating u
212
213            u(3,0) := u(2,0);
214            u(2,0) := u(1,0);
215            u(1,0) := u(0,0);
216
```

```
217             ----- Updating error
218
219             erro(2,0) := erro(1,0);
220             erro(1,0) := erro(0,0);
221
222             end if;
223
224           end if;
225
226       end process;
227
228       process(clk, n_reset)
229       begin
230          if(n_reset = '0') then
231             state_reg <= idle;
232             q_reg <= 0;
233          elsif (clk'event and clk = '1')  then
234             state_reg <= state_next;
235             q_reg <= q_next;
236          end if;
237       end process;
238
239       -- COUNTER FOR TIMING: integers
240       q_next <= 0 when q_reset = '1' else
241       q_reg + 1 when q_add = '1' else  q_reg;
242
243       process(q_reg, state_reg, sample_trig)
244       begin
245
246          q_reset <= '0';
247          q_add <= '0';
248          sub_states <= '0'; -- deltax and delta_tracking
249          build_matrix <= '0'; -- build Xf
250          mul_coefs1 <= '0'; -- build eta
251          mul_coefs2 <= '0'; -- build deltau
252          constraints1 <= '0'; -- deltau constraints
253          constraints2   <= '0'; -- u constraints
254          trunc_out <= '0'; -- build xmMaisHum
255          output_flag <= '0'; -- read the output value
256          update_val <= '0'; -- update values
257          filter_done <= '0'; -- end of filtering
258
259          case state_reg is
260
261          when idle =>
262
263             if(sample_trig = '1') then
264                state_next <= run;
265             else
266                state_next <= idle;
267             end if;
268
269          when run =>
270
271            if( q_reg < 1) then
272                q_add <= '1';
273                state_next <= run;
274             elsif( q_reg < 5) then
275                sub_states <= '1';
276                q_add <= '1';
277                state_next <= run;
278             elsif( q_reg < 7) then
279                build_matrix <= '1';
280                q_add <= '1';
281                state_next <= run;
282             elsif( q_reg < 11) then
283                mul_coefs1 <= '1';
284                q_add <= '1';
285                state_next <= run;
286             elsif( q_reg < 15) then
287                mul_coefs2 <= '1';
288                q_add <= '1';
289                state_next <= run;
```

```
290             elsif( q_reg < 17) then
291                constraints1 <= '1';
292                q_add <= '1';
293                state_next <= run;
294             elsif( q_reg < 19) then
295                constraints2 <= '1';
296                q_add <= '1';
297                state_next <= run;
298             elsif( q_reg < 27) then
299                trunc_out <= '1';
300                q_add <= '1';
301                state_next <= run;
302             elsif( q_reg < 28) then
303                output_flag <= '1';
304                q_add <= '1';
305                state_next <= run;
306             elsif( q_reg < 29) then
307                update_val <= '1';
308                q_add <= '1';
309                state_next <= run;
310             else
311                q_reset <= '1';
312                filter_done <= '1';
313                state_next <= idle;
314             end if;
315
316          end case;
317       end process;
318
319       process(clk, sub_states)
320       begin
321          if(n_reset = '0') then
322             Deltax := zeros(2,1);
323             Derro := zeros(3,1);
324          elsif(clk'event and clk = '1') then
325             if(sub_states = '1') then
326             Deltax := resize(x*a,N1,-N2,fixed_saturate,fixed_round);
327             Derro(0,0) := resize(erro(0,0),N1,-N2,fixed_saturate,fixed_round);
328             Derro(1,0) := resize(erro(0,0) - erro(1,0),N1,-N2,fixed_saturate,fixed_round);
329             Derro(2,0) := resize(erro(0,0) - to_sfixed(2.0, N1,-N2)*erro(1,0) + ↵
    erro(2,0),N1,-N2,fixed_saturate,fixed_round);
330             end if;
331          end if;
332       end process;
333
334       process(clk, build_matrix)
335       begin
336          if(n_reset = '0') then
337             Xf := zeros(5,1);
338          elsif(clk'event and clk = '1') then
339             if(build_matrix = '1') then
340                BuildMatrix(Derro,Xf,0,0);
341                BuildMatrix(Deltax,Xf,3,0);
342             end if;
343          end if;
344       end process;
345
346       process(clk, mul_coefs1)
347       begin
348          if(n_reset = '0') then
349             eta <= zeros(2,1);
350             xm <= zeros(2,1);
351          elsif(clk'event and clk = '1') then
352             if(mul_coefs1 = '1') then
353                eta <= resize(gain*Xf,N1,-N2,fixed_saturate,fixed_round);
354                xm(0,0) <= x(0,0);
355                xm(1,0) <= x(1,0);
356             end if;
357          end if;
358       end process;
359
360       process(clk, mul_coefs2, constraints2)
361       begin
```

```
362         if(n_reset = '0') then
363            deltau1 <= zeros(1,1);
364            u(0,0) := to_sfixed(0.0, N1, -N2);
365            deltau2 <= zeros(1,1);
366            delta_MV <= (others => '0');
367         elsif(clk'event and clk = '1') then
368
369            if(mul_coefs2 = '1') then
370
371            deltau1 <= resize(L_m1*eta,N1,-N2,fixed_saturate,fixed_round);
372            u(0,0) := resize(deltau1(0,0) + a(2,0)*u(1,0) - a(2,0)*u(2,0) + ↵
       u(3,0),N1,-N2,fixed_saturate,fixed_round);
373            deltau2(0,0) <= resize(u(0,0)-u(1,0),N1,-N2,fixed_saturate,fixed_round);
374
375            end if;
376
377            if (constraints1 = '1') then
378               if (deltau2(0,0) > deltau_max(0,0)) then
379                  deltau2(0,0) <= deltau_max(0,0);
380               elsif (deltau2(0,0) < deltau_min(0,0)) then
381                  deltau2(0,0) <= deltau_min(0,0);
382               end if;
383            u(0,0) := resize(u(1,0)+deltau2(0,0),N1,-N2,fixed_saturate,fixed_round);
384            end if;
385
386            if (constraints2 = '1') then
387               if u(0,0) > u_max(0,0) then
388                  deltau2(0,0) <= ↵
       resize(u_max(0,0)-u(1,0),N1,-N2,fixed_saturate,fixed_round);
389                  u(0,0) := u_max(0,0);
390               elsif u(0,0) < u_min(0,0) then
391                  deltau2(0,0) <= ↵
       resize(u_min(0,0)-u(1,0),N1,-N2,fixed_saturate,fixed_round);
392                  u(0,0) := u_min(0,0);
393               end if;
394            delta_MV <= to_slv(resize(deltau2(0,0),2,-13,fixed_saturate,fixed_round));
395            end if;
396
397         end if;
398      end process;
399
400      process(clk, trunc_out)
401      begin
402         if(n_reset = '0') then
403            xmMaisHum := zeros(2,1);
404         elsif(clk'event and clk = '1') then
405            if (trunc_out = '1') then
406               xmMaisHum := resize(Ad*xm+Bd*u(0,0),N1,-N2,fixed_saturate,fixed_round);
407            end if;
408         end if;
409      end process;
410
411      process(clk, output_flag)
412      begin
413         if(n_reset = '0') then
414            Mv <= (others => '0');
415         elsif(clk'event and clk = '1') then
416            if (output_flag = '1') then
417               Mv <= to_slv(resize(u(0,0),3,-12,fixed_saturate,fixed_round));
418            end if;
419         end if;
420      end process;
421
422   end arch;
```