Silvane Cristina de Melo Schons

IMPLEMENTAÇÃO DE ESTRATÉGIA DE CONTROLE PREDITIVO EM CASCATA EM SISTEMA EMBARCADO DE TEMPO REAL E FPGA

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas para a obtenção do Grau de Mestre em Engenharia de Automação e Sistemas.
Orientador
UFSC: Prof. Dr. Rodolfo César Costa Flesch
Coorientador
UFSC: Prof. Dr. Julio Elias Normey-Rico

Florianópolis

2017

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Schons, Silvane Cristina de Melo
Implementação de estratégia de controle preditivo em cascata em sistema embarcado de tempo real e FPGA / Silvane Cristina de Melo Schons ; orientador, Rodolfo César Costa Flesch; coorientador, Julio Elias Normey-Rico - SC, 2017.
129 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2017.

Inclui referências.

1. Engenharia de Automação e Sistemas. 2. GPC. 3. PID. 4. myRIO. 5. LabVIEW. I. Flesch, Rodolfo César Costa. II. Normey-Rico, Julio Elias . III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. IV. Título.

Silvane Cristina de Melo Schons

IMPLEMENTAÇÃO DE ESTRATÉGIA DE CONTROLE PREDITIVO EM CASCATA EM SISTEMA EMBARCADO DE TEMPO REAL E FPGA

Esta Dissertação foi julgada aprovada para a obtenção do Título de "Mestre em Engenharia de Automação e Sistemas", e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Florianópolis, 29 de maio 2017.

Prof. Dr. Daniel Ferreira Coutinho
Coordenador do curso
UFSC

Banca Examinadora:

Prof. Dr. Rodolfo César Costa Flesch
Orientador
UFSC

Prof. Dr. Julio Elias Normey-Rico
Coorientador
UFSC

Prof. Dr. Eugênio de Bona Castelan Neto
UFSC

Prof. Dr. Douglas Wildgrube Bertol
UDESC

Dr. Paulo Renato da Costa Mendes
UFSC

AGRADECIMENTOS

A Deus, por ter me guiado e me dado forças para superar os desafios.
Aos meus pais, por todo carinho e compreensão, pois não mediram esforços para me ajudar em todos os momentos.
Ao meu namorado e companheiro, Julio, que sempre esteve ao meu lado e viu em mim o meu melhor.
Ao meu orientador, Rodolfo, e meu coorientador, Julio, por toda sua dedicação e todo o conhecimento que me passaram no decorrer do mestrado.
Aos membros da banca examinadora desta dissertação.
A todos que contribuíram direta ou indiretamente para realização deste trabalho e que não foram explicitamente citados nas linhas acima.

RESUMO

Este trabalho apresenta a implementação de uma estrutura de controle em cascata, muito utilizada em sistemas que possuem dois laços de controle com diferentes constantes de tempo e acesso ao sinal intermediário. O sistema de controle proposto é implementado em um *hardware* de alto desempenho, o myRIO, que possui um arranjo de portas programável em campo (FPGA) customizável e um processador que trabalha com um sistema operacional de tempo real, garantindo confiabilidade e determinismo ao sistema. Na malha interna da estrutura em cascata, é aplicado um controlador proporcional-integral (PI) implementado em FPGA e, na malha externa, é aplicado um controlador preditivo generalizado (GPC) implementado para execução no sistema operacional de tempo real do myRIO. A estrutura de controle foi desenvolvida de maneira que possa ser aplicada em diferentes sistemas monovariáveis, contudo foi escolhida uma planta de laboratório de controle de velocidade e corrente de um motor de corrente contínua como estudo de caso. Os resultados experimentais validaram a implementação em ambas as camadas. O período de amostragem que foi alcançado na implementação do PI no FPGA foi inferior a 10 μs e o período de amostragem da implementação do GPC no sistema operacional de tempo real foi de 13 ms (valor do tempo máximo observado), com período médio de execução inferior a 5 ms.

Palavras-chave: PID. GPC. myRIO. LabVIEW. Motor de corrente contínua.

ABSTRACT

This work presents the implementation of a cascade control structure, widely used in systems that have two control loops with different time constants and access to the intermediate signal. The proposed control system is implemented in a high performance hardware, myRIO, wich has a customizable field programmable gate array (FPGA) and a processor that works with a real-time operating system, guaranteeing reliability and determinism to the system. In the internal loop of the cascade structure, a proportional-integral controller (PI) implemented in FPGA is applied and, in the external loop, a generalized predictive controller (GPC) is implemented for execution in the real-time operating system of myRIO. The control structure was developed in a way that it can be applied to different monovariable systems, however a laboratory plant that consists of the control of speed and current of a direct current motor was used as a case study. The experimental results validated the implementation in both layers. The sampling period that was achieved in the implementation of the PI FPGA-embedded was less than 10 μs and the sampling period for GPC implementation in the real-time operating system was 13 ms (maximum observed value), with average execution period of less than 5 ms.

Keywords: PID. GPC. myRIO. LabVIEW. Direct current motor.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ACGPC	controle preditivo generalizado adaptativo em cascata - do inglês *adaptive cascade generalized predictive control*
A/D	analógico/digital
CARIMA	auto-regressivo com média móvel e integrador e entrada controlada - do inglês *controlled autoregressive integrated moving average*
CC	corrente contínua
D/A	digital/analógico
DMC	controle por matriz dinâmica - do inglês *dynamic matrix control*
E/S	entradas/saídas
IDE	ambiente de desenvolvimento integrado - do inglês *integrated development environment*
FPGA	arranjo de portas programável em campo - do inglês *field programmable gate array*
GPC	controle preditivo generalizado - do inglês *generalized predictive control*
HDL	linguagem de descrição de *hardware* - do inglês *hardware description language*
I^2C	*inter-integrated circuit*
MPC	controle preditivo baseado em modelo - do inglês *model predictive control*
MOSFET	transistor de efeito de campo de semicondutor de óxido metálico - do inglês *metal oxide semiconductor field effect transistor*
MSP	*minisystems port*
MXP	*myRIO expansion ports*
NI	*National Instruments*

OPC	controle preditivo ótimo - do inglês *optimum predictive control*
PC	*personal computer*
PID	proporcional-integral-derivativo
PFC	controle preditivo funcional - do inglês *predictive functional control*
PWM	modulação por largura de pulso - do inglês *pulse width modulation*
RMPCT	controle preditivo baseado em modelo robusto - do inglês *robust model predictive control technology*
RT	tempo real - do inglês *real-time*
RTOS	sistema de operação em tempo real - do inglês *real time operating system*
SOPC	*system-on-a-programmable-chip*
SPI	*serial peripheral interface*
Tg	representação do tacogerador
UART	*universal asynchronous receiver/transmitter*
USB	barramento serial universal - do inglês *universal serial bus*
VI	instrumentos virtuais - do inglês *virtual instruments*

LISTA DE SÍMBOLOS

Símbolos numéricos e latinos

$\vec{1}$	vetor com todos os elementos iguais a um
a	representa o valor do polo de $G_e(s)$
a_i	n-ésimo coeficiente de um polinômio associado com a dinâmica da planta
$\tilde{a}_i$	negativo do i-ésimo coeficiente do polinômio $\tilde{A}(z)$
A	matriz que determina o tipo de restrição
$A(z)$	polinômio referente ao denominador do modelo da planta
$\tilde{A}(z)$	polinômio associado com a dinâmica da planta, obtido pela multiplicação de Δ pelo polinômio $A(z)$
$\vec{b}$	vetor que determina o tipo de restrição
b_i	coeficiente do polinômio $B(z)$
$B(z)$	polinômio referente ao numerador do modelo da planta
$C(z^{-1})$	polinômio que define a função de transferência do modelo da perturbação no GPC
$C_e(s)$	controlador da malha externa
$C_i(s)$	controlador da malha interna
$C_{id}(s)$	representação ideal (acadêmica) de um controlador PID
$C_p(s)$	representação paralela de um controlador PID
$C_s(s)$	representação em série de um controlador PID
c_{med}	valor do sinal de controle médio na função objetivo
d	atraso de transporte em tempo discreto
D_{C_i}	denominador do controlador da malha interna
D_{G_i}	denominador do modelo da planta da malha interna

$e_e(s)$	erro entre a referência e o valor medido do sistema da malha externa
$e_i(s)$	erro entre a referência e o valor medido do sistema da malha interna
e_{med}	valor do erro médio na função objetivo
$\vec{f}$	vetor de resposta livre de um sistema
$f(k+j)$	elemento do vetor de resposta livre no instante $k+j$
$\vec{f_o}^T$	vetor de coeficientes do termo afim do funcional de um problema de programação quadrática
G	matriz dos coeficientes da resposta ao degrau de um sistema (em alguns casos é utilizado como ganho de um amplificador)
$G_e(s)$	modelo da planta da malha externa
$G_i(s)$	modelo da planta da malha interna
$G_\omega(s)$	modelo da relação entre o sinal PWM e a velocidade angular do motor
H	matriz Hessiana de um problema de programação quadrática representada por $2(G^T Q_\delta G + Q_\lambda)$ no caso do GPC
$H_i(s)$	função de transferência em malha fechada do sistema da malha interna
Hz	frequência (Hz)
I	matriz identidade (em alguns casos é utilizado como representação de corrente elétrica ou o ganho integrador de um controlador PID)
j	variável complexa (em alguns casos é utilizada como índice em somatórios, matrizes e vetores)
J	função objetivo a ser minimizada no GPC
k	tempo discreto (múltiplo do período de amostragem)
k_c	constante definida no intervalo $(2 \leq k_c \leq 4)$
K_c	constante proporcional de um controlador PID

K_p	ganho proporcional de um controlador PID
K_i	ganho integral de um controlador PID
K_d	ganho derivativo de um controlador PID
l	*offset* da função objetivo
m	ordem de coeficientes do denominador do modelo do sistema
n	ordem de coeficientes do numerador do modelo do sistema
n_b	número de *bits* de um conversor
N	tamanho do horizonte de predição
N_1	início do horizonte de predição
N_2	final do horizonte de predição
N_{C_i}	numerador do controlador da malha interna
N_G	número de dentes da engrenagem conectada ao eixo do gerador
N_{G_i}	numerador do modelo da planta da malha interna
N_M	número de dentes da engrenagem conectada ao eixo do motor
N_u	horizonte de controle
p	valor do polo desejado de malha fechada
Q_δ	matriz diagonal composta pelos termos de ponderação $\delta_{(j)}$
Q_λ	matriz diagonal composta pelos termos de ponderação $\lambda_{(j)}$
$R(s)$	transformada de Laplace do sinal de referência do sistema
R	resolução de uma conversão (em alguns casos é utilizado como representação de resistência elétrica)
R_G	resistência de um resistor responsável pelo ajuste do ganho de um amplificador
R_{SHUNT}	resistência de um resistor de precisão para medição de corrente elétrica
s	variável complexa (transformada de Laplace)

T	matriz triangular inferior de números 1
$t_{5\%}$	tempo de assentamento para que a resposta do sistema atinja a faixa de 95 % do seu valor final
t_a	tempo decorrido entre duas amostras
T_s	tempo de amostragem do sistema da malha externa
T_{s_i}	tempo de amostragem do sistema da malha interna
T_z	representação de matriz *Toeplitz*
$\underline{u}$	limite inferior do sinal de controle
$\overline{u}$	limite superior do sinal de controle
$\underline{U}$	limite inferior do sinal de controle com a restrição aplicada
$\overline{U}$	limite superior do sinal de controle com a restrição aplicada
$u_e(s)$	sinal de controle gerado por $C_e(s)$
$u_i(s)$	sinal de controle gerado por $C_i(s)$
$u(k)$	sinal de controle aplicada à planta no instante k
V	tensão elétrica
V_{REF}	tensão de referência de um amplificador
ω	velocidade angular do motor
$\vec{w}$	vetor de referências futuras do sistema
$w(k+j)$	sinal de referência futura no instante $k+j$
$\vec{y}$	vetor da saída predita do sistema
$Y_e(s)$	transformada de Laplace do sinal de saída do sistema de malha externa
$Y_i(s)$	transformada de Laplace do sistema de malha interna
$y(k)$	saída do sistema a ser controlado no instante k
$\hat{y}(k+j\|k)$	valor predito, no instante k, da saída do sistema em $k+j$
z	variável discreta (transformada Z)

z^{-d}	operador de atraso

Símbolos gregos e especiais

$(\cdot)^T$	matriz transposta de $(\cdot)$
$\delta_{(j)}$	ponderação do erro de seguimento de referência
Δ	operador diferença $(\Delta = 1 - z^{-1})$
$\Delta\vec{u}$	vetor dos incrementos de controle dentro do horizonte de controle
$\Delta u(k)$	incremento de controle no instante atual
ΔV	variação de tensão máxima do processo de conversão
$\lambda_{(j)}$	ponderação do esforço do incremento de controle
ω_G	velocidade angular do tacogerador CC
ω_M	velocidade angular do motor CC
ω_P	velocidade angular do potenciômetro
$\sigma(s)$	polinômio característico do sistema de malha fechada
τ	tempo de assentamento para que a resposta do sistema de primeira ordem atinja a faixa de $63,2$ % do seu valor final
$\xi(k)$	sinal de ruído branco com média nula

SUMÁRIO

1 INTRODUÇÃO

Existem duas possibilidades para implementação de controladores: digital e analógica. O controlador digital possui maior versatilidade e mais recursos em relação ao analógico, o que faz com que sua aceitação nos dias atuais seja muito superior à apresentada pelos controladores puramente analógicos (OGATA, 2011). Entretanto, toda essa melhoria traz limitações ao controlador em relação ao tempo de resposta. Um sistema de controle digital executa suas tarefas de forma sequencial, e cada uma dessas tarefas requer um certo tempo de processamento (SANTANA, 2006). Dessa forma, algoritmos de controle mais complexos, que apresentam grande número de tarefas, tendem a demorar mais para executar e isso faz com que o período de amostragem tenha que ser grande, o que limita o tempo de resposta do sistema em malha fechada.

Controladores preditivos baseados em modelo (MPC) são um exemplo típico de método de controle que sofre com esse problema. De acordo com Camacho e Bordons (1999), o MPC não é apenas uma estratégia, mas sim um conjunto de métodos de controle com características específicas, ou seja, os controladores preditivos baseados em modelo fazem o uso explícito de um modelo do processo a fim de predizer o comportamento futuro das variáveis de interesse e, então, calcular uma sequência de ações de controle a partir da minimização de uma determinada função objetivo. A limitação do MPC é que, a cada período de amostragem, um problema de otimização deve ser resolvido, gerando um atraso do tempo de cálculo do controlador preditivo, que é ainda maior quando o sistema possui alguma restrição, seja ela no sinal de controle ou na saída do processo. Assim, em sistemas que possuem dinâmicas muito rápidas pode ser inviável a aplicação de um controlador desse tipo (FLESCH, 2012).

Existem diferentes técnicas que permitem acelerar este tempo de cálculo, como é apresentado em (WANG; BOYD, 2008). Este artigo traz exemplos de métodos com a finalidade de resolver o problema de otimização do MPC de uma maneira mais rápida e eficaz, através da realização dos cálculos de forma *online*. Apesar de os resultados obtidos serem satisfatórios, mesmo com essas técnicas, a implementação de controladores preditivos tipicamente ainda está limitada a períodos de amostragem de milissegundos.

Em paralelo com as técnicas de cômputo rápido há o avanço do poder computacional, tornando claro que a Lei de Moore vem sendo respeitada há algumas décadas. De acordo com Morimoto (2005), Gordon Earl Moore, cofundador de uma das maiores empresas fabricantes de processadores do mundo (Intel Corporation), previu que o número de transistores em um processador, em média, dobraria a cada 18 meses mantendo o mesmo custo e espaço. A

partir disso, a indústria de semicondutores percebeu que era possível atingir a meta estipulada pela lei de Moore, passando a investir mais em pesquisa e desenvolvimento, resultando em uma evolução constante no que diz respeito à tecnologia computacional.

Para que se possa fazer uso dos benefícios oferecidos pelos controladores preditivos em plantas de dinâmica ainda mais rápida que alguns milissegundos, pensou-se em associar um controlador preditivo com controladores digitais que possam ser implementados em tempos compatíveis com as dinâmicas desejadas.

O controle em cascata é uma das estruturas mais populares para controle de processos, sendo muito utilizada em sistemas que possuem restrições e perturbações. Na maioria dos casos, a estrutura consiste em dois laços de controle, um interno e um externo, com diferentes constantes de tempo. Um cuidado a ser tomado em relação ao controle em cascata é que os dois controladores devem ser sintonizados em conjunto, já que a modificação de um parâmetro de um dos controladores irá influenciar diretamente sobre o outro (TAO; XIUYING, 2014).

Dessa forma, o controlador preditivo pode ser implementado na malha externa, rodando com períodos de alguns milissegundos, enquanto que na malha interna é empregado um controlador proporcional-integral-derivativo (PID). O PID é a estratégia mais frequentemente utilizada em arquiteturas em cascata, inclusive no âmbito industrial, devido a sua simplicidade de projeto e implementação, baixo custo e aplicabilidade em diversos tipos de sistemas (AKKAYA; AKBATI; GÖRGÜN, 2014). A fim de resolver um problema de controle em sistemas de dinâmica bastante rápida, existem *hardwares* dedicados que tornam esse cálculo mais rápido, como é o caso do FPGA (arranjo de portas programável em campo, do inglês *field programmable gate array*). Além disso, os controladores implementados em processadores de propósito geral podem não apresentar determinismo, característica que é atingida ao ser realizada a implementação em um sistema de tempo real, que garante a entrega do sinal de controle num prazo máximo estipulado e, consequentemente, garante o determinismo.

Neste trabalho, foi proposta uma estrutura em cascata em que, na malha interna, é aplicado um controlador proporcional-integral (PI) e, na malha externa, um controlador preditivo generalizado (GPC, do inglês *generalized predictive control*), que é responsável por definir as referências a serem aplicadas no sistema. Portanto, neste caso, a estrutura de controle em cascata permite levar o sistema à estabilidade e rejeitar perturbações através do controle PI em baixo nível e, em uma camada superior, a definição das referências que levem o sistema a apresentar um comportamento ótimo, através da aplicação do controlador preditivo. Ou seja, o controle em cascata une as vantagens

de um controlador PI, que possui um algoritmo mais simples e de rápida execução, ao controlador preditivo baseado em modelo, que permite que um comportamento ótimo seja atingido (MENGHUI; YIQUN; WEI, 2007).

O sistema de controle foi implementado em um *hardware* de alto desempenho desenvolvido pela empresa National Instruments, o myRIO, que é baseado em quatro componentes: um processador de tempo real, um FPGA, entradas e saídas e um *software* de projeto gráfico, o LabVIEW. O myRIO é capaz de garantir determinismo ao sistema, além de fornecer a vantagem de que toda a estratégia de controle em cascata pode ser implementada em apenas um *hardware*.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo deste trabalho é implementar uma estrutura de controle em cascata, em que a malha externa contitui-se de um controlador preditivo executado em um sistema de tempo real, que é responsável por definir os valores das referências para um controlador PI, na malha interna, implementado em um FPGA.

1.1.2 Objetivos Específicos

Dentre os objetivos específicos desta pesquisa, encontram-se:

- estudar técnicas de implementação de controladores PI em dispositivos FPGA;
- estudar técnicas de implementação de controladores preditivos em sistemas operacionais de tempo real;
- implementar os controladores propostos em sistemas embarcados;
- avaliar os resultados através de experiências em uma planta física/real..

1.2 DELIMITAÇÃO DO TRABALHO

A princípio, a estratégia de controle proposta neste trabalho não define delimitações para o processo real, mas sim para o modelo a ser utilizado, que

deve ser linear e invariante no tempo para a implementação dos controladores no sistema de tempo real e no FPGA. Caso o processo tenha características não lineares ou variantes no tempo, é possível obter um modelo dentro da delimitação do trabalho fazendo-se a linearização em torno de um ponto de operação da planta real, desde que a região de operação seja limitada à vizinhança desse ponto.

1.3 ESTRUTURA DO TRABALHO

O capítulo 2 explora os algoritmos de controle PID e GPC, além da estratégia de controle em cascata e a implementação desses controladores em *hardwares* de alto desempenho. A primeira seção apresenta os conceitos básicos que envolvem o ajuste do controlador PID, assim como alguns métodos de sintonia existentes na literatura. A segunda seção traz uma introdução do que é o controle preditivo e sua estrutura básica, além de apresentar uma introdução sobre a forma de implementação da função objetivo no caso da técnica de controle GPC, com estudos da literatura contendo o efeito de cada termo sobre o controlador e a sua representação matricial. Outro ponto importante tratado nessa seção diz respeito às restrições, que podem ser implementadas em conjunto com o controlador GPC. A terceira seção traz uma introdução sobre o controle em cascata, apresentando as vantagens dessa estratégia em relação ao controle em malha fechada convecional com uma malha, a sua arquitetura geral de implementação, além dos métodos de sintonia dos controladores e alguns trabalhos da literatura que implementam esta técnica. A quarta seção apresenta as vantagens da implementação de um controlador digital, assim como a sua realização por meio de *hardwares* de alto desempenho. O ponto principal dessa seção é a implementação de controladores em processadores que possuem um sistema operacional de tempo real, apresentando as principais vantagens em sua implementação a partir de trabalhos já desenvolvidos na área. A quinta e última seção traz um estudo sobre a implementação de controladores em FPGA, as vantagens em sua escolha no que diz respeito a tempo de processamento, assim como um estudo da literatura sobre os resultados obtidos com essa implementação.

O capítulo 3 apresenta os três principais materiais e métodos empregados neste trabalho. Primeiramente, é apresentado um estudo sobre a plataforma de desenvolvimento myRIO, trazendo a arquitetura do *hardware*, os seus principais elementos, formas de aplicação e comunicação com o restante do sistema, além das vantagens em sua utilização. A seguir, são mostrados os conceitos básicos envolvendo o ambiente de desenvolvimento LabVIEW, além de suas vantagens sobre outras linguagens de programação para o desenvolvimento

deste trabalho. Também são apresentados os conceitos básicos que envolvem a sua programação, assim como uma breve introdução sobre a sua integração com *targets* de *hardware*, através dos módulos LabVIEW Real-Time e LabVIEW FPGA. E, por fim, o capítulo trata do objeto de estudo de caso deste trabalho, o DIGIAC 710. Na terceira seção do capítulo são apresentados os seus componentes principais, assim como as suas respectivas características mecânicas e elétricas.

O capítulo 4 mostra a implementação da estratégia de controle proposta, apresentando a sua estrutura geral. Na seção 4.1, é exibido como foi realizada a programação no LabVIEW FPGA para a implementação de um controlador PI, assim como a aplicação de um controle GPC no LabVIEW Real-Time, ambos implementados para um caso geral. Na seção 4.2, são apresentados os esquemas de circuitos elétricos, necessários para a implementação do controle em cascata no estudo de caso escolhido (DIGIAC 710), que envolvem a medição da velocidade angular e da corrente, além do acionamento do motor. Por último, é mostrado o projeto da placa de circuito impresso.

O capítulo 5 traz os experimentos práticos realizados neste trabalho, assim como os resultados obtidos. Primeiramente, é mostrado como foram definidos os modelos dos processos, tanto da malha interna quanto da malha externa. Em seguida, são apresentados os métodos utilizados para o ajuste dos ganhos do controlador PI e os parâmetros do GPC. Então, são apresentados, por meio de gráficos, os resultados obtidos com a implementação dos controladores, verificando principalmente o seguimento da referência desejada e o cumprimento das retrições estabelecidas. A partir disso, são mostrados os resultados do sistema quando ocorrem pertubações, como variação de corrente ou inserção de carga no eixo do motor. Por fim, são apresentados os resultados que dizem respeito ao tempo de processamento do sistema.

O capítulo 6 apresenta as considerações finais e conclusões obtidas com o trabalho. Adicionalmente, são listadas propostas para trabalhos futuros, com o intuito de complementar ou aprimorar o que foi desenvolvido neste trabalho de mestrado.

2 REVISÃO DA LITERATURA

A engenharia de controle tem como preocupação entender e ser capaz de controlar materiais e forças da natureza, geralmente chamados de sistemas, a fim de beneficiar a sociedade. Segundo Dorf e Bishop (2001, p. 2), um sistema de controle pode ser definido como "uma interconexão de componentes formando uma configuração de sistema que produzirá uma resposta desejada do sistema".

Em um sistema de controle em malha fechada, a ação do controlador depende do comportamento da variável a ser controlada, realizando uma comparação entre o seu valor de referência e a saída real do processo. Isso é realizado através de um sinal de realimentação (*feedback*), gerando um sinal de erro. Portanto, com base no algoritmo de controle, o atuador realiza as modificações necessárias no processo a fim de reduzir ou, de preferência, eliminar esse erro (DORF; BISHOP, 2001).

Dentre os diversos algoritmos de controle existentes na literatura, dois deles serão explorados nesta revisão da literatura: controle proporcional-integral-derivativo (PID) e controle preditivo baseado em modelo (MPC, do inglês *model predictive control*).

2.1 CONTROLE PID

Na teoria de controle clássico, um algoritmo que tem demonstrado grande eficácia e praticidade na sua aplicação em processos industriais é o controlador PID (VIDAL et al., 2013). O ganho proporcional faz com que o controlador trabalhe proporcionalmente ao erro do sistema, ou seja, para a correção de erros maiores devem ser aplicados maiores sinais de controle e para os casos de erros pequenos devem ser aplicados sinais de controle de menor magnitude (CASTRUCCI; BITTAR; SALES, 2011). A ação integral está relacionada à redução para zero do erro em estado estacionário, resolvendo os problemas de resposta oscilatória, típicos em controladores liga-desliga, e do erro em regime permanente, associado a um controlador puramente proporcional (OGATA, 2011). A ação derivativa pode ser entendida como uma parcela que atua em uma predição do erro em instantes de tempo futuros (aproximação dada pela primeira derivada da função erro), configurando um grande potencial na melhoria do desempenho de controle. Entretanto, existem alguns aspectos negativos que tornam a sua aplicação menos frequente na prática, sendo o mais comum associado ao fato de que o ganho de alta frequência da ação derivativa pura é responsável pela amplificação do ruído de medição da variável

manipulada (VISIOLI, 2006). A fim de contornar este problema, tipicamente a ação derivativa é filtrada como forma de limitar o ganho em altas frequências.

Para que o processo responda de maneira desejada à ação de um controlador PID, é necessário a determinação dos valores dos ganhos Proporcional, Integral e Derivativo. De acordo com Visioli (2006), o controlador PID pode ser representado de diferentes maneiras, tais como ideal, série ou paralelo. Na forma ideal (também chamada de acadêmica), o controlador é descrito pela função de transferência

$$C_{id}(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right), \tag{2.1}$$

onde K_P representa a ação proporcional, T_i e T_d são as constantes de tempo integral e derivativa, respectivamente (SARAIVA, 2011). A representação em série, diferentemente da acadêmica, justificada pelo fato de o valor da constante de tempo derivativa T_d afetar a ação integral do controlador PID, pode tornar-se um problema em casos onde é necessário que haja independência entre as ações de controle. Entretanto, esta estrutura é mais fácil de aplicar no que diz respeito à sintonia manual do controlador (ÅSTRÖM; HÄGGLUND, 1995). Sua função de transferência é dada por

$$C_s(s) = K_p\left(1 + \frac{1}{T_i s}\right)(T_d s + 1). \tag{2.2}$$

Outra maneira de implementar um controlador PID é no formato paralelo, no qual as três ações de controle estão completamente separadas:

$$C_p(s) = K_p + \frac{K_i}{s} + K_d s, \tag{2.3}$$

onde K_i representa o ganho da ação integral e K_d o ganho do termo derivativo. Essa forma de representação do controlador PID, escolhida para aplicação no trabalho, é a mais geral dentre todas, já que ela permite desabilitar uma das ações sem que isso afete as demais (SARAIVA, 2011).

Existem diferentes técnicas de ajuste para encontrar os valores dos ganhos do controlador PID, que vão desde um ajuste manual até um projeto detalhado do controlador com base em um modelo da planta. Um método muito utilizado foi proposto por Ziegler e Nichols em 1940, principalmente para casos em que o modelo matemático do sistema a ser controlado é desconhecido, o que não impede a sua aplicação em sistemas com modelos conhecidos também. Entretanto, de acordo com Ogata (2011), como a proposta de sintonia de Ziegler e Nichols tem por objetivo atenuar rapidamente possíveis perturbações que sejam introduzidas no sistema, esse método pode resultar em

um controle com um sobressinal elevado no regime transitório de seguimentos de referência, tornando-se necessários novos ajustes para se obter uma resposta mais aceitável. Para tanto, existem diversas versões modificadas do método de Ziegler-Nichols na literatura, além de diversos outros métodos propostos por diferentes autores, como Cohen e Coon (1953), Tyreus e Luyben (1992) e Tan et al. (2012).

Outro método de sintonia de controladores PID é o Lugar Geométrico das Raízes, desenvolvido por Walter R. Evans em 1950. Segundo Ogata (2011), essa técnica determina o caminho percorrido pelos polos de malha fechada graficamente de acordo com a variação dos parâmetros do sistema, sendo considerado um método rápido para obtenção de respostas aproximadas. Apesar de esse método ser bastante geral e ter aplicação no projeto de diferentes estruturas de controle, ele pode ser também empregado para definição dos ganhos de controladores PID.

Na prática industrial, muitas vezes são empregados métodos manuais de ajuste para encontrar os ganhos do controlador, entretanto, existem alguns estudos mostrando que cerca de 85% das malhas de controle baseadas em algoritmo PID estão mal sintonizadas. Isso pode ocasionar diversos problemas, como o aumento no consumo de energia elétrica e desgaste de válvulas, resultando em manutenções mais frequentes, que fazem com que o processo seja interrompido com mais frequência, ocasionando a diminuição na qualidade do produto final (DILLENBURG, 2011).

Segundo Visioli (2006), podem ocorrer algumas situações em que a amplitude do sinal de controle irá saturar, permanecendo fora da região linear do atuador. Assim sendo, pode haver uma divergência entre o sinal calculado pelo controlador e o sinal efetivamente aplicado na planta, fenômeno conhecido como *windup*. De acordo com Neto (2005), a dinâmica em malha fechada ou até mesmo o desempenho do sistema podem ser deteriorados em consequência disso, já que o termo integral do controlador pode atingir valores elevados devido ao fato de que o erro do sistema continua a ser integrado. Isso acaba gerando um sobressinal elevado, pois o erro deve permanecer negativo por um período de tempo muito longo a fim de trazer o termo integral de volta ao estado estacionário. Para corrigir este problema, alguma técnica *anti-windup* deve ser aplicada ao controlador PID, a qual pode ter caráter linear ou não-linear.

De acordo com Trentelman e Willems (2012), qualquer sistema está sujeito a restrições do atuador, assim como muitas vezes ele deve satisfazer mais de um objetivo e, consequentemente, necessita operar em diferentes modos de controle. Se as condições de operação exigirem uma mudança de modo, um esquema de seleção utiliza aquele mais apropriado à situação. Em adição ao fenômeno *windup*, nesta comutação, a desigualdade entre as saídas

de diferentes controladores causa uma descontinuidade na entrada da planta do sistema, requerendo uma transição mais suave (*bumpless*) entre os diferentes modos de operação. Dessa forma, caso existam variações nos parâmetros ou na estrutura do controlador, é necessário também aplicar técnicas de transferência *bumpless*.

Nos últimos anos, processadores que realizam cálculos em alta velocidade estão sendo cada vez mais desenvolvidos, possibilitando o uso de sistemas de controle implementados de modo digital. Segundo Nise (2002), as vantagens do controle digital sobre o analógico podem ser divididas em três: redução de custos ao serem implementados algoritmos mais complexos, flexibilidade para modificação de parâmetros no projeto e um grau maior de imunidade a ruídos e interferências em relação ao controlador analógico.

2.2 CONTROLE PREDITIVO

Uma ampla variedade de estratégias de controle têm se tornado populares e muito eficientes desde o fim da década de setenta, devido ao fato de fazerem uso explícito de um modelo do processo a fim de obter o sinal de controle que consiga minimizar uma função objetivo. Essas estratégias fazem parte do que é denominado controle preditivo baseado em modelo (MPC) (CAMACHO; BORDONS, 1999).

De acordo com Camacho e Bordons (1999), todos os controladores preditivos baseados em modelo possuem três conceitos básicos:

- uso explícito de um modelo a fim de prever a saída do processo real em instantes futuros de tempo num determinado horizonte de predição;
- cálculo de uma sequência de controle que seja capaz de minimizar uma função objetivo;
- a cada intervalo de tempo, é gerado um conjunto de sinais de controle, porém o horizonte de controle é deslocado para o futuro e apenas o primeiro elemento da sequência é aplicado à entrada do sistema.

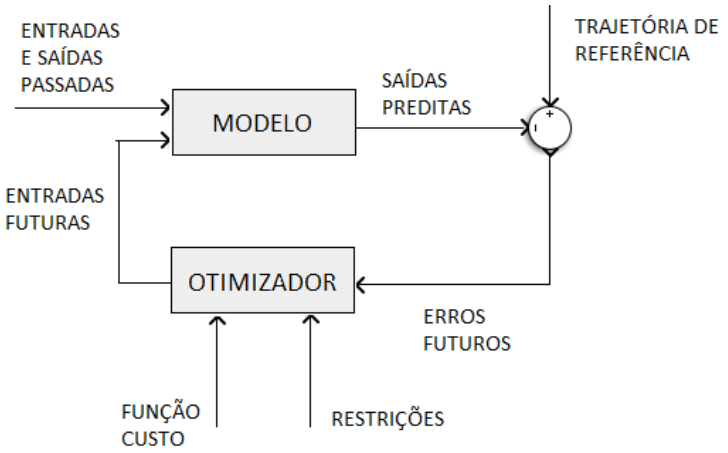
Atualmente, o controle preditivo, apesar de ter sido idealizado inicialmente para a indústria petroquímica, têm sido aplicado cada vez mais em outras áreas. Isso deve-se ao fato de que o controle preditivo baseado em modelo possui diversas características atrativas para aplicação (CAMACHO; BORDONS, 1999), tais como:

- pode ser aplicado em diversos processos, desde os mais simples até mesmo os mais complexos;

- é capaz de incluir um controle de pré-alimentação de perturbação (*feed forward*) quando a mesma pode ser medida;
- dependendo das necessidades do projeto, é possível adicionar explicitamente restrições para serem tratadas pelo controlador, que podem ser, por exemplo, no valor da saída do processo, no incremento de controle e no sinal de controle;
- é capaz de tratar de forma fácil referências futuras conhecidas para antecipar a resposta do sistema;
- possui uma metodologia baseada em alguns princípios básicos, mas que permite ao projetista estender a sua aplicação, se necessário.

A estrutura básica de um MPC (Figura 1) é formada por um modelo matemático do processo, a trajetória de referência que define qual será o comportamento futuro desejado da saída do processo, um preditor que fornece uma predição de como a saída se comportará tendo como base o seu valor atual e o modelo da planta, e um otimizador que minimiza uma função objetivo a cada instante de tempo, obtendo assim um sinal de controle que consiga garantir um desempenho desejado do sistema.

Figura 1 – Estrutura básica de um MPC.



Fonte: adaptado de Camacho e Bordons, 1999.

Atualmente, existem algumas técnicas de controle preditivo que possuem grande impacto no meio industrial, sendo consideradas as representantes

do atual estado da arte da tecnologia MPC (Camacho e Bordons, 1999), entre as quais é possível citar o controle por matriz dinâmica (*dynamic matrix control* - DMC) (Cutler e Ramaker, 1979), controle preditivo generalizado (*generalized predictive control* - GPC) (Clarke, Mohtadi e Tuffs, 1987a) (Clarke, Mohtadi e Tuffs, 1987b), controle preditivo funcional (*predictive functional control* - PFC) (Aström, Richalet e O'Donovan, 2009), controle preditivo ótimo (*optimum predictive control* - OPC) (Berlin e Frank, 1992) e tecnologia de controle preditivo baseado em modelo robusto (*robust model predictive control technology* - RMPCT) (Bemporad e Morari, 1999), apesar de existirem outras.

2.2.1 Controle Preditivo Generalizado

Um dos controladores mais difundidos na literatura é o GPC, proposto em (Clarke, Mohtadi e Tuffs, 1987a) e (Clarke, Mohtadi e Tuffs, 1987b), o qual tem se tornado um dos métodos MPC mais populares, tanto no âmbito acadêmico quanto no industrial, apresentando um bom desempenho e um certo grau de robustez. De acordo com Clarke, Mohtadi e Tuffs (1987c), esta técnica possui vantagens em relação às propostas anteriores de MPC, como o fato de poder ser aplicado a sistemas instáveis em malha aberta.

A diferença entre os algoritmos de controle preditivo está essencialmente no tipo de modelo utilizado e na forma da função objetivo. O controlador GPC representa um modelo do sistema na forma CARIMA (do inglês, *controlled auto-regressive integrating moving-average*), trazendo vantagens como a eliminação de *offsets* (CLARKE; MOHTADI; TUFFS, 1987a, 1987b). De acordo com Camacho e Bordons (1999), o modelo CARIMA é dado por:

$$A(z^{-1})y(k) = B(z^{-1})z^{-d}u(k-1) + C(z^{-1})\frac{\xi(k)}{\Delta}$$

ou, equivalentemente,

$$A(z^{-1})\Delta y(k) = B(z^{-1})z^{-d}\Delta u(k-1) + C(z^{-1})\xi(k) \tag{2.4}$$

com

$$\Delta = 1 - z^{-1} \tag{2.5}$$

garantindo uma lei de controle integral, onde $y(k)$ representa a sequência da saída do sistema a ser controlado, $u(k)$ é a sequência de controle aplicada à planta, $\xi(k)$ é um sinal de ruído branco com média nula e d representa o atraso

do sistema. Os polinômios $A(z^{-1})$, $B(z^{-1})$ e $C(z^{-1})$ são definidos como:

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_{na} z^{-na}$$

$$B(z^{-1}) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_{nb} z^{-nb}$$

$$C(z^{-1}) = 1 + c_1 z^{-1} + c_2 z^{-2} + \ldots + c_{nc} z^{-nc}$$

Geralmente, por simplicidade, o polinômio $C(z^{-1})$ é definido com um valor constante igual a 1 (CAMACHO; BORDONS, 1999). Com isso, o modelo CARIMA é empregado para calcular a predição da saída em um determinado horizonte de tempo à frente (LLOYD, 2011).

O cálculo da ação de controle é realizado através da minimização de um critério formado por uma função quadrática medindo o erro entre a saída predita do sistema e a trajetória de referência futura mais uma função quadrática medindo o esforço de controle ao longo de um horizonte de controle, que não precisa ser igual ao horizonte de predição, necessariamente. Matematicamente, tem-se o custo J expresso como:

$$J = \sum_{j=N_1}^{N_2} \delta_{(j)} [w(k+j) - \hat{y}(k+j|k)]^2 + \sum_{j=1}^{N_u} \lambda_{(j)} [\triangle u(k+j)]^2 \qquad (2.6)$$

onde:

- N_1 e N_2 são os valores mínimo e máximo, respectivamente, de horizonte de predição;
- N_u é o horizonte de controle;
- $\delta_{(j)}$ e $\lambda_{(j)}$ são ponderações escalares para um j definido em cada parcela da função objetivo;
- $\hat{y}(k+j|k)$ é o valor predito, no instante k, da saída do sistema no instante $(k+j)$;
- $w(k+j)$ é o valor da referência futura do sistema no instante $(k+j)$;
- $\triangle u(k+j)$ é o incremento de controle no instante $(k+j)$.

Em um controlador GPC existem cinco parâmetros de ajuste, sendo eles N_1, N_2, N_u, $\delta_{(j)}$ e $\lambda_{(j)}$. Apesar de ser difícil encontrar uma relação analítica entre os parâmetros de ajuste e as características da resposta para uma planta qualquer, os parâmetros de ajuste possuem singificado físico bastante claro e o projeto tipicamente é realizado com base nesse significado. Tipicamente, o

valor de N_1 é igual a 1, pois a ação de controle tomada no instante zero não influencia a saída instantaneamente, portanto o erro só pode ser observado no próximo intervalo de tempo. De acordo com Pereira (1997), se a escolha para N_1 for maior que 1 e a planta não tiver atraso, o processo de otimização deixa de levar em consideração os dados do início do comportamento transitório.

Entretanto, quando o sistema possui um atraso de transporte d, o início e fim do horizonte de predição passam a ser $N_1 = (d+1)$ e $N_2 = (d+N)$, já que a tomada de ações de controle no instante k terá efeito na saída somente após a decorrência do tempo $(d+1)$, ou seja, no instante $(k+d+1)$. Isso garante que não haja carga computacional adicional. Em relação aos horizontes de controle e predição, N_u e $N = (N_2 - N_1)$, existem vantagens que envolvem o processamento de dados quando N_u é menor, isto é, a complexidade de um problema irá depender do tamanho escolhido para o horizonte de controle (JAHAGIRDAR; DAN, 2015).

Para aplicação em sistemas estáveis em malha aberta, em alguns casos chega-se a empregar $N_u = 1$, apesar de ser mais comum o emprego de valores um pouco maiores, como forma de oferecer mais liberdade ao controlador considerando um horizonte de decisão maior. O horizonte de predição, por outro lado, tipicamente é escolhido de forma que seja observado todo o transitório e parte da resposta em regime permanente, ou seja, tipicamente escolhe-se um valor maior que o tempo de acomodação esperado para a resposta em malha fechada.

A determinação dos valores das ponderações escalares $\delta_{(j)}$ e $\lambda_{(j)}$ influencia diretamente na importância dada à minimização do erro e seguimento de referência ou do esforço de controle. Tipicamente, os valores de $\delta_{(j)}$ e $\lambda_{(j)}$ são constantes ao longo dos horizontes. Na prática, é comum que os valores de N e N_u sejam definidos em um primeiro momento em função das características do processo a ser controlado e, posteriormente, utilizar apenas $\delta_{(j)}$ e $\lambda_{(j)}$ como parâmetros de ajuste até que seja alcançada a resposta do sistema desejada.

De acordo com Rossiter (2003), o efeito de pequenas variações de $\lambda_{(j)}$ não é significativo quando N possui um valor elevado, já que a função custo é fortemente dominada pela parcela referente ao erro. Porém, quando o parâmetro N é definido como um valor menor, a ponderação $\lambda_{(j)}$ possui grande impacto na função, já que o custo não é mais dominado por erros de rastreamento. Tipicamente em casos monovariáveis, o parâmetro $\delta_{(j)}$ é assumido como possuindo valor unitário, já que o que influencia no valor do sinal de controle ótimo é a razão entre $\delta_{(j)}$ e $\lambda_{(j)}$, quando constantes ao longo dos horizontes.

Sendo $N = N_2 - N_1$, é possível escrever a função objetivo J em notação matricial, onde a referência futura $\vec{w} \in \mathbb{R}^N$, a saída predita $\vec{\hat{y}} \in \mathbb{R}^N$, o esforço de controle $\vec{\triangle u} \in \mathbb{R}^{N_u}$ e as matrizes de ponderação $Q_\delta \in \mathbb{R}^{N \times N}$ e $Q_\lambda \in \mathbb{R}^{N_u \times N_u}$

tornam-se

$$\vec{w} = \begin{bmatrix} w(k+N_1) \\ w(k+N_1+1) \\ \vdots \\ w(k+N_2) \end{bmatrix}, \ \vec{\hat{y}} = \begin{bmatrix} \hat{y}(k+N_1|k) \\ \hat{y}(k+N_1+1|k) \\ \vdots \\ \hat{y}(k+N_2|k) \end{bmatrix},$$

$$\triangle\vec{u} = \begin{bmatrix} \triangle u(k) \\ \triangle u(k+1) \\ \vdots \\ \triangle u(k+N_u-1) \end{bmatrix}, \ Q_\delta = \begin{bmatrix} \delta_{N_1} & 0 & 0 & 0 & \dots & 0 \\ 0 & \delta_{N_1+1} & 0 & 0 & \dots & 0 \\ 0 & 0 & \delta_{N_1+2} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \dots & \delta_{N_2} \end{bmatrix} \text{ e}$$

$$Q_\lambda = \begin{bmatrix} \lambda_0 & 0 & 0 & 0 & \dots & 0 \\ 0 & \lambda_1 & 0 & 0 & \dots & 0 \\ 0 & 0 & \lambda_2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \dots & \lambda_{N_u-1} \end{bmatrix}.$$

A saída predita do modelo do processo, $\vec{\hat{y}}$, é composta por duas respostas: forçada e livre. A primeira refere-se à parcela da resposta provocada pelas ações futuras de controle e a segunda resposta é constituída por elementos que não são influenciados por tais fatores, ou seja, como o processo se comportaria a partir do momento em que $\triangle u(k+j) = 0$, para todo e qualquer $j \geq 0$ (CAMACHO; BORDONS, 1999). Então:

$$\vec{\hat{y}} = G\triangle\vec{u} + \vec{f}, \tag{2.7}$$

onde

$$G = \begin{bmatrix} g_1 & 0 & 0 & \dots & 0 \\ g_2 & g_1 & 0 & \dots & 0 \\ g_3 & g_2 & g_1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ g_{N_2} & g_{N_2-1} & g_{N_2-2} & \dots & \dots \end{bmatrix} \quad \text{e} \quad \vec{f} = \begin{bmatrix} f(k+N_1) \\ f(k+N_1+1) \\ \vdots \\ f(k+N_2) \end{bmatrix},$$

em que g_i representa os valores das respostas do sistema a uma entrada do tipo degrau unitário i instantes de tempo depois de passado o atraso e $f(k+j)$ se refere à resposta livre no instante $k+j$. Portanto, como a saída predita do

sistema depende diretamente do sinal de controle, a função objetivo J torna-se

$$\begin{aligned} J &= (\vec{w}-\vec{\hat{y}})^T Q_\delta (\vec{w}-\vec{\hat{y}}) + \triangle\vec{u}^T Q_\lambda \triangle\vec{u} \\ &= (\vec{w}-G\triangle\vec{u}-\vec{f})^T Q_\delta (\vec{w}-G\triangle\vec{u}-\vec{f}) + \triangle\vec{u}^T Q_\lambda \triangle\vec{u}, \end{aligned} \tag{2.8}$$

a qual pode ser reescrita de forma a seguir os parâmetros impostos pela programação quadrática, que é definida por um problema de otimização com função objetivo quadrática, ou seja,

$$J = \frac{1}{2}\vec{x}^T H\vec{x} + \vec{f_o}^T\vec{x} + l. \tag{2.9}$$

Portanto, tendo como base a equação (2.9), fazendo-se os cálculos aritméticos necessários e reagrupando termos em comum na equação (2.8), tem-se que

$$J = \frac{1}{2}\triangle\vec{u}^T[2(G^T Q_\delta G + Q_\lambda)]\triangle\vec{u} + [2(\vec{f}-\vec{w})^T Q_\delta G]\triangle\vec{u} + l. \tag{2.10}$$

Por semelhança entre as equações (2.9) e (2.10), $H = 2(G^T Q_\delta G + Q_\lambda)$ na função objetivo da equação (2.8) e $\vec{f_o}^T = 2(\vec{f}-\vec{w})^T Q_\delta G$. O termo l, chamado de *offset* da função, pode ser descartado dos cálculos pelo fato de não influenciar em relação à obtenção do ponto mínimo. Por esse motivo não foi explicitamente desenvolvido na equação (2.10). Finalmente, a partir da equação (2.11), é possível minimizar a função objetivo:

$$J = \frac{1}{2}\triangle\vec{u}^T H\triangle\vec{u} + \vec{f_o}^T\triangle\vec{u}. \tag{2.11}$$

A partir deste cálculo, são obtidos os valores dos incrementos de controle futuros como solução do problema de programação quadrática. É importante salientar que somente o primeiro elemento do vetor $\triangle\vec{u}$ é aplicado no instante k e que este procedimento de cálculo é repetido a cada período de amostragem (DUTRA, 2003).

2.2.2 Restrições

Geralmente, o projeto de controladores tradicionais é realizado sem levar em consideração, de maneira explícita, alguma possível limitação nos sinais envolvidos. Entretanto, na prática, qualquer processo possui algum tipo de restrição, que pode ser de segurança, física, tecnológica, entre outras, dependendo da necessidade individual de cada sistema. Quando houver restrições no

processo, essas devem ser respeitadas pelo controlador, evitando que o sistema se comporte de modo indesejado em malha fechada (PEREIRA, 1997), podendo até mesmo ocasionar paradas emergenciais e, consequentemente, tempo não produtivo da planta (LIMA, 2013).

De acordo com Lima (2013), o uso de restrições aumenta a complexidade do cálculo de controle, um problema que pode surgir tanto no período transitório quanto em regime permanente. No primeiro caso, quando existirem perturbações ou uma grande mudança no valor de referência, o sistema torna-se incapaz de responder satisfatoriamente a tais circunstâncias, já que não é possível fazer com que a resposta retorne à região permitida durante um determinado período. No segundo caso, o sistema de controle torna-se infactível quando os objetivos são inexecutáveis (CAMACHO; BORDONS, 1999).

Existem três restrições mais comuns, representadas como restrições de desigualdade, em controladores GPC:

- restrição no incremento de controle, $\triangle u(k+j)$;
- restrição no sinal de controle, $u(k+j)$;
- restrição na variável de processo, $\hat{y}(k+j|k)$.

Para realizar o cômputo do sinal de controle em um controlador GPC, resolve-se um problema de programação quadrática, que deve ter restrições afins em função das variáveis de decisão. Dessa forma, todas as restrições devem ser expressas em função do incremento de controle (PEREIRA, 1997), como se segue:

$$A \triangle \vec{u} \leq \vec{b} \qquad \mapsto \qquad \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ \vdots \end{bmatrix} \triangle \vec{u} \leq \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \end{bmatrix}. \tag{2.12}$$

Em relação à restrição no incremento de controle, $\triangle u(k+j)$ deve permanecer entre um valor mínimo $\underline{u}$ e um valor máximo $\overline{u}$. Sendo assim, considerando-se a restrição em todo o horizonte de controle N_u, tem-se

$$\underline{u} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{N_u} \leq \begin{bmatrix} \triangle u(k) \\ \triangle u(k+1) \\ \vdots \\ \triangle u(k+N_u-1) \end{bmatrix} \leq \overline{u} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{N_u}.$$

Expressando a restrição em função de $\triangle\vec{u}$, tem-se que

$$\begin{bmatrix} I \\ -I \end{bmatrix} \triangle \vec{u}(k) \leq \begin{bmatrix} \vec{1}\overline{u} \\ -\vec{1}\underline{u} \end{bmatrix}, \tag{2.13}$$

onde I é uma matriz identidade e $\vec{1}$ é um vetor composto apenas por números 1 com dimensão N_u.

A partir do conceito apresentado anteriormente, ao aplicar-se uma restrição no sinal de controle $u(k+j)$, este deve permanecer entre um valor mínimo $\underline{U}$ e um máximo $\overline{U}$. Portanto,

$$\underline{U} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{N_u} \leq \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N_u-1) \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{N_u} \overline{U}.$$

Sabendo-se que o valor do sinal de controle atual é igual ao sinal de controle no instante anterior mais o incremento de controle atual, isto é,

$$u(k) = u(k-1) + \triangle u(k),$$

então

$$\underline{U}\ \vec{1} \leq \begin{bmatrix} u(k-1) + \triangle u(k) \\ u(k-1) + \triangle u(k) + \triangle u(k+1) \\ u(k-1) + \triangle u(k) + \triangle u(k+1) + \triangle u(k+2) \\ \vdots \end{bmatrix} \leq \vec{1}\ \overline{U}.$$

A fim de encontrar uma maneira de expressar essa restrição de acordo com a equação (2.12), isola-se $\triangle\vec{u}(k)$ na matriz central:

$$(\underline{U} - u(k-1))\ \vec{1} \leq \underbrace{\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ 1 & 1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}}_{T} \triangle\vec{u} \leq \vec{1}\ (\overline{U} - u(k-1)),$$

portanto

$$\begin{bmatrix} T \\ -T \end{bmatrix} \triangle \vec{u} \leq \begin{bmatrix} (\overline{U} - u(k-1))\ \vec{1} \\ -(\underline{U} - u(k-1))\ \vec{1} \end{bmatrix}, \tag{2.14}$$

onde T é uma matriz triangular inferior de números 1 e dimensão $N_u \times N_u$.

A terceira restrição de desigualdade a ser tratada é imposta no sinal de saída do sistema. Como não se tem acesso aos valores futuros da saída do sistema, empregam-se as predições, $\hat{y}(k+j)$, ou seja,

$$\underline{y} \leq \hat{y}(k+j) \leq \overline{y}, \quad \forall \quad N_1 \leq j \leq N_2.$$

Sabendo-se que

$$\vec{\hat{y}} = G \triangle \vec{u} + \vec{f},$$

então

$$\vec{1}\,\underline{y} \leq G \triangle \vec{u} + \vec{f} \leq \vec{1}\,\overline{y}.$$

Portanto, utilizando-se da formulação em (2.12),

$$\begin{bmatrix} G \\ -G \end{bmatrix} \triangle \vec{u} \leq \begin{bmatrix} \vec{1}\,\overline{y} - \vec{f} \\ -(\vec{1}\,\underline{y} - \vec{f}) \end{bmatrix}, \tag{2.15}$$

onde G é a matriz contendo os elementos da resposta ao degrau de dimensão $(N \times N_u)$ e, consequentemente, o vetor de números 1 passa a ter dimensão N.

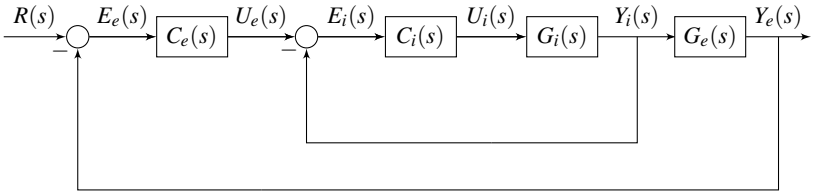
Se o modelo utilizado para representar o processo real não for construído de modo rigoroso, não há como garantir que essa restrição será obedecida pela planta. Uma restrição imposta sobre o sinal e incremento de controle pode ser garantida, já que há domínio total sobre as ações do controlador, ao contrário da saída do sistema, visto que neste caso é necessário confiar em sua predição.

2.3 CONTROLE EM CASCATA

Dentre as estruturas de controle em malha fechada, a mais convencional é composta por uma malha, que contém um controlador apenas. Porém, a sua desvantagem é que no caso de ocorrência de uma perturbação, o controle irá atuar sobre o sistema somente após a perturbação ser percebida na variável de processo. Em aplicações industriais é comum a implementação de um sistema em malha fechada com apenas um controlador, que consegue satisfatoriamente atender aos requisitos de projeto. Contudo, à medida que os requisitos do sistema passam a ser mais exigentes, o controle em malha fechada convencional é inaceitável, já que torna-se incapaz de atender a todas as necessidades do processo, reduzindo o seu desempenho consideravelmente em relação ao desempenho que pode ser obtido com o emprego de estruturas de controle mais complexas (SMITH; CORRIPIO, 1997). A arquitetura em cascata é um dos métodos mais importantes e eficientes para o desenvolvimento de um sistema

de controle, assim como para tratar perturbações. A estrutura básica de um controle em cascata é mostrada na Figura 2, sendo representada no domínio da transformada de Laplace.

Figura 2 – Arquitetura geral do controle em cascata.



Fonte: adaptado de Castrucci, Bittar e Sales, 2011.

Observando a Figura 2, que apresenta um sistema contínuo no tempo:

- $R(s)$ é a representação da referência do sistema (*set point*);
- $E_e(s)$ é o erro entre a referência e o valor medido do sistema da malha externa;
- $C_e(s)$ é o controlador da malha externa;
- $U_e(s)$ é o sinal de controle gerado por $C_e(s)$, mas também o valor de referência para o sistema da malha interna;
- $E_i(s)$ é o erro entre a referência e o valor medido do sistema da malha interna;
- $C_i(s)$ é o controlador da malha interna;
- $U_i(s)$ é o sinal de controle gerado por $C_i(s)$;
- $G_i(s)$ é a planta, ou seja, o modelo do sistema da malha interna;
- $Y_i(s)$ é a saída do sistema de malha interna;
- $G_e(s)$ é a planta do sistema da malha externa;
- $Y_e(s)$ é a saída do sistema de malha externa e, consequentemente, a saída total do sistema de controle em cascata.

Como é possível observar na Figura 2, a arquitetura em cascata é formada por duas malhas fechadas de controle, número que pode ser estendido dependendo das necessidades do projeto, chamadas de malha interna e malha externa. Segundo Smith e Corripio (1997), a consideração mais importante em um controle em cascata é que a malha interna deve ter o tempo de assentamento significativamente mais rápido que o da malha externa. Quanto maior for a diferença entre esses tempos melhor, visto que essa diferença entre as constantes de tempo das duas malhas permite o projeto separado dos controladores interno e externo.

O ajuste dos controladores tipicamente é realizado de dentro para fora, ou seja, o controlador da malha interna é o primeiro a ser projetado. Como o controlador interno é construído em cascata, esta acaba se tornando uma prática para checar o seu funcionamento antes de proceder para o projeto do controlador externo. O controlador da malha externa tem como planta toda a malha interna em série com a planta externa. Caso o controlador da malha interna tenha ação integral e tenha uma dinâmica muito mais rápida que a desejada para a malha externa, toda a malha interna pode ser aproximada por um ganho estático unitário, visto que o controlador da malha interna fará a saída seguir a referência que for passada pelo controlador externo (SMITH; CORRIPIO, 1997). Isso torna claro a eficácia de implementação de um controle em cascata, ou seja, quaisquer perturbações que possam ocorrer no processo da malha interna, serão efetivamente compensadas antes de afetarem o sistema por completo (VISIOLI, 2006).

Enquanto o controlador da malha interna está encarregado de levar a resposta do processo interno a um valor desejado e rejeitar perturbações rapidamente, o controlador da malha externa é responsável por controlar a planta externa, enquanto define os valores de referência para o laço interno, melhorando o desempenho do sistema.

A estratégia de controle em cascata é aplicada em diversas áreas do conhecimento que vão desde processos químicos a sistemas elétricos e mecânicos. Camacho e Bordons (2015) trazem um exemplo de aplicação de uma arquitetura em cascata em seu artigo, tendo como foco a aplicação de controle GPC em um processo dentro de uma fábrica de açúcar. Este caso combina as vantagens do MPC com a facilidade de implementação de controladores comumente encontrados na indústria. Devido ao fato de os requisitos computacionais serem simples, no caso descrito no artigo, dois algoritmos GPC puderam ser implementados como controladores em cascata. A aplicação proposta foi testada e implementada em uma indústria de açúcar, onde o objetivo principal era torná-la auto suficiente no que diz respeito ao consumo de água. Com base nisso, os autores Camacho e Bordons (2015) desenvolveram uma formulação GPC, fácil de implementar e fazer possíveis ajustes, que é válida

para grande parte dos processos industriais. Como a estratégia foi executada em plantas reais, em alguns casos, viu-se a necessidade de modificações no algoritmo original a fim de atender a todas as incertezas do processos. Com isso, os autores deixaram claro que é possível utilizar, de maneira simples e robusta, uma estrutura em cascata implementando controladores preditivos baseados em modelo, nesse caso, o GPC.

Já em Xiuying et al. (2016) é proposto um novo método de controle, definido como Controle Preditivo Generalizado Adaptativo em Cascata (do inglês, *Adaptive Cascade Generalized Predictive Control* - ACGPC) que, segundo os autores, é mais simples e muito mais eficaz. O ACGPC é uma estratégia baseada em GPC, onde os controladores interno e externo são substituídos por um controlador preditivo generalizado em cascata que pode se adaptar ao longo do tempo. No artigo, são apresentados dois exemplos de implementação dessa estratégia a fim de comprovar que todas as propriedades vantajosas de um controle em cascata estão mantidas nesse método e que os controladores podem ser auto ajustáveis, característica interessante para aplicações industriais nas quais é necessária uma resposta rápida do sistema de controle e existe variação do modelo ao longo do tempo ou do ponto de operação.

A estrutura de controle em cascata também pode ser aplicada a sistemas que possuem atrasos de transporte, como é o caso apresentado em (Padhan e Reddy, 2016). O artigo sugere uma estrutura em cascata contendo um esquema com um filtro compensador de tempo morto incorporado à malha externa do sistema. O critério de estabilidade de Routh Hurwitz é empregado para projetar o controlador da malha externa, enquanto que o controle da malha interna é desenvolvido por aproximações. Os resultados apresentados foram satisfatórios, demonstrando bom desempenho no caso nominal e robustez no caso da variação paramétrica.

Uma das principais áreas de aplicação da estratégia de controle em cascata é o controle de sistemas elétricos. O artigo de Garcia, Rodriguez e Silva (2016) valida experimentalmente um novo esquema de controle para sistemas de acionamento elétrico, nomeado como controle preditivo em cascata de velocidade e corrente, implementando também um controlador preditivo baseado em modelo nas duas malhas. Nesse caso, os autores realizam o controle da velocidade de um motor de corrente alternada utilizando um inversor de frequência, sendo que a malha interna controla a corrente do estator, enquanto que a malha externa controla a velocidade. Os resultados experimentais mostraram que a estratégia proposta forneceu um desempenho ao sistema que é comparável às estratégias de controle clássicas, mas que está livre de sobrecarga e proporciona uma melhor resposta temporal. Eles concluíram que esta estratégia possui um bom desempenho para muitas aplicações, mas passa a ter

algumas limitações de largura de banda que podem se tornar um problema em casos nos quais a resposta dinâmica é considerada um requisito crítico.

2.4 IMPLEMENTAÇÃO DE CONTROLADORES EM SISTEMAS OPERACIONAIS DE TEMPO REAL

Existem duas maneiras de implementar um controlador em um sistema em malha fechada: analógica e digital. A vantagem mais importante que o controle digital possui sobre o analógico é a flexibilidade em realizar modificações na lei de controle mesmo após o *hardware* projetado ter sido fixado, já que o cálculo do sistema de controle é realizado no *software*, significando uma otimização no tempo de processamento. Para o controle de plantas contínuas no tempo, o controlador digital faz uso de conversores A/D (analógico/digital) e D/A (digital/analógico), tornando o sistema em malha fechada híbrido, ou seja, contendo tanto sinais analógicos quanto digitais (FRANKLIN; POWELL; EMAMI-NAEINI, 2002).

A implementação de um controlador digital pode ser realizada por meio da utilização de diferentes ferramentas. Nos últimos anos, os *hardwares* de alto desempenho avançaram de modo significativo, acompanhando a própria evolução dos microcontroladores. Com isso, novos *hardwares* surgem com algumas vantagens, como o oferecimento de sistemas operacionais de tempo real e a possibilidade de implementação do controle em um único circuito integrado, através do uso de FPGAs (SANTANA, 2006).

De acordo com Tekin (2010), um sistema em tempo real é aquele em que a melhoria de uma resposta não depende apenas da correção dos cálculos realizados, mas também da entrega do sinal com um limite de tempo máximo predefinido. Existem diferentes maneiras de medir o desempenho e robustez do sistema de controle, como o tempo de ciclo do laço de controle, o determinismo e o *jitter*. Este último, geralmente utilizado em sistemas de controle, segundo Yamamoto e Azevedo (2013, p. 2), "é a variação do tempo real do ciclo do *loop* e do tempo desejado". Em sistemas operacionais de propósito geral, a estabilidade do sistema de controle não pode ser garantida, já que o *jitter* é ilimitado, entretanto, em um sistema de controle implementado em um processador de tempo real, é possível garantir um *jitter* do laço de controle menor que 100 milissegundos (YAMAMOTO; AZEVEDO, 2013).

É comum associar o desempenho de tempo real com o aumento da velocidade de execução de um programa, já que esse tipo de processamento fornece uma temporização mais precisa e previsível. Na maioria das vezes, as aplicações de tempo real são embarcadas, incorporando características de temporização limitada e atuando como um *hardware* dedicado (YAMAMOTO;

AZEVEDO, 2013). A definição de sistemas em tempo real para a implementação de controladores digitais vem sendo muito estudada e aplicada, devido às vantagens fornecidas por esse tipo de prática.

Um exemplo disso é apresentado pelos autores Sigarev, Kuzmina e Krasilnikov (2016), que propõem um sistema de controle em tempo real aplicado a um motor de corrente contínua. O sistema de controle foi dividido em duas partes: o projeto do motor e do *hardware* utilizado e o desenvolvimento do controle implementado através de uma ferramenta (*toolbox*) disponível no *software* MATLAB, Matlab Real-Time Windows Target Toolbox, que permite testar, simular e ajustar parâmetros em tempo real de um sistema. Com isso, os autores verificaram que o modelo do motor utilizado para a simulação do sistema de controle em tempo real produzia uma dinâmica de resposta muito parecida com a implementação do controlador no sistema real, concluindo que essa arquitetura fornece a habilidade de construir e analisar qualquer algoritmo de controle implementado no MATLAB.

A implementração de controladores em sistemas de tempo real traz uma vantagem importante no que diz respeito a uma economia significativa no tempo de execução da atuação do controlador sobre o processo, tornando-se capaz de atender até mesmo a requisitos mais críticos de tempo. O *software* geralmente utilizado por projetistas para desenvolver um sistema de controle é o MATLAB, o qual possui diversas opções que auxiliam na simulação de processos. Entretanto, o *software* LabVIEW possui conjuntos de ferramentas (*toolkits*) que vêm facilitando implementações em tempo real, por exemplo, sendo muito empregado no âmbito industrial (TEKIN, 2010).

Como exemplo, em (Liu, Zhang e Wang, 2009) é implementado um controlador PID da posição de um servo motor no *software* LabVIEW, aproveitando-se dos recursos de um módulo para sistemas em tempo real chamado Real-Time (RT). Todo o código de programação é desenvolvido no ambiente LabVIEW RT e, em seguida, é carregado para executar aplicações embarcadas em um dispositivo alvo. Esta implementação traz benefícios em relação ao tipo de programação que, diferentemente do MATLAB, no LabVIEW é realizada através de diagramas de blocos, tornando-a mais intuitiva e sem necessidade de obter conhecimento mais avançado sobre programação. A escolha entre os dois *softwares* depende dos requisitos do sistema a ser implementado, assim como a preferência do próprio projetista.

O artigo de Tekin (2010) apresenta uma comparação do uso dos dois *softwares* MATLAB e LabVIEW para o desenvolvimento de um controle de movimento. Primeiramente, foi encontrado o modelo matemático do sistema e projetado um controlador implementado no MATLAB e, em seguida, foi definido um *hardware*, o CompactRIO, adequado para aplicações industriais. Depois, foi implementado um controle em tempo real do sistema utilizando-se

do *toolkit* já mencionado anteriormente, LabVIEW RT, e para tratamento dos dados recebidos foi utilizado o módulo LabVIEW FPGA. O autor Tekin (2010) conclui o trabalho reafirmando o fato de que o uso do MATLAB ou LabVIEW depende da aplicação desejada. Neste caso, o MATLAB foi designado para modelar, projetar o controlador e analisar o sistema, enquanto que os módulos LabVIEW RT e FPGA eram os mais apropriados para uma aplicação em tempo real, já que o LabVIEW é compatível com diversos *hardwares* em muitas aplicações, assim como possui uma *interface* com o usuário mais amigável. Além disso, dependendo do *hardware* utilizado, módulos alternativos podem ser adicionados ao LabVIEW, aumentando ainda mais o número de aplicações possíveis.

2.5 IMPLEMENTAÇÃO DE CONTROLADORES EM FPGA

O FPGA foi desenvolvido pela primeira vez na década de 1980 pela empresa *Xilinx*, sendo que a sua arquitetura depende de três fatores fundamentais: tecnologia de programação, arquitetura de blocos lógicos e arquitetura de roteamento (QUEIROZ, 2005). De acordo com Yamamoto e Azevedo (2013), um FPGA é composto por uma matriz de circuitos lógicos contendo um arranjo de portas reconfiguráveis que utiliza um *hardware* dedicado para lógica de processamento, não possuindo sistema operacional.

O método de processamento implementado por um FPGA é realizado de forma paralela, ou seja, ele possui a capacidade de executar mais de um processo ao mesmo tempo e com uma velocidade relativamente alta se comparado aos demais *hardwares*, fazendo com que a funcionalidade do sistema aumente significativamente (SANTANA, 2006). Por conta disso, a velocidade de execução de programas paralelos em um FPGA não é influenciada pelo número de processos.

Segundo Bezerra (2010), o projeto de um controlador digital implementado em FPGA fornece praticidade, ampla capacidade de armazenamento, menor consumo de energia e redução de custos, permitindo ao projetista a possibilidade de futuras modificações no sistema, já que ele permite a reconfiguração dos seus circuitos internos, trazendo a confiabilidade de um *hardware* dedicado. Os FPGAs vêm sendo utilizados na implementação de qualquer sistema de controle industrial, como controle de processos de dinâmica contínua no tempo, de lógica discreta ou máquinas de estado.

Como exemplo, o trabalho apresentado pelos autores Mekonnen, Katcha e Parker (2012) traz a modelagem de um sistema eletrônico de potência e um controle digital baseado em FPGA com uma alternativa ao HDL (do inglês, *Hardware Description Language*), linguagem de programação que geralmente

é utilizada em FPGA, mas que acaba sendo muito complexa, criando uma barreira para o desenvolvedor. Foi utilizada uma ferramenta fornecida pela empresa Altera, chamada DSP Builder, que é acessada através do ambiente de desenvolvimento Simulink/MATLAB. A compilação do código faz com que o próprio programa gere um arquivo em HDL para este ser enviado ao FPGA. Os autores apresentaram bons resultados com a implementação de um controlador baseado em FPGA, garantindo uma operação paralela, assim como a isolação funcional das ações, consequentemente, diminuindo o tempo de processamento dos dados.

O artigo de Akkaya, Akbati e Görgün (2014) traz a aplicação de um controle PID digital baseado em um FPGA, o Altera De0 Nano, utilizando-se da linguagem HDL para isso. Para a implementação prática foi escolhido um robô de cinco eixos, a fim de comprovar que o FPGA pode ser uma solução flexível para esse tipo de sistema.

Em (Ali, Hussein e Ismael, 2010) é mostrada uma nova metodologia de projeto de controladores digitais, onde o módulo LabVIEW FPGA é usado para a implementação do sistema, que inclui a captura de valores do mundo real, ou seja, a variável de processo e o valor da referência, além de um controlador PID e a geração de um sinal PWM para controlar um motor. O módulo LabVIEW FPGA, desenvolvido pela empresa National Instruments, usa a tecnologia embarcada do LabVIEW para extender o desenvolvimento gráfico para *hardwares* FPGA, permitindo ao usuário a criação de um *hardware* customizado sem utilizar-se da linguagem HDL, execução de múltiplas tarefas de modo simultâneo e determinístico, solução de diversas aplicações, controle em alta velocidade, processamento digital de sinais e qualquer outra aplicação que exiga alta confiabilidade do *hardware* e determinismo do processo.

Além da alta confiabilidade fornecida por um FPGA, é possível desenvolver controles determinísticos de malha fechada a uma taxa muito alta, podendo exceder o valor de 1 MHz, já que a frequência de execução em um sistema baseado em FPGA não é limitada pelo seu processamento, mas por outros fatores, como sensores e atuadores (YAMAMOTO; AZEVEDO, 2013). Apesar de existir necessidade de um *clock* externo em um circuito com FPGA, tipicamente ele não representa um limitante, pois é comum o emprego de ciclos de execução de nanossegundos ou menos.

Ainda de acordo com Yamamoto e Azevedo (2013), o laço de controle não compartilha os recursos de *hardware* com outras tarefas em uma aplicação implementada em FPGA, possibilitando temporização de maior precisão. É fato que o *jitter* em um FPGA depende da precisão da sua fonte de *clock*, por exemplo, num FPGA com um *clock* de 40 MHz o *jitter* fica na faixa de picossegundos.

2.6 COMENTÁRIOS FINAIS

Neste capítulo foram abordados os algoritmos de controle PID e GPC. Em relação ao PID, foram apresentadas as suas características e métodos de sintonia. No caso do controlador GPC, foram tratadas as principais diferenças e vantagens relacionadas às demais técnicas de controle preditivo baseado em modelo. Também foram apresentados os parâmetros de ajuste do GPC e quais aspectos devem ser levados em consideração no projeto do controlador, além das três restrições mais comuns em controladores GPC. Em seguida, foi abordada a estrutura de controle em cascata e suas principais vantagens em aplicações, além dos aspectos que devem ser levados em consideração para o ajuste dos controladores. Sabendo-se que existem duas formas de implementar um controlador em um sistema em malha fechada (analógica e digital), foram apresentados dois tópicos com o objetivo de tratar os ganhos que podem ser alcançados, relacionados principalmente a tempo de processamento, ao aplicar um controlador em sistemas operacionais de tempo real ou em um FPGA. Os conceitos apresentados neste capítulo serão utilizados como base para a implementação de um controlador PI em FPGA e um GPC em um sistema operacional de tempo real. O capítulo 3 apresenta as ferramentas utilizadas para essa implementação, assim como o sistema escolhido como estudo de caso para sua validação experimental.

3 MATERIAIS E MÉTODOS EMPREGADOS

Placas de aquisição de dados multifunção tipicamente são compostas por diversos elementos, tais como entradas e saídas analógicas e digitais, contadores e temporizadores, que têm por finalidade ler e escrever valores de sinais com características analógicas do mundo real. Além dos elementos que implementam as entradas e saídas, há um *hardware* que é responsável pelo gerenciamento das aquisições e escritas de dados. Entretanto, as placas multifunção convencionais apresentam o que é chamado de *hardware* fixo, ou seja, que não pode ser reconfigurado pelo usuário, trazendo consigo desvantagens relacionadas a desempenho e flexibilidade (DINIZ, 2008).

Na seção 3.1 é apresentada uma forma de solução para esses problemas a partir da plataforma de desenvolvimento myRIO, assim como as suas principais características de arquitetura e entradas/saídas disponíveis. Na seção 3.2 é apresentada uma visão geral sobre o ambiente de desenvolvimento gráfico LabVIEW, além de uma introdução sobre as vantagens de implementação utilizando-se dos módulos compatíveis ao myRIO. Por fim, a seção 3.3 traz as principais características da planta de laboratório escolhida como estudo de caso neste trabalho, o DIGIAC 710.

3.1 PLATAFORMA DE DESENVOLVIMENTO MYRIO

Nas últimas décadas, surgiram placas de aquisição reconfiguráveis combinando a velocidade do *hardware* com a flexibilidade do *software*. A tecnologia de uma plataforma reconfigurável permite que a sua arquitetura seja modificada em tempo real para melhor se adequar à aplicação que será implementada, possibilitando ao usuário escrever o código que gerenciará a aquisição (COELHO; FERTIG; FERTIG, 2016). Exemplos dessas plataformas reconfiguráveis são os dispositivos RIO multifuncionais da série R da empresa National Instruments (NATIONAL INSTRUMENTS, 2016c), que oferecem um FPGA de alto desempenho para controle imediato de todas a entradas/saídas (E/S), e alguns *hardwares* da fabricante dSpace (DSPACE, 2016).

De acordo com Diniz (2008), essas arquiteturas podem ser híbridas, que utilizam os modelos de *hardware* fixo em conjunto com modelos de *hardware* reconfigurável, ou puramente reconfiguráveis, com a finalidade de empregar seu próprio sistema operacional e um FPGA programável para gerenciamento das aquisições de dados. No contexto dos FPGAs essa abordagem recebe o nome de *System-on-a-Programmable-Chip* (SOPC), caracterizando-se por possuir todos os recursos de *software* e *hardware* em um único *chip* programável,

fornecendo uma programação paralela na execução das funções (COELHO; FERTIG; FERTIG, 2016). Isso significa que, em um dispositivo puramente reconfigurável, é possível usar as portas de E/S para gerar um sinal PWM (do inglês, *Pulse Width Modulation*) ou ler dados em um protocolo de comunicação serial, por exemplo. Isso é inviável em uma placa multifunção de *hardware* fixo, visto que todo o processamento deve ser realizado por *software* e os tempos de comunicação entre o *software* escrito pelo usuário e o *hardware* inviabilizam operações em altas frequências, como as exigidas para geração de um sinal PWM ou emulação de um nó em uma comunicação serial.

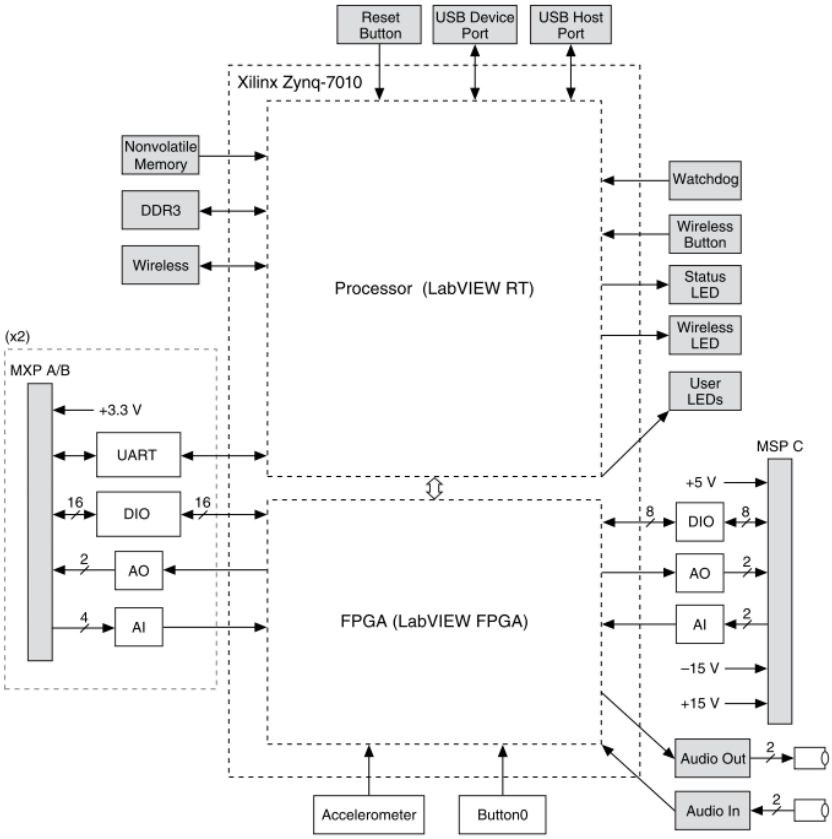
O myRIO, desenvolvido pela National Instruments, é um exemplo desses sistemas reconfiguráveis, que foi criado com o objetivo de auxiliar estudantes de engenharia a desenvolver projetos complexos com maior rapidez e facilidade (NATIONAL INSTRUMENTS, 2016b). O NI myRIO é composto pelo *chip* Zynq-7010 da fabricante Xilinx, que possui um processador ARM Cortex-A9 *dual-core* totalmente programável, que trabalha com um sistema operacional de tempo real, e um FPGA customizável. O myRIO funciona de forma que o FPGA é responsável pelo gerenciamento de E/S, permitindo programação para tratá-las, enquanto que o sistema de tempo real, que possui comunicação com o FPGA, também apresenta uma *interface* de comunicação com o *software* de programação. A Figura 3 apresenta a arquitetura do *hardware* do myRIO.

A conexão entre o myRIO e o computador pode ser realizada através de USB 2.0 (do inglês *Universal Serial Bus*) ou por comunicação sem fio com protocolo IEEE 802.11 b,g,n. Como é possível observar na Figura 3, o NI myRIO fornece alimentação em níveis de $3,3$ V e 5 V, 10 entradas e 6 saídas analógicas, 40 entradas/saídas digitais, conectores de áudio, um acelerômetro de três eixos, além de alguns LEDs e botões para pequenos testes, tudo incluso em um único dispositivo embarcado. Também há alguns protocolos de comunicação disponíveis no *firmware* fornecido de fábrica, que são UART (do inglês *Universal Asynchronous Receiver/Transmitter*), SPI (do inglês *Serial Peripheral Interface*) e I^2C (do inglês *Inter-Integrated Circuit*), porém novos protocolos podem ser adicionados através da programação do FPGA. A Figura 4 apresenta a visão geral do NI myRIO.

O módulo NI myRIO possui três conectores (A, B e C), sendo que os dois primeiros são idênticos e levam o nome de MXP A e MXP B (do inglês, *myRIO Expansion Ports*) e o terceiro é chamado de MSP C (do inglês, *miniSystems port*). Essas portas são responsáveis por receber e enviar sinais a circuitos externos associados a elas (NATIONAL INSTRUMENTS, 2013). A Figura 5 mostra os sinais dos conectores MXP A e B, onde é possível notar que alguns pinos carregam funções primárias e secundárias.

Nesses conectores estão presentes as alimentações de $3,3$ V e 5 V,

Figura 3 – Arquitetura do *hardware* do NI myRIO.

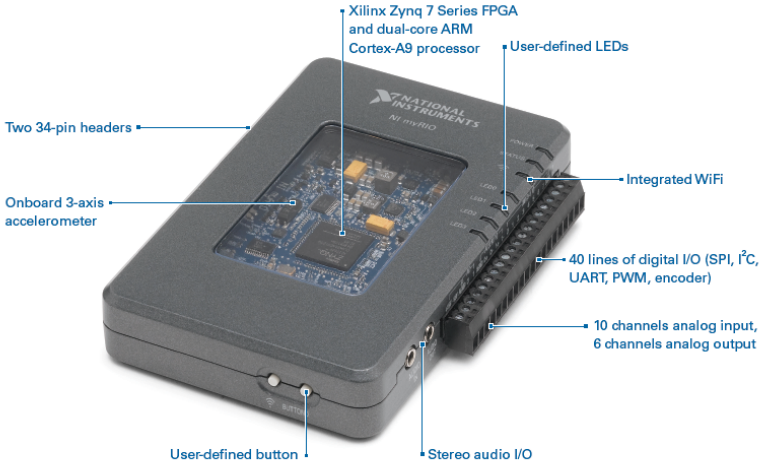


Fonte: NATIONAL INSTRUMENTS, 2014.

além de 4 entradas e 2 saídas analógicas por conector, que podem ser usadas para medir sinais de (0 a 5) V. Também encontram-se disponíveis 16 canais de entradas/saídas digitais de propósito geral por conector com tensões de 3,3 V para as saídas e 3,3 V a 5 V para as entradas, sendo que as funções digitais secundárias desses canais incluem comunicação SPI, I^2C e PWM de até 100 kHz, entre outros. Os conectores MXP A e B também possuem o sinal de referência para os pinos de alimentação e os sinais digitais.

A Figura 6 apresenta os sinais referentes ao conector MSP C, que

Figura 4 – Visão geral do NI myRIO.

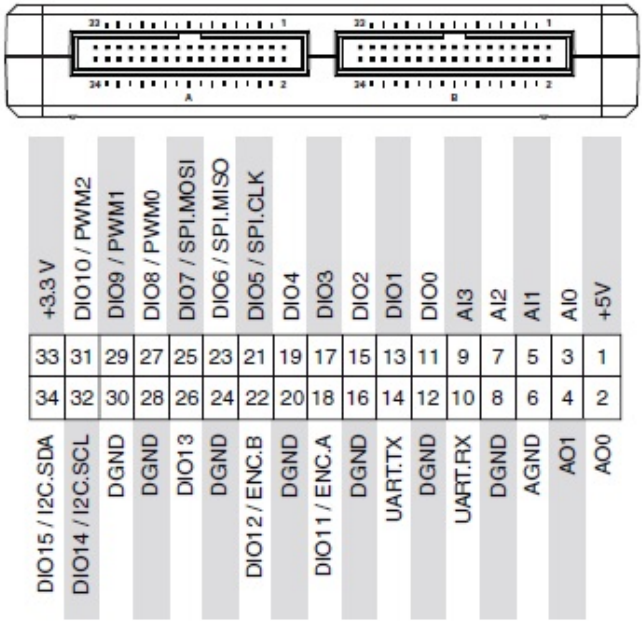


Fonte: NATIONAL INSTRUMENTS, 2014.

oferece opções adicionais ao myRIO, como 2 entradas e 2 saídas analógicas, 8 entradas/saídas digitais, além de alimentação em diferentes níveis. As entradas analógicas nesse conector possuem níveis diferenciais de medição de sinais no valor de ± 10 V, assim como as saídas são capazes de gerar tensões também de ± 10 V. As entradas/saídas de propósito geral do MSP C possuem sinais de $3,3$ V para as saídas e entradas compatíveis com níveis de tensão nos padrões digitais e $3,3$ V e 5 V. O conector MSP C ainda oferece pinos de alimentação de 5 V e ± 15 V, assim como dois canais de referência, sendo que o primeiro é para as entradas e saídas analógicas e o sinal de ± 15 V, enquanto que o segundo serve para as linhas digitais e o pino de 5 V. Esse conector também disponibiliza sinais secundários de PWM.

Os dispositivos da família Zynq-7000 possuem em seu encapsulamento conversores D/A (digital/analógico) dedicados para cada saída analógica do NI myRIO, tornando possível a atualização de valores dessas saídas simultânea, sendo que a máxima frequência de geração é de 345 kS/s. O Zynq também integra dois conversores A/D (analógico/digital) e um multiplexador interno, que suporta até 17 canais de entradas analógicas. Com isso, as entradas analógicas do myRIO são multiplexadas para um conversor A/D, responsável

Figura 5 – Conectores MXP A e MXP B.

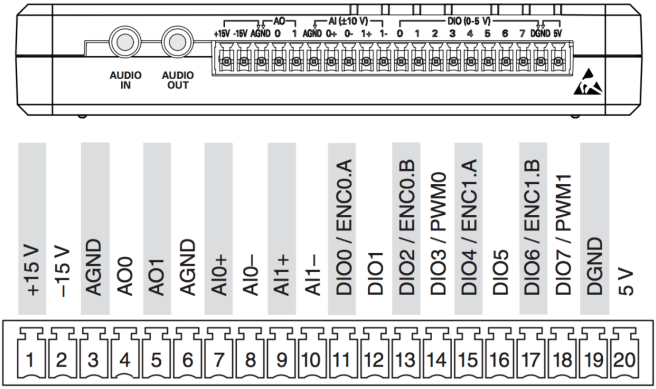


Fonte: NATIONAL INSTRUMENTS, 2013.

por amostrá-las a uma máxima frequência de amostragem no valor de 500 kS/s, sendo que os dois conversores podem ser configurados para amostrar dois canais de entrada simultaneamente. Quando um conversor A/D é empregado para realizar a aquisição de mais de um canal, a taxa máxima de aquisição de cada canal é razão da taxa máxima quando apenas um canal é utilizado pelo número de canais empregados, ou seja, a taxa agregada máxima de aquisição é de 500 kHz.

Cada conversor A/D do NI myRIO possui uma resolução de 12 *bits*, utilizada para a conversão de valores de tensões aplicadas em cada um dos canais para valores digitais correspondentes. Por exemplo, para os canais de entradas analógicas dos conectores MXP A e B, em que a variação de tensão é de 0 V a 5 V, o valor da resolução é de $R = 1,221$ mV. A resolução pode ser determinada através do emprego da equação (3.1), onde ΔV é a variação de tensão máxima do processo de conversão A/D e n_b é o número de *bits* do

Figura 6 – Conector MSP C.



Fonte: NATIONAL INSTRUMENTS, 2013.

conversor.

$$\mathrm{R} = \frac{\Delta V}{2^{n_b} - 1} \tag{3.1}$$

A resolução dos conversores D/A também é de 12 *bits*, portanto a resolução do sinal gerado por esses conversores também é de $1,221$ mV.

Existem diversas aplicações que podem se beneficar dos seus recursos, dentre as quais é possível citar sistemas de controle, mecatrônica, robótica e sistemas embarcados, além de existirem inúmeras outras (NATIONAL INSTRUMENTS, 2015a). O dispositivo myRIO pode ser programado em C/C++, na IDE (do inglês, *Integrated Development Environment*) Eclipse, VHDL ou ainda no *software* de desenvolvimento gráfico LabVIEW. Entretanto, existem vantagens que somente o LabVIEW fornece em implementações com o NI myRIO, as quais serão apresentadas na seção 3.2.

3.2 AMBIENTE DE DESENVOLVIMENTO LABVIEW

O LabVIEW é um ambiente de desenvolvimento criado pela empresa National Instruments com o foco principal de acelerar a produtividade de engenheiros em seu trabalho, apresentando duas grandes diferenças se comparado às demais linguagens de programação. A primeira delas diz respeito ao tipo de programação que, ao contrário das linguagens tradicionais que possuem uma programação textual, baseia-se em uma representação gráfica realizada pela conexão entre blocos gráficos em um diagrama para, em seguida, ser

compilada diretamente em código de máquina e executada. O segundo diferencial é que o código em LabVIEW não é executado de forma sequencial como nas abordagens tradicionais, mas sim de acordo com as regras do fluxo de dados que determinam a ordem da execução, ou seja, cada nó do programa executa sua lógica somente após ter todos os dados de entrada válidos para então produzir novos dados de saída (NATIONAL INSTRUMENTS, 2015c). Esses fatos trazem a vantagem de tornar a programação em LabVIEW mais simples e intuitiva, além de permitirem a criação de trechos de código que sejam executados em paralelo de forma fácil.

Os programas gráficos do LabVIEW são chamados de VIs (instrumentos virtuais, do inglês *virtual instruments*), devido ao fato de o *software* fornecer aparência e modo de operação similares a instrumentos físicos. Cada VI é composto por dois ambientes: diagrama de blocos e painel frontal. O diagrama de blocos, que contém o código fonte gráfico do programa LabVIEW, possui o conceito de separar, de uma maneira lógica, o código fonte e a *interface* com o usuário, representada pelo painel frontal. Cada objeto em um dos ambientes possui o seu correspondente no outro, assim como cada modificação realizada em um irá influenciar diretamente sobre o outro (NATIONAL INSTRUMENTS, 2015d).

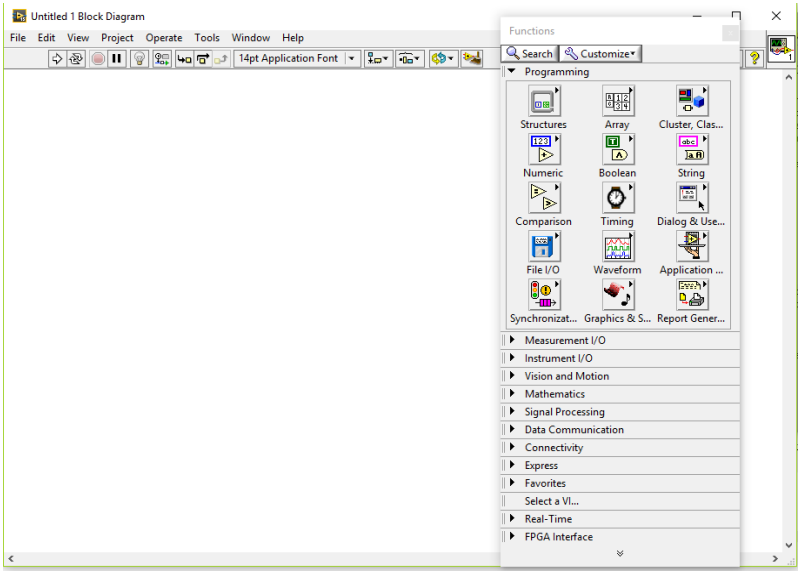
O diagrama de blocos possui diversas opções de blocos gráficos para implementação, tais como funções, vetores e matrizes, estruturas, constantes, além de existirem aqueles específicos para aplicações como medição, processamento de sinal, matemática, controle e simulação, entre outras. O LabVIEW também permite ao usuário que, após a criação de um VI, ele possa ser utilizado em outro VI, sendo chamado, a partir do diagrama de blocos, em forma de um subVI. A Figura 7 mostra como é aparência do diagrama de blocos, assim como a paleta de funções disponíveis no programa.

O painel frontal, mostrado na Figura 8, possui uma paleta de controle utilizada para a criação da *interface* com o usuário, com a opção de enviar sinais através de controles e recebê-los com a ajuda de indicadores, fornecendo esses dados ao programa a partir da comunicação com o diagrama de blocos.

O LabVIEW traz maior versatilidade e confiabilidade ao projetista fornecendo integração com diversos tipos de *targets* de *hardware*, como instrumentos convencionais de medição, dispositivos de aquisição de dados, controladores de motores, sistemas de aquisição e processamento de imagem, FPGAs, entre outros. Para isso, o *software* faz uso de módulos, também desenvolvidos pela National Instruments ou por terceiros, que são integrados ao LabVIEW e compatíveis aos *hadwares* de interesse, como por exemplo, LabVIEW Real-Time e LabVIEW FPGA.

De acordo com dados da National Instruments (NATIONAL INSTRUMENTS, 2016a), o módulo NI LabVIEW Real-Time é uma solução completa,

Figura 7 – Diagrama de blocos do LabVIEW.



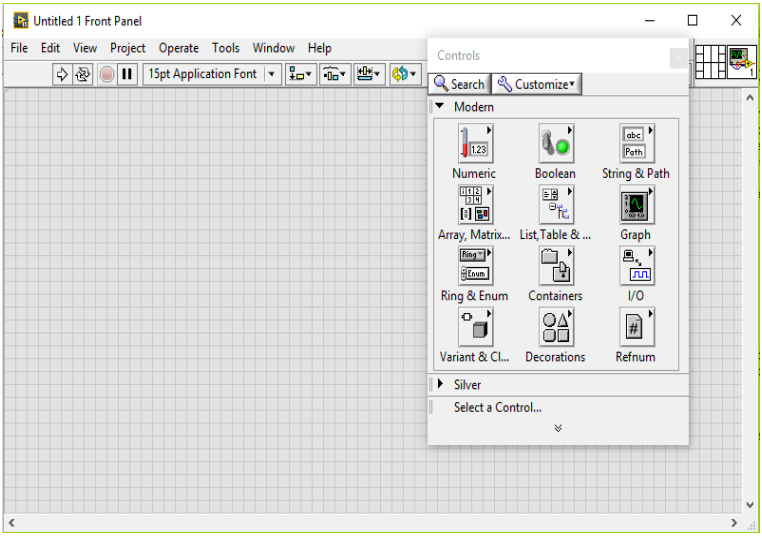
Fonte: Elaborado pelo autor.

capaz de criar sistemas embarcados autônomos de alta confiabilidade que, em conjunto com um *hardware* com sistema operacional de tempo real, oferece desempenho determinístico. Como mencionado anteriormente, o sistema operacional de tempo real (RTOS, do inglês *Real Time Operating System*), da empresa National Instruments, é baseado em Linux e possui suporte total do ambiente de desenvolvimento NI LabVIEW com o módulo LabVIEW Real-Time. Ainda de acordo com a National Instruments, existem três razões principais para usar o módulo LabVIEW Real-Time:

- incorporação da programação gráfica do LabVIEW a um sistema embarcado;
- confiabilidade e temporização precisa que somente um sistema operacional de tempo real pode oferecer em aplicações que operam sob requisitos críticos de tempo;
- grande número de bibliotecas disponíveis no LabVIEW para aplicação em sistemas embarcados.

O módulo NI LabVIEW FPGA permite a utilização do ambiente gráfico

Figura 8 – Painel frontal do LabVIEW.



Fonte: Elaborado pelo autor.

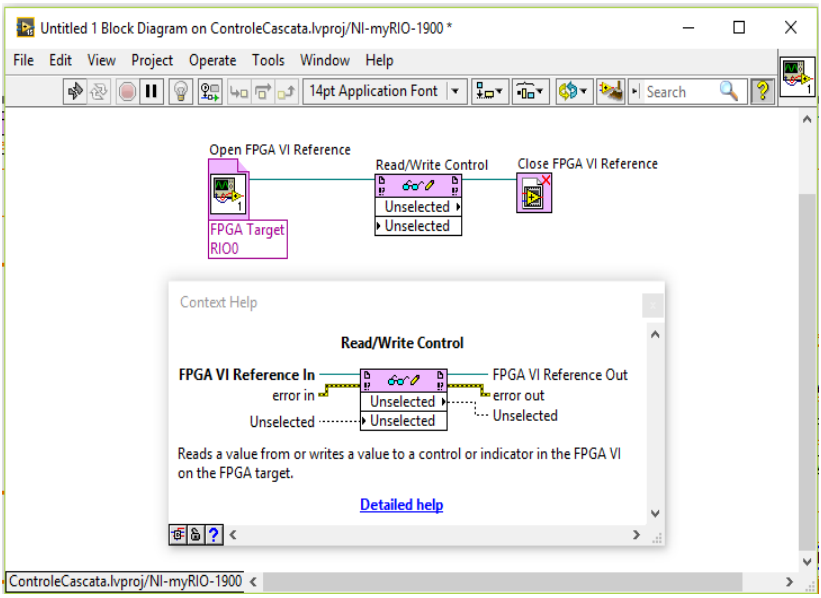
LabVIEW em projetos que incluem *targets* FPGA de *hardware* de entradas/saídas reconfiguráveis da National Instruments. Isso oferece ao projetista a capacidade de implementar sistemas mais complexos com alta eficiência onde faz-se necessário determinismo, confiabilidade e rápido tempo de resposta, justificando o uso da tecnologia FPGA. A empresa National Instruments também destaca três motivos principais para a utilização do módulo NI LabVIEW FPGA:

- abordagem gráfica para programação em FPGA, simplificando a tarefa de comunicação e interfaceamento de entradas e saídas;
- bibliotecas prontas com funções básicas, assim como a reutilização de códigos HDL existentes com o auxílio de um bloco de integração;
- desenvolvimento rápido de algoritmos, executando simultaneamente as tarefas paralelas e contendo ferramentas de depuração e recursos de simulação, a fim de encontrar possíveis erros de implementação antes da compilação do programa.

Em conjunto com os módulos NI LabVIEW Real-Time e LabVIEW FPGA, dispositivos NI RIO, como é o caso do myRIO, fornecem uma pla-

taforma flexível para a criação de projetos mais complexos, que vão desde aquisições de dados a sistemas de controle. A integração entre o LabVIEW, os módulos e o myRIO é relativamente simples e intuitiva. Quando o myRIO é conectado ao computador, o NI LabVIEW já o reconhece, assim, ao criar um novo projeto no programa, existe a opção de escolher um específico para a implementação no myRIO e que será programado nos módulos Real-Time e FPGA. Também existe a possibilidade de comunicação direta entre o Real-Time e o FPGA, através dos comandos apresentados na Figura 9.

Figura 9 – Integração LabVIEW Real-Time e FPGA.



Fonte: Elaborado pelo autor.

Como observado na Figura 9, no diagrama de blocos referente ao módulo LabVIEW Real-Time é iniciada uma referência de comunicação com o módulo LabVIEW FPGA, com a *target* do *hardware* já reconhecida pelo programa. A partir disso, é possivel escrever valores em controles ou lê-los de indicadores presentes no VI do FPGA, fechando a comunicação logo em seguida.

3.3 DIGIAC 710

O sistema escolhido como estudo de caso para a implementação do controle em cascata é o DIGIAC 710. Ele é composto por três componentes principais: um potenciômetro multivoltas, um motor de corrente contínua (CC) e um tacogerador CC, acoplados por meio de uma relação de engrenagens. A Figura 10 mostra a estrutura física do DIGIAC 710, em dois ângulos diferentes de visão: frontal e superior.

Figura 10 – (a) Visão frontal e (b) visão superior - DIGIAC 710.

(a) Visão frontal

(b) Visão superior

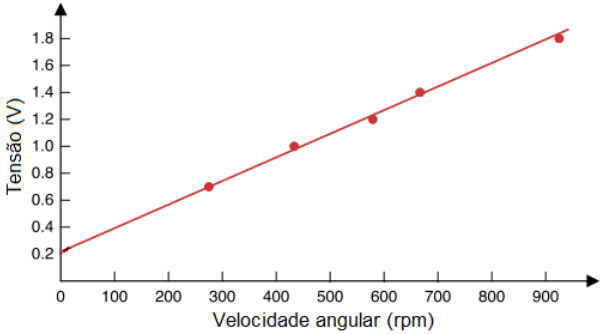
Fonte: Imagem obtida de www.ebay.com

O potenciômetro, que permite medir a posição do eixo do motor, foi fabricado pela empresa Waters MFG, modelo MP 109, e possui uma resistência regulável no valor de 10 kΩ. O motor CC foi desenvolvido pela Servo-Tek Products, entretanto, não foi possível encontrar dados suficientes para o seu acionamento, sendo que a única informação disponível é uma indicação de tensão no valor de $27,5$ V em sua carcaça. O tacogerador CC, também fabricado pela Servo-Tek Products, que pode ser implementado para a medição de velocidade angular, possui uma relação média tensão/(rotações por minuto) de 7 V/1000 rpm.

Existem, no mínimo, três não linearidades de um motor CC, tais como zona morta, saturação e folga na engrenagem (do inglês, *back lash*). A Figura 11 apresenta o comportamento geral de um motor CC, tendo como base a relação entre tensão e velocidade angular.

Como pode ser observado, quando uma tensão de nível muito baixo é aplicada ao motor, o número de rotações é nulo. À medida que a tensão de alimentação aumenta, o motor passa a adquirir maior velocidade angular.

Figura 11 – Relação tensão x velocidade angular de um motor DC.



Fonte: adaptado de Scarpino, 2015.

Esse fato, por si só, é capaz de comprovar a não linearidade de um motor CC, pois indica que ele apresenta uma zona morta, que é um elemento não linear. Entretanto, é possível utilizar algumas técnicas de linearização a fim de utilizar um modelo que seja capaz de reproduzir o comportamento desse sistema de maneira aproximada em torno de uma região de interesse. Um exemplo é adquirir dados de um determinado ponto de operação do sistema e aproximar o comportamento do sistema não linear por um modelo linear, para que o projeto de técnicas lineares de controle torne-se mais evidente.

A relação de redução de engrenagens entre o motor e o tacogerador no DIGIAC 710 baseia-se na equação geral, que expressa a relação entre a velocidade angular e o número de dentes das engrenagens, ou seja,

$$\frac{\omega_M}{\omega_G} = \frac{N_G}{N_M}, \tag{3.2}$$

onde ω_G e ω_M são a velocidade angular do tacogerador e do motor, respectivamente, N_G é o número de dentes da engrenagem conectada ao eixo do tacogerador e N_M ao eixo do motor.

No caso do DIGIAC 710, N_G é igual a 120 dentes e N_M a 30 dentes. Portanto, a relação de engrenagens entre o motor e o tacogerador é igual a

$$\frac{\omega_M}{\omega_G} = \frac{120}{30},$$

ou seja,

$$\omega_M = 4\omega_G. \tag{3.3}$$

Com base nisso, é possível afirmar que a velocidade angular do motor é quatro vezes maior que do tacogerador. Para este caso especificamente, o potenciômetro não foi utilizado. Porém, a relação de redução de engrenagens entre o potenciômetro e o tacogerador é de

$$\omega_G = 16\omega_P, \tag{3.4}$$

asssim como a relação entre o potenciômetro e o motor é

$$\omega_M = 64\omega_P. \tag{3.5}$$

3.4 COMENTÁRIOS FINAIS

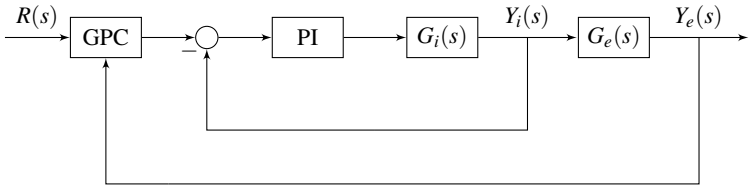
Este capítulo tratou dos principais materiais e métodos empregados para a implementação deste trabalho. Primeiramente, foi apresentada a plataforma de desenvolvimento myRIO, as suas vantagens em relação às placas multifunção de *hardware* fixo, seus componentes principais, assim como as portas responsáveis pela comunicação com circuitos externos. Como o myRIO pode ser programado pelo *software* de desenvolvimento gráfico LabVIEW, em seguida, foram abordadas as suas características básicas, como a composição do diagrama de blocos e painel frontal, além de introduzir conceitos importantes em relação aos módulos LabVIEW Real-Time e LabVIEW FPGA e a comunicação entre eles e com o próprio myRIO. Por fim, foi descrito o funcionamento do sistema escolhido como estudo de caso, o DIGIAC 710, os seus componentes, as características específicas de cada um e a relação de engrenagens entre o motor, o tacogerador e o potenciômetro. Tomando como base o que foi apresentado neste capítulo, o capítulo 4 apresenta a implementação do sistema de controle no myRIO e a construção do *hardware* necessário para a validação do trabalho.

4 IMPLEMENTAÇÃO

Neste trabalho, foi implementado um sistema de controle em cascata, contendo somente duas malhas: interna e externa. Para a sua execução, foi utilizado o *hardware* myRIO, garantindo determinismo e confiabilidade aos controladores.

Na malha interna, foi desenvolvido um controlador PI, que foi compilado para a *target* de *hardware* FPGA, isso significa que todo o sistema de controle PI foi programado utilizando-se do módulo NI LabVIEW FPGA e posteriormente convertido em uma implementação de *hardware* reconfigurável. Para a malha externa, entretanto, a implementação do controlador GPC foi realizada utilizando-se do módulo NI LabVIEW Real-Time para, em seguida, ser compilada no processador com sistema de tempo real do *hardware* myRIO. A Figura 12 apresenta a estrutura em questão.

Figura 12 – Controle em cascata: PI (malha interna) e GPC (malha externa).



Fonte: Elaborado pelo autor.

É importante salientar que o sistema de controle em cascata implementado no LabVIEW foi pensado e desenvolvido de maneira que essa estrutura, proposta no trabalho, possa ser aplicada em diferentes sistemas monovariáveis. Para isso, é necessário modificar apenas o ajuste de cada controlador, que é feito com base na planta a qual o sistema de controle é aplicado.

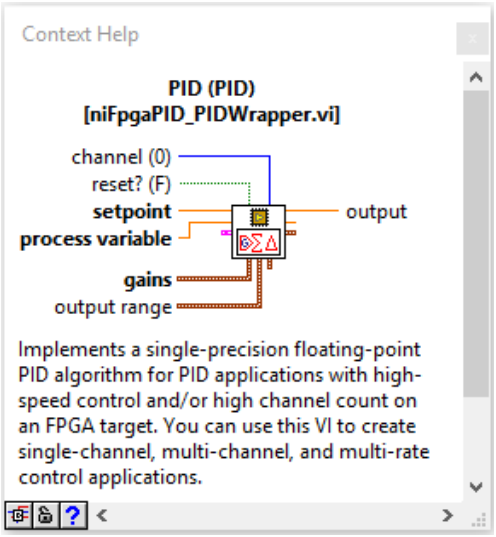
4.1 ALGORITMOS DE CONTROLE

4.1.1 Controle PI implementado em FPGA

O módulo NI LabVIEW FPGA possui, em sua paleta de funções, um bloco pronto para implementação de um controlador PID representado na

forma paralela, que pode ser observado na Figura 13.

Figura 13 – Bloco de controle PID no módulo LabVIEW FPGA.



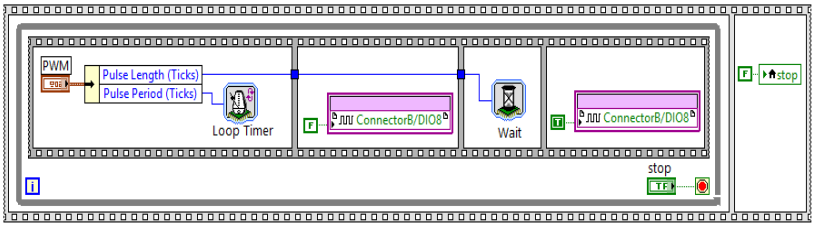
Fonte: Elaborado pelo autor.

A partir disso, é possível definir os principais parâmetros necessários para sua execução, ou seja, o valor de referência a ser seguido pelo controlador (*setpoint*), a variável de processo (*process variable*) que é a variável a ser controlada, o valor dos ganhos do controlador K_P, K_i e K_d, assim como os valores máximo e mínimo que o sinal de controle pode assumir (*output range*), gerando então o sinal de controle u_i (*output*). O bloco PID no LabVIEW FPGA também permite a escolha entre controladores PID, PI e PD, caso o desenvolvedor opte por não utilizar os três ganhos em conjunto.

Entretanto, o módulo LabVIEW FPGA não fornece um bloco específico para a aplicação de um sinal PWM, geralmente utilizado para o acionamento de atuadores com características analógicas, como motores e resistores de potência, por exemplo. Com isso, faz-se necessário a criação desse sinal a partir de outras funções disponíveis e que são compatíveis com a *target* FPGA. A Figura 14 apresenta um exemplo de como o PWM pode ser implementado no LabVIEW FPGA.

A partir da definição de uma base de *clock*, que pode ser em milissegundos, microssegundos ou *ticks*, sendo que o último refere-se ao próprio ciclo de varredura da *target* do FPGA, é possível determinar o tempo que uma saída

Figura 14 – Bloco de sinal PWM no módulo LabVIEW FPGA.



Fonte: Elaborado pelo autor.

digital, definida pelo projetista, deve permanecer em nível alto ou baixo, ou seja, controlar a sua razão cíclica. Por padrão, a base de *clock tick* do FPGA equivale a 40 MHz, isto é, o valor de um *tick* corresponde a 25 ns.

A Figura 15 apresenta o diagrama de blocos da implementação do controlador PI no módulo NI LabVIEW FPGA proposto neste trabalho, enquanto que a Figura 16 mostra o seu respectivo painel frontal.

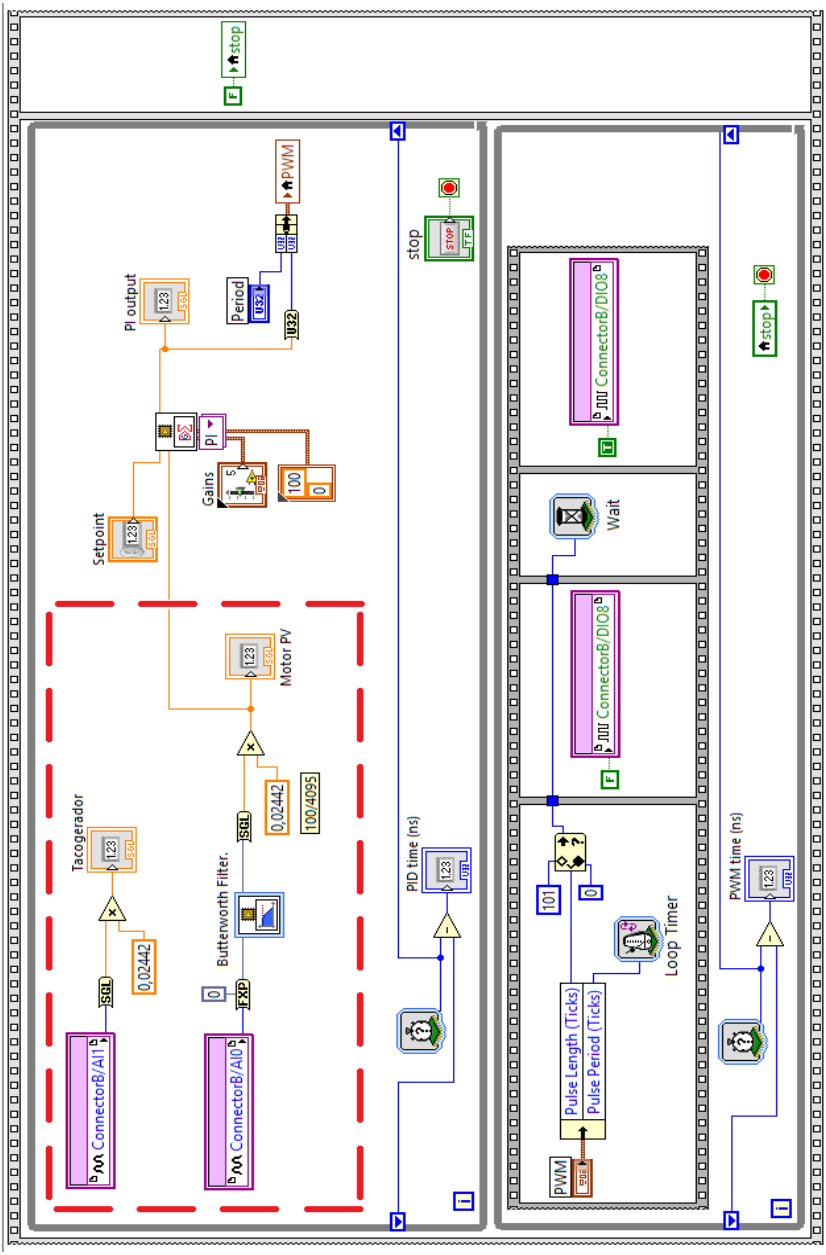
Na Figura 15, é importante ressaltar que a área destacada em vermelho é referente ao modelo do sistema ao qual o controlador foi aplicado, sendo que esse é o único trecho que deve ser modificado se a planta for alterada, além dos parâmetros do controlador PI, que variam de acordo com o modelo implementado. Essa região do código baseia-se em filtragem e conversão dos dados medidos na planta, isto é, a saída do bloco de leitura analógica é um número inteiro de 0 a 4095 (12 *bits*) e antes de o valor ser informado ao controlador ele é filtrado e convertido para um valor que representa a variável de processo.

Neste VI, foram implementadas as funções *flat sequence*, responsável por executar os quadros de maneira sequencial, e *while loop*, que repete o código em seu interior até que alguma condição específica ocorra, neste caso, até que o botão *stop* seja acionado. Com isso, foram definidos os parâmetros do bloco PI, assim como o sinal de controle foi delimitado como entrada para gerar o sinal PWM que, neste caso, foi aplicado ao circuito externo através de uma porta digital I/O do conector MXP B.

O painel frontal, na Figura 16, mostra a *interface* com o usuário, trazendo parâmetros para definição de valores, assim como o tempo de execução do programa e variáveis que permitem acompanhar o processo.

Como o foco deste trabalho é apresentar resultados significativamente melhores do que sistemas de controle tradicionais no que diz respeito ao tempo de processamento, foi realizada uma medição para comprovação utilizando-se

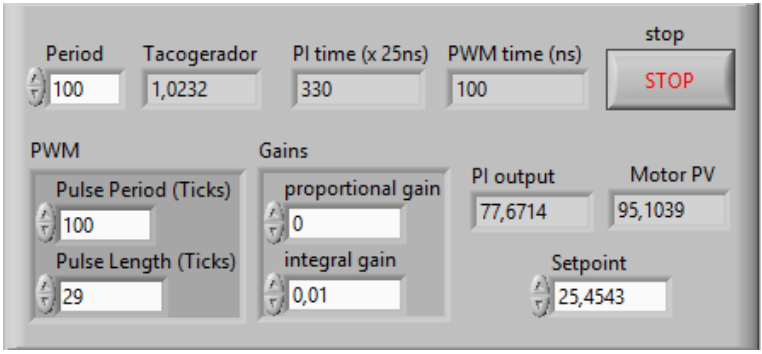
Figura 15 – Diagrama de blocos do controle PI no LabVIEW FPGA.



Fonte: Elaborado pelo autor.

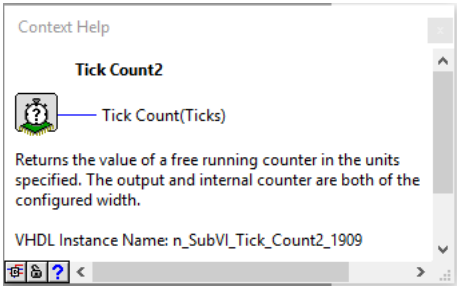
do componente *tick count*, mostrado na Figura 17, que é responsável por retornar o valor do tempo de execução do laço em questão. Essa medição é realizada como apresentado na Figura 15, ou seja, é utilizado um registrador de deslocamento que retém o valor adquirido na última iteração do laço e, em seguida, calcula-se a diferença entre esse tempo e o atual.

Figura 16 – Painel frontal do controle PI no LabVIEW FPGA.



Fonte: Elaborado pelo autor.

Figura 17 – Contador de tempo de execução.



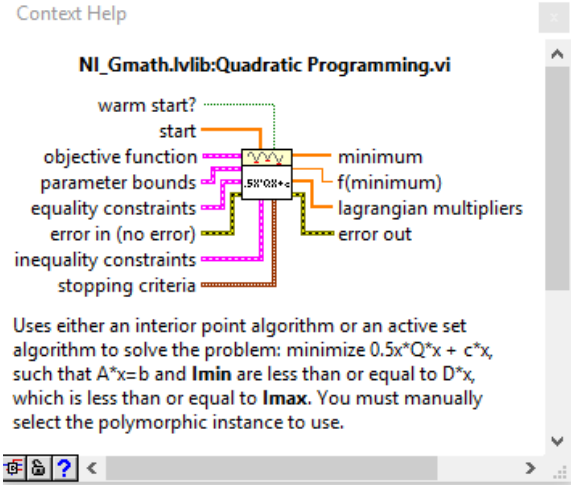
Fonte: Elaborado pelo autor.

4.1.2 Controle GPC implementado em sistema de tempo real

O módulo NI LabVIEW Real-Time possui, em sua paleta de funções, um bloco chamado *Quadratic Programming VI*, que é responsável pela resolução

de um problema de otimização do tipo programação quadrática, mostrado na Figura 18.

Figura 18 – Bloco de programação quadrática no LabVIEW Real-Time.

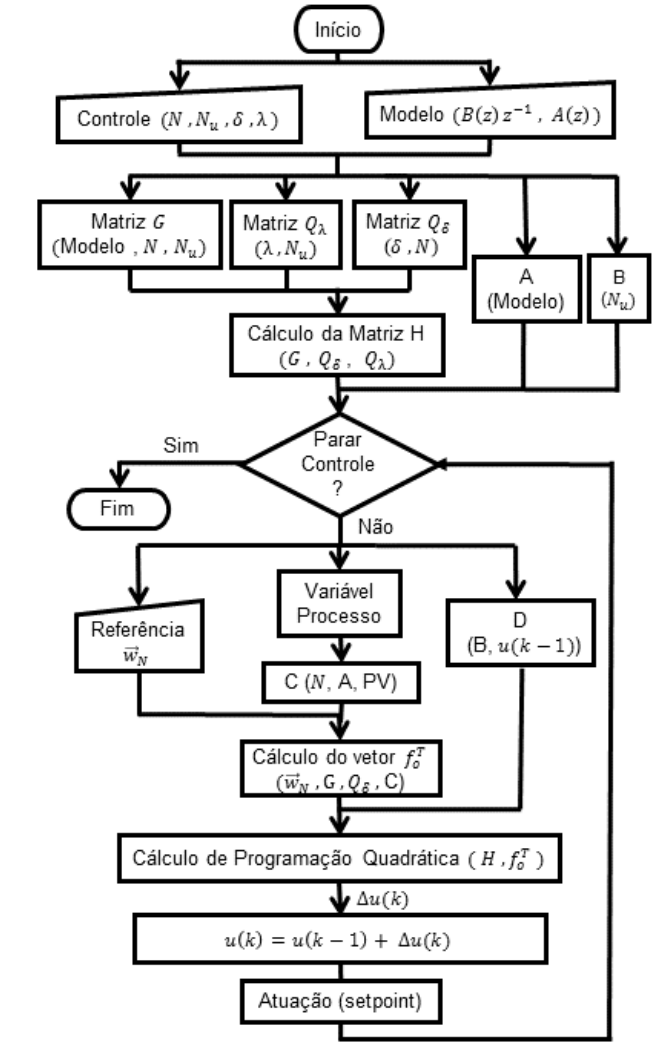


Fonte: Elaborado pelo autor.

Essa função recebe alguns parâmetros de entrada, como coeficientes específicos da função objetivo, limites máximo e mínimo e restrições de igualdade e desigualdade, além de retornar valores de saída como o vetor $\Delta\vec{u}$, elemento de interesse nesse caso. Os coeficientes da função objetivo que esse bloco espera receber são um termo quadrático representado pela matriz $H = 2(G^T Q_\delta G + Q_\lambda)$ e um termo linear na forma do vetor $\vec{f_o}^T = 2(\vec{f} - \vec{w})^T Q_\delta G$. Com isso, essas matrizes foram construídas com a finalidade de que o controle GPC pudesse ser implementado em um sistema monovariável, ou seja, apenas o modelo utilizado para representação da planta deve ser alterado via *interface* gráfica no LabVIEW Real-Time, além dos parâmetros de ajuste do controlador (horizontes e ponderações).

A Figura 19 apresenta um fluxograma com as etapas referentes à implementação do controlador GPC no LabVIEW Real-Time. As funções dos blocos exemplificados pelas letras A, B, C e D são descritas abaixo do fluxograma. A programação do controlador GPC no LabVIEW Real-Time foi realizada utilizando-se de um laço *while*. Entretanto, para que o sistema de controle tivesse o menor tempo possível de processamento, valores constantes foram inicializados apenas uma vez fora do laço, como as matrizes G, Q_δ,

Figura 19 – Fluxograma contendo as etapas da implementação do GPC no LabVIEW RT.

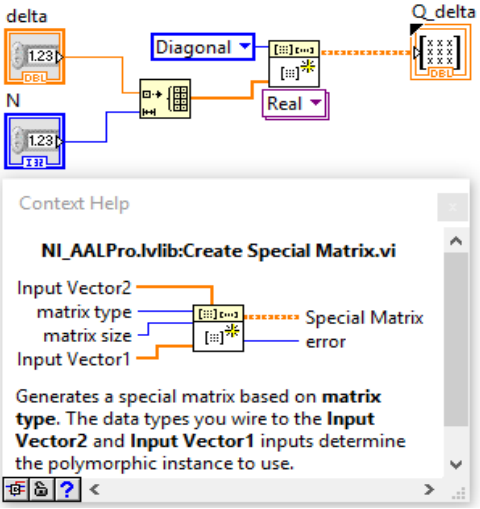


Fonte: Elaborado pelo autor.

Q_λ, H e a de restrições, além dos vetores constantes referentes ao cálculo da resposta livre. A comunicação entre o *Real-Time* e o FPGA também foi iniciada fora do laço. Os termos que necessitavam de atualização a cada instante de tempo, como o vetor de resposta livre $\vec{f}$, o cálculo do valor do sinal de controle, o cálculo do vetor $\vec{f_o}$ e dos vetores referentes à matriz de restrição, como também os valores lidos e escritos no LabVIEW FPGA, foram dispostos no interior do laço *while*.

Como visto na seção 2.2, a matriz Q_δ possui dimensão $(N \times N)$ e foi elaborada da forma apresentada na Figura 20, ou seja, um vetor com dimensão N foi inicializado com os elementos contendo o valor da ponderação $\delta_{(j)}$ para, em seguida, criar uma matriz especial do tipo diagonal a partir disso. É importante salientar que o valor da ponderação $\delta_{(j)}$ foi assumido constante ao longo do horizonte neste caso.

Figura 20 – Matriz de ponderação Q_δ no LabVIEW Real-Time.

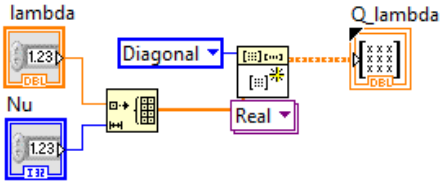


Fonte: Elaborado pelo autor.

Da mesma forma, a matriz Q_λ, que possui dimensão $(N_u \times N_u)$, foi construída como é mostrado na Figura 21, isto é, um vetor com essa dimensão é inicializado com os elementos iguais a $\lambda_{(j)}$, assumida como constante ao longo do horizonte neste caso, e então uma matriz diagonal é criada com esses valores.

O vetor contendo as referências futuras para o controlador GPC, $\vec{w}$, possui dimensão N e, neste caso, a referência é assumida constante ao longo do

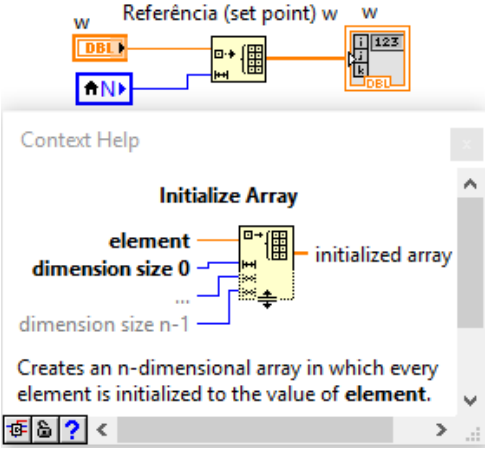
Figura 21 – Matriz de ponderação Q_λ no LabVIEW Real-Time.



Fonte: Elaborado pelo autor.

horizonte de predição, mas isso pode ser facilmente alterado se as referências futuras forem conhecidas. A Figura 22 apresenta o vetor $\vec{w}$ construído no LabVIEW Real-Time.

Figura 22 – Vetor de referência $\vec{w}$ no LabVIEW Real-Time.



Fonte: Elaborado pelo autor.

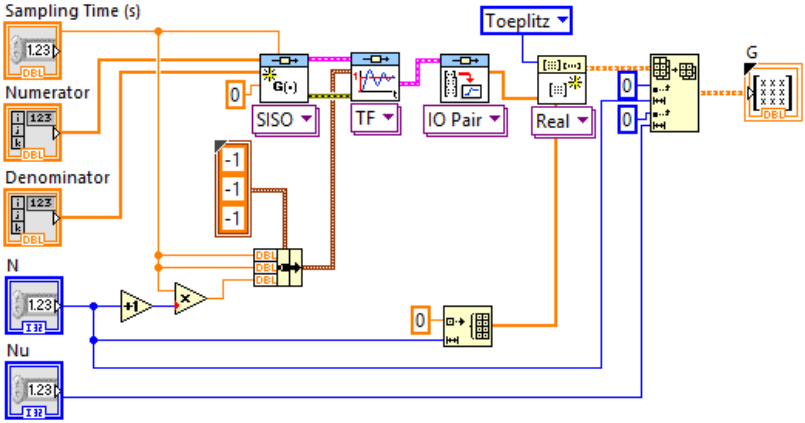
A matriz G, que possui dimensão $N \times N_u$, foi construída como mostra a Figura 23. Primeiramente, a partir do modelo do sistema a ser controlado, são definidos o numerador, o denominador e o tempo de amostragem para a construção da função de transferência, que deve estar representada no domínio de tempo discreto para a aplicação do controle GPC. Então, é obtida a resposta do sistema no tempo quando sua entrada é excitada por um degrau unitário e, em seguida, esses valores são transformados em um vetor de elementos. A

seguir, é inicializada uma matriz especial do tipo *Toeplitz*, ou seja,

$$T_z = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & \dots & \dots & a_{-n+1} \\ a_1 & a_0 & a_{-1} & \ddots & \ddots & \vdots \\ a_2 & a_1 & a_0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & a_{-1} & a_{-2} \\ \vdots & \ddots & \ddots & a_1 & a_0 & a_{-1} \\ a_{n-1} & \dots & \dots & a_2 & a_1 & a_0 \end{bmatrix}.$$

O bloco responsável pela criação dessa matriz possui como entrada dois vetores, sendo o primeiro referente aos coeficientes positivos da matriz e o segundo aos negativos, ou seja, neste caso implementado, os termos negativos são definidos como nulos já que a entrada relativa ao segundo vetor encontra-se vazia. A partir dessa construção, resulta a matriz G com dimensão $N \times N_u$.

Figura 23 – Matriz G no LabVIEW Real-Time.



Fonte: Elaborado pelo autor.

A construção do vetor de resposta livre $\vec{f}$, com dimensão N, foi realizada em duas etapas: inicialização das entradas e cálculo da resposta livre propriamente dita. Na primeira etapa, tomam-se como base o numerador e

denominador do modelo do sistema, ou seja,

$$G(z) = \frac{y(k)}{u(k)} = \frac{B(z)z^{-1}}{A(z)} = \frac{(b_0 + b_1 z^{-1} + \cdots + b_n z^{-n})z^{-1}}{1 + a_1 z^{-1} + \cdots + a_m z^{-m}}. \tag{4.1}$$

Assim,

$$A(z)y(k) = B(z)z^{-1}u(k), \tag{4.2}$$

onde $y(k)$ é o sinal de saída do sistema, adquirido da planta real, e $u(k)$ é o sinal de controle. Levando-se em consideração o modelo CARIMA na equação (2.4) com $C(z) = 1$ e o valor atribuído a Δ em (2.5), para a implementação da resposta livre no LabVIEW Real-Time, a equação (4.2) pode ser escrita na forma

$$\tilde{A}(z)y(k) = B(z)z^{-1}\Delta u(k-1), \tag{4.3}$$

onde $\tilde{A}$ corresponde a $\tilde{A}(z)\Delta = A(z)(1-z^{-1})$. Em notação matricial, a equação (4.3) é representada por

$$\begin{bmatrix} 1 & \tilde{a}_1 & \tilde{a}_2 & \dots & \tilde{a}_{m+1} \end{bmatrix} \begin{bmatrix} y(k) \\ y(k-1) \\ \vdots \\ y(k-m-1) \end{bmatrix} = \begin{bmatrix} b_0 & b_1 & \dots & b_n \end{bmatrix} \begin{bmatrix} \Delta u(k-1) \\ \Delta u(k-2) \\ \vdots \\ \Delta u(k-n-1) \end{bmatrix},$$

onde $\tilde{a}_i$ são os coeficientes do polinômio $\tilde{A}(z)$ e b_i os coeficientes de $B(z)$. Isolando $y(k)$ em (4.3), tem-se

$$y(k) = B(z)\Delta u(k-1) - [(\tilde{A}(z) - 1)y(k)]. \tag{4.4}$$

Em notação matricial:

$$y(k) = \begin{bmatrix} -\tilde{a}_1 & -\tilde{a}_2 & \dots & -\tilde{a}_{m+1} \end{bmatrix} \begin{bmatrix} y(k-1) \\ y(k-2) \\ \vdots \\ y(k-m-1) \end{bmatrix} + \begin{bmatrix} b_0 & b_1 & \dots & b_n \end{bmatrix} \begin{bmatrix} \Delta u(k-1) \\ \Delta u(k-2) \\ \vdots \\ \Delta u(k-n-1) \end{bmatrix}.$$

A resposta livre é calculada como a predição da resposta do sistema a uma entrada constante e igual ao último valor que foi aplicado à planta. Portanto, a segunda etapa, responsável por esse cálculo, baseia-se na seguinte

configuração, que calcula o primeiro elemento do vetor de resposta livre:

$$f(k+1) = \begin{bmatrix} -\tilde{a}_1 & -\tilde{a}_2 & \dots & -\tilde{a}_{m+1} \end{bmatrix} \begin{bmatrix} y(k) \\ y(k-1) \\ \vdots \\ y(k-m) \end{bmatrix} + \begin{bmatrix} b_0 & b_1 & \dots & b_n \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \Delta u(k-1) \\ \vdots \\ \Delta u(k-n) \end{bmatrix}.$$

Neste caso, $\Delta u(k)$ é considerado nulo, pois a resposta livre assume que não há variação no sinal de controle. Em seguida, o segundo termo do vetor é calculado da seguinte forma:

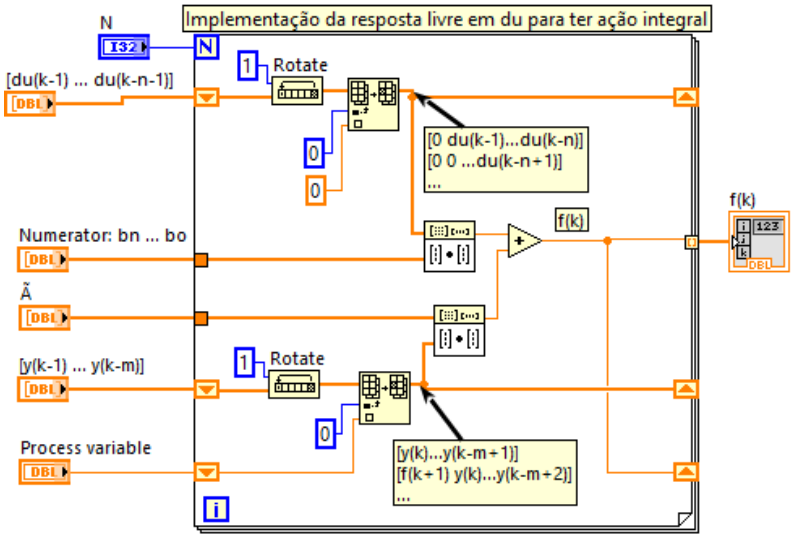
$$f(k+2) = \begin{bmatrix} -\tilde{a}_1 & -\tilde{a}_2 & \dots & -\tilde{a}_{m+1} \end{bmatrix} \begin{bmatrix} f(k+1) \\ y(k) \\ y(k-1) \\ \vdots \\ y(k-m+1) \end{bmatrix} + \begin{bmatrix} b_0 & b_1 & \dots & b_n \end{bmatrix} \begin{bmatrix} \Delta u(k+1) \\ \Delta u(k) \\ \vdots \\ \Delta u(k-n+1) \end{bmatrix},$$

aqui considerando-se $\Delta u(k)$ e $\Delta u(k+1)$ nulos, e assim por diante.

Esse cálculo foi implementado no LabVIEW Real-Time como mostra a Figura 24. Com essas entradas definidas, foi possível realizar os cálculos necessários para a obtenção da matriz H (Figura 25) e do vetor $\vec{f}_o$ (Figura 26). A matriz H e o vetor $\vec{f}_o$ foram unidos por um *cluster* e aplicados à entrada *objective function* do bloco *Quadratic Programming*, como apresentado na Figura 27.

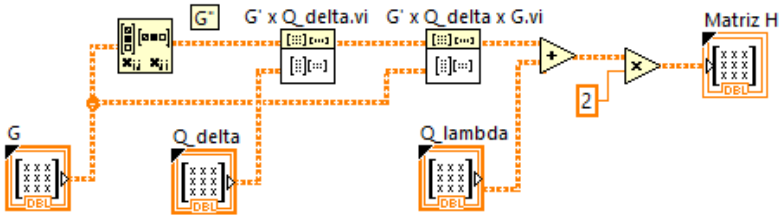
Neste caso, foi implementada uma restrição de desigualdade, que foi conectada à entrada *inequality constraints* do bloco de programação quadrática, para que o sinal de controle $u(k)$ se mantivesse entre os limites 0 % e 100 % em todos os instantes do horizonte de controle. A sua construção foi baseada na equação (2.14) e possui duas etapas: inicialização e cálculo da restrição. A primeira etapa tem como entrada a dimensão da matriz de restrição, N_u, e apresenta como saída a matriz de restrições e os vetores contendo os valores

Figura 24 – Vetor de resposta livre $\vec{f}(k+j)$ no LabVIEW Real-Time.



Fonte: Elaborado pelo autor.

Figura 25 – Matriz H no LabVIEW Real-Time.

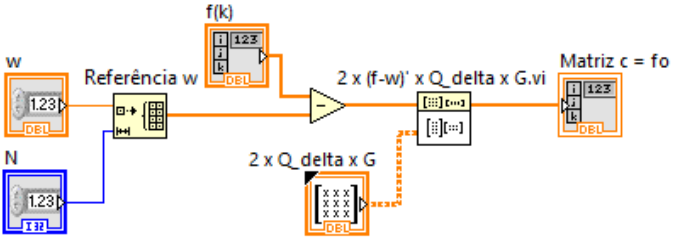


Fonte: Elaborado pelo autor.

mínimo e máximo do sinal de controle. Isso pode ser verificado na Figura 28.

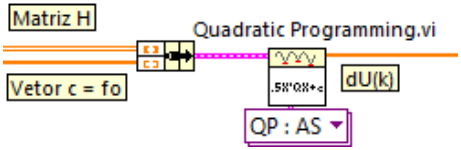
Na segunda etapa, foi construída a matriz de restrição para o vetor de sinal de controle $[u(k)\dots u(k+N_u-1)]$ de acordo com a equação (2.14), como mostra a Figura 29. A saída do bloco de programação quadrática retorna um vetor de dimensão N_u, que é atualizado a cada período de amostragem, contendo os valores do incremento de controle presente e futuros ($[\Delta u(k)\dots\Delta u(k+N_u-1)]$). O sinal de controle no instante atual $u(k)$ é calcu-

Figura 26 – Vetor $\vec{fo}$ no LabVIEW Real-Time.



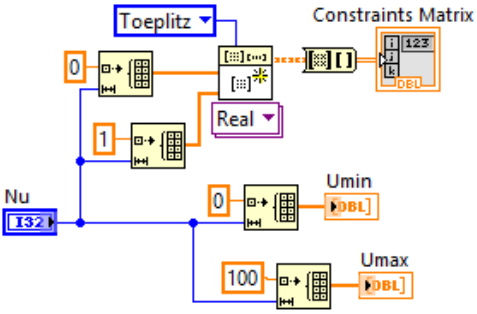
Fonte: Elaborado pelo autor.

Figura 27 – *Cluster* de entrada do bloco QP no LabVIEW Real-Time.



Fonte: Elaborado pelo autor.

Figura 28 – Inicialização de valores da matriz de restrição no LabVIEW Real-Time.



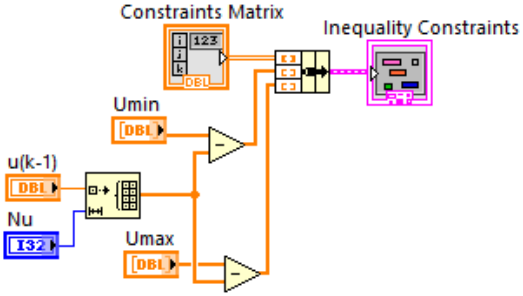
Fonte: Elaborado pelo autor.

lado a partir da equação (4.5) e foi construído no LabVIEW Real-Time como

mostra a Figura 30.

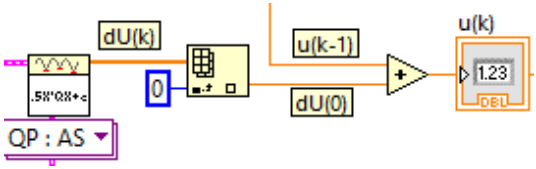
$$u(k) = \Delta u(k) + u(k-1) \tag{4.5}$$

Figura 29 – Restrição de desigualdade no LabVIEW Real-Time.



Fonte: Elaborado pelo autor.

Figura 30 – Cálculo do sinal de controle $u(k)$ no LabVIEW Real-Time.



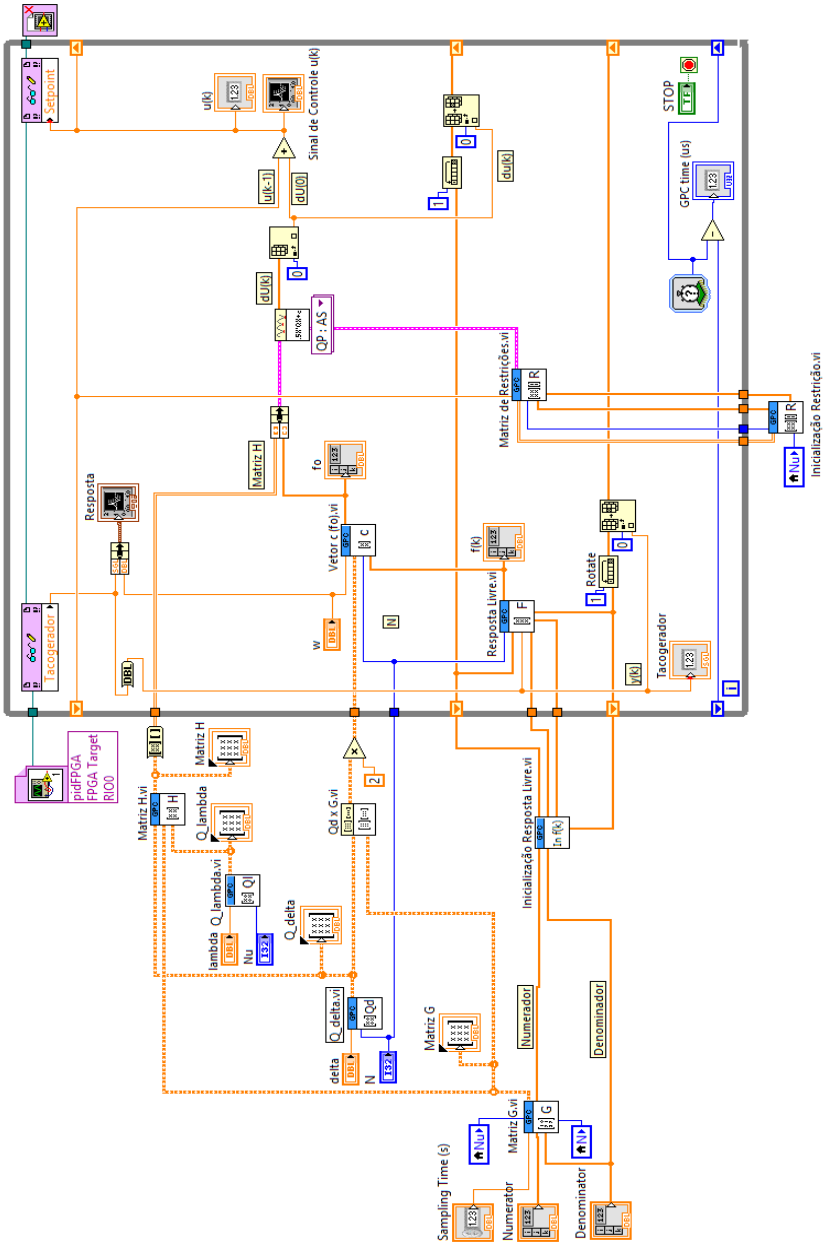
Fonte: Elaborado pelo autor.

Na Figura 31 é apresentado o diagrama de blocos referente ao controle GPC implementado no LabVIEW Real-Time.

É possível observar a criação de diversas subVIs com o propósito de tornar o código mais legível e interessante para visualização, como os cálculos referentes às matrizes G, Q_δ, Q_λ, H e de restrições, a inicialização e os vetores de resposta livre $\vec{f}$ e de restrições, assim como o cálculo do vetor $\vec{c} = \vec{f}_o$. Também é possível verificar a utilização do bloco *Tick Count* para calcular o tempo de execução de uma iteração do laço *while* a fim de avaliar as melhorias em relação ao tempo de processamento para a implementação do controlador GPC.

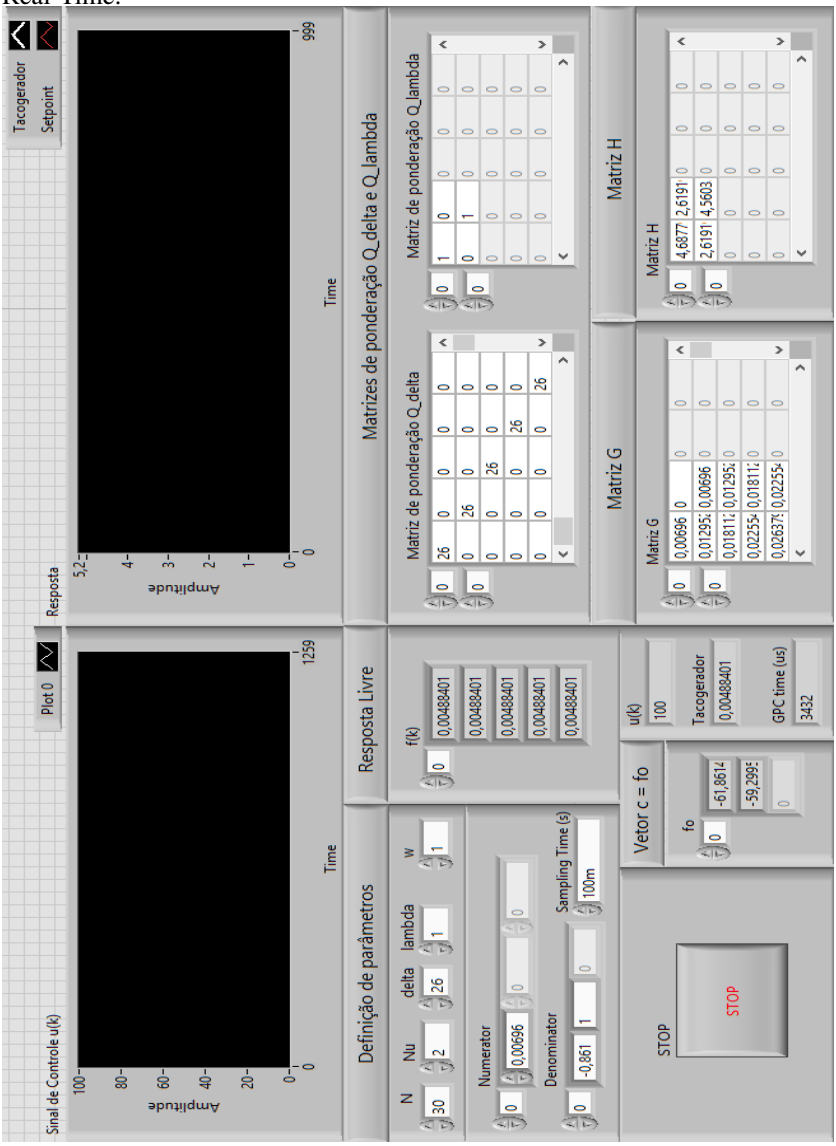
A Figura 32 mostra o painel frontal do controle GPC implementado no LabVIEW Real-Time.

Figura 31 – Diagrama de blocos do controle GPC no LabVIEW Real-Time.



Fonte: Elaborado pelo autor.

Figura 32 – Painel Frontal do controle GPC implementado no LabVIEW Real-Time.



Fonte: Elaborado pelo autor.

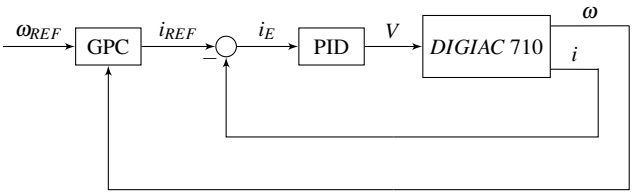
Essa *interface* permite ao usuário a definição de parâmetros importantes, como os horizontes de controle (N_u) e predição (N), as ponderações δ e λ, o numerador e denominador do modelo a ser utilizado, assim como o tempo de amostragem e o valor da referência a ser seguido. Também é possível verificar as matrizes G, Q_δ, Q_λ e H e os vetores $\vec{f}$ e $\vec{f_o}$. O painel ainda permite o acompanhamento, em tempo real, de variáveis importantes como o valor atual do sinal de controle, da variável de processo e do tempo de execução de uma iteração. São apresentados dois gráficos para melhor visualização do processo, em que um mostra o sinal de controle e o outro faz uma comparação entre o valor desejado (*setpoint*) e a variável de processo, permitindo ao usuário maior rapidez e facilidade no monitoramento e controle do processo.

Na implementação do controlador em cascata, em que é aplicado um controle PI na malha interna e um controlador GPC na malha externa, o valor adquirido de $u(k)$, programado no LabVIEW Real-Time, atualiza, a cada período de amostragem, um bloco de controle numérico presente no LabVIEW FPGA, em forma de *setpoint*, através da referência de comunicação existente entre os dois módulos.

4.2 HARDWARE PARA VALIDAÇÃO

Como apresentado na seção 2.3, neste trabalho, foi desenvolvida uma estratégia de controle em cascata, em que, na malha interna, foi implementado um controle PI e, na malha externa, um controlador GPC. Esse sistema de controle foi aplicado ao DIGIAC 710, sendo que o objetivo é controlar duas variáveis, velocidade angular e corrente elétrica, a partir da atuação sobre a tensão aplicada no motor. A Figura 33 apresenta o esquema de controle implementado.

Figura 33 – Controle em cascata: PI (malha interna) e GPC (malha externa).



Fonte: Elaborado pelo autor.

De acordo com a figura:

- ω_{REF} é a referência de velocidade angular do motor;
- ω é a medição da velocidade angular do motor;
- i_{REF} é a referência de corrente elétrica no motor;
- i é a medição da corrente no motor;
- i_E é o sinal de erro gerado pela comparação entre i_{REF} e i;
- V representa a tensão aplicado ao motor.

Neste caso, todos os intervalos de variação referentes aos sinais de controle gerados e aos valores adquiridos das saídas dos sistemas foram normalizados para que se mantivessem na faixa entre 0 % e 100 %.

Portanto, este trabalho baseia-se na implementação de um controlador GPC com a finalidade de controlar a velocidade angular do motor, sendo que o sinal de controle, gerado pelo GPC, é aplicado à malha interna como variável de processo para um controlador PI, que tem como objetivo controlar a corrente drenada pelo motor.

4.2.1 Medição de velocidade angular

Como visto na seção 3.3, o sistema DIGIAC 710 possui um motor, um tacogerador e um potenciômetro. Para medição da velocidade angular do motor foi empregado o tacogerador do sistema, que é acoplado ao motor através de uma relação de engrenagens.

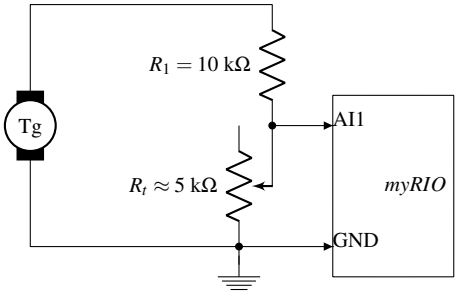
Como explicado na seção 3.3, o tacogerador possui uma relação de 7 V/1000 rpm. Com isso, a variação de tensão medida pelo tacogerador foi adquirida experimentalmente, resultando no intervalo de (0 a 15) V. Isso significa que a tensão é igual a 0 V quando o motor encontra-se parado, porém, quando o motor atinge a velocidade angular máxima a vazio, a tensão no tacogerador é de 15 V. Isso significa que a máxima velocidade angular que o motor pode atingir é de, aproximadamente,

$$\omega \approx \frac{15\ \text{V} \times 1000\ \text{rpm}}{7\ \text{V}} \approx 2142\ \text{rpm}. \tag{4.6}$$

Entretanto, como foi apresentado na seção 3.1, a plataforma de desenvolvimento myRIO possui entradas para medir sinais analógicos de tensão no intervalo de (0 a 5) V, tornando-se necessário o condicionamento do sinal proveniente da medição realizada pelo tacogerador, a fim de proteger o conector de entrada do myRIO contra tensões acima do suportado pelo mesmo. Portanto, para obter uma variação de (0 a 5) V referente à medição de velocidade

do tacogerador, foi implementado um divisor de tensão com o auxílio de um resistor ajustável, conhecido como *trimpot*. A Figura 34 apresenta o circuito implementado com esse objetivo.

Figura 34 – Condicionamento de sinal para leitura da saída do tacogerador.

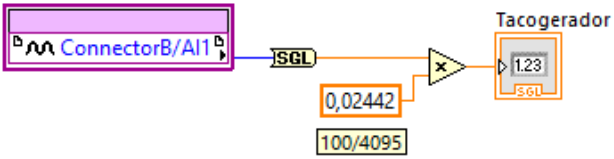


Fonte: Elaborado pelo autor.

De acordo com a figura, Tg representa o tacogerador, R_1 é um resistor com o valor escolhido igual a 10 kΩ e R_t é o *trimpot* ajustado em um valor que permitiu a leitura de tensão dentro dos limites desejados, neste caso, aproximadamente 5 kΩ. A leitura da diferença de tensão sobre o R_t foi realizada através de uma porta analógica AI1 e o GND do conector MXP B no myRIO.

Como o FPGA trabalha somente com valores digitais, a leitura vinda do tacogerador é convertida para um dado digital, através do conversor A/D com resolução de 12 *bits*. Isso significa que a entrada analógica do myRIO possui uma variação de 0 a 4095. Esse valor foi condicionado novamente, no LabVIEW FPGA, a fim de obter o intervalo da normalização de (0 a 100) % da velocidade angular medida pelo tacogerador, como mostra a Figura 35.

Figura 35 – Condicionamento da leitura do tacogerador no LabVIEW FPGA.



Fonte: Elaborado pelo autor.

4.2.2 Medição de corrente

Como apresentado na seção 3.1, as entradas analógicas do myRIO possuem intervalo de medição de (0 a 5) V. Assim, foi necessário encontrar uma relação entre tensão e corrente, já que o objetivo, neste caso, é a medição da corrente no motor do DIGIAC 710. Com isso, a maneira mais simples de encontrar essa relação é a partir da inserção de um resistor, conhecido como *shunt*, que é conectado em série com o motor, a fim de representar um amperímetro. De acordo com a Lei de Ohm, $V = R \cdot I$, e como o valor escolhido para o *shunt* é igual a 1 Ω, o valor da corrente no resistor, em amperes, é igual ao valor da tensão medida sobre o mesmo, em volts. Como o resistor *shunt* e o motor estão em série, a corrente drenada pelo motor é igual à corrente medida no resistor *shunt*. É importante salientar que, quanto menor o valor escolhido para o *shunt*, menor a sua influência sobre o restante do circuito, porém mais difícil torna-se a medição, uma vez que o valor da tensão obtida sobre o resistor *shunt* torna-se menor.

A corrente medida sobre o resistor *shunt* possui uma variação muito baixa, na faixa de (0 a 400) mA, aproximadamente, que considera a máxima corrente que o motor é capaz de drenar na sua partida ou quando o seu eixo encontra-se totalmente travado. Sendo assim, foi necessário realizar o condicionamento do sinal medido para efetuar a sua leitura na entrada do myRIO. Para isso, o valor de corrente adquirido a partir da medição foi amplificado com a utilização do *INA*122, um amplificador de instrumentação de precisão (TEXAS INSTRUMENTS, 1997). De acordo com a sua folha de dados, o ganho do amplificador é igual a

$$G = 5 + \frac{200\ \text{k}\Omega}{R_G}, \tag{4.7}$$

onde R_G é o resistor responsável pelo ajuste do ganho desejado.

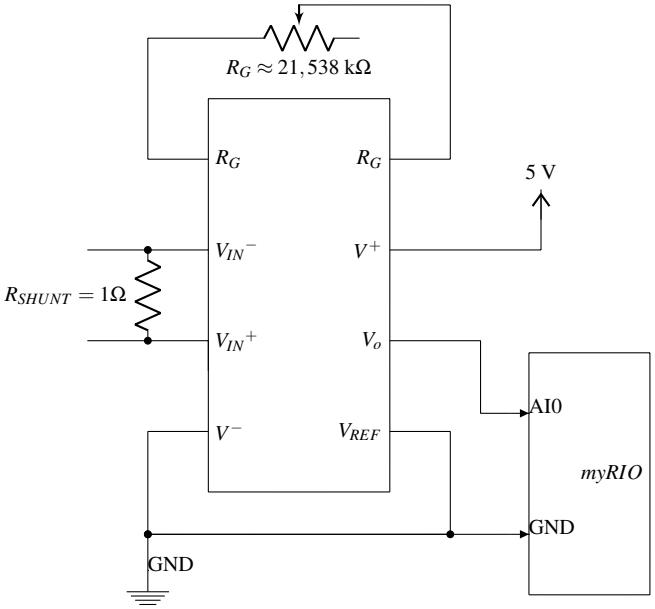
Com o objetivo de amplificar um valor de corrente de (0 a 400) mA para o intervalo de (0 a 5) V, o ganho do amplificador é igual a

$$G = 11,111. \tag{4.8}$$

Assim, o resistor R_G foi ajustado até que esse valor fosse medido na saída do amplificador. A Figura 36 apresenta o circuito responsável pela amplificação do valor medido de corrente no motor. A ligação entre o motor e o resistor *shunt* não é mostrada aqui, mas sim na subseção 4.2.3, que apresenta o circuito de acionamento do motor.

Como visto na subseção 4.2.1, o myRIO converte a variação de (0 a 5) V, medida em sua entrada analógica, em um dado digital variando de (0 a 4095).

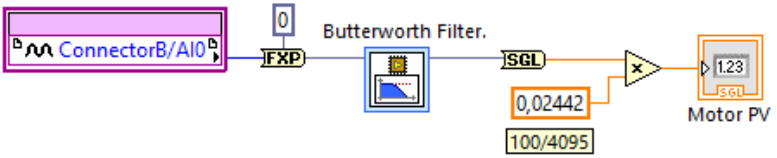
Figura 36 – Condicionamento de sinal para leitura da corrente no motor.



Fonte: Elaborado pelo autor.

Com isso, o sinal referente à medição de corrente fluindo sobre o motor foi condicionado novamente no LabVIEW FPGA, a fim de que o valor de leitura estivesse entre os limites de (0 a 100) % para a implementação do controle. Isso é apresentado na Figura 37.

Figura 37 – Condicionamento da leitura da corrente no motor no LabVIEW FPGA.



Fonte: Elaborado pelo autor.

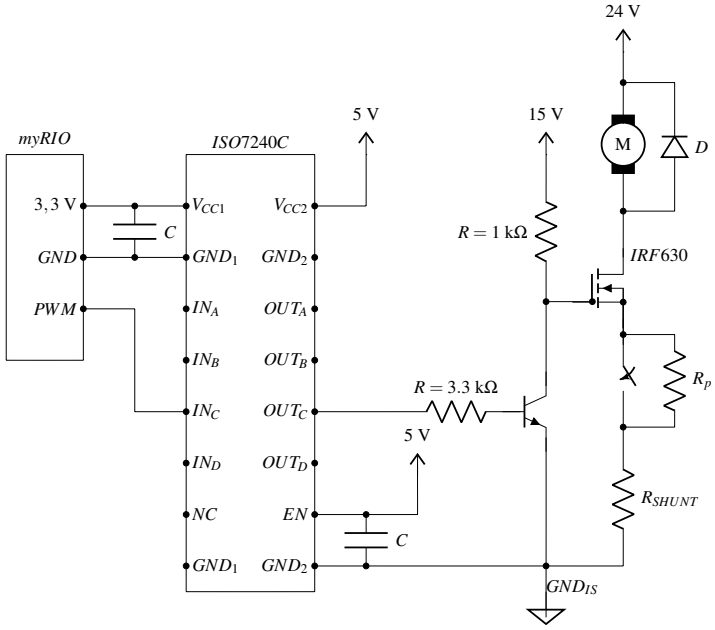
Como o sinal adquirido de um motor CC está sujeito a diferentes fontes

de ruídos e interferências, sejam eles do ambiente ou pela vibração do próprio motor, foi implementado um filtro, no LabVIEW FPGA, responsável por gerar sinais com melhor relação sinal/ruído para serem enviados ao controlador.

4.2.3 Acionamento do motor

O acionamento do motor foi realizado com o auxílio de uma fonte de alimentação, regulada em um valor que segue o padrão industrial, neste caso, 24 V. Já que o motor possui uma indicação de tensão de $27,5$ V, ele deve ser devidamente bem acionado com 24 V, porém, isso não garante que ele vá trabalhar em sua velocidade angular nominal. A Figura 38 apresenta o circuito construído com este objetivo, onde $C = 100$ nF.

Figura 38 – Circuito de acionamento do motor.



Fonte: Elaborado pelo autor.

Para o acionamento do motor por um sinal PWM, resultante da programação no LabVIEW FPGA e adquirido através de uma saída digital do myRIO,

viu-se a necessidade de isolação entre o myRIO e o restante do circuito eletrônico, pois qualquer sinal gerado que tenha um valor acima do máximo permitido pelo myRIO poderia danificar permanentemente as suas entradas ou saídas. Para tanto, foi implementado o isolador digital $ISO7240C$ (TEXAS INSTRUMENTS, 2007), que trabalha com alimentação de $3,3$ V e 5 V.

O MOSFET (transistor de efeito de campo de semicondutor de óxido metálico, do inglês *Metal Oxide Semiconductor Field Effect Transistor*) $IRF630$, neste caso, trabalha com a função de chaveamento a fim de acionar o motor CC. Como o sinal PWM proveniente do myRIO possui uma tensão de 5 V, foi necessário a inserção de um transistor auxiliar, neste caso o $2N2222$, com o objetivo de elevar o nível de tensão para 15 V, já que com uma tensão de 5 V no pino *gate* não se pode garantir que o MOSFET permanecerá fortemente polarizado em todas as situações de operação do motor, principalmente na partida e nos casos de altas cargas no eixo. Assim, ao pino de porta (*gate*) do MOSFET foi aplicada uma tensão de 15 V. Portanto, a partir desse circuito, o motor é acionado em uma determinada frequência, definida pela razão cíclica do sinal PWM.

A função do resistor R_{SHUNT}, que está exemplificado neste circuito, foi apresentada na subseção 4.2.2. O esquema envolvendo uma chave e o resistor R_p, em série com o resistor R_{SHUNT}, foi implementado para simular uma perturbação na corrente drenada pelo motor. O resistor R_p, igual a 27 Ω, foi escolhido de forma que, quando a chave estiver aberta e permitir o fluxo de corrente sobre ele, esse resistor exerça alguma influência no valor de corrente medido sobre o R_{SHUNT} e, consequentemente, sobre o motor. Com isso, é possível verificar se o controlador é capaz de rejeitar perturbações em regime permanente. Essa alteração de corrente é interessante, porque há uma malha interna (rápida) de controle de corrente. Logo, perturbações de corrente praticamente não devem influenciar na velocidade angular do motor.

4.3 COMENTÁRIOS FINAIS

Este capítulo apresentou a implementação de controle proposta neste trabalho. Relatou-se como foi realizada a programação no módulo LabVIEW FPGA para a implementação do controlador PI, incluindo os blocos e funções necessários e a *interface* gráfica construída neste caso. Em seguida, foram descritos os passos seguidos para o desenvolvimento do controlador GPC, compilado no processador com sistema de tempo real do myRIO, em que foi usado um algoritmo de programação quadrática de propósito geral, já disponível no LabVIEW RT, para obter o sinal de controle a cada iteração. Com isso, foram apresentados o diagrama de blocos e painel frontal implementados neste

trabalho. Nos dois módulos foi aplicado um método de medição do tempo de execução de um laço com o objetivo de medir o tempo de processamento do sistema. A última seção foi responsável por apresentar a implementação da estratégia de controle proposta na planta de laboratório escolhida DIGIAC 710, com o objetivo de controlar a velocidade angular e a corrente elétrica do motor a partir da atuação sobre a tensão aplicada sobre o mesmo. Então, foi mostrado como foram realizadas as medições da velocidade angular e da corrente, assim como o acionamento do motor. Com isso, o capítulo 5 apresentará os resultados experimentais obtidos a partir da implementação do sistema de controle proposto neste trabalho.

5 VALIDAÇÃO EXPERIMENTAL

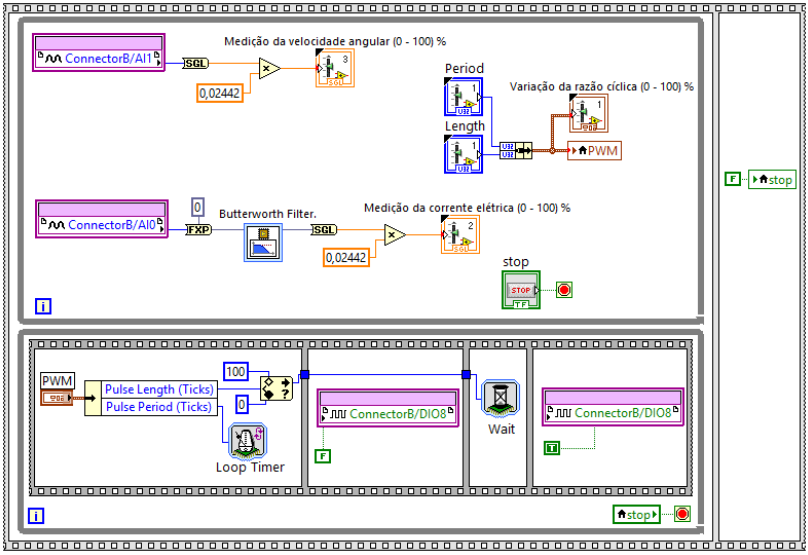
A partir da implementação dos controladores PI e GPC no ambiente LabVIEW, foi necessário realizar testes experimentais a fim de validar o funcionamento do sistema de controle aplicado ao estudo de caso. Como o objetivo do trabalho é o controle de duas variáveis, foram considerados dois modelos diferentes, definidos aqui como $G_i(s)$ e $G_e(s)$. O primeiro modelo, $G_i(s)$, baseia-se na relação entre a variação da razão cíclica de um sinal PWM e a corrente drenada pelo motor CC, ambos dentro dos limites de (0 a 100) %, a fim de obter um controle de corrente com a implementação do controlador PI. Por sua vez, o segundo modelo, $G_e(s)$, é determinando a partir da relação entre a corrente, adquirida do primeiro modelo, e a medição da velocidade angular do motor, variando de (0 a 100) % também, com o objetivo de controlá-la com um GPC. O controle de corrente em um motor CC é uma estratégia bastante adotada na prática como forma de compensar rapidamente perturbações nas condições de alimentação sem a necessidade de aguardar os efeitos dessas perturbações aparecerem na velocidade, justificando desta forma a aplicação do controle em cascata.

Na seção 5.1 deste capítulo é apresentado como foi realizada a determinação dos modelos citados, assim como as funções de transferência obtidas em cada caso. Em seguida, na seção 5.2, é mostrado como os controladores PI e GPC foram ajustados de modo a obter uma resposta satisfatória do sistema. Os resultados experimentais foram divididos em três partes: implementação dos controladores (seção 5.3), análise de rejeição de perturbações (seção 5.4) e resultados de tempo de cômputo do sinal de controle (seção 5.5). Na primeira parte, os ensaios realizados têm como objetivo comprovar que os controladores são capazes de garantir que as variáveis manipuladas sigam os valores de referência, assim como que os sinais de controle aplicados não ultrapassem os limites pré-definidos em projeto. A segunda parte baseia-se na constatação de que o sistema consegue rejeitar perturbações em regime permanente. A terceira e última parte traz os resultados experimentais no que diz respeito ao tempo de processamento de cada controlador, tanto no LabVIEW Real-Time quanto no FPGA, a fim de comprovar as vantagens de implementação dessa estratégia aplicada no myRIO.

5.1 DEFINIÇÃO DOS MODELOS PARA IDENTIFICAÇÃO DOS SISTEMAS

Neste trabalho, a determinação dos modelos foi realizada com o auxílio do módulo FPGA do LabVIEW, sendo que o diagrama de blocos referente a esse ensaio é apresentado na Figura 39.

Figura 39 – Diagrama de blocos de ensaio realizado para obtenção dos modelos.



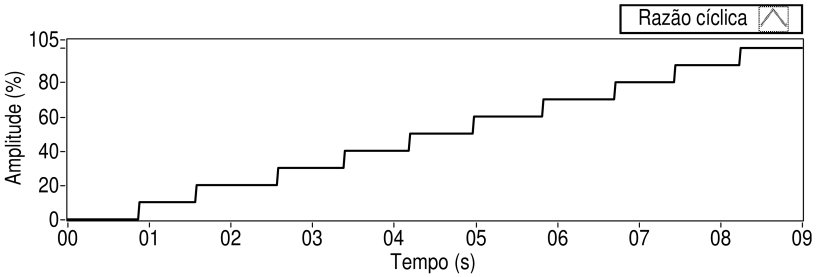
Fonte: Elaborado pelo autor.

Como é possível observar, o sinal PWM é gerado da mesma maneira como foi apresentado na subseção 4.1.1, com a diferença de que, neste caso, a variação da razão cíclica é definida manualmente ao longo do tempo do ensaio de identificação. As leituras relativas às medições de corrente e velocidade angular estão normalizadas no intervalo (0 a 100) % e são acompanhadas a partir de gráficos no painel frontal do LabVIEW.

O sinal de PWM e a corrente drenada pelo motor, que foram obtidos com o objetivo de definir o primeiro modelo, são apresentados nas Figuras 40 e 41, respectivamente.

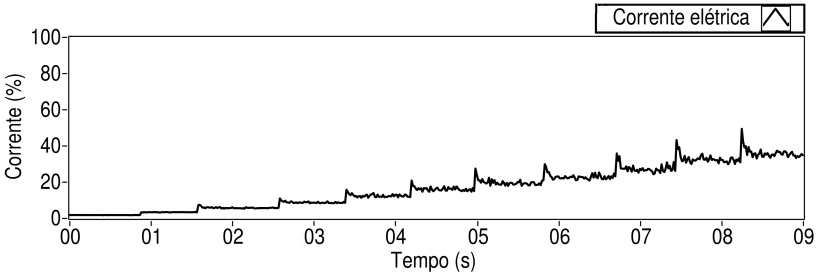
Neste caso, o valor da razão cíclica foi incrementado em passos de 10 %

Figura 40 – Variação da razão cíclica aplicada ao processo de identificação de $G_i(s)$ $(0\ -\ 100)$ %.



Fonte: Elaborado pelo autor.

Figura 41 – Medição da corrente elétrica drenada pelo motor para identificar $G_i(s)$ $(0\ -\ 100)$ %.



Fonte: Elaborado pelo autor.

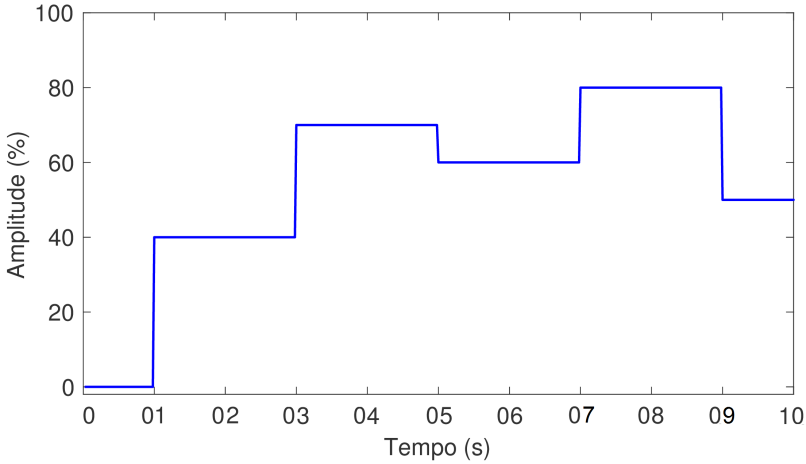
dentro do intervalo total, entre 0 % e 100 %. Assim, foi possível identificar o sistema relacionado a cada variação da razão cíclica, verificando-se que as constantes de tempo e os ganhos estáticos praticamente não se alteram em função do ponto de operação. Com base nisso, uma média da soma dos dados obtidos em cada identificação foi calculada e, por fim, definida a função de transferência que seja capaz de representar o comportamento de $G_i(s)$, a qual configura a relação entre o sinal PWM como entrada e a medição da corrente elétrica como saída:

$$G_i(s) = \frac{0,39 \cdot (7,5\ s+1)}{3\ s+1} = \frac{0,975 \cdot (s+0,13333)}{s+0,3333}, \tag{5.1}$$

onde o zero da função, em $(s = -0,13333)$, é responsável por representar a parcela de resposta instantânea. Essa função de transferência foi utilizada para o ajuste de ganhos do controlador PI.

Com o objetivo de comprovar que o modelo encontrado em (5.1) é capaz de representar o comportamento do sistema real, foi realizado um ensaio levando-se em consideração diferentes valores de amplitude do sinal PWM aplicado à entrada do sistema. A Figura 42 apresenta a variação da razão cíclica nesse caso.

Figura 42 – Sinal PWM aplicado.



Fonte: Elaborado pelo autor.

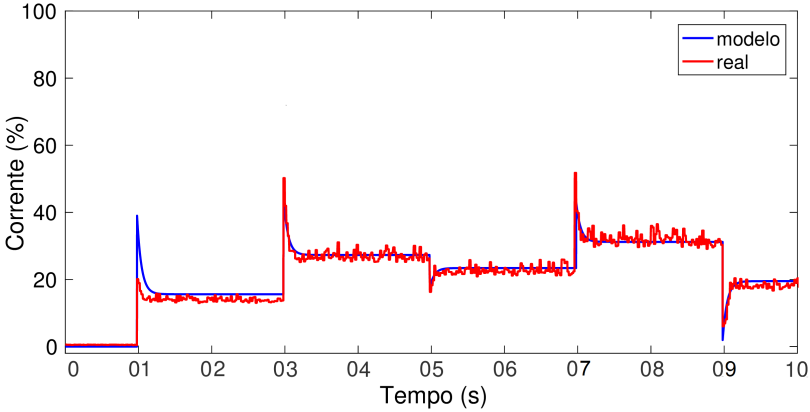
A Figura 43 mostra a comparação entre a resposta do modelo $G_i(s)$ e o comportamento do sistema real.

É possível observar que o modelo encontrado representa o processo real razoavelmente bem na maioria das faixas de operação. Como foi apresentado na seção 3.3, um motor CC pode apresentar não linearidades que podem afetar no seu comportamento, como mostrado na Figura 11. Observando a Figura 43, os pontos em que o comportamento real diverge do modelo de $G_i(s)$ devem-se às características não lineares do motor.

Já o modelo do processo $G_e(s)$ foi definido a partir da relação entre a corrente medida na Figura 41 e a velocidade angular do motor, cujo resultado é mostrado na Figura 44.

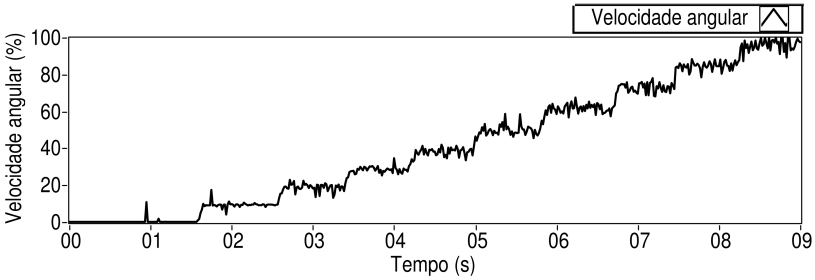
Entretanto, para essa identificação, inicialmente, foi determinado um modelo de representação, $G_\omega(s)$, da relação entre o sinal PWM como entrada e a medição da velocidade angular como saída. Neste caso, as constantes

Figura 43 – Comparação entre o modelo tensão-corrente e o comportamento do sistema real.



Fonte: Elaborado pelo autor.

Figura 44 – Medição da velocidade angular do motor para identificar $G_e(s)$ $(0 - 100)$ %.



Fonte: Elaborado pelo autor.

de tempo e os ganhos estáticos relativos a cada variação também possuem valores próximos de forma independente do ponto de operação. Portanto, a função de transferência responsável por representar o comportamento de $G_\omega(s)$ foi obtida a partir da média das somas dos valores encontrados em cada identificação, resultando em

$$G_\omega(s) = \frac{0,8075}{(s+1)\cdot(0,1\,s+1)} = \frac{8,075}{(s+1)\cdot(s+10)}\,. \tag{5.2}$$

Tomando como base as funções de transferência definidas em (5.1) e (5.2), é possível determinar o modelo de $G_e(s)$, ou seja, a relação entre a corrente elétrica como entrada e a velocidade angular como saída, a partir do quociente definido por:

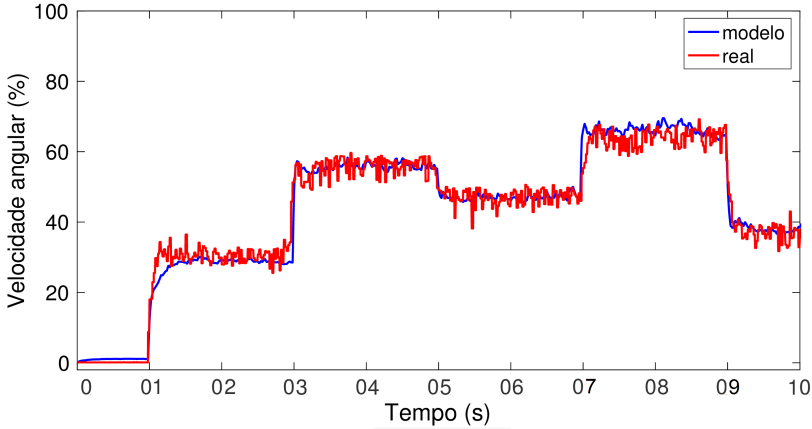
$$G_e(s) = \frac{G_\omega(s)}{G_i(s)} = \frac{0,8075}{(s+1)\cdot(0,1\,s+1)} \cdot \frac{(3\,s+1)}{0,39\cdot(7,5\,s+1)}\,, \tag{5.3}$$

com isso,

$$G_e(s) = \frac{2,0705\cdot(3\,s+1)}{(s+1)\cdot(0,1\,s+1)\cdot(7,5\,s+1)}\,. \tag{5.4}$$

Com o objetivo de comprovar que o modelo encontrado aproxima-se do comportamento do sistema real, o valor do sinal real correspondente à corrente elétrica, adquirido no ensaio de identificação apresentado na Figura 43, foi definido como o sinal de entrada de $G_e(s)$. Assim, foi possível realizar a comparação entre a velocidade angular encontrada a partir disso e o comportamento do sinal real da velocidade angular do sistema, como mostra a Figura 45.

Figura 45 – Comparação entre o modelo corrente-velocidade e o comportamento do sistema real.



Fonte: Elaborado pelo autor.

Neste caso, também observou-se que o modelo encontrado consegue representar o processo real razoavelmente bem na maioria das faixas de operação. Os pontos em que o comportamento real diverge do modelo de $G_e(s)$ devem-se

às características não lineares do motor.

Como o controlador GPC trabalha apenas com modelos de tempo discreto, foi necessário determinar uma representação de $G_e(s)$ no domínio z. Sabe-se que, neste caso, o tempo de assentamento para que a resposta do sistema em malha aberta atinja a faixa de 95 % do seu valor final ($t_{5\%}$) é de 2 s, aproximadamente. A determinação do período de amostragem considera o $t_{5\%}$ de malha fechada desejado, que foi definido da ordem de 50 % do valor de $t_{5\%}$ de malha aberta, ou seja, um tempo de assentamento de 1 s. Tipicamente, o período de amostragem é escolhido de modo a ser 20 vezes menor que o $t_{5\%}$ de malha fechada, portanto

$$T_s = \frac{1}{20} = 0,05\ s. \tag{5.5}$$

A partir disso, o modelo discreto de $G_e(s)$ com o segurador de ordem zero é representado por

$$G_e(z) = \frac{0,0087 \cdot (z-0.9835) \cdot (z+0.8355)}{(z-0,9934) \cdot (z-0.9512) \cdot (z-0.6065)}. \tag{5.6}$$

5.2 AJUSTE DOS CONTROLADORES

5.2.1 PI

Como foi apresentado na seção 2.1, existem três formas de representar um controlador PID: acadêmica (equação (2.1)), série (equação (2.2)) e paralela (equação (2.3)). Para a aplicação do controlador PI, que tem como objetivo atuar sobre o valor da corrente drenada pelo motor, foi levada em consideração a implementação dessa função no LabVIEW FPGA, onde os ganhos K_p e K_i estão representados na forma paralela e em tempo discreto.

Inicialmente, o controlador foi desenvolvido em tempo contínuo. Sendo o tempo de assentamento ($t_{5\%}$) de $G_i(s)$ de, aproximadamente, $1,5$ s, a partir disso, foi definida uma resposta de malha fechada desejada sem sobressinal e com um $t_{5\%}$ igual a 3 ms. Assim, o valor da constante de tempo para que a resposta do sistema de malha fechada atinja a faixa de $63,2$ % do seu valor final (τ) foi determinado da seguinte forma:

$$\begin{aligned} t_{5\%} &= 3 \cdot \tau \\ \tau &= \frac{t_{5\%}}{3} = 1 \text{ ms}. \end{aligned} \tag{5.7}$$

Com isso, o polo desejado de malha fechada é igual a

$$p = -\frac{1}{\tau} = -\frac{1}{1 \cdot 10^{-3}} = -1000 \ . \tag{5.8}$$

Usando o lugar das raízes, verificou-se que um controlador I de ganho integral K_i, com ganho alto, pode deslocar o polo do modelo da planta para a posição desejada e forçar um par polo-zero de baixa frequência, que quase não tem efeito na resposta. Assim,

$$C_i(s) = \frac{K_i}{s} \tag{5.9}$$

é suficiente para o controle.

Sabe-se que a função de transferência de malha fechada contendo o controlador $C_i(s)$ é

$$H_i(s) = \frac{C_i(s) \cdot G_i(s)}{1 + C_i(s) \cdot G_i(s)} = \frac{\frac{N_{C_i}}{D_{C_i}} \cdot \frac{N_{G_i}}{D_{G_i}}}{1 + \frac{N_{C_i}}{D_{C_i}} \cdot \frac{N_{G_i}}{D_{G_i}}} = \frac{N_{C_i} \cdot N_{G_i}}{D_{C_i} \cdot D_{G_i} + N_{C_i} \cdot N_{G_i}} \ , \tag{5.10}$$

onde N_{C_i} e N_{G_i} são os numeradores e D_{C_i} e D_{G_i} os denominadores de $C_i(s)$ e $G_i(s)$, respectivamente. O polinômio característico do sistema de malha fechada ($\sigma(s)$) é igual ao denominador de $H_i(s)$, então, tomando-se como base $G_i(s)$ em (5.1) e $C_i(s)$ em (5.9), tem-se que:

$$\sigma(s) = s \cdot (s + 0,3333) + K_i \cdot 0,975 \cdot (s + 0,13333) \ . \tag{5.11}$$

Igualando (5.11) a zero e substituindo s pelo valor de p encontrado em (5.8),

$$s \cdot (s + 0,3333) + K_i \cdot 0,975 \cdot (s + 0,13333) = 0$$
$$(-1000) \cdot ((-1000) + 0,3333) + K_i \cdot 0,975 \cdot ((-1000) + 0,13333) = 0 \ ,$$

obtém-se o valor do ganho do controlador:

$$K_i \approx 1025 \ . \tag{5.12}$$

Com isso,

$$C_i(s) = \frac{1025}{s} \ . \tag{5.13}$$

O sistema de malha fechada resultante da implementação de $C_i(s)$ é

igual a

$$H_i(s) = \frac{999.375 \cdot (s + 0,13333)}{(s + 999.575) \cdot (s + 0,13327)} \ . \tag{5.14}$$

Com base na equação (5.1), o zero em $s = -0,13333$ pertence à planta $G_i(s)$ e, consequentemente, aparecerá como zero de malha fechada tanto na resposta de seguimento de referência quanto de rejeição de perturbação de carga. Entretanto, quando há um polo e um zero muito próximos um do outro, sabe-se que os seus efeitos se compensam. Com isso, a partir da equação (5.14), apesar da dinâmica do polo em $s = -0,13327$ ser muito mais lenta que a dinâmica do polo em $s = -999.575$, a sua influência praticamente não é percebida, já que está ocorrendo um quase cancelamento do polo dominante com o zero em $s = -0,13333$.

A partir de dados experimentais, sabe-se que o número de ciclos de varredura do FPGA necessários para executar uma iteração do laço de cálculo do controlador PI é de 330 *ticks*. Como foi explicado na subseção 4.1.1, cada *tick* corresponde a 25 ns, logo o tempo de amostragem T_{s_i} deve ser de pelo menos

$$T_{s_i} = (330 \cdot 25) \text{ ns} = 8,25 \ \mu\text{s} \ . \tag{5.15}$$

Para a implementação desse controlador no LabVIEW FPGA, o valor de K_p foi definido igual a zero e foi necessário encontrar a representação de K_i em tempo discreto a partir do tempo de amostragem definido em (5.15). Sendo assim,

$$\begin{aligned} K_p &= K_c = 0 \ , \\ K_i &= K_i \cdot T_{s_i} = 1025 \cdot 8,25 \cdot 10^{-6} = 0,00845 \ . \end{aligned} \tag{5.16}$$

5.2.2 GPC

A sintonia do GPC consistiu na definição dos parâmetros de ajuste de forma a modificar a dinâmica da resposta referente à velocidade angular do motor, sendo eles: N_u, N, δ e λ. As ponderações de erro e do esforço de controle foram definidas como valores constantes ao longo dos horizontes, portanto serão expressas apenas como δ e λ, respectivamente, sem a dependência do instante j em relação ao início do horizonte. Além disso, será assumido $\delta = 1$, visto que o processo é monovariável, logo o que interessa é a relação entre as ponderações e não seus valores absolutos.

Como apresentado na subseção 2.2.1, existem algumas vantagens no que diz respeito à complexidade do problema quando o valor de N_u é pequeno. Portanto, neste caso, foi escolhido $N_u = 5$, que permite ao controlador ter

liberdade para ajuste do sinal de controle dentro de um horizonte razoável sem comprometer o tempo de cálculo de maneira a inviabilizar o período de amostragem pretendido. O valor de N foi determinado com base no $t_{5\%}$ de malha aberta e no período de amostragem, definido em (5.5). Com isso, de forma a possibilitar observar toda a resposta transitória, além de uma parcela da resposta a partir do momento em que ela atinge o regime permanente, foi escolhido $N = 20$.

Com esses parâmetros definidos, o ajuste de δ e λ foi realizado com a finalidade de adequar a relação de importância entre erro e esforço de controle, para que haja um equilíbrio entre os dois termos da função custo (2.6). Portanto, assumiu-se que o erro possui uma variação que vai desde 10 % até 0 % dentro do horizonte de predição N, ou seja, o valor médio do erro é da ordem de

$$e_{med} = 5\%.$$

Como o valor do ganho estático de $G_e(s)$ é igual a $2,07$, estima-se que a variação do controle é igual a

$$c_{med} = \frac{5\%}{2,07} \approx 2,4\%.$$

Com isso, a função objetivo em (2.6) pode ser reescrita da seguinte maneira:

$$J = \sum_{j=1}^{20} \delta (e_{med})^2 + \sum_{j=1}^{5} \lambda (c_{med})^2. \tag{5.17}$$

Assumindo o valor de $\delta = 1$, ou seja,

$$J = \sum_{j=1}^{20} (1)(5)^2 + \sum_{j=1}^{5} \lambda (2,4)^2,$$

é possível encontrar um λ inicial, como se segue, para ter um peso equivalente dos erros e do controle na função custo:

$$\lambda (5)(2,4)^2 = (20)(1)(5)^2$$

$$\lambda = (1)\left(\frac{20}{5}\right)\left(\frac{5}{2,4}\right)^2$$

$$\lambda = 17,36\ . \tag{5.18}$$

Valores maiores ou menores que esse podem ser então ajustados para buscar uma resposta adequada regulando-se o peso da ação de controle.

5.3 RESULTADOS EXPERIMENTAIS DE CONTROLE

A partir do ajuste dos controladores, foram realizados ensaios com o objetivo de validar o funcionamento do sistema de controle em cascata. O primeiro ensaio foi executado a fim de comprovar que tanto o controlador PI quanto o GPC são capazes de garantir que as variáveis manipuladas, corrente e velocidade angular, sigam o valor determinado pela referência de cada malha.

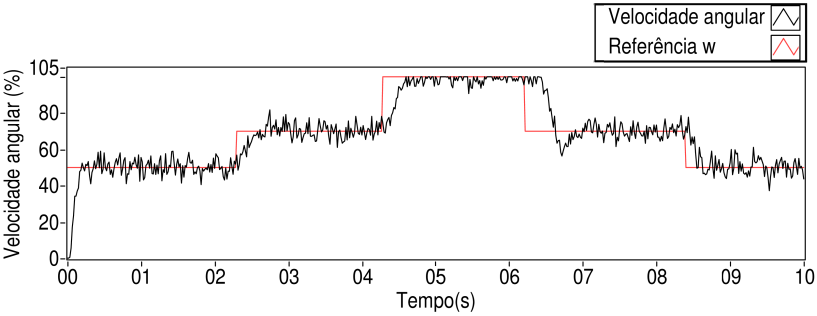
A Figura 46 apresenta a comparação entre o valor de referência e a medição da velocidade angular do motor, ambos com variação de (0 a 100) %, que é o resultado da implementação do controle GPC no LabVIEW Real-Time. É importante salientar que, de acordo com a equação (4.6), o limite máximo (100 %) que a velocidade do motor pode atingir é de 2142 rpm. A Figura 47 mostra o sinal de controle $u(k)$ gerado para controlar a resposta do sistema neste caso.

No instante inicial do ensaio apresentado na Figura 46, o motor encontrava-se em repouso e a referência $w(k)$ (em vermelho) foi definida em um valor igual a 50 % ($\approx$ 1072 rpm). A partir do momento em que o motor foi alimentado com a fonte de 24 V, a medição da velocidade angular (em preto) começou a aumentar até atingir o valor de referência. Com o objetivo de comprovar que o sitema da malha externa era capaz de trabalhar em diferentes níveis de amplitude, posteriormente, foram definidas novas referências em 70 % ($\approx$ 1500 rpm) e 100 % ($\approx$ 2142 rpm). Em seguida, foram definidos valores decrescentes de referência, sendo possível confirmar que a resposta do sistema atendeu a todos os requisitos nestes casos também. Conforme foi apresentado na seção 4.2, o sinal de controle $u(k)$ da Figura 47, gerado pelo controlador GPC, foi aplicado à malha interna em forma de variável de processo com o objetivo de controlar a corrente drenada pelo motor através da implementação do controlador PI.

Com isso, o ensaio realizado para verificação do comportamento da resposta de malha interna é apresentado na Figura 48 e na Figura 49 é mostrado o sinal de controle aplicado ao sistema da malha interna.

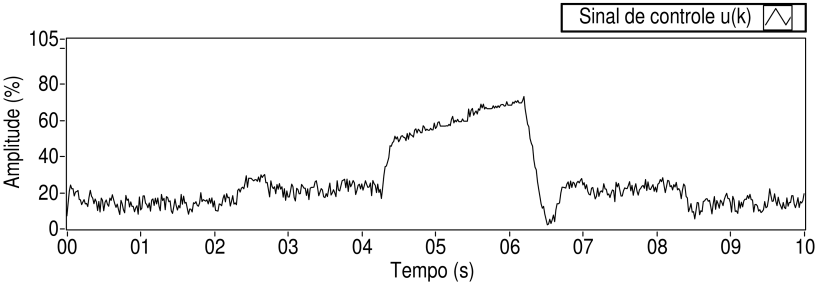
É possível verificar que o controle PI foi capaz de garantir que a corrente conseguisse seguir o valor da variável manipulada da malha externa nas condições em que isso era fisicamente possível. É importante salientar que a corrente atinge um valor de cerca de 165 mA com o motor rodando a vazio. Entretanto, de acordo com a subseção 4.2.2, o limite máximo (100 %) que a corrente pode alcançar é de, aproximadamente, 400 mA no momento de partida do motor ou então quando o seu eixo estiver totalmente travado. Portanto, como pode ser observado na Figura 48, mesmo que o sinal de controle $u(k)$ alcance o limite máximo, a medição da corrente elétrica não será capaz de atingir esse valor se não houver carga no eixo do motor. Além disso,

Figura 46 – Comparação entre referência e medição da velocidade angular do motor $(0\ -\ 100)$ %.



Fonte: Elaborado pelo autor.

Figura 47 – Sinal de controle $u(k)$ $(0\ -\ 100)$ %.

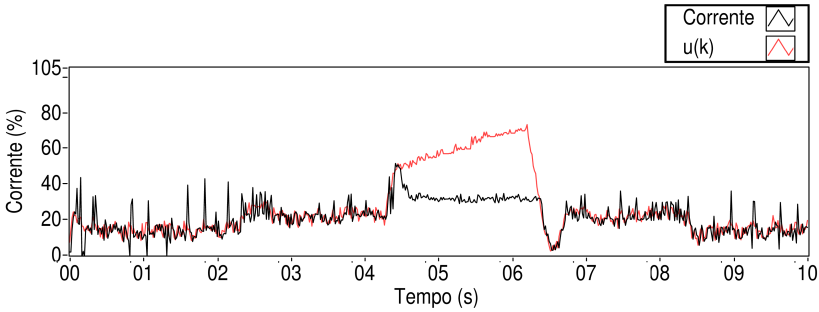


Fonte: Elaborado pelo autor.

constata-se que o sinal de controle PI atendeu aos limites pré-definidos do sinal PWM a ser aplicado no sistema, ou seja, manteve-se entre 0 % e 100 %.

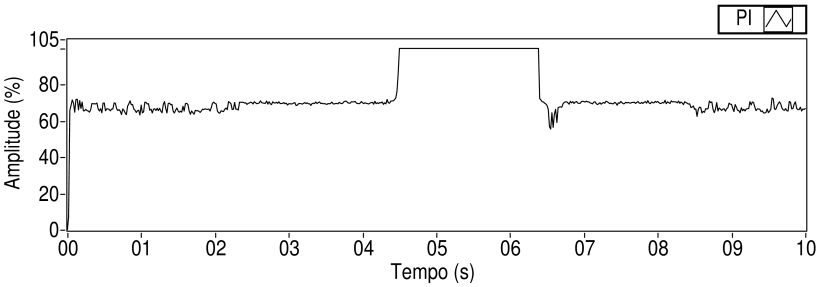
É possível observar que está ocorrendo *windup* da ação integral do GPC. O controlador GPC verifica que a velocidade angular não atingiu o valor desejado de 100 % e atua para que a referência de corrente aumente. Entretanto, a corrente não é capaz de acompanhar o valor de referência enviado pela malha externa, dado que o sinal de controle da malha interna está saturado. Com isso, não ocorrem mudanças na velocidade angular, mas a referência continua crescendo e só diminui quando ocorre a mudança de referência para um valor menor, como se observa na figura logo após o tempo de 6 s. Assim, tem-se um efeito de acumulação da ação integral do GPC (ou *windup*). Na

Figura 48 – Comparação entre referência e medição da corrente no motor $(0 - 100)$ %.



Fonte: Elaborado pelo autor.

Figura 49 – Sinal de controle PI $(0 - 100)$ %.



Fonte: Elaborado pelo autor.

Figura 46 se observa então que é necessário descarregar esta integral depois de mudada a referência de velocidade angular, atrasando a mudança da referência de corrente.

Para contornar este problema, o controlador GPC deveria conhecer no seu modelo de predição o efeito que o comportamento da variável que realmente satura, que é o sinal de controle da malha interna, tem sobre a variável de decisão da malha do controlador GPC, que é a referência de corrente. Ou seja, a solução do problema passaria por realizar um mapeamento da restrição da variável da malha interna para a externa, para o qual é necessário construir um modelo do motor que permita levar em consideração seu estado de carga. Este problema, apesar de identificado, não foi tratado neste trabalho,

tornando-se uma sugestão para trabalhos futuros.

5.4 ANÁLISE DE REJEIÇÃO DE PERTURBAÇÕES

Perturbações são variáveis que não estão incluídas no modelo que descreve o sistema, mas que são capazes de afetar o seu comportamento. Elas podem ser consideradas determinísticas, ou seja, aquelas que podem ser previstas, ou estocásticas, aquelas cujo estado é indeterminado, como ruídos em sensores ou atuadores (VISIOLI, 2006). De acordo com Cholakkal (2009), o fato de considerar perturbações no projeto de um controlador torna o sistema mais rápido no que diz respeito à rejeição das mesmas. Com isso, neste trabalho, foram consideradas duas perturbações, sendo a primeira através da inserção de uma carga no eixo do motor e a segunda a partir da variação da corrente drenada pelo motor.

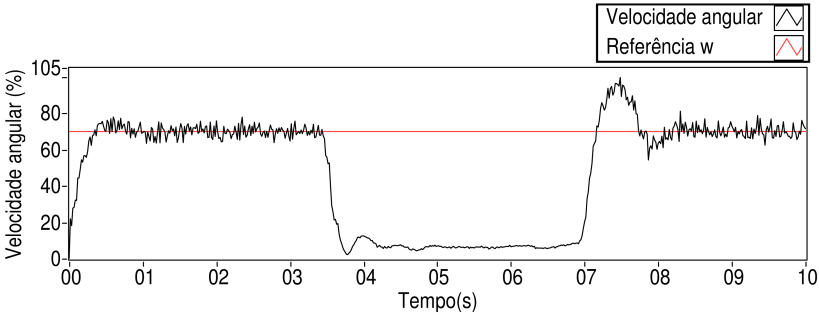
Conforme foi explicado na subseção 4.2.2, a variação da corrente atinge o valor máximo de, aproximadamente, 400 mA quando o seu eixo encontra-se totalmente travado. Assim, foram realizados dois ensaios, sendo o primeiro para analisar o comportamento do sistema quando o eixo do motor encontra-se quase totalmente travado e o segundo com o objetivo de verificar se o sistema de controle implementado é capaz de rejeitar uma perturbação através da inserção de uma carga leve, ou seja, com menos torque resistivo, aplicada no eixo do motor.

No primeiro ensaio, portanto, inicialmente, a referência de velocidade angular do motor, $w(k)$, foi definida em 70 %. Após um certo período, foi inserida uma carga no sistema, travando o eixo do motor quase completamente, fazendo com que a velocidade angular diminuísse. Imediatamente, o controlador GPC atuou no sistema, aumentando o sinal de controle $u(k)$ em seu valor máximo (100 %) a fim de buscar a referência de velocidade. Em seguida, a carga foi retirada, ou seja, o motor passou a rodar a vazio outra vez, fazendo com que a sua velocidade angular aumentasse. Quando a velocidade angular do motor aproximou-se do valor de referência, o controlador diminuiu a ação de controle, assentando novamente o sistema na referência desejada. A resposta referente à medição da velocidade angular neste ensaio é apresentada na Figura 50. A Figura 51 apresenta a comparação entre o sinal de controle $u(k)$ e a medição da corrente elétrica drenada pelo motor.

Como é possível observar, no instante em que ocorreu a perturbação de carga, a medição da corrente se elevou até alcançar o valor máximo de 400 mA, equivalente ao 100 % da amplitude de corrente. Com isso, quando o eixo do motor encontra-se travado, a corrente consegue seguir o valor de referência estipulado por $u(k)$ sem ocorrer a limitação apresentada na Figura

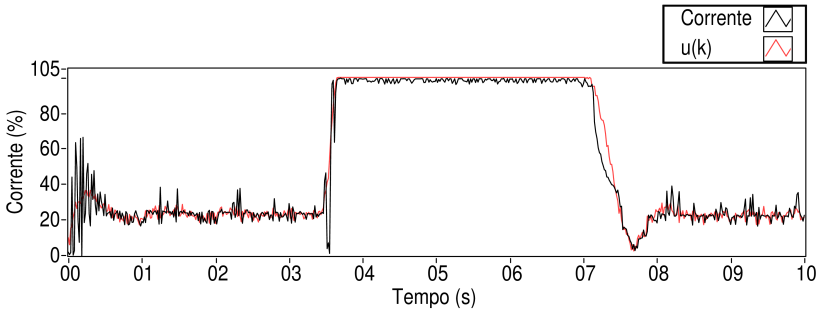
48. Neste caso, a restrição no sinal de controle do GPC é importante, pois evita que a referência passe de 100 % e, consequentemente, o fenômeno *windup* também é evitado. A Figura 52 apresenta o sinal de controle PI aplicado neste caso, o qual também foi capaz de se manter entre 0 % e 100 %.

Figura 50 – Medição da velocidade angular devido ao travamento do eixo do motor.



Fonte: Elaborado pelo autor.

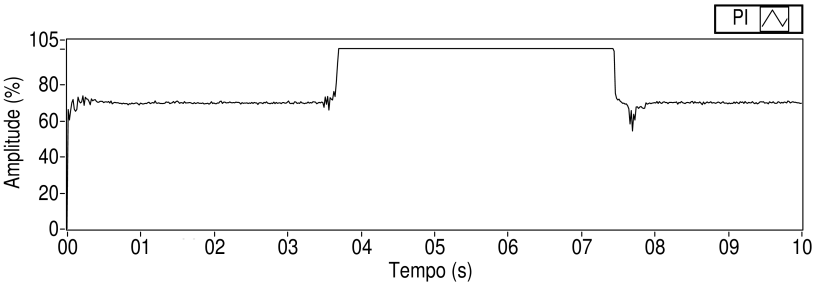
Figura 51 – Comparação entre sinal de controle $u(k)$ e medição da corrente devido ao travamento do eixo do motor.



Fonte: Elaborado pelo autor.

Para o segundo ensaio, foi definida a referência inicial de $w(k) = 80$ %. A partir de um certo período, foi inserida uma carga leve no eixo do motor. A Figura 53 traz a medição da velocidade angular e, como é possível constatar, o sistema de controle implementado conseguiu rejeitar a perturbação, já que a velocidade angular foi mantida na referência mesmo

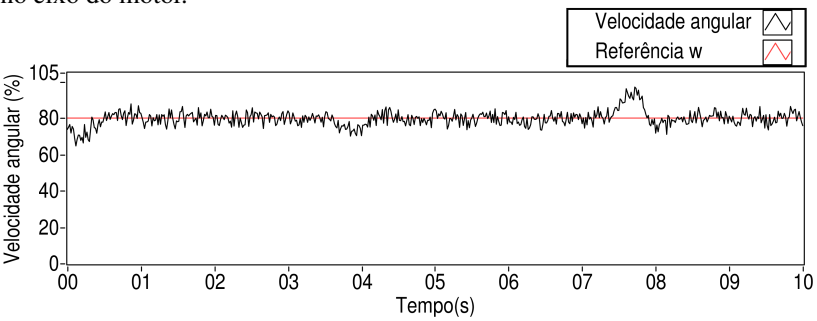
Figura 52 – Sinal de controle PI devido ao travamento do eixo do motor.



Fonte: Elaborado pelo autor.

durante a atuação da perturbação. Os únicos pontos em que há uma divergência entre a referência $w(k)$ e a velocidade angular do motor são aqueles nos quais a perturbação é inserida ou retirada (aproximadamente em 4 s e $7,5$ s). A Figura 54 apresenta o sinal de controle $u(k)$ que foi aplicado ao sistema para que a medição da velocidade angular do motor se mantivesse seguindo a referência e, consequentemente, rejeitasse a perturbação.

Figura 53 – Medição da velocidade angular devido a uma perturbação de carga no eixo do motor.

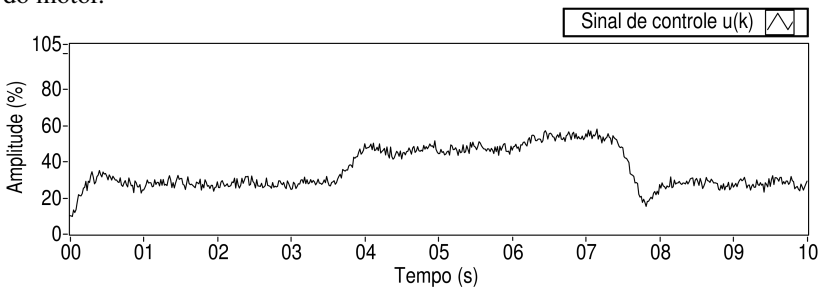


Fonte: Elaborado pelo autor.

A Figura 55 mostra o sinal de medição da corrente elétrica, que foi capaz de seguir a referência dada pelo controlador GPC.

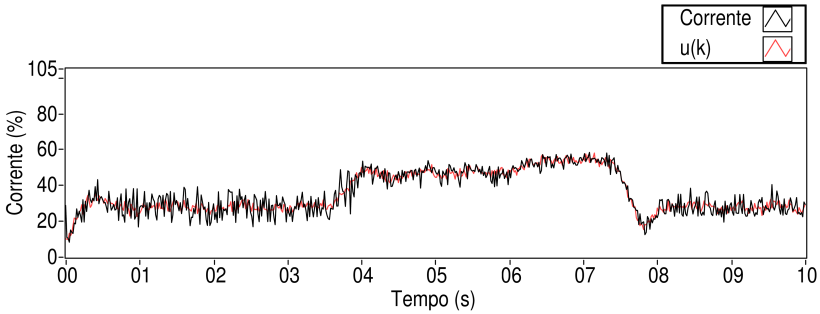
O próximo ensaio foi realizado com o objetivo de verificar se o sistema implementado é capaz de rejeitar uma perturbação na corrente drenada pelo motor. Como foi apresentado na subseção 4.2.3, para este experimento, foi

Figura 54 – Sinal de controle $u(k)$ devido a uma perturbação de carga no eixo do motor.



Fonte: Elaborado pelo autor.

Figura 55 – Comparação entre sinal de controle $u(k)$ e medição da corrente devido a uma perturbação de carga no eixo do motor.



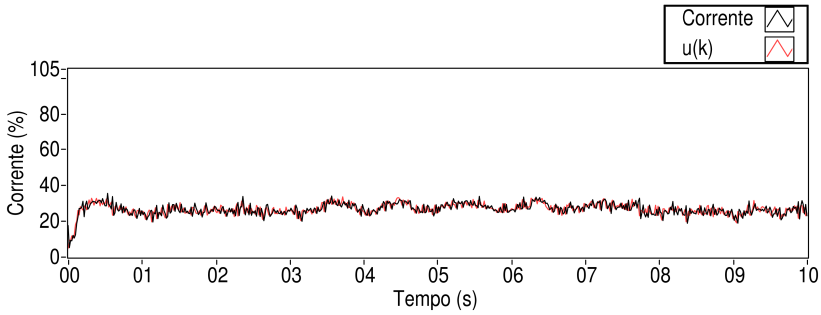
Fonte: Elaborado pelo autor.

inserido um resistor em série com o R_{SHUNT} e com o motor através do acionamento de uma chave para que ele exerça alguma influência no valor de corrente.

Então, para este ensaio, foi definida uma referência inicial de velocidade angular do motor $w(k) = 80$ % e, a partir de um certo tempo, foi acionada a chave inserindo o resistor no circuito. A Figura 56 apresenta o comportamento da corrente elétrica devido a uma perturbação e a Figura 57 traz o sinal de controle PI aplicado para que a malha interna fosse capaz de rejeitar esta perturbação.

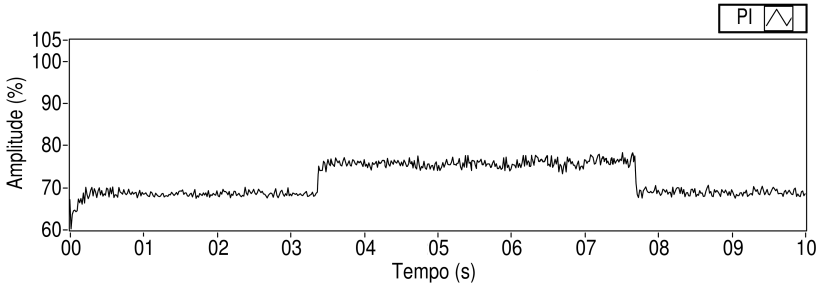
Como é possível observar, o controlador PI atua rapidamente modifi-

Figura 56 – Comparação entre sinal de controle $u(k)$ e medição da corrente devido a uma perturbação de corrente.



Fonte: Elaborado pelo autor.

Figura 57 – Sinal de controle PI devido a uma perturbação de corrente.



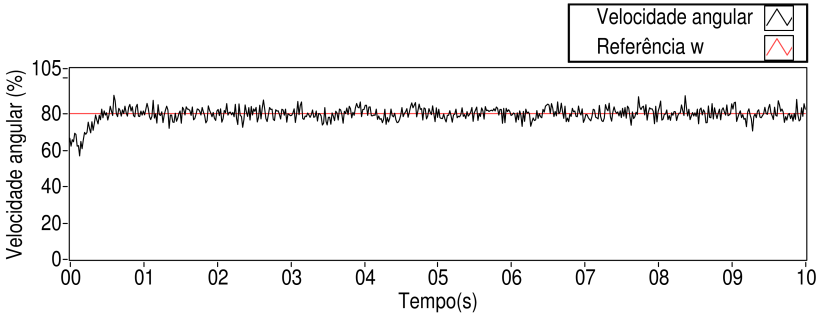
Fonte: Elaborado pelo autor.

cando o valor da razão cíclica a fim de compensar a queda de corrente medida quando a chave é ativada.

Como foi explicado na subseção 4.2.3, devido à presença de uma malha interna rápida de controle de corrente, uma perturbação na corrente não deve apresentar grandes alterações na velocidade angular e, consequentemente, no sistema de controle da malha externa. Com isso, a Figura 58 mostra que, mesmo durante a ocorrência da perturbação, a medição manteve-se seguindo a referência, sem alterações no valor da velocidade angular do motor.

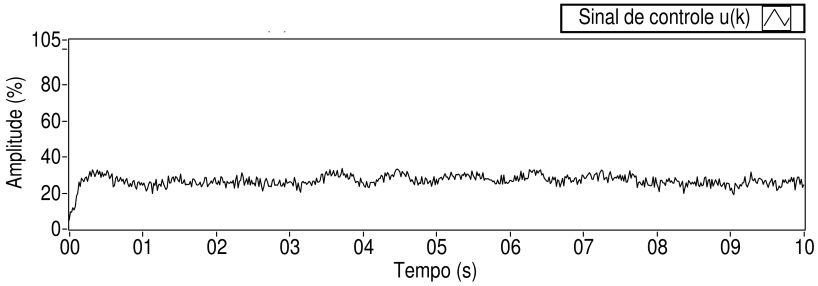
A Figura 59 apresenta o sinal de controle da malha externa, $u(k)$, o qual também não apresentou mudanças com a aplicação da perturbação de corrente. Isso significa que o sistema de controle da malha interna foi capaz

Figura 58 – Medição da velocidade angular devido a uma perturbação de corrente.



Fonte: Elaborado pelo autor.

Figura 59 – Sinal de controle $u(k)$ aplicado devido a uma perturbação de corrente.



Fonte: Elaborado pelo autor.

de rejeitar uma perturbação na corrente de alimentação do motor antes que ela apresentasse alguma influência sobre a velocidade angular do motor.

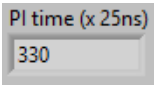
5.5 RESULTADOS EXPERIMENTAIS DE TEMPO DE CÔMPUTO

5.5.1 PI

Com o objetivo de analisar as vantagens que um controlador PI implementado em FPGA apresenta sobre a implementação em um microcontrolador

no que diz respeito ao tempo de processamento, foi realizado um ensaio para verificar o tempo de cômputo do controlador PI, como apresentado nas Figuras 15 e 17, na subseção 4.1.1. A Figura 60 apresenta dados adquiridos do Painel Frontal do LabVIEW FPGA contendo o tempo de cálculo do controlador PI.

Figura 60 – Tempo de cômputo do controlador PI no LabVIEW FPGA.



Fonte: Elaborado pelo autor.

De acordo com a equação (5.15), é possível afirmar que o tempo de cômputo do controlador PI implementado neste trabalho é de $8,25\ \mu$s, constante ao longo do ensaio experimental, o que permite controlar sistemas com dinâmicas ainda mais rápidas que a corrente do motor. Esse valor na unidade de microssegundos é esperado devido à implementação estar embarcada em um FPGA.

Em microcontroladores, o tempo de aquisição e conversão de dados depende da frequência de *clock* de cada um. Por exemplo, o microcontrolador ATmega32 (ATMEL, 2011) possui uma frequência de 16 MHz e uma variação do tempo de conversão A/D de (13 a 260) μs, já no MSP430 (TEXAS INSTRUMENTS, 1999) o clock é de 1 MHz e o tempo da conversão A/D de 12 μs. Em relação à distribuição de tempo de cálculo em cada iteração nas tarefas básicas realizadas pelo controlador, neste caso, o FPGA estipulou cerca de $(80 \cdot 25)$ ns $=$ 2 μs para aquisição dos dados referentes as entradas analógicas do myRIO, além da conversão A/D desses dados, e $(250 \cdot 25)$ ns $=$ $6,25\ \mu$s para o cômputo do sinal de controle efetivamente. Tendo em vista que apenas o tempo da aquisição de dados realizada por um microcontrolador já é maior que o tempo de toda a implementação do controlador PI no FPGA, incluindo a aquisição de dados, conversão A/D e cálculo do sinal de controle propriamente dito, neste caso, é possível afirmar que o sistema de controle em FPGA é capaz de rodar algumas dezenas de vezes mais rápido que um microcontrolador poderia fazer, já que o FPGA trabalha como um *hardware* dedicado e os cálculos são realizados diretamente no *hardware* que gerencia as aquisições, poupando bastante tempo de comunicação.

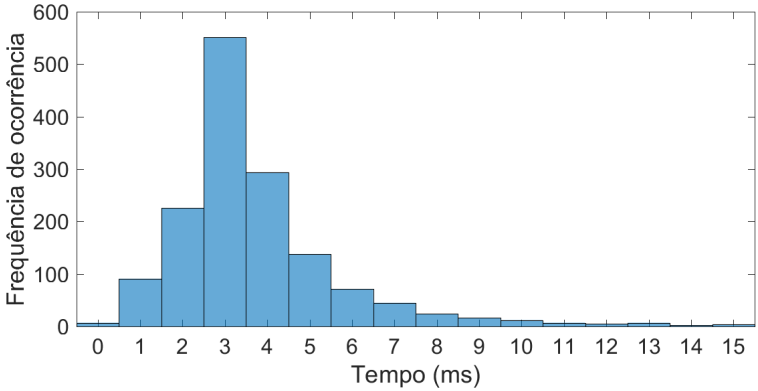
5.5.2 GPC

Para análise de tempo de cômputo do controlador GPC implementado no LabVIEW Real-Time, foram eliminados os gráficos de resultados do Painel

Frontal, já que eles são atualizados a cada período de amostragem e, consequentemente, influenciam no tempo de processamento do controlador. Neste trabalho, como é implementado um algoritmo de otimização de propósito geral, o tempo de cálculo depende do estado do sistema, principalmente o fato de haver ou não uma restrição ativa. Sendo assim, foram realizados dois ensaios para verificar o tempo de cômputo da lei de controle.

No primeiro, é definida uma referência inicial de $w(k) = 60\ \%$ para a medição da velocidade angular, sendo que a restrição de desigualdade no sinal de controle não é ativada neste caso. Na Figura 61, o histograma apresenta o tempo de aquisição de dados somado ao tempo de cálculo do controlador GPC em cada amostra ao longo de 1500 amostras.

Figura 61 – Tempo de aquisição de dados mais cômputo do controlador GPC no LabVIEW RT sem restrição ativa.



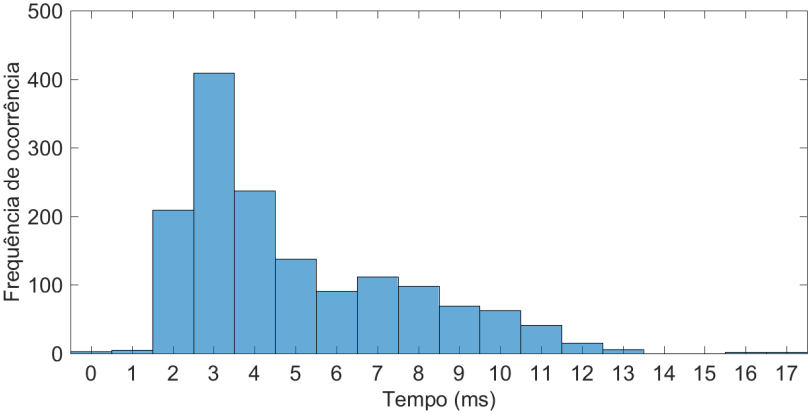
Fonte: Elaborado pelo autor.

Como é possível observar, o valor mínimo é igual a 0 ms, ou seja, em 7 amostras o tempo de cálculo do GPC é da faixa de microssegundos, e o valor máximo é de 15 ms, com ocorrência em 4 amostras. A média de todas as amostras de tempo é igual a $3,716$ ms.

O segundo ensaio foi realizado levando-se em consideração uma situação em que a restrição no sinal de controle está ativa. A Figura 62 apresenta o histograma referente a esse caso, no qual foram coletadas 1500 amostras, assim como ocorreu no primeiro ensaio.

De acordo com a figura, é possível observar que o intervalo de tempo de aquisição de dados mais o tempo de cômputo do controlador é maior em relação ao caso no qual não há restrição no sinal de controle ativa. O valor mínimo é da ordem de microssegundos, em 3 amostras, já o tempo máximo

Figura 62 – Tempo de aquisição de dados mais cômputo do controlador GPC no LabVIEW RT com restrição ativa.



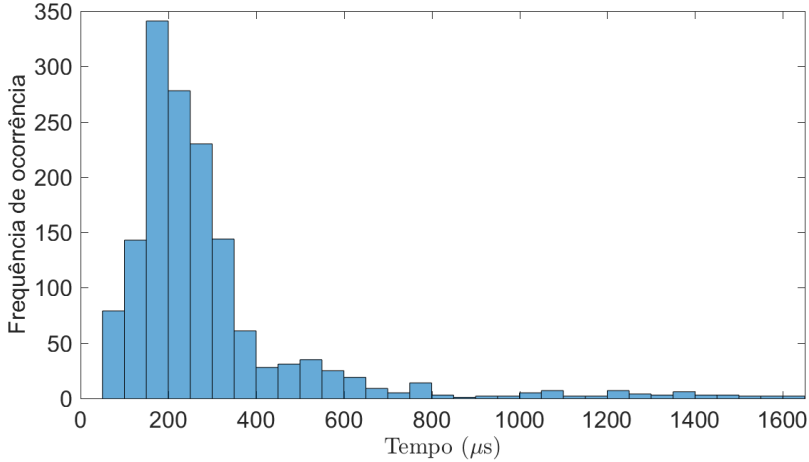
Fonte: Elaborado pelo autor.

de cálculo é de 17 ms, com ocorrência em 2 amostras. A escolha do mínimo período de amostragem do controlador deve ser feita por esse valor. A média do total de amostras de tempo é $4,952$ ms, que confirma o fato de que a presença de uma restrição no sistema demanda mais tempo para o controlador realizar os seus cálculos.

Também foi possível verificar neste ensaio que o tempo demandado somente para a aquisição de dados é bem menor quando comparado ao tempo de cômputo do controlador. A Figura 63 apresenta o histograma referente a esses dados, em que o valor mínimo é de 100 μs, o máximo é igual a $1,6$ ms e a média dos tempos é de $296,88$ μs.

A fim de estabelecer um ponto de referência para comparação dos tempos, o mesmo *software* responsável por implementar o controlador GPC no LabVIEW Real-Time foi desenvolvido também em um computador pessoal (PC, do inglês *Personal Computer*) com sistema operacional Windows 7, processador Intel Core i5 e memória RAM de 8 GB, a partir do LabVIEW padrão, otimizado da mesma forma que o controlador que executa no Real-Time e também com divisão em processos para fazer melhor uso de sistemas com múltiplos processadores. Além disso, foi desenvolvido um outro *software* específico para determinação do tempo de aquisição de dados em um PC, que foi realizada com o auxílio de uma placa multifunção USB, modelo NI USB 6009 (NATIONAL INSTRUMENTS, 2015b). Para essa comparação, em ambos os casos, foram empregadas configurações extremamente restritivas de parada para o algoritmo de programação quadrática, com tolerância absoluta de 10^{-8}

Figura 63 – Tempo de aquisição de dados no LabVIEW RT com restrição ativa.



Fonte: Elaborado pelo autor.

e um limite máximo de 10^4 iterações, os mesmos empregados para gerar todos os dados apresentados nesta seção. A Figura 64 apresenta o histograma referente ao tempo de aquisição de dados somado ao tempo de cálculo do controlador GPC no PC, contendo 1500 amostras também.

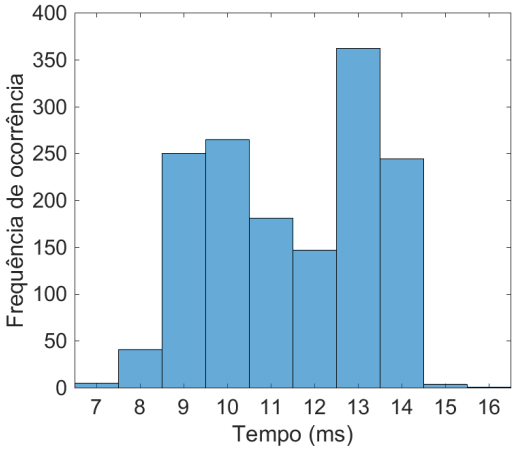
De acordo com a figura, o tempo mínimo é de 7 ms, com frequência de ocorrência em 5 amostras, e o máximo valor é igual a 16 ms, em apenas 1 amostra. A média dos tempos é de $11,88$ ms.

Neste ensaio, foi possível verificar que o tempo de cálculo do controlador GPC é bem inferior se comparado ao tempo de aquisição de dados, como mostra a Figura 65, sendo o mínimo de 350 μs e o máximo de 590 μs, com uma média de tempos igual a $403,64$ μs.

De forma a facilitar a comparação entre a implementação no PC e no myRIO, a Tabela 1 apresenta os dados adquiridos referentes aos ensaios descritos nesta seção. É importante ressaltar que os dados da implementação no PC foram obtidos em ensaios separados, entretanto a tabela apresenta a soma dos dois tempos, aquisição de dados e cômputo do controlador GPC, em um único termo.

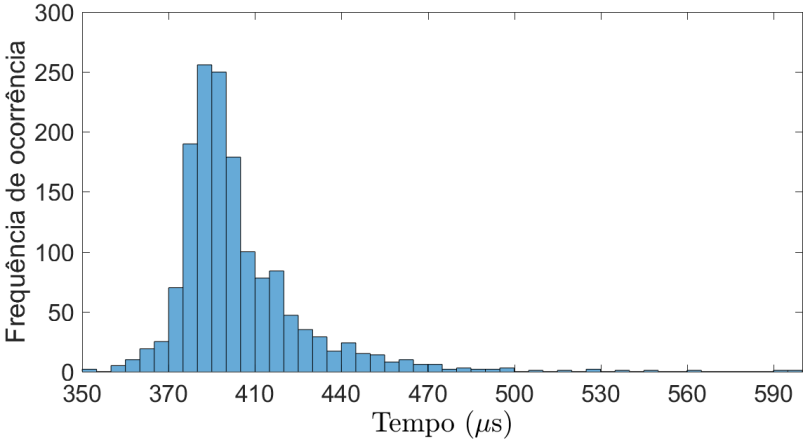
A partir desses valores, nota-se que o tempo médio de processamento apresenta uma vantagem para a implementação no sistema de tempo real, porém a variabilidade em torno do valor médio é maior nesse sistema, o que faz com que o tempo máximo, tipicamente empregado para definição

Figura 64 – Tempo de aquisição de dados mais cômputo do controlador GPC no LabVIEW padrão - PC.



Fonte: Elaborado pelo autor.

Figura 65 – Tempo de cômputo do controlador GPC no LabVIEW padrão - PC.



Fonte: Elaborado pelo autor.

do período de amostragem, seja maior que o obtido no PC. Esse aparente contrassenso de a implementação em tempo real apresentar maior variabilidade pode ser explicado pelas análises da distribuição do tempo de cálculo em cada

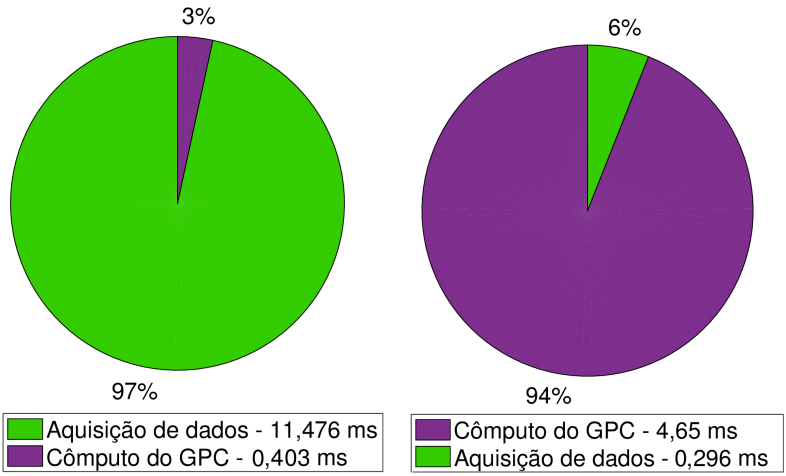
Tabela 1 – Comparação entre a implementação no PC e no myRIO em relação ao tempo de processamento.

Hardware	Tempo de um ciclo de controle (ms)	
	Médio	Máximo
myRIO	4,95	17,00
PC	11,88	16,59

Fonte:Elaborado pelo autor.

iteração nas tarefas básicas realizadas pelo controlador. A Figura 66 apresenta a comparação referente a essa distribuição do tempo de cálculo no PC e no myRIO.

Figura 66 – Distribuição de tempo de cálculo no (a) PC e no (b) myRIO.



(a) PC (b) myRIO

Fonte: Elaborado pelo autor.

Com base na figura, é possível afirmar que, no PC, o tempo é dividido em 97 % para aquisição de dados e apenas 3 % para cômputo do sinal de controle. Já no myRIO/ a proporção de tempo é de, aproximadamente, 6 % para aquisição e 94 % para cálculo do sinal de controle. Além disso, observou-se que a tarefa que domina o tempo de cálculo do sinal de controle nos dois

casos é a resolução do problema de programação quadrática.

Como o problema de programação quadrática é resolvido com um algoritmo iterativo que tem um tempo de execução variável em função das condições do problema, essa variabilidade aparece de forma acentuada no myRIO, porém não é expressiva no PC, dado que em todos os casos o cálculo é realizado de forma rápida. Com isso, é possível afirmar que se for implementado um algoritmo de cômputo rápido e não um otimizador de propósito geral, como neste caso, os ganhos serão desprezíveis no PC, já que quem domina o tempo é a aquisição dos dados. Por outro lado, no myRIO isso pode trazer ganhos enormes, pois o ponto crítico é a falta de poder de processamento, então o gargalo está nos cálculos matemáticos intensivos em processamento, ou seja, na otimização.

Uma forma bastante simples de melhorar esse tempo de processamento é através da alteração das condições de parada do problema de otimização no bloco de programação quadrática, visto que tipicamente os problemas de controle ótimo não necessitam de condições de parada tão restritivas quanto outras classes de problemas (WANG; BOYD, 2010). A partir disso, em um novo ensaio, o número máximo de iterações do otimizador foi limitado em apenas 20 iterações, diferentemente do valor padrão igual a 10^4, citado anteriormente nesta subseção.

A Figura 67 apresenta uma comparação da resposta temporal referente à medição da velocidade angular do motor sem e com a redução no número máximo de iterações do otimizador.

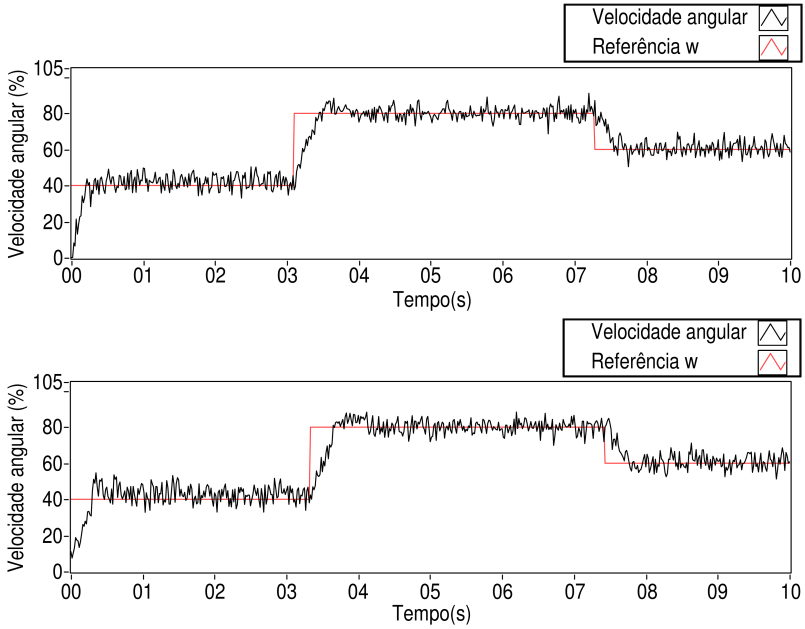
Como é possível observar, o fato de implementar esta técnica de cômputo rápido através da redução da condição de parada do problema de otimização não prejudicou a resposta de malha fechada de maneira expressiva.

A Figura 68 apresenta um histograma que compara a implementação realizada no PC e no LabVIEW RT com o número reduzido de iterações do otimizador, referentes ao tempo de aquisição de dados mais cômputo do controlador GPC, ao longo de 1500 amostras.

Para o caso da implementação no PC, é possível confirmar que a alteração de parâmetros do problema de otimização não apresenta ganhos expressivos já que, no geral, o tempo para aquisição de dados somado ao tempo de cômputo do controlador GPC permanece praticamente o mesmo se comparado ao apresentado na Figura 64. Neste caso, a média total dos tempos é de $11,86$ ms.

Já no caso da implementação em tempo real no myRIO, o valor mínimo de tempo é de 1 ms e o máximo igual a 13 ms, sendo que a média total dos tempos é de $4,653$ ms. Comparando esse resultado ao adquirido no ensaio referente à Figura 62, é possível comprovar que a implementação de técnicas de cômputo rápido são capazes de diminuir o tempo de processamento em

Figura 67 – Medição da velocidade angular (a) sem número reduzido de iterações e (b) com número reduzido de iterações.



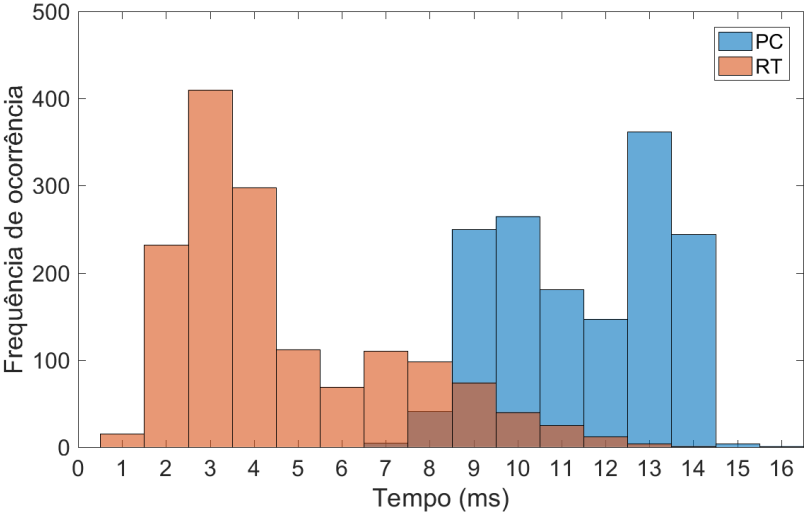
Fonte: Elaborado pelo autor.

uma implementação realizada em um sistema de tempo real, como neste caso que houve uma diminuição de aproximadamente 10 % do tempo médio. Outro fator importante é que, em um PC, a velocidade de processamento dependerá também do número de aplicações que estiverem executando de forma paralela com o controlador. Em contrapartida, quando a implementação é realizada em um sistema embarcado como o myRIO, ele atua como um *hardware* dedicado com o objetivo de atender a um único propósito específico, sendo capaz de garantir determinismo na execução de suas tarefas.

A Tabela 2 apresenta os dados adquiridos referentes à implementação com o número de iterações reduzido. Assim como a Tabela 1, neste caso também é apresentada a soma dos dois tempos (aquisição de dados e cômputo do controlador GPC) em um único elemento.

É importante salientar que a adoção de técnicas de cômputo rápido da literatura tem potencial para diminuir bastante o tempo de processamento, visto que, neste trabalho, foi apresentado um caso simples de limitação do número de iterações do otimizador, que poderia ser ainda mais reduzido dado

Figura 68 – Comparação entre o tempo de aquisição de dados mais cômputo do controlador GPC no PC e no myRIO com número reduzido de iterações.



Fonte: Elaborado pelo autor.

Tabela 2 – Comparação entre a implementação no PC e no myRIO em relação ao tempo de processamento com o número de iterações reduzido.

Hardware	Tempo de um ciclo de controle (ms)	
	Médio	Máximo
myRIO	4,65	13,00
PC	11,86	16,51

Fonte: Elaborado pelo autor.

que o desempenho do sistema não foi muito prejudicado com o limite de 20 amostras.

5.6 COMENTÁRIOS FINAIS

Este capítulo apresentou a validação experimental da estratégia de controle em cascata proposta neste trabalho. Inicialmente, definiu-se o sistema da malha interna a partir da relação entre a variação da razão cíclica de um sinal

PWM e a corrente drenada pelo motor CC do DIGIAC 710, a fim de controlar a corrente com a aplicação de um controlador PI. Em seguida, foi determinado o modelo do sistema da malha externa baseado na relação entre a corrente, adquirida do primeiro modelo, e a velocidade angular do motor, a fim de obter um controle da velocidade angular com a implementação do GPC, que trabalha apenas com modelos de tempo discreto, daí a determinação do modelo no domínio z. Definidos os modelos, o próximo passo foi apresentar o ajuste dos controladores PI e GPC. O primeiro tomou como base o sobressinal e tempo de acomodação desejados, além de ser necessário encontrar a representação em tempo discreto para a sua implementação no LabVIEW FPGA. Já o controlador GPC foi desenvolvido a partir do ajuste dos parâmetros N_u, N, δ e λ, tomando como base conceitos apresentados em capítulos anteriores.

Os resultados experimentais foram divididos em três: controle, rejeição de perturbações e tempo de cômputo. No primeiro caso, foi possível verificar que tanto o GPC quanto o PI são capazes de assegurar o seguimento da referência de velocidade angular e de corrente, respectivamente. Neste ponto, observou-se que ocorreu o fenômeno *windup*, o qual não foi tratado neste trabalho pois a abordagem mais simples de reduzir o limite de corrente de referência (saída do GPC) no problema de otimização poderia acarretar problemas relacionados à partida do motor ou rejeição de perturbações de carga em alguns casos. Um estudo mais aprofundado deste problema considerando um mapeamento de restrições da malha interna para a externa é sugerido para trabalhos futuros.

Na análise de rejeição de perturbações, foram realizados diversos ensaios com a finalidade de simular situações que possam ocorrer na prática. No primeiro ensaio, o eixo do motor foi quase totalmente travado, sendo possível verificar que o valor de corrente conseguiu seguir a referência de controle $u(k)$, assim como o fato de que foi respeitada a restrição no sinal de controle determinada pelo controlador GPC e o sinal de controle PI manteve-se abaixo do limite máximo definido pelo sinal PWM. O segundo ensaio mostrou que o sistema consegue rejeitar uma perturbação através da inserção de uma carga leve no eixo do motor, sendo que os únicos pontos nos quais a velocidade angular não segue a referência $\vec{w}$ é nos quais a perturbação é inserida ou retirada. O último ensaio, com o objetivo de analisar a rejeição de uma perturbação no valor da corrente, mostrou que o controlador PI atuou rapidamente quando um resistor foi inserido em série com o motor e rejeitou essa perturbação, fazendo com que ela não apresentasse influência no sistema de controle da malha externa.

Em relação aos resultados de tempo de cômputo, verificou-se que apenas o tempo de aquisição de dados em um microcontrolador ultrapassa o valor encontrado de toda a implementação do controlador PI no FPGA ($8,25\ \mu$s),

que inclui aquisição e conversão A/D dos dados e o cálculo do sinal de controle propriamente dito, comprovando assim as vantagens de implementação em um FPGA. Para análise de tempo de cômputo do controlador GPC, observou-se que o fato de haver uma restrição ativa no sistema demanda mais tempo de cálculo do controlador. Além disso, para comprovar as vantagens que uma implementação em um sistema de tempo real no myRIO apresenta sobre uma implementação em um PC, foram desenvolvidos dois *softwares* no PC, um somente para aquisição de dados e outro responsável por implementar o controlador GPC, igual àquele desenvolvido no LabVIEW Real-Time. A partir dos resultados mostrados na Tabela 1, comprovou-se que a soma dos tempos médios de aquisição de dados e de cômputo do controlador GPC implementado no myRIO apresenta uma considerável vantagem em relação ao PC, entretanto a variabilidade em torno do valor médio é maior no myRIO. Contudo, foi possível observar que o tempo demandado para a aquisição de dados é muito maior que o tempo de cálculo do controlador GPC em uma aplicação no PC, já no myRIO ocorre o contrário. Com isso, foram realizados novos ensaios através da diminuição do número máximo de iterações do problema de otimização, comprovando-se que a implementação de técnicas de cômputo rápido são capazes de diminuir o tempo de processamento em uma implementação no myRIO, diferentemente do PC, que não apresenta ganhos expressivos.

6 CONSIDERAÇÕES FINAIS

Neste capítulo final da dissertação são apresentadas tanto as principais conclusões e ganhos obtidos com o trabalho quanto propostas para trabalhos futuros.

6.1 CONCLUSÕES

Neste trabalho, foi implementada uma estratégia de controle em cascata e, em seguida, realizada uma avaliação experimental utilizando-se uma planta de laboratório, o DIGIAC 710. Na malha externa dessa estrutura, foi desenvolvido um controlador GPC com o objetivo de controlar a velocidade angular do motor presente no DIGIAC 710, sendo que o sinal de controle gerado neste caso, $u(k)$, foi aplicado à malha interna como valor de referência para um controlador PI, que é implementado a fim de controlar a corrente drenada pelo motor.

A partir dos resultados experimentais de controle, foi possível constatar que o algoritmo proposto é capaz de garantir o seguimento das referências de corrente elétrica e velocidade angular do motor. Em relação aos resultados referentes à aplicação de perturbações no sistema, comprovou-se que o controlador GPC rejeitou a perturbação resultante da inserção de uma carga leve no eixo do motor, assim como o controlador PI foi capaz de atuar rapidamente para compensar uma queda na corrente drenada pelo motor antes mesmo de essa perturbação gerar alguma alteração na malha externa de controle. Em outro ensaio realizado, no qual o eixo do motor foi quase totalmente travado, verificou-se que a restrição no sinal de controle foi respeitada pelo sistema, ou seja, $u(k)$ manteve-se abaixo do limite máximo pré-determinado de 100 %.

Por fim, com o objetivo de comparar os tempos de execução do sistema de controle implementado no myRIO frente a uma implementação convencional, foram realizados diversos ensaios. No caso do controle PI implementado em FPGA, foram adquiridos dados referentes a outros microcontroladores e, com isso, foi confirmado que o sistema de controle em FPGA é capaz de executar algumas dezenas de vezes mais rápido que implementação equivalente em um microcontrolador, já que alcançou um período de amostragem inferior a 10 μs, permitindo a sua aplicação até mesmo em sistemas que possuem dinâmica mais rápida que a da corrente do motor empregado neste caso. Em relação ao controlador GPC implementado em um processador com sistema operacional de tempo real, foi aplicado o mesmo *software* de controle em um PC para a comparação referente ao tempo de cômputo do controlador GPC,

assim como um outro *software* específico para aquisição de dados. A partir dos dados experimentais, confirmou-se que a implementação realizada no sistema de tempo real obteve o menor tempo de processamento, além de ser possível constatar que a distribuição do tempo de cálculo em cada iteração é realizada de maneira diferente no PC, onde a aquisição de dados demanda mais tempo, e no myRIO, em que o tempo de cálculo do controlador é responsável por ocupar a maior parte do tempo. Com isso, a partir de testes experimentais através da alteração das condições de parada do problema de otimização, foi mostrado que, no caso avaliado, a utilização de estratégias de cômputo rápido não apresenta qualquer ganho em uma implementação no PC. Por outro lado, no myRIO os ganhos são enormes referentes à redução no tempo de processamento do sistema, ou seja, foi possível obter um tempo máximo de um ciclo de controle igual a 13 ms, com o tempo médio de execução inferior a 5 ms.

6.2 PROPOSTAS PARA TRABALHOS FUTUROS

Portanto, a primeira proposta para trabalho futuro é a implementação de diferentes estratégias de cômputo rápido, aliadas à apresentada neste trabalho, com o objetivo de obter maior redução no tempo de processamento em relação àquela já alcançada.

Outra sugestão de trabalho futuro é tratar do problema de *windup*, que foi detectado e citado na seção 5.3, em que deve ser considerado um mapeamento de restrições da malha interna para a externa.

REFERÊNCIAS

AKKAYA, S.; AKBATI, O.; GÖRGÜN, H. Multiple closed loop system control with digital PID controller using FPGA. In: **IEEE Conference on Control, Decision and Information Technologies (CoDIT)**. Turquia: IEEE, 2014. p. 764–769.

ALI, F. H.; HUSSEIN, M. M.; ISMAEL, S. M. LabVIEW FPGA implementation of a PID controller for D.C. motor speed control. In: **IEEE Conference on Energy, Power and Control (EPC-IQ)**. Basrah: IEEE, 2010. p. 139–144.

ÅSTRÖM, K.; HÄGGLUND, T. **PID Controllers**. EUA: International Society for Measurement and Control, 1995.

ASTRÖM, K.; RICHALET, J.; O'DONOVAN, D. **Predictive Functional Control**: Principles and industrial applications. Londres, Reino Unido: Springer-Verlag London, 2009.

ATMEL. **Microcontroller with 32KBytes In-System Programmable Flash**. 2011. Disponível em: <http://www.atmel.com/images/doc2503.pdf>. Acesso em: 29 mar. 2017.

BEMPORAD, A.; MORARI, M. Robust model predictive control: A survey. **Robustness in Identification and Control**, v. 245, p. 207–226, 1999.

BERLIN, F.; FRANK, P. M. Advanced optimal predictive control. In: **American Control Conference**. EUA: IEEE, 1992. p. 2027–2031.

BEZERRA, M. S. **Projeto, Implementação e Ensaios de um Controlador PID utilizando FPGA**. 68 p. Monografia — Curso de Engenharia Elétrica - Universidade Federal do Ceará, Fortaleza, 2010.

CAMACHO, E. F.; BORDONS, C. **Model Predictive Control**. Inglaterra: Springer London, 1999.

CAMACHO, E. F.; BORDONS, C. Application of simple cascade GPC with robust behaviour to a sugar refinery. In: **IEEE Conference on Control Conference (ECC), European**. Espanha: IEEE, 2015. p. 1405–1410.

CASTRUCCI, P. de L.; BITTAR, A.; SALES, R. M. **Controle Automático**. Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora S.A., 2011.

CHOLAKKAL, S. **Load disturbance torque estimation for motor drive systems with application to electric power steering system**. Dissertação (Mestrado) — Universidade de Windsor, Canadá, 2009.

CLARKE, D. W.; MOHTADI, C.; TUFFS, P. S. Generalized predictive control–part i. the basic algorithm. **Automatica**, Elsevier, v. 23, n. 2, p. 137–148, 1987.

CLARKE, D. W.; MOHTADI, C.; TUFFS, P. S. Generalized predictive control–part ii: extensions and interpretations. **Automatica**, Elsevier, v. 23, n. 2, p. 149–160, 1987.

CLARKE, D. W.; MOHTADI, C.; TUFFS, P. S. **Industrial Digital Control Systems**. Londres, Reino Unido: Peter Peregrinus, 1987.

COELHO, L. A.; FERTIG, K. S.; FERTIG, K. S. Aplicações de FPGA em robótica. 2016. Disponível em: <http://wiki.sj.ifsc.edu.br/wiki/images/a/a2/DLP29006-AE1-Tema2-2016-1.pdf>. Acesso em: 09 dez. 2016.

COHEN, G. H.; COON, G. A. Theoretical consideration of retarded control. **Transactions of the ASME**, v. 75, p. 827–834, 1953.

CUTLER, C. R.; RAMAKER, B. L. Dynamic matrix control – A computer control algorithm. **Proceedings of AIChE 86th National Meeting**, Houston, 1979.

DILLENBURG, M. R. **Indo além do controle PID**. 2011. Disponível em: <http://www.novus.com.br/downloads/Arquivos/indoalemdocontrolepid.pdf>. Acesso em: 15 fev. 2016.

DINIZ, A. R. M. **Arquitetura de *hardware* reconfigurável paralela dedicada para a implementação da SA-DCT**. Dissertação (Mestrado) — Pontifícia Universidade Católica de Minas Gerais, Programa de pós-graduação em Engenharia Elétrica, 2008.

DORF, R. C.; BISHOP, R. H. **Sistemas de controle modernos**. Rio de Janeiro: LTC, 2001.

DSPACE. **Products**. 2016. Disponível em: <https://www.dspace.com/en/inc/home/products/products.cfm>. Acesso em: 09 dez. 2016.

DUTRA, C. B. S. **Controle Preditivo Multiobjetivo para processos com atraso**. Tese (Doutorado) — Universidade Federal de Santa Catarina, Programa de Pós-Graduação em Engenharia Elétrica, 2003.

FLESCH, R. C. C. **Contribuições ao controle de sistemas monovariáveis e multivariáveis com atraso de transporte**. Tese (Doutorado) — Universidade Federal de Santa Catarina, Programa de Pós-Graduação em Engenharia de Automação e Sistemas, 2012.

FRANKLIN, G. F.; POWELL, J. D.; EMAMI-NAEINI, A. **Feedback Control of Dynamic Systems**. EUA: Prentice Hall, 2002.

GARCIA, C.; RODRIGUEZ, J.; SILVA, C. Full predictive cascaded speed and current control of an induction machine. **IEEE Transactions on Energy Conversion**, IEEE, EUA, v. 31, p. 1059–1067, 2016.

JAHAGIRDAR, A. C.; DAN, T. K. Effect of tuning parameters on performance of first-order plus dead-time processes using generalized predictive control. In: **IEEE Conference on Industrial Instrumentation and Control (ICIC)**. India: IEEE, 2015. p. 692–696.

LIMA, D. M. **Sistema Embarcado de Controle Preditivo para Processos Industriais**. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, Programa de Pós-Graduação em Engenharia de Automação e Sistemas, 2013.

LIU, J.; ZHANG, P.; WANG, F. Real-Time DC servo motor position control by PID controllers using LabVIEW. In: **IEEE Conference on Intelligent Human-Machine Systems and Cybernetics**. China: IEEE, 2009. p. 206–209.

LLOYD, J. W. **Generalized Predictive Control Parameter Adaptation Using a Fuzzy Logic Approach**. Tese (Doutorado) — Virginia Polytechnic Institute and State University, EUA, 2011.

MEKONNEN, E. T.; KATCHA, J.; PARKER, M. An FPGA-based digital control development method for power electronics. In: **IECON - 38th Annual Conference on IEEE Industrial Electronics Society**. EUA: IEEE, 2012. p. 222–226.

MENGHUI, S.; YIQUN, W.; WEI, Z. Research on cascade predictive control in hydraulic agc of cold rolling mill. In: **IEEE Conference on Industrial Electronics and Applications**. China: IEEE, 2007. p. 2775–2780.

MORIMOTO, C. E. Lei de Moore. 2005. Disponível em: <http://www.hardware.com.br/termos/lei-de-moore>. Acesso em: 28 nov. 2016.

NATIONAL INSTRUMENTS. **NI myRIO Workshop**. 14.04. ed. Austin, EUA, 2013.

NATIONAL INSTRUMENTS. **From Student to Engineer**: Preparing future innovators with the ni labview rio architecture. 2014. Disponível em: <http://www.ni.com/white-paper/52093/en/>. Acesso em: 30 nov. 2016.

NATIONAL INSTRUMENTS. **Aplicações do myRIO**. 2015. Disponível em: <http://www.ni.com/myrio/applications/pt/>. Acesso em: 30 nov. 2016.

NATIONAL INSTRUMENTS. **NI USB-6009 Device Specifications**. 2015. Disponível em: <http://www.ni.com/pdf/manuals/375296a.pdf>. Acesso em: 30 mar. 2017.

NATIONAL INSTRUMENTS. **Os benefícios da programação gráfica do LabVIEW**. fev 2015. Disponível em: <http://www.ni.com/white-paper/14556/pt/>. Acesso em: 02 dez. 2016.

NATIONAL INSTRUMENTS. **Tutorial**: Block diagram. out 2015. Disponível em: <http://www.ni.com/tutorial/7565/en/>. Acesso em: 02 dez. 2016.

NATIONAL INSTRUMENTS. **Módulo LabVIEW Real-Time**. 2016. Disponível em: <http://www.ni.com/labview/realtime/pt/>. Acesso em: 15 dez. 2016.

NATIONAL INSTRUMENTS. **NI myRIO Hardware at a Glance**. set. 2016. Disponível em: <http://www.ni.com/product-documentation/14604/en/>. Acesso em: 30 nov. 2016.

NATIONAL INSTRUMENTS. **Série R de dispositivos RIO multifuncionais**. 2016. Disponível em: <http://sine.ni.com/nips/cds/view/p/lang/pt/nid/11829>. Acesso em: 09 dez. 2016.

NETO, A. H. **Técnicas anti-windup em estruturas de controle PID, RST e GPC**. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, Programa de Pós-Graduação em Engenharia Elétrica, 2005.

NISE, N. **Engenharia de sistemas de controle**. Rio de Janeiro, RJ: LTC, 2002.

OGATA, K. **Engenharia de controle moderno**. Rio de Janeiro: Prentice / Hall do Brasil, 2011.

PADHAN, D. G.; REDDY, B. R. A new tuning rule of cascade control scheme for processes with time delay. In: **IEEE Conference on Power, Control, Communication and Computational Technologies for Sustainable Growth (PCCCTSG)**. India: IEEE, 2016. p. 102–105.

PEREIRA, W. F. A. **Projeto e Avaliação do Controlador Preditivo Generalizado Sujeito a Restrições via Métodos de Otimização de Pontos Interiores**. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, Programa de Pós-Graduação em Engenharia Elétrica, 1997.

QUEIROZ, D. C. de. **Implementação do barramento on–chip AMBA baseada em computação reconfigurável**. Dissertação (Mestrado) — Universidade de São Paulo, Instituto de Ciências Matemáticas e Computação, 2005.

ROSSITER, J. **Model-Based Predictive Control**: A practical approach (control series). EUA: CRC Press, 2003.

SANTANA, M. **Uma Metodologia de Desenvolvimento do Controle Digital de Conversores Estáticos Utilizando FPGA**. Dissertação (Mestrado) — Universidade Federal de Itajubá, Programa de Pós-Graduação em Engenharia Elétrica, 2006.

SARAIVA, F. de A. **Métodos de sintonia em controladores PID**. 47 p. Monografia — Curso de Bacharelado em Engenharia de Telecomunicações - Centro Universitário La Salle - Unilasalle, Rio Grande do Sul, 2011.

SCARPINO, M. **Motors for Makers**: A guide to steppers, servos, and other electrical machines. Indiana, EUA: Que Publishing, 2015.

SIGAREV, V.; KUZMINA, T.; KRASILNIKOV, A. Real-Time control system for a DC motor. In: **NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW)**. Rússia: IEEE, 2016. p. 689–690.

SMITH, C. A.; CORRIPIO, A. B. **Principles and Practice of Automatic Process Control**. 2. ed. EUA: John Wiley & Sons, Inc., 1997.

TAN, K. K. et al. **Advances in PID Control**. Inglaterra: Springer London, 2012.

TAO, G.; XIUYING, C. **Cascade Generalized Predictive Control with constraints**. Canadá: IEEE, 2014. 371-375 p.

TEKIN, R. MATLAB and LabVIEW in modeling, analysis and Real Time control of a motion control system. In: **IEEE Conference on Control and Automation (ICCA)**. Turquia: IEEE, 2010. p. 2077–2081.

TEXAS INSTRUMENTS. **Single Supply, MicroPower, Instrumentation Amplifier**. 1997. Disponível em: <http://www.ti.com/lit/ds/symlink/ina122.pdf>. Acesso em: 03 jan. 2017.

TEXAS INSTRUMENTS. **Architecture and Function of the MSP430 14-Bit ADC**. 1999. Disponível em: <http://www.ti.com/lit/an/slaa045/slaa045.pdf>. Acesso em: 29 mar. 2017.

TEXAS INSTRUMENTS. **ISO724x High-Speed, Quad-Channel Digital Isolators**. 2007. Disponível em: <http://www.ti.com/lit/ds/symlink/iso7240cf.pdf>. Acesso em: 04 jan. 2017.

TRENTELMAN, H. L.; WILLEMS, J. C. **Essays on Control**: Perspectives in the theory and its applications. Suiça: Birkhäuser Basel, 2012.

TYREUS, B. D.; LUYBEN, W. L. Tuning of PI controllers for integrator/deadtime processes. **Industrial and Engineering Chemistry Research**, v. 31, p. 2625–2628, 1992.

VIDAL, L. de C. et al. Controle de vazão utilizando PID desenvolvido em linguagem gráfica LabVIEW e transmissor virtual. In: **Simpósio de Excelência em Gestão e Tecnologia**. Resende, RJ: SEGeT, 2013.

VISIOLI, A. **Practical PID Control**. Inglaterra: Springer London, 2006.

WANG, Y.; BOYD, S. Fast model predictive control using online optimization. In: **17th World Congress The International Federation of Automatic Control**. EUA: IFAC, 2008. p. 6974–6979.

WANG, Y.; BOYD, S. Fast model predictive control using online optimization. **IEEE Transactions on Control Systems Technology**, v. 18, n. 2, p. 267–278, 2010.

XIUYING, C. et al. Adaptive cascade generalized predictive control. In: **IEEE Conference on Control and Decision Conference (CCDC)**. China: IEEE, 2016. p. 494–499.

YAMAMOTO, G. K.; AZEVEDO, R. A. M. de. Controle baseado em FPGA: Milhões de transistores ao seu comando. 2013. Disponível em: <http://www.sabereletronica.com.br/artigos-2/3454-controle-baseado-em-fpga-milhoes-de-transistores-ao-seu-comando>. Acesso em: 22 mar. 2016.