UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Vinícius Berndsen Peccin

Controle Preditivo Baseado em Modelo de Cômputo Rápido: Contribuições em Algoritmos Embarcados com Otimização Online para Formulações Entrada-Saída

Florianópolis
2021

Vinícius Berndsen Peccin

Controle Preditivo Baseado em Modelo de Cômputo Rápido: Contribuições em Algoritmos Embarcados com Otimização Online para Formulações Entrada-Saída

Tese submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina para a obtenção do título de doutor em Engenharia de Automação e Sistemas.
Orientador: Prof. Julio Elias Normey Rico, Dr.
Coorientadores: Prof. Rodolfo César Costa Flesch, Dr. e Prof. Daniel Martins Lima, Dr.

Florianópolis
2021

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

Peccin, Vinícius Berndsen
Controle preditivo baseado em modelo de cômputo rápido : contribuições em algoritmos embarcados com otimização online para formulações entrada-saída / Vinícius Berndsen Peccin ; orientador, Julio Elias Normey-Rico, coorientador, Rodolfo César Costa Flesch, coorientador, Daniel Martins Lima, 2021.
145 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2021.

Inclui referências.

1. Engenharia de Automação e Sistemas. 2. Controle de Processos. 3. Controle preditivo baseado em modelo. 4. Otimização embarcada. I. Normey-Rico, Julio Elias. II. Flesch, Rodolfo César Costa. III. Lima, Daniel Martins IV. Universidade Federal de Santa Catarina. Programa de Pós Graduação em Engenharia de Automação e Sistemas. V. Título.

Vinícius Berndsen Peccin

Controle Preditivo Baseado em Modelo de Cômputo Rápido: Contribuições em Algoritmos Embarcados com Otimização Online para Formulações Entrada-Saída

O presente trabalho em nível de doutorado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Ricardo Hiroshi Caldeira Takahashi, Dr.
Universidade Federal de Minas Gerais

Prof. Douglas Wildgrube Bertol, Dr.
Universidade do Estado de Santa Catarina

Prof. Eduardo Camponogara, Dr.
Universidade Federal de Santa Catarina

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de doutor em Engenharia de Automação e Sistemas.

Documento assinado digitalmente
Werner Kraus Junior
Data: 24/09/2021 09:19:31-0300
CPF: 531.085.239-53
Verifique as assinaturas em https://v.ufsc.br

Coordenação do Programa de
Pós-Graduação

Documento assinado digitalmente
Julio Elias Normey Rico
Data: 24/09/2021 07:58:50-0300
CPF: 762.840.859-15
Verifique as assinaturas em https://v.ufsc.br

Prof. Julio Elias Normey Rico, Dr.
Orientador

Florianópolis, 2021.

Para Tati, Cora e Maya, com amor.

AGRADECIMENTOS

À minha esposa Tatiana e as minhas filhas Cora e Maya pelo apoio, carinho e compreensão.

Aos meus pais pela base, valores e oportunidades que me proporcionaram.

Ao professor Julio por ser um modelo e inspiração como profissional, pelos ensinamentos desde a graduação, que me despertaram a paixão pela área de controle, e pela orientação desse trabalho.

Aos professores Rodolfo e Daniel, meus coorientadores, pelas orientações e pelo compromisso com a excelência.

À banca examinadora da qualificação e da presente tese pela avaliação e contribuições ao trabalho.

Ao Instituto Federal de Santa Catarina pelo período de afastamento que permitiu a dedicação à pesquisa.

RESUMO

As principais contribuições inéditas desta tese são os algoritmos de controle preditivo baseado em modelo (MPC) de cômputo rápido com otimização online e a proposição de arquiteturas de implementação em computação paralela. Foram abordados o controle preditivo generalizado (GPC) e o controle por matriz dinâmica (DMC), que são as formulações entrada-saída de MPC mais utilizadas na indústria. Dos seis algoritmos propostos, cinco deles utilizam a formulação GPC e o outro a formulação DMC. Em cada algoritmo é utilizado um método de otimização com características distintas. São também propostas generalizações para o caso de múltiplas entradas e múltiplas saídas. Os algoritmos GPCGPAD e GPCFAMA foram os que apresentaram o melhor desempenho no quesito do tempo de cômputo. Para esses dois métodos são também fornecidos meios de se calcular o limite superior do número de iterações do método de otimização, de modo a descobrir o pior caso de tempo de execução. Os algoritmos propostos foram testados e comparados em simulação. Os estudos de caso considerados nas simulações foram um inversor de frequência trifásico com filtro LCL e conectado à rede e uma coluna de destilação de laboratório. Quanto às arquiteturas, foram desenvolvidos métodos de implementação dos algoritmos em um FPGA, explorando as vantagens de paralelismo da descrição direta em hardware. Foi proposta uma solução paralela para a multiplicação matricial em aritmética de ponto fixo, por ser a operação computacional mais custosa. Outra contribuição inédita é a proposição de um precondicionamento do problema de otimização resultante do GPC que obteve uma diminuição de até 11% no tempo de cômputo na arquitetura de implementação proposta. Os testes em FPGA apresentaram resultados rápidos com o tempo de cômputo do sinal de controle da ordem de microssegundos. A tese está formatada como uma coleção de artigos. São apresentados cinco artigos, diretamente ligados à pesquisa, organizados em cada capítulo.

Palavras-chave: Controle de processos. Controle preditivo baseado em modelo. Otimização embarcada. FPGA

ABSTRACT

The main novel contributions of this thesis are the proposition of fast model predictive control (MPC) algorithms with online optimization and the proposition of implementation architectures with parallel computing. Generalized predictive control (GPC) and dynamic matrix control (DMC) were addressed, which are the most popular input-output formulations of the MPC in industry. Six algorithms were proposed, with five of them using a GPC formulation, and the other one a DMC formulation. In each algorithm, an optimization method with different characteristics is used. Generalizations are also proposed for multiple-inputs and multiple-outputs systems. The GPCGPAD and GPCFAMA algorithms have the best performance in terms of computation time. For these two algorithms, methods to calculate the upper bound on the number of iterations are provided to discover the worst case execution time. The proposed algorithms were tested in simulation. The case studies stand out in a three-phase grid-connected LCL-filtered inverter, and a laboratory distillation column. The implementation architectures are proposed with the methods to implement the algorithms in an FPGA, exploring the advantages of parallelism of the direct description in hardware. A parallel solution for matrix multiplication in fixed-point arithmetic has been proposed as it is the most costly operation. Another novel contribution is the proposal of a preconditioning operation of the optimization problem resulting from the GPC, which obtained a decrease of up to 11% in the computation time in the proposed implementation architecture. The tests in FPGA showed fast results, with the computation time of the control signal in microseconds. The thesis is organized as a collection of papers. Five papers directly linked to the research are presented in each chapter.

Keywords: Control processes. Embedded optimization. FPGA. Model predictive control.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

A/D	Analógico para digital.
ACA	Ação de controle média, do inglês *Average Control Action*.
ADMM	Método dos multiplicadores em direções alternadas, do inglês *Alternating Direction Method of Multipliers*.
AS	Conjunto ativo, do inglês *Active Set*.
CARIMA	Autorregressivo com média móvel, integrador e entrada controlada, do inglês *Controlled Autoregressive Integrated Moving Average*
CCS-MPC	MPC com conjunto contínuo de controle, do inglês *Continuous Control Set-MPC*
DMC	Controle por matriz dinâmica, do inglês *Dynamic Matrix Control*.
FAMA	Algoritmo rápido de minimização alternada, do inglês *Fast Alternating Minimization Algorithm*.
FCS-MPC	MPC com conjunto de controle finito, do inglês *Finite Control Set-MPC*
FPGA	Arranjo de portas programáveis em campo, do inglês *Field Programmable Gate Array*.
GPAD	Gradiente projetado acelerado dual.
GPC	Controle preditivo generalizado, do inglês *Generalized Model Predictive Control*.
GPM	Método de projeção de gradiente, do inglês *Gradient Projection Method*.
HIL	Hardware no laço, do inglês *Hardware-In-the-Loop*.
IAE	Integral do erro absoluto, do inglês *Integral Absolute Error*.
IGBT	Transistor bipolar de porta isolada, do inglês *Insulated Gate Bipolar Transistor*.
IP	Ponto interior, do inglês *Interior Point*.
ISE	Integral do erro quadrático, do inglês *Integral Squared Error*.
KKT	Karush-Kuhn-Tucker.
LQR	Regulador quadrático linear, do inglês *Linear Quadratic Regulator*.
MFD	descrição matricial fracionária, do inglês *Matrix Fraction Description*).
MIMO	Múltiplas entradas e múltiplas saídas, do inglês *Multiple Input Multiple Output*.
MPC	Controle preditivo baseado em modelo, do inglês *Model Predictive Control*.
mp-QP	Programação Quadrática multiparamétrica, do inglês *multiparametric Quadratic Programming*.
OSM	Método do operador de partição, do inglês *Operator Split Method*.

PFQP	Programação quadrática sem projeção, do inglês *Projection-free Quadratic Programming*.
PID	Proproporcional-integral-derivativo.
PWM	Modulador por largura de pulso, do inglês *Pulse Width Modulation*.
QP	Programação Quadrática, do inglês *Quadratic Programming*.
RMSE	Raíz do erro quadrático médio, do inglês *Root Mean Square Error*.
SISO	Uma entrada e uma saída, do inglês *Single Input Single Output*.
ssMPC	MPC em espaço de estados, do inglês *State Space Model Predictive Control*.
WCET	Pior caso de tempo de execução, do inglês *Worst Case Execution Time*.

LISTA DE SÍMBOLOS

Notação

$\mathbf{0}$	Matriz com todos os elementos iguais a 0 e dimensão apropriada
$\mathbf{1}$	Matriz com todos os elementos iguais a 1 e dimensão apropriada
$A(.)$	Letras maiúsculas e itálicas denotam polinômios
argmin(.)	Argumento que minimiza
$\mathbf{dom} f$	Domínio da função f
$\mathbf{I}_{n\times m}$	Matriz identidade de dimensão $n \times m$
$\mathbf{M}$	Letras maiúsculas em negrito denotam matrizes
$\mathbb{R}$	Conjunto dos números reais
$\mathbb{R}_+$	Conjunto dos números reais positivos
$\mathbb{R}_-$	Conjunto dos números reais negativos
$\mathbf{v}$	Letras minúsculas em negrito denotam vetores
$v^{(i)}$	Denota a i-ésima instância da variável v
$\mathbf{X}_{i,:}$	Representa a i-ésima linha da matriz $\mathbf{X}$
$x(k)$	É o valor no k-ésimo instante amostrado do sinal x
$\lambda_{\max}(\mathbf{M})$	Denota o maior autovalor da matriz $\mathbf{M}$
$\lambda_{\min}(\mathbf{M})$	Denota o menor autovalor da matriz $\mathbf{M}$
∇f	Gradiente da função f
$\nabla^2 f$	Hessiana da função f
$\|.\|$	Norma Euclidiana de um vetor
$[\mathbf{M}]_{ij}$	Elemento da linha i e coluna j da matriz $\mathbf{M}$
$\circ$	Produto Hadamard entre dois vetores (multiplicação entre elementos na mesma posição)
$\triangleq$	Por definição
$\leftarrow$	Atribuição de um valor a uma variável

Variáveis

$\mathbf{A}$	Matriz de estados (espaço de estados)
$A(z^{-1})$	Polinômio da saída no modelo CARIMA
$\mathbf{b}$	Matriz de de ponderação da parcela linear do QP
$\mathbf{B}$	Matriz de entrada (espaço de estados)
$B(z^{-1})$	Polinômio de entrada no modelo CARIMA
$\mathbf{C}$	Matriz de saída (espaço de estados)
d	Tempo morto do sistema
$E_j(z^{-1})$	Polinômio da equação Diofantina utilizada no GPC
$\mathbf{f}$	Vetor de resposta livre
$F_j(z^{-1})$	Polinômio da equação Diofantina utilizada no GPC

$\mathcal{F}$	Conjunto factível de um QP
$\mathbf{G}$	Matriz de resposta ao degrau
$\mathbf{H}$	Matriz Hessiana que pondera a parcela quadrática do QP
k	Tempo discreto amostrado
l	Comprimento de passo
L	Constante Lipschitz
$\mathbf{L}$	Matriz diagonal com os elementos não nulos sendo os comprimentos de passo
$\mathcal{L}$	Lagrangiano
$\mathcal{L}_\rho$	Lagrangiano aumentado
$\bar{\mathbf{M}}$	Matriz de restrições de igualdade
$\bar{\mathbf{m}}$	Vetor de restrições de igualdade
N	Horizonte de predição
N_1	Início do horizonte de predição
N_2	Final do horizonte de predição
n_i	Quantidade de entradas do sistema
n_o	Quantidade de saídas do sistema
n_{req}	Quantidade de restrições de igualdade
n_{rin}	Quantidade de restrições de desigualdade
n_s	Quantidade de variáveis de estados
N_{ss}	Quantidade de amostras necessárias para o sistema estável atingir o regime permanente
N_u	Horizonte de controle
$\mathbf{Q}$	Matriz de ponderação nos estados (espaço de estados)
$\mathbf{R}$	Matriz de ponderação no esforço de controle (espaço de estados)
$\bar{\mathbf{R}}$	Matriz de restrições de desigualdade
$\bar{\mathbf{r}}$	Vetor de restrições de desigualdade
$\mathbf{S}$	Matriz de ponderação cruzada (espaço de estados)
t	Parâmetro de exatidão da barreira logarítmica
$T(z^{-1})$	Polinômio de ponderação das incertezas no modelo CARIMA
$u(k)$	Sinal de controle
$\mathbf{u}(k)$	Vetor de sinais de controle
$w(k+j)$	Referência futura no instante $k+j$
$\mathbf{w}$	Vetor de referências atual e futuras
$\mathbf{x}(k)$	Vetor de variáveis de estado
$y(k)$	Resposta amostrada de variável controlada no instante k
$\mathbf{y}$	Vetor de saídas no instante k
$\hat{\mathbf{y}}$	Vetor de predições das saídas da planta
$\mathbf{z}$	Vetor de variável particionada do ADMM

δ	Ponderação no erro
$\Delta u(k)$	Incremento de controle no instante k
$\Delta \mathbf{u}$	Vetor de incrementos de controle atual(is) e futuros
ϵ	Tolerância para os métodos de otimização
ϵ_{pri}	Tolerância no resíduo primal
ϵ_{dual}	Tolerância no resíduo dual
η	Função barreira logarítmica
λ	Ponderação no esforço de controle
μ	Incremento em t que representa a barreira logrítmica do IP
$\boldsymbol{\nu}$	Vetor dos multiplicadores de Lagrange associado às restrições de igualdade
$\boldsymbol{\rho}$	Variável de ponderação do Lagrangiano aumentado
$\boldsymbol{\phi}$	Variável de acúmulo de resíduo no ADMM escalonadao
$\boldsymbol{\psi}$	Vetor dos multiplicadores de Lagrange associado às restrições de desigualdade
Ω	Subconjunto genérico

SUMÁRIO

1 INTRODUÇÃO

O controle preditivo baseado em modelo (MPC, do inglês *Model Predictive Control*) é reconhecidamente uma das técnicas de controle avançado mais utilizadas na indústria (CAMACHO; BORDONS, 2004). Possui muitas vantagens em sua aplicação, tais como tratamento explícito de restrições, aplicação tanto em sistemas de uma entrada e uma saída (SISO, do inglês *Single Input Single Output*) quanto em de múltiplas entradas e múltiplas saídas (MIMO, do inglês *Multiple Input Multiple Output*) e compensação intrínseca de tempo morto (NORMEY-RICO; CAMACHO, 2007). Entretanto, é bem difundido que o MPC, em sua formulação convencional, só é aplicável para processos com dinâmica lenta, com período de amostragem da ordem de segundos ou minutos (WANG; BOYD, 2010). Essa característica advém do fato de que cada nova ação de controle é obtida por meio do cômputo de um problema de otimização online com restrições em um horizonte predefinido. Com os avanços da indústria, é latente a necessidade de controladores avançados nos níveis mais baixos da pirâmide de produção que, consequentemente, devem operar com períodos de amostragem da ordem de micro ou milissegundos. Apesar de ter sido desenvolvido há décadas, o PID ainda é a técnica com mais impacto na indústria, seguida pelo MPC, e grande parte disso se deve a um distanciamento entre a comunidade acadêmica e a aplicação prática das técnicas propostas (SAMAD, 2017). Dessa forma, é importante que os algoritmos propostos também levem em consideração características relevantes para a aplicação prática em cenários mais abrangentes. Assim, pesquisas para implementação de MPC de computo rápido, levando em conta algoritmos de otimização eficientes aplicados a hardwares de alto de desempenho, são de grande interesse.

Existem diversas formulações de MPC, que se diferenciam pelo modelo utilizado na predição, pelas perturbações consideradas e pela função objetivo (NORMEY-RICO; CAMACHO, 2007). Entre as formulações mais utilizadas na indústria, destacam-se o controle preditivo generalizado (GPC, do inglês *Generalized Predictive Control*) e o controle por matriz dinâmica (DMC, do inglês *Dynamic Matrix Control*) (CAMACHO; BORDONS, 2004). Para o cálculo das predições, o GPC baseia-se na função de transferência discreta e o DMC nos coeficientes da resposta ao degrau. A formulação do MPC em espaço de estados (ssMPC, do inglês *State Space Model Predictive Control*) é bastante difundida na comunidade acadêmica, devido à facilidade na extensão para o caso MIMO e por permitir uma abordagem mais fácil para demonstrar garantias de estabilidade e robustez do sistema em malha fechada. Entretanto, as formulações baseadas em funções de transferência são bastante utilizadas na indústria, uma vez que conceitos como tempo morto, constantes de tempo e ganhos são geralmente mais familiares no contexto industrial do que os conceitos de espaço de estados (CAMACHO; BORDONS, 2004). Outro ponto fundamental é que o tempo morto, frequente em

processos industriais, é mais facilmente representado em funções de transferência do que em outras abordagens (NORMEY-RICO; CAMACHO, 2007). Como é apresentado no capítulo 2, percebe-se que a grande maioria dos trabalhos que exploram algoritmos de MPC de cômputo rápido empregam a formulação em espaço de estados. Dessa forma, há uma lacuna entre os algoritmos do estado da arte de otimização e as formulações amplamente utilizadas na indústria. Apesar de as formulações poderem ser equivalentes em alguma medida, elas ainda guardam algumas particularidades de implementação, como o número de variáveis para modelar o problema, a esparsidade e dimensão das matrizes, a necessidade de observador de estados etc. É possível reescrever o DMC em uma formulação de espaço de estados, entretanto, devido ao uso do modelo de resposta ao degrau, o número de estados cresce consideravelmente. Um sistema SISO representado dessa forma teria tantos estados quanto coeficientes de resposta ao degrau (LEE *et al.*, 1994). Para um sistema MIMO, o número de estados torna a implementação DMC em espaço de estados impraticável para sistemas rápidos.

Em geral, o tempo consumido para resolver o problema de programação quadrática (QP, do inglês *Quadratic Programming*) para se obter a ação de controle a cada iteração é o principal obstáculo para acelerar o cômputo do MPC. Portanto, a escolha do método de resolução do QP do MPC é crítico para se obter um algoritmo eficiente. Uma abordagem é reformular o QP como um problema de programação paramétrica e posteriormente calcular uma solução offline baseada em regiões. Essa abordagem é chamada de MPC explícito (PISTIKOPOULOS *et al.*, 2015). Entretanto, nessa abordagem o número de regiões cresce muito rapidamente com o número de estados do sistema e, consequentemente, o mesmo ocorre com a memória necessária para o armazenamento dos dados. Em Borrelli *et al.* (2010) e em Kvasnica *et al.* (2015), soluções de ssMPC explícito sem regiões são apresentadas para mitigar essa questão. Nesses trabalhos, a geração de regiões críticas é evitada e substituída pela enumeração de conjuntos ativos ótimos, o que reduz a memória requerida para armazenamento. Entretanto, seu uso ainda é limitado a problemas com poucas restrições, uma vez que o número de possíveis combinações de restrições ativas aumenta exponencialmente com o número de restrições (AHMADI-MOSHKENANI *et al.*, 2018). Entre as soluções de computação online, o método de conjunto ativo (CIMINI; BEMPORAD, 2017; HERCEG *et al.*, 2015), o método de ponto interior (ROLDAO-LOPES *et al.*, 2009; WILLS *et al.*, 2011) e os métodos de primeira ordem merecem ser destacados. Para os métodos de primeira ordem, o trabalho de Patrinos e Bemporad (2014) apresenta uma solução baseada no método de gradiente projetado acelerado dual (GPAD) para a formulação dual do QP do MPC. Uma solução via o método do operador de partição chamada método dos multiplicadores em direção alternada (ADMM, do inglês *Alternating Direction Method of Multipliers*) é apresentada no trabalho de O'Donoghue *et al.*

(2013) e o trabalho de Pu *et al.* (2017) utiliza o método FAMA. Outra abordagem por métodos de primeira ordem é apresentada em Cairano *et al.* (2013), na qual é utilizado um algoritmo de otimização sem projeção baseado no método de programação quadrática paralela (PFQP, do inglês *Projection-free Quadratic Programming*). Os métodos de primeira ordem requerem poucos recursos computacionais e são relativamente simples de implementar quando comparados com abordagens tradicionais de otimização online. Entretanto, uma desvantagem comum dos métodos de primeira ordem é que o desempenho numérico é mais dependente das características do problema a ser resolvido (FERREAU *et al.*, 2017).

Com o foco em propor algoritmos adequados à implementação prática, é necessário também avaliar as soluções viáveis e mais adequadas para a implementação de técnicas avançadas de controle. Para as soluções de hardware, em geral, existem pesquisas de implementação de MPC de cômputo rápido embarcado em processadores e em arranjos de portas programáveis em campo (FPGAs, do inglês *Field Programmable Gate Arrays*) (PEYRL *et al.*, 2014). Por se tratar de uma implementação em hardware, a execução das tarefas no FPGA é paralela e há uma garantia de determinismo. De forma geral, os processadores com um núcleo executam os programas sequencialmente através de um escalonador do sistema operacional e dependem de sistemas operacionais de tempo real para garantir o determinismo (MA *et al.*, 2014). Na linha de FPGA, uma solução que vem sendo pesquisada é implementar o otimizador diretamente em hardware. Entretanto, o ganho de velocidade em hardware traz consigo o ônus da limitação de recursos dos mesmos. Em Patrinos e Bemporad (2014) e Ferreau *et al.* (2017) foram apresentados alguns requisitos comuns para que um algoritmo de otimização aplicado a controle possa ser embarcado:

1. computar uma ação de controle em um intervalo de amostragem definido em um hardware relativamente simples, como um microcontrolador ou FPGA, por exemplo;

2. necessitar de pouca memória para guardar os dados que definem o problema de otimização e o código que implementa a solução;

3. resultar em um código de controle que seja simples o suficiente para gerar um software capaz de ser verificado/validado/certificado, fácil de entender e com sinais claros de por que falhou em determinadas situações, especialmente em aplicações críticas;

4. ter um pior caso de tempo de execução (WCET, do inglês *Worst Case Execution Time*) previsível, de modo a atender os requisitos *hard* de tempo real.

Portanto, a partir dos requisitos evidenciam-se os desafios de se propor métodos de controle ótimo embarcáveis. Esses requisitos serviram como uma linha base para os algoritmos propostos, assim como são representados nas métricas escolhidas para a comparação entre os métodos.

1.1 OBJETIVOS

1.1.1 Objetivo geral

O objetivo geral da pesquisa de doutorado é explorar de forma teórico-experimental formulações e implementações de GPC e DMC com restrições, de forma integrada aos algoritmos de programação quadrática do estado da arte, com vias a obter um cômputo rápido, eficiente e confiável da ação de controle e com boas características para a implementação em hardwares de alto desempenho.

1.1.2 Objetivos específicos

De modo a esquematizar o objetivo geral e relacioná-lo às etapas da pesquisa, foram definidos como objetivos específicos:

- revisar as formulações de GPC e DMC com restrições, visando o cômputo rápido da ação de controle;
- revisar as técnicas de otimização para MPC embarcado de cômputo rápido;
- adaptar técnicas do estado da arte de otimização embarcada, que geralmente são desenvolvidas para a formulação MPC em espaço de estados, para as formulações GPC e DMC;
- comparar as técnicas propostas com um otimizador de uso geral;
- propor algoritmos integrados do controlador com o otimizador e com características para um cômputo rápido, eficiente e confiável da ação de controle;
- propor uma arquitetura paralela para a implementação dos algoritmos propostos;
- testar as implementações em um hardware de alto desempenho, em cenários semelhantes aos encontrados na indústria.

1.2 PRINCIPAIS CONTRIBUIÇÕES E ARTIGOS PUBLICADOS

As principais contribuições inéditas da pesquisa, detalhadas nos próximos capítulos, são:

- a proposição de cinco algoritmos GPC de cômputo rápido com restrições e otimização online, moldados para requerer apenas recursos computacionais adequados à sistemas embarcados, baseados nos métodos GPAD, FAMA, ADMM, PFQP e de barreira;
- a proposição de um algoritmo MIMO DMC com restrições de cômputo rápido e otimização online;
- a proposição de arquiteturas de implementação dos algoritmos descritos em hardware de forma paralela para FPGA;
- a proposição de um precondicionamento do problema de otimização resultante que permite uma diminuição de até 11% no tempo de cômputo na arquitetura de implementação proposta.

No total foram produzidos seis artigos ligados diretamente à pesquisa de doutorado. Dois deles foram publicados nos periódicos indexados *IET Control Theory & Applications* e *IEEE Latin America Transactions*. Outros três artigos foram publicados e apresentados em congressos, sendo um no congresso internacional *12th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems* (IFAC-DYCOPS) e dois no XXII e XXIII Congresso Brasileiro de Automática (CBA). O último foi submetido para avaliação no periódico *Elsevier Computers & Chemical Engineering*. Os títulos dos artigos seguem listados abaixo:

- *Fast Algorithms for Constrained Generalized Predictive Control with On-line Optimization* (PECCIN *et al.*, 2021a). *IET Control Theory & Applications* (2021).

- *Fast Constrained Generalized Predictive Control with ADMM Embedded in an FPGA* (PECCIN *et al.*, 2020b). *IEEE Latin America Transactions* (2020).

- *Fast Generalized Predictive Control Based on Accelerated Dual Gradient Projection Method* (PECCIN *et al.*, 2019). *12th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems* (DYCOPS 2019).

- Algoritmos GPC de Cômputo Rápido com Métodos de Ponto Interior e Programação Quadrática Sem Projeção (PECCIN *et al.*, 2020a). XXIII Congresso Brasileiro de Automática (CBA 2020).

- Implementação de GPC e DMC para Sistemas Rápidos Usando ADMM (PECCIN *et al.*, 2018). XXII Congresso Brasileiro de Automática (CBA 2018).

- *Fast Constrained Dynamic Matrix Control Algorithm with Online Optimization* (PECCIN *et al.*, 2021b). *Elsevier Computers & Chemical Engineering* (submetido para avaliação).

1.3 ESTRUTURA DO TRABALHO

O trabalho foi montado como uma coleção de artigos. Foram omitidas as introduções e as revisões para evitar repetições de conteúdo ao leitor. No capítulo 2 e no capítulo 3 são apresentadas revisões da literatura e fundamentações teóricas, mais amplas e aprofundadas, abordando as formulações MPC estudadas nesta pesquisa e os principais métodos de otimização para programação quadrática, respectivamente. No artigo do capítulo 4 é apresentado um algoritmo de cômputo rápido do GPC, para o caso SISO, com o método de projeção de gradiente acelerado dual. No artigo do capítulo 5 é apresentado um algoritmo GPC com o método de multiplicadores em direções alternadas. São também abordadas questões de implementação do algoritmo em arquiteturas de computação paralela. No artigo do capítulo 6 são apresentados os algoritmos de GPC com os métodos de projeção de gradiente acelerado dual e algoritmo rápido de minimização alternada, ambos para o caso MIMO e com uma estimação do limite superior do número de iterações. É também apresentado um precondicionamento do QP do GPC de modo a diminuir o tempo de cômputo do sinal de controle na arquitetura de implementação proposta. O estudo de caso abordado é um inversor trifásico com filtro LCL e conectado à rede. No artigo do capítulo 7 são propostos os algoritmos GPC com os métodos de barreira logarítmica e o método de programação quadrática sem projeção. No artigo do capítulo 8 é apresentado um algoritmo do DMC com o método de projeção de gradiente acelerado dual. O estudo de caso considerado é uma coluna de destilação de laboratório e os resultados são comparados com abordagens de MPC explícito. O capítulo 9 traz uma conclusão geral da tese.

2 CONTROLE PREDITIVO BASEADO EM MODELO

O controle preditivo baseado em modelo (MPC, do inglês *Model Predictive Control*) é um tipo de controle ótimo. De forma geral, o MPC calcula a ação de controle a ser aplicada no instante k, a partir da predição do comportamento da planta, baseado em um modelo dinâmico, em um horizonte finito N. São computados N_U controles futuros por meio da otimização de uma função objetivo que representa algum critério de comportamento do sistema. Apenas a primeira ação de controle é aplicada ao sistema e o MPC recalcula uma nova sequência de sinais de controle a cada novo instante de amostragem, uma vez que novas informações da planta são adquiridas (CAMACHO; BORDONS, 2004). O conceito básico do MPC pode ser visto na Figura 1, na qual são apresentados 3 instantes de tempo nos quais a janela de predição é computada (em vermelho) e é aplicado apenas o sinal de controle no tempo atual.

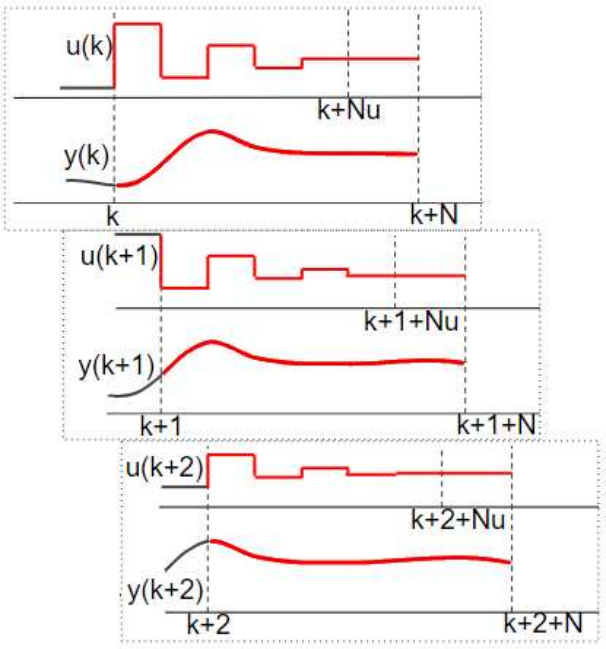


Figura 1 – Conceito básico do MPC onde é computada uma janela de predição (em vermelho) e aplicado apenas o sinal de controle no tempo atual.

Como o cômputo da ação de controle do MPC passa por um processo de otimização, podem ser definidas de forma explícita restrições nas variáveis do sistema. Outra característica importante é que, se as referências futuras da planta forem conhecidas, é possível ter ações de controle antes mesmo de a mudança de referência realmente acontecer (NORMEY-RICO; CAMACHO, 2007). Um diagrama de blocos simplificado representando o MPC, a planta e os sinais envolvidos podem ser vistos na Figura 2. Dentro do bloco do MPC estão presentes o modelo utilizado para gerar as predições, o otimizador, que calcula as ações futuras de controle minimizando a função custo e respeitando as restrições impostas e um bloco de memória responsável por

armazenar informações como saídas e controles passados de acordo com o modelo utilizado.

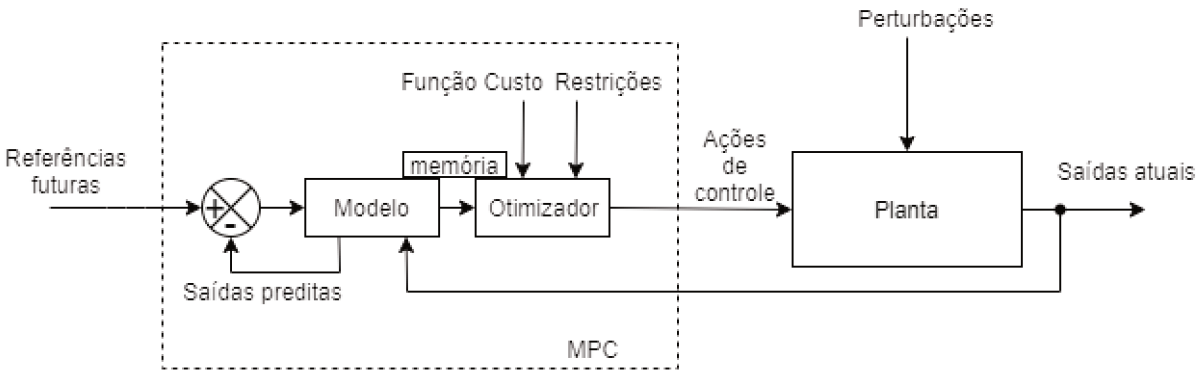


Figura 2 – Diagrama de blocos simplificado representando o MPC, a planta e os sinais envolvidos.

2.1 FUNDAMENTOS

O MPC não é uma técnica específica, mas uma família de soluções. Entretanto, as diferentes técnicas de MPC possuem uma estrutura comum (CAMACHO; BORDONS, 2004):

- modelo de predição;
- função objetivo;
- procedimento para a obtenção da lei de controle.

O que diferencia cada tipo de técnica são as perturbações consideradas, o modelo utilizado na predição e a função objetivo.

2.1.1 Modelo de Predição

O modelo de predição varia de acordo com o algoritmo MPC. Por exemplo, o DMC utiliza resposta ao degrau e o controle algorítmico baseado em modelo (MAC, do inglês *Model Algorithm Control*) utiliza a resposta ao impulso. Já o GPC baseia-se em modelos de função de transferência e o controle funcional preditivo (PFC, do inglês *Predictive Functional Control*) baseia-se em modelos de espaço de estados.

2.1.2 Função objetivo

A função objetivo varia de acordo com a técnica MPC. Uma expressão bastante utilizada na prática como função objetivo, para o caso SISO (apresentada aqui por

simplicidade), é dada por (CAMACHO; BORDONS, 2004):

$$J = \sum_{j=N_1}^{N_2} q_\delta(j) \left[\hat{y}(k+j|k) - w(k+j)\right]^2 + \sum_{j=1}^{N_u} q_\lambda(j) \left[\Delta u(k+j-1)\right]^2, \quad (1)$$

onde $\hat{y}(k+j|k)$ é uma predição da saída em um tempo futuro $k+j$, dadas as informações em k, $w(k+j)$ é a referência em $k+j$, $\Delta u(k+j-1)$ é o incremento de controle em $k+j-1$, $q_\delta(j)$ e $q_\lambda(j)$ ponderam o comportamento futuro do erro e do esforço de controle, respectivamente, N_1 e N_2 definem o horizonte de predição e N_u é o horizonte de controle.

2.1.3 Obtenção da lei de controle

A obtenção da ação de controle do MPC passa por um processo de otimização da função objetivo do tipo:

$$\begin{aligned} \min_{\Delta \mathbf{u}} \quad & J \\ \text{s.a.} \quad & \bar{\mathbf{R}}\Delta\mathbf{u} \leq \bar{\mathbf{r}} \\ & \bar{\mathbf{M}}\Delta\mathbf{u} = \bar{\mathbf{m}}, \end{aligned} \quad (2)$$

onde $\bar{\mathbf{R}}\Delta\mathbf{u} \leq \bar{\mathbf{r}}$ e $\bar{\mathbf{M}}\Delta\mathbf{u} = \bar{\mathbf{m}}$ representam as restrições de desigualdade e igualdade, respectivamente.

Para os casos com restrições, geralmente são utilizadas técnicas de otimização. Esse processo de otimização é um ponto crítico na questão de custo computacional, que depende das restrições, do tamanho do horizonte de predição e das entradas e saídas do processo.

2.2 CONTROLE PREDITIVO GENERALIZADO

O método chamado controle preditivo generalizado foi proposto por Clarke *et al.* (1987) e se tornou um dos métodos mais populares tanto na indústria quanto na academia (CAMACHO; BORDONS, 2004). O índice a ser otimizado pelo GPC é a solução de uma função quadrática que mede a distância entre a saída predita do sistema e alguma sequência de referências futuras previstas sobre um horizonte mais uma função quadrática do esforço de controle. O GPC permite uma solução analítica, na ausência de restrições, e pode tratar plantas instáveis e de fase não mínima (CAMACHO; BORDONS, 2004).

2.2.1 Formulação do GPC

Os modelos lineares são amplamente utilizados para representar plantas SISO. Um dos possíveis modelos lineares para representar a planta e as suas perturbações é o chamado modelo autorregressivo com média móvel, integrador e entrada

controlada (CARIMA, do inglês *Controlled Autoregressive Integrated Moving Average*) (CAMACHO; BORDONS, 2004). O modelo CARIMA é dado por:

$$A(z^{-1})y(k) = B(z^{-1})z^{-d}u(k-1) + C(z^{-1})\frac{e(k)}{\Delta}, \tag{3}$$

com

$$\begin{aligned} A(z^{-1}) &= 1 + a_1 z^{-1} + a_2 z^{-2} + ... + a_{na} z^{-na}, \\ B(z^{-1}) &= b_0 + b_1 z^{-1} + b_2 z^{-2} + ... + b_{nb} z^{-nb}, \\ C(z^{-1}) &= 1 + c_1 z^{-1} + c_2 z^{-2} + ... + c_{nc} z^{-nc}, \\ \Delta &= 1 - z^{-1}, \end{aligned} \tag{4}$$

onde $u(k)$ e $y(k)$ são as sequências de ações de controle e saídas da planta, respectivamente, $e(k)$ é um ruído branco de média zero, $A(z^{-1})$, $B(z^{-1})$ e $C(z^{-1})$ são polinômios no operador de deslocamento para trás z^{-1}, d é o tempo morto do sistema e k representa as amostras do tempo discreto. O termo Δ é utilizado para uma melhor modelagem de perturbações não estacionárias, que são muito presentes em aplicações industriais (CLARKE *et al.*, 1987). Já o polinômio $C(z^{-1})$, que é difícil de ser obtido através de dados da planta, pode ser utilizado como um parâmetro de sintonização, pois afeta diretamente a robustez de malha fechada (ROSSITER, 2003). Para uma análise simplificada do GPC, tipicamente utiliza-se $C(z^{-1}) = 1$.

A função custo típica do GPC, para o caso SISO, é a mesma definida em (1). Como pode-se perceber, a função custo necessita da predição de saída y no horizonte como função das ações de controle futuras. Para se obter a predição de y a partir do modelo CARIMA, pode-se definir a seguinte equação Diofantina:

$$1 = E_j(z^{-1})\tilde{A}(z^{-1}) + z^{-j}F_j(z^{-1}), \tag{5}$$

com $\tilde{A}(z^{-1}) = \Delta A(z^{-1})$. Os polinômios E_j e F_j podem ser obtidos a partir de um procedimento de divisão de 1 por $\tilde{A}(z^{-1})$. O procedimento segue até que o resto da divisão possa ser fatorado como $z^{-j}F_j(z^{-1})$. O quociente resultante representa $E_j(z^{-1})$. Para fins de utilização em algoritmos, a solução pela equação Diofantina pode também ser obtida recursivamente como demonstrado em Clarke *et al.* (1987). Existem outras abordagens que obtêm a predição de saída sem utilizar o recurso das equações Diofantinas, como, por exemplo, a apresentada em Rossiter (2003).

A partir do modelo CARIMA, em (3), multiplicado por $\Delta E_j(z^{-1})z^j$ obtém-se:

$$\tilde{A}(z^{-1})E_j(z^{-1})y(k+j) = E_j(z^{-1})B(z^{-1})\Delta u(k+j-d-1) + E_j(z^{-1})e(k+j). \tag{6}$$

Ao isolar-se a equação Diofantina em (5) como $\tilde{A}(z^{-1})E_j(z^{-1}) = 1 - z^{-j}F_j(z^{-1})$ e substituí-la em (6), obtém-se:

$$(1 - z^{-j}F_j(z^{-1}))y(k+j) = E_j(z^{-1})B(z^{-1})\Delta u(k+j-d-1) + E_j(z^{-1})e(k+j), \tag{7}$$

que pode ser reescrita como:

$$y(k+j) = F_j(z^{-1})y(k) + E_j(z^{-1})B(z^{-1})\Delta u(k+j-d-1) + E_j(z^{-1})e(k+j). \tag{8}$$

Como o grau de $E_j(z^{-1})$ é $j-1$ e o sinal $e(k+j)$ representa um ruído branco, o comportamento de $e(k+j)$ não é conhecido em k. Como se trata de um ruído branco com média zero, o valor esperado é zero, logo a predição de saída em $k+j$ é dada por:

$$\hat{y}(k+j|k) = G_j(z^{-1})\Delta u(k+j-d-1) + F_j(z^{-1})y(k), \tag{9}$$

com $G_j(z^{-1}) = E_j(z^{-1})B(z^{-1})$.

Com o modelo de predição de saída obtido em (9), as predições dentro do horizonte N devem ser escritas como funções dos valores conhecidos até k e dos sinais de controle futuros dentro do horizonte de controle. Vale ressaltar que o início do horizonte, em geral, é definido como um passo discreto após o atraso do sistema, ou seja, $N_1 = d+1$ e, consequentemente, $N_2 = d+N$. De forma genérica, as predições de N_1 até N_2 são dadas por:

$$\begin{aligned}
\hat{y}(k+N_1|k) &= G_{N_1}(z^{-1})\Delta u(k) + F_{N_1}(z^{-1})y(k) \\
\hat{y}(k+N_1+1|k) &= G_{N_1+1}(z^{-1})\Delta u(k+1) + F_{N_1+1}(z^{-1})y(k) \\
&\vdots \\
\hat{y}(k+N_2|k) &= G_{N_2}(z^{-1})\Delta u(k+N-1) + F_{N_2}(z^{-1})y(k),
\end{aligned} \tag{10}$$

que podem ser reescritas, em forma matricial, como:

$$\mathbf{y} = \mathbf{G}\Delta\mathbf{u} + \mathbf{F}(z^{-1})y(k) + \underleftarrow{\mathbf{G}}(z^{-1})\Delta u(k-1), \tag{11}$$

onde $\mathbf{y} \in \mathbb{R}^N$, $\Delta\mathbf{u} \in \mathbb{R}^{N_u}$, $\mathbf{G} \in \mathbb{R}^{N\times N_u}$, $\mathbf{F}(z^{-1})$ de dimensões $N \times n_a + 1$ e $\underleftarrow{\mathbf{G}}(z^{-1})$ de dimensões $N \times n_b$ são dados por:

$$\mathbf{y} = \begin{bmatrix} \hat{y}(k+N_1|k) \\ \hat{y}(k+N_1+1|k) \\ \vdots \\ \hat{y}(k+N_2|k) \end{bmatrix} \qquad \Delta\mathbf{u} = \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_u-1) \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} g_{N_1} & g_{N_1-1} & \cdots & g_{N_1-N_u+1} \\ g_{N_1+1} & g_{N_1} & \cdots & g_{N_1-N_u+2} \\ \vdots & \vdots & \vdots & \vdots \\ g_{N_2} & g_{N_2-1} & \cdots & g_{N_2-N_u+1} \end{bmatrix}, \qquad \mathbf{F}(z^{-1}) = \begin{bmatrix} F_{N_1}(z^{-1}) \\ F_{N_1+1}(z^{-1}) \\ \vdots \\ F_{N_2}(z^{-1}) \end{bmatrix} \tag{12}$$

$$\underleftarrow{\mathbf{G}}(z^{-1}) = \begin{bmatrix} \underleftarrow{\mathbf{G}}_{N_1}(z^{-1}) \\ \underleftarrow{\mathbf{G}}_{N_1+1}(z^{-1}) \\ \vdots \\ \underleftarrow{\mathbf{G}}_{N_2}(z^{-1}) \end{bmatrix}.$$

Cabe ressaltar que a estrutura apresentada de $\mathbf{G}$, para sistemas causais, resulta em uma matriz triangular inferior, visto que $g_{N_1-j} = 0$ para $j > 0$. Pode-se perceber que os últimos termos da equação (11) dependem apenas do passado e podem ser agrupados, formando um vetor de reposta livre $\mathbf{f}$, tal que $\mathbf{f} = \mathbf{F}(z^{-1})y(k) + \underleftarrow{\mathbf{G}}(z^{-1})\Delta u(k-1)$, e portanto:

$$\mathbf{y} = \mathbf{G}\Delta\mathbf{u} + \mathbf{f}. \tag{13}$$

2.2.2 Cômputo do sinal de controle

Para o cômputo do sinal de controle, pode-se aplicar a predição de saída obtida em (13) na função custo do GPC, definida em (1). Dessa forma, definindo o vetor de referências futuras como $\mathbf{w} = [w(k+N_1) \;\; w(k+N_1+1) \;...\; w(k+N_2)]^T$ e os coeficientes de ponderação $q_\lambda(j)$ e $q_\delta(j)$ assumidos constantes e na forma matricial diagonal, tal que $\mathbf{Q}_\delta = \text{diag}(q_\delta(j)) \in \mathbb{R}^{N\times N}$ e $\mathbf{Q}_\lambda = \text{diag}(q_\lambda(j)) \in \mathbb{R}^{N_u\times N_u}$, tem-se:

$$J_{GPC} = (\mathbf{G}\Delta\mathbf{u} + \mathbf{f} - \mathbf{w})^T\mathbf{Q}_\delta(\mathbf{G}\Delta\mathbf{u} + \mathbf{f} - \mathbf{w}) + \Delta\mathbf{u}^T\mathbf{Q}_\lambda\Delta\mathbf{u}, \tag{14}$$

que pode ser reescrito como:

$$J_{GPC} = \frac{1}{2}\Delta\mathbf{u}^T\mathbf{H}\Delta\mathbf{u} + \mathbf{b}^T\Delta\mathbf{u} + f_0, \tag{15}$$

onde

$$\begin{aligned}\mathbf{H} &= 2(\mathbf{G}^T\mathbf{Q}_\delta\mathbf{G} + \mathbf{Q}_\lambda),\\ \mathbf{b}^T &= 2(\mathbf{f}-\mathbf{w})^T\mathbf{Q}_\delta\mathbf{G},\\ f_0 &= (\mathbf{f}-\mathbf{w})^T\mathbf{Q}_\delta(\mathbf{f}-\mathbf{w}).\end{aligned}$$

Por fim, o vetor de sinais de controle $\Delta\mathbf{u}$ do GPC pode ser computado, a cada iteração, a partir do problema de otimização dado por:

$$\begin{aligned}\min_{\Delta\mathbf{u}} \quad & \frac{1}{2}\Delta\mathbf{u}^T\mathbf{H}\Delta\mathbf{u} + \mathbf{b}^T\Delta\mathbf{u}\\ \text{s.a.} \quad & \bar{\mathbf{R}}\Delta\mathbf{u} \leq \bar{\mathbf{r}}\end{aligned} \tag{16}$$

com a matriz de restrições $\bar{\mathbf{R}} \in \mathbb{R}^{N_r\times N_u}$, o vetor $\bar{\mathbf{r}} \in \mathbb{R}^{N_r}$ e N_r o número de restrições. As restrições de igualdade em (2) foram omitidas, pois podem ser representadas como desigualdades.

Caso não sejam utilizadas restrições, o GPC possui solução analítica que é dada por:

$$\Delta\mathbf{u} = -\mathbf{H}^{-1}\mathbf{b} = (\mathbf{G}^T\mathbf{Q}_\delta\mathbf{G} + \mathbf{Q}_\lambda)^{-1}\mathbf{G}^T\mathbf{Q}_\delta(\mathbf{w}-\mathbf{f}).$$

Vale ressaltar que, em ambos os casos, apenas o primeiro incremento de controle será realmente aplicado à planta. Dessa forma:

$$\Delta u(k) = [1 \quad 0 \quad \cdots \quad 0]_{1\times N_u}\Delta\mathbf{u}$$

e a ação de controle no instante k é dada por:

$$u(k) = u(k-1) + \Delta u(k).$$

2.2.3 Algoritmo recursivo

A forma de obtenção da lei de controle utilizando a equação Diofantina, apresentada na Seção 2.2.1, nem sempre é a mais adequada para a utilização em algoritmos embarcados, devido à operação de divisão requerida. Dessa forma, neste trabalho utilizou-se uma formulação direta, na qual a matriz $\mathbf{G}$ e o vetor $\mathbf{f}$ podem ser calculadas recursivamente (CAMACHO; BORDONS, 2004, p. 51, 56–57). Os elementos g_i e f_i podem ser computados por:

$$\begin{aligned} g_i &= z(1 - A(z^{-1}))g_{i-1} + B(z^{-1})\bar{u}(i), \\ f_i &= z(1 - \tilde{A}(z^{-1}))f_{i-1} + B(z^{-1})\Delta u(k - d + i - 1), \end{aligned} \tag{17}$$

com $i = 1, ..., N_2$, $\bar{u}(i)$ sendo um degrau unitário, $g_i = 0$ para $i \leq 0$, $f_i = y(k + i)$ para $i \leq 0$ e $\Delta u(k + j) = 0$ para $j \geq 0$. Após o cômputo dos elemento g_i e f_i, a matriz $\mathbf{G}$ é definida da mesma forma de (12) e o vetor $\mathbf{f}$ pode ser escrito como:

$$\mathbf{f} = \begin{bmatrix} f_{N_1} \\ f_{N_1+1} \\ \vdots \\ f_{N_2} \end{bmatrix}.$$

Vale ressaltar que em casos práticos, com sistemas estáveis, a matriz $\mathbf{G}$ frequentemente é obtida de forma experimental a partir dos coeficientes da resposta ao degrau em malha aberta que compõem a primeira coluna $\mathbf{G}$. O algoritmo do GPC recursivo, que é utilizado na continuação do trabalho, pode ser visto no Algoritmo 1.

2.2.4 Generalização para o caso MIMO

De modo a obter-se uma generalização da formulação GPC para sistemas MIMO, é necessário, primeiro, generalizar a representação de função de transferência do caso SISO. Uma forma de representar sistemas MIMO a partir de funções de transferência é por meio do emprego de uma matriz de transferência. A matriz de transferência é uma matriz composta por funções de transferência que modelam a relação dinâmica entre todas as entradas e saídas de um dado sistema Normey-Rico e Camacho (2000). A matriz de transferência genérica para um sistema MIMO de n_i entradas e n_o saídas pode ser representada por:

$$\mathbf{M}_t = \begin{bmatrix} \frac{N_{11}(z^{-1})}{D_{11}(z^{-1})}z^{-d_{11}} & \cdots & \frac{N_{1n_i}(z^{-1})}{D_{1n_i}(z^{-1})}z^{-d_{1n_i}} \\ \vdots & \ddots & \vdots \\ \frac{N_{n_o1}(z^{-1})}{D_{n_o1}(z^{-1})}z^{-d_{n_o1}} & \cdots & \frac{N_{n_on_i}(z^{-1})}{D_{n_on_i}(z^{-1})}z^{-d_{n_on_i}} \end{bmatrix}_{n_o \times n_i}, \tag{18}$$

Algoritmo 1 Controle preditivo generalizado

Entrada: A, B, d, $\bar{\mathbf{R}}$, $\bar{\mathbf{r}}$
Saída: $u(k)$
Dados: $\mathbf{Q}_\lambda$, $\mathbf{Q}_\delta$, N_u, N_1, N_2, $u(-1)$
início
  $\tilde{A}(z^{-1}) \leftarrow (1 - z^{-1})A(z^{-1})$;
  para $i = 0 : N_2$ **faça**
    $g_i \leftarrow z(1 - A(z^{-1}))g_{i-1} + B(z^{-1})\bar{u}_i$;
  para $i = 1 : N_u$ **faça**
    $\mathbf{G}_{i:N,i} \leftarrow \mathbf{g}_{N_1:N_2+1-i}$;
  $\mathbf{H} \leftarrow 2(\mathbf{G}^T\mathbf{Q}_\delta\mathbf{G} + \mathbf{Q}_\lambda)$;
  $k \leftarrow 0$;
  enquanto *modo automatico* **faça**
    se *tempo de amostragem* **então**
      Adquire saída $y(k)$ e referência $w(k)$;
      $f_0 \leftarrow y(k)$;
      para $i = 1 : N_2$ **faça**
        $f_i \leftarrow z(1 - \tilde{A}(z^{-1}))f_{i-1} + B(z^{-1})\Delta u(k - d + i - 1)$;
      $\mathbf{b}^T \leftarrow 2(\mathbf{f} - \mathbf{w})^T\mathbf{Q}_\delta\mathbf{G}$;
      Obtém $\Delta\mathbf{u}$ resolvendo (16);
      $\Delta u(k) \leftarrow [1 \quad 0 \quad \cdots \quad 0]_{1\times N_u}\Delta\mathbf{u}$;
      $u(k) \leftarrow u(k-1) + \Delta u(k)$;
      $k \leftarrow k + 1$;
fim

onde $\mathbf{M}_t \in \mathbb{R}^{n_o \times n_i}$ e $N_{pl}(z^{-1})$, $D_{pl}(z^{-1})$ e d_{pl} representam o numerador, o denominador e o atraso de transporte discreto da função de transferência que relaciona a saída p com a entrada l, respectivamente.

Com o intuito de generalizar a representação do modelo CARIMA para o caso MIMO, pode-se decompor a matriz $\mathbf{M}_t$ na representação chamada descrição matricial fracionária (MFD, do inglês *Matrix Fraction Description*) da seguinte forma:

$$\mathbf{M}_t = \mathbf{A}(z^{-1})^{-1}\mathbf{B}(z^{-1}),$$

onde a matriz polinomial $\mathbf{A}(z^{-1})$ tem dimensões $n_o \times n_o$ e a matriz polinomial $\mathbf{B}(z^{-1})$ tem dimensões $n_o \times n_c$. A matriz $\mathbf{A}(z^{-1})$ pode ser obtida a partir da formação de uma matriz diagonal com os elementos $pl : p = l$ sendo os mínimos múltiplos comuns entre os denominadores da linha correspondente p de $\mathbf{M}_t$. A matriz $\mathbf{B}(z^{-1})$, por sua vez, pode ser calculada por $\mathbf{B}(z^{-1}) = \mathbf{A}(z^{-1})\mathbf{M}_t$.

Para representar as predições de n_o saídas, os vetores associados a cada saída podem ser agrupados em um único vetor $\hat{\mathbf{y}}_m \in \mathbb{R}^{\sum_{p=1}^{n_o} N_2^{(p)} - N_1^{(p)} + 1}$ e os n_i sinais

de controle em um vetor $\mathbf{u}_m \in \mathbb{R}^{\sum_{l=1}^{n_i} N_u^{(l)}}$, tal que:

$$\hat{\mathbf{y}}_m = \left[\hat{\mathbf{y}}^{(1)^T} \ ... \ \hat{\mathbf{y}}^{(n_o)^T}\right]^T, \ \mathbf{u}_m = \left[\mathbf{u}^{(1)^T} \ ... \ \mathbf{u}^{(n_i)^T}\right]^T. \tag{19}$$

A partir das representações das matrizes polinomiais $\mathbf{A}(z^{-1})$ e $\mathbf{B}(z^{-1})$ e dos vetores generalizados para o caso MIMO, $\hat{\mathbf{y}}_m$ e $\mathbf{u}_m$, a representação do modelo CARIMA para sistemas MIMO:

$$\mathbf{A}(z^{-1})\hat{\mathbf{y}}_m(k) = \mathbf{B}(z^{-1})\mathbf{D}(z^{-1})\mathbf{u}_m(k-1) + \mathbf{C}(z^{-1})\frac{e(k)}{\Delta}, \tag{20}$$

onde a matriz polinomial $\mathbf{C}(z^{-1})$ tem dimensões $n_o \times n_o$ e $\mathbf{D}(z^{-1})$ é uma matriz polinomial diagonal de dimensões $n_o \times n_o$ com cada elemento representando o menor tempo morto entre as entradas relacionadas às correspondentes saídas.

Com o modelo CARIMA generalizado, o modelo de predição segue os mesmos passos desenvolvidos para o caso SISO e obtém-se a mesma forma de (13). No caso MIMO, a matriz $\mathbf{G}_m$ é formada por outras matrizes $\mathbf{G}^{(p,l)}$, onde $\mathbf{G}^{(p,l)}$ é uma matriz de coeficientes da resposta ao degrau que relaciona a saída p com a entrada l. Portanto, a matriz $\mathbf{G}_m$ é dada por:

$$\mathbf{G}_m = \begin{bmatrix} \mathbf{G}^{(1,1)} & \dots & \mathbf{G}^{(1,n_i)} \\ \mathbf{G}^{(2,1)} & \dots & \mathbf{G}^{(2,n_i)} \\ \vdots & \ddots & \vdots \\ \mathbf{G}^{(n_o,1)} & \dots & \mathbf{G}^{(n_o,n_i)} \end{bmatrix}.$$

Para se obter o vetor de resposta livre generalizado $\mathbf{f}_m$, as respostas livres para cada saída do sistema são agrupadas como:

$$\mathbf{f}_m = \left[\mathbf{f}^{(1)^T} \ \dots \ \mathbf{f}^{(n_o)^T}\right]^T,$$

onde $\mathbf{f}_m^{(i)}$ tem a mesma forma de (17). Portanto, o vetor de resposta livre para cada saída p pode ser obtido recursivamente como:

$$f_i^{(p)} = z(1 - \tilde{\mathbf{A}}_{pp}(z^{-1}))f_{i-1}^{(p)} + \sum_{l=1}^{n_i} \mathbf{B}_{pl}(z^{-1})\Delta u^{(l)}(k - \mathbf{D}_{pp} + i - 1), \tag{21}$$

com $i = 1, ..., N_2^{(p)}$, $p = 1, ..., n_o$, $l = 1, ..., n_i$, $f_i^{(p)} = y^{(p)}(k+i)$ para $i \leq 0$ e $\Delta u^{(l)}(k+j) = 0$ para $j \geq 0$.

Seguindo a mesma lógica para se obter a lei de controle para o caso SISO e generalizando $\mathbf{w}$ a partir do agrupamento das referências futuras para cada saída predita similarmente a (19), o problema de otimização para o caso MIMO pode ser representado da mesma forma padrão de (16).

2.2.5 Representação de restrições

Uma característica importante do GPC, assim como de outras formulações MPC, é o tratamento explícito de restrições. Em geral, as restrições modeladas no problema de controle são utilizadas para garantir a operação dentro dos limites físicos da planta. Graças à utilização da predição, o sistema de controle pode antever uma violação futura de restrições e agir de forma apropriada para que ela não aconteça. As restrições mais comuns são a limitação da amplitude do sinal de controle, a limitação da taxa de variação do sinal de controle e a limitação da saída da planta. A seguir, é apresentada uma forma de representação dessas restrições na forma compacta do QP, definido em (16). Para um aprofundamento na utilização e representação de outros tipos de restrições, sugere-se a leitura de Camacho e Bordons (2004) e Rossiter (2003).

Uma forma de representar a limitação na amplitude do sinal de controle é:

$$\underline{u} \leq u(j) \leq \bar{u}, \quad \forall j.$$

Já para a varição do sinal de controle pode-se escrever:

$$\underline{\Delta u} \leq u(j) - u(j-1) \leq \overline{\Delta u}, \quad \forall j,$$

e para a saída predita da planta as restrições são representadas por:

$$\underline{y} \leq \hat{y(j)} \leq \bar{y}, \quad \forall j.$$

Para um sistema com restrições sobre um horizonte N ou N_U, essas restrições podem ser representadas de forma matricial por:

$$\begin{aligned} \mathbf{1}\underline{\Delta u} &\leq \Delta \mathbf{u} \leq \mathbf{1}\overline{\Delta u} \\ \mathbf{1}\underline{u} &\leq \mathbf{T}\Delta \mathbf{u} + u(k-1)\mathbf{1} \leq \mathbf{1}\overline{u} \\ \mathbf{1}\underline{y} &\leq \mathbf{G}\Delta \mathbf{u} + \mathbf{f} \leq \mathbf{1}\overline{y} \end{aligned}$$

onde $\mathbf{1}$ é um vetor de dimensão apropriada e formado por elementos iguais a 1 e $\mathbf{T} \in \mathbb{R}^{N_u \times N_u}$ é uma matriz triangular inferior cujos elementos não nulos são iguais a 1. As restrições podem então ser representadas na forma compacta apresentada em (16):

$$\bar{\mathbf{R}}\Delta \mathbf{u} \leq \bar{\mathbf{r}}$$

com:

$$\bar{\mathbf{R}} = \begin{bmatrix} \mathbf{I}_{N_u \times N_u} \\ -\mathbf{I}_{N_u \times N_u} \\ \mathbf{T} \\ -\mathbf{T} \\ \mathbf{G} \\ -\mathbf{G} \end{bmatrix}, \qquad \bar{\mathbf{r}} = \begin{bmatrix} 1\overline{\Delta u} \\ -1\underline{\Delta u} \\ 1\overline{u} - 1u(k-1) \\ -1\underline{u} + 1u(k-1) \\ 1\overline{y} - \mathbf{f} \\ -1\underline{y} + \mathbf{f} \end{bmatrix}.$$

2.3 CONTROLE POR MATRIZ DINÂMICA

O controle por matriz dinâmica (DMC, do inglês *Dynamic Matrix Control*) foi proposto no fim dos anos setenta por Cutler e Ramaker (1980) da companhia petrolífera Shell. O DMC utiliza a resposta ao degrau como modelo de predição e se tornou um método de controle bastante popular no contexto industrial, aplicável para plantas estáveis na sua versão original e permitindo a consideração de modos integradores no caso estendido. A adesão da indústria deu-se, em grande parte, devido à familiaridade na obtenção do modelo por processos conhecidos de identificação da planta e a capacidade de lidar com plantas multivariáveis de maneira simples, principalmente no processo de obtenção do modelo (CAMACHO; BORDONS, 2004). A pesquisa em aplicações industriais do DMC continua ativa e alguns trabalhos recentes podem ser citados, por exemplo, o trabalho de Fernandes *et al.* (2020), onde o DMC foi utilizado em uma planta integrada de separação de ar e ciclo Allam. Uma estratégia com DMC para controlar uma usina termelétrica a carvão é apresentada em He e Lima (2019), e em Wang *et al.* (2020) o DMC é utilizado no contexto de controle preditivo distribuído, aplicado a um processo reator e separador.

2.3.1 Formulação do DMC

O modelo de processo utilizado para a predição do sistema no cômputo do DMC é baseado na resposta ao degrau, com perturbações consideradas constantes ao longo do horizonte. A resposta ao degrau de um sistema com condições iniciais nulas e sem considerar possíveis perturbações (malha aberta), pode ser dado por:

$$y_0(k) = \sum_{i=1}^{\infty} g_i \Delta u(k-i). \tag{22}$$

O DMC utiliza o modelo da equação (22) para fazer uma predição da saída de um sistema como:

$$\hat{y}(k+j|k) = \sum_{i=1}^{\infty} g_i \Delta u(k+j-i) + \hat{n}(k+j|k), \tag{23}$$

com $\hat{n}(k+j|k)$ representando uma predição da perturbação feita no instante k e que atuará em $k+j$ (CAMACHO; BORDONS, 2004). Por conveniência, pode-se separar

a predição de saída em duas parcelas, sendo uma que depende de informações do passado e outra de dados futuros. Sendo assim, a equação (23) pode ser reescrita como:

$$\hat{y}(k+j|k) = \sum_{i=1}^{j} g_i \Delta u(k+j-i) + \sum_{i=j+1}^{\infty} g_i \Delta u(k+j-i) + \hat{n}(k+j|k). \tag{24}$$

Devido ao fato de as perturbações futuras serem desconhecidas, pode-se considerar que as mesmas serão constantes e portanto $\hat{n}(k+j|k) = \hat{n}(k|k) = y(k) - \hat{y}(k|k)$, onde $y(k)$ é o valor medido da saída no instante k. Dessa forma, a equação (24) se torna:

$$\hat{y}(k+j|k) = \sum_{i=1}^{j} g_i \Delta u(k+j-i) + \sum_{i=j+1}^{\infty} g_i \Delta u(k+j-i) + y(k) - \sum_{i=1}^{\infty} g_i \Delta u(k-i), \tag{25}$$

que, por sua vez, pode ser reescrita de forma mais compacta:

$$\hat{y}(k+j|k) = \sum_{i=1}^{j} g_i \Delta u(k+j-i) + f(k+j), \tag{26}$$

onde $f(k+j)$ é a resposta livre do sistema, ou seja, a parte da resposta que não depende das ações de controle futuras, e é dada por:

$$f(k+j) = y(k) + \sum_{i=1}^{\infty} (g_{j+i} - g_i) \Delta u(k-i). \tag{27}$$

Para sistemas assintoticamente estáveis, a resposta ao degrau tende a se tornar constante após um número suficiente de amostras N_{ss}. Portanto, $(g_{j+i} - g_i) \approx 0,\ \forall i > N_{ss}$ e o somatório de (27) pode ser truncado em N_{ss} e ser reapresentado como:

$$f(k+j) = y(k) + \sum_{i=1}^{N_{ss}} (g_{j+i} - g_i) \Delta u(k-i). \tag{28}$$

As predições de saídas, equação (26), podem ser representadas de forma matricial e adquirir uma formulação idêntica àquela adotada para o GPC, com:

$$\hat{\mathbf{y}} = \mathbf{G}\Delta\mathbf{u} + \mathbf{f}. \tag{29}$$

onde $\mathbf{G} \in \mathbb{R}^{N \times N_u}$ é a matriz de coeficientes da resposta ao degrau e $\mathbf{f} \in \mathbb{R}^{N}$ a resposta livre do sistema, dados por:

$$\mathbf{G} = \begin{bmatrix} g_{N_1} & g_{N_1-1} & \cdots & g_{N_1-N_u+1} \\ g_{N_1+1} & g_{N_1} & \cdots & g_{N_1-N_u+2} \\ \vdots & \vdots & \vdots & \vdots \\ g_{N_2} & g_{N_2-1} & \cdots & g_{N_2-N_u+1} \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} f_{N_1} \\ f_{N_1+1} \\ \vdots \\ f_{N_2} \end{bmatrix}.$$

A matriz $\mathbf{G}$ é a mesma matriz do GPC e a estrutura apresentada resulta em uma matriz triangular inferior, visto que $g_{N_1-j} = 0$ para $j > 0$. No caso do DMC, e em aplicações práticas do GPC, $\mathbf{G}$ pode ser obtida diretamente, uma vez que o vetor de coeficientes da resposta ao degrau $\mathbf{g}$ é o modelo fornecido para a predição. Já para o cômputo de $\mathbf{f}$, pode-se separar entre os termos que dependem dos N_{SS} incrementos de controle passados $\underleftarrow{\Delta\mathbf{u}}$ e da predição de saída em $k+d$, dada por $\hat{\mathbf{y}}(k+d|k)$. Dessa forma,

$$\mathbf{f} = \mathbf{F}\underleftarrow{\Delta\mathbf{u}} + \mathbf{1}\hat{y}(k+d|k), \tag{30}$$

com $\mathbf{1} \in \mathbb{R}^N$ sendo um vetor formado por elementos iguais a 1 e $\mathbf{F} \in \mathbb{R}^{N\times N_{ss}-1}$ e $\underleftarrow{\Delta\mathbf{u}} \in \mathbb{R}^{N_{ss}-1}$ dados por:

$$\mathbf{F} = \begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,N_{ss}-2} & f_{1,N_{ss}-1} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,N_{ss}-2} & f_{2,N_{ss}-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ f_{N,1} & f_{N,2} & \cdots & f_{N,N_{ss}-2} & f_{N,N_{ss}-1} \end{bmatrix}, \ \underleftarrow{\Delta\mathbf{u}} = \begin{bmatrix} \Delta u(k-1) \\ \Delta u(k-2) \\ \vdots \\ \Delta u(k-N_{ss}-1) \end{bmatrix},$$

com

$$f_{i,j} = g_{i+j} - g_j, \ i+j \leq N_{ss},$$
$$f_{i,j} = 0, \ i+j > N_{ss}.$$

Percebe-se que essa formulação exige que os sinais de controle passados $\underleftarrow{\Delta\mathbf{u}}$ sejam armazenados. A formulação recursiva, que é apresentada na Seção 2.3.3, traz uma solução mais eficiente e que necessita de menos dados armazenados.

2.3.2 Cômputo do sinal de controle

O cômputo do sinal de controle do DMC, a partir da predição definida em (29) e da função custo de (14) segue o mesmo desenvolvimento do GPC e recai na solução do QP apresentado em (16).

2.3.3 DMC recursivo

O DMC pode ser formulado a partir da utilização de recursão, cujas características se tornam mais adequadas para a implementação em sistemas embarcados. O DMC recursivo foi apresentado em Cutler e Ramaker (1980) e requer menos espaço de memória para armazenar variáveis. Para desenvolver a predição recursiva do DMC, parte-se da representação das predições de malha aberta em $k+j$ dados em k e $k-1$, que respectivamente são dados por:

$$\hat{y}(k+j|k) = \sum_{i=j+1}^{\infty} g_i \Delta u(k+j-i), \tag{31}$$

$$\hat{y}(k+j|k-1) = \sum_{i=j+2}^{\infty} g_i \Delta u(k+j-i). \tag{32}$$

Subtraindo a equação (32) de (31), tem-se:

$$\hat{y}(k+j|k) - \hat{y}(k+j|k-1) = \sum_{i=j+1}^{\infty} g_i \Delta u(k+j-i) - \sum_{i=j+2}^{\infty} g_i \Delta u(k+j-i),$$

que pode ser rearranjada como,

$$\hat{y}(k+j|k) = \hat{y}(k+j|k-1) + g_{j+1}\Delta u(k-i) + \sum_{i=j+2}^{\infty} g_i \Delta u(k+j-i) - \sum_{i=j+2}^{\infty} g_i \Delta u(k+j-i),$$

e, por fim, obtém-se uma predição de saída em malha aberta recursiva dada por:

$$\hat{y}(k+j|k) = \hat{y}(k+j|k-1) + g_{j+1}\Delta u(k-1). \tag{33}$$

A abordagem do DMC recursivo pode ser utilizada para computar a resposta livre do sistema mais facilmente. No começo de cada período de amostragem há um vetor de predições de saída do sistema desatualizado,

$$\hat{\mathbf{f}}(k-1) = [\hat{y}(k-1|k-1)\ \hat{y}(k|k-1)\ ...\ \hat{y}(k+N_{ss}|k-1)].$$

O vetor atualizado é computado com:

$$\hat{\mathbf{f}}(k) = \hat{\mathbf{f}}(k-1) + \begin{bmatrix} g_1 \\ \vdots \\ g_{N_{ss}} \end{bmatrix} \Delta u(k-1). \tag{34}$$

O primeiro elemento de $\hat{\mathbf{f}}$ é então $\hat{y}(k|k)$, o qual é utilizado para computar o erro de predição $e(k|k)$. Esse valor é então descartado do vetor uma vez que apenas as predições de $k+1$ até $k+N_{ss}$ são agora necessárias. Assumindo que a planta é inicializada em regime permanente, $\hat{\mathbf{f}}(0)$ tem todos os seus elementos iguais a $y(0)$. O Vetor $\hat{\mathbf{f}}$ deve também ser deslocado a cada iteração de controle devido ao horizonte deslizante. Entretanto, quando o vetor é deslocado, o elemento $\hat{y}(k|k)$ é descartado e, para manter as dimensões apropriadas, o valor de $\hat{y}(k+N_{ss}-1|k)$ é repetido na última posição do vetor. Essa abordagem pode ser adotada porque, para sistemas estáveis, $\hat{y}(k+N_{ss}-1|k) \approx \hat{y}(k+N_{ss}|k)$ para um N_{ss} suficientemente grande (LIMA *et al.*, 2014). Dessa forma $\hat{\mathbf{f}}(k)$ atualizado é dado por:

$$\hat{\mathbf{f}}(\mathbf{k}) = \begin{bmatrix} \hat{y}(k+1|k) \\ \hat{y}(k+2|k) \\ \vdots \\ \hat{y}(k+N_{ss}-1|k) \\ \hat{y}(k+N_{ss}-1|k) \end{bmatrix}.$$

A partir dos desenvolvimentos apresentados, um algoritmo para o DMC recursivo pode ser proposto, como apresentado no Algoritmo 2.

Algoritmo 2 Controle por matriz dinâmica recursivo

Entrada: $\mathbf{g}$, $\bar{\mathbf{R}}$, $\bar{\mathbf{r}}$
Saída: $u(k)$
Dados: $\mathbf{Q}_\delta$, $\mathbf{Q}_\lambda$, N_u, N_1, N_2, N_{ss}, $u(-1)$
início
  $N \leftarrow N_2 - N_1 + 1$;
  para $i = 1 : N_u$ **faça**
    $\mathbf{G}_{i:N,i} \leftarrow \mathbf{g}_{N_1:N_2+1-i}$;
  $\mathbf{H} \leftarrow 2(\mathbf{G}^T\mathbf{Q}_\delta\mathbf{G} + \mathbf{Q}_\lambda)$;
  $\hat{\mathbf{f}}(-1) \leftarrow \mathbf{1}_{N_{ss}} y(k)$;
  $k \leftarrow 0$;
  enquanto *modo automático* **faça**
    se *tempo de amostragem* **então**
      Adquire saída $y(k)$ e referência $w(k)$;
      $\hat{\mathbf{f}}(k) \leftarrow \hat{\mathbf{f}}(k-1) + \mathbf{g}\Delta u(k-1)$;
      $e \leftarrow (y(k) - \hat{\mathbf{f}}_0)$;
      $\hat{\mathbf{f}}(k) \leftarrow [\hat{\mathbf{f}}_1\ \hat{\mathbf{f}}_2\ ...\ \hat{\mathbf{f}}_{N_{ss}-1}\ \hat{\mathbf{f}}_{N_{ss}-1}]^T$;
      $\mathbf{f} \leftarrow \hat{\mathbf{f}}_{N1:N2} + \mathbf{1}_{N\times 1} e$;
      $\mathbf{b}^T \leftarrow 2(\mathbf{f} - \mathbf{w})^T\mathbf{Q}_\delta\mathbf{G}$;
      Obtém $\Delta\mathbf{u}$ resolvendo (16);
      $\Delta u(k) \leftarrow [1 \quad 0 \quad \cdots \quad 0]_{1\times N_u}\Delta\mathbf{u}$;
      $u(k) \leftarrow u(k-1) + \Delta u(k)$;
      $k \leftarrow k + 1$;
fim

2.4 MPC POR VARIÁVEIS DE ESTADO

Considerando uma planta com comportamento linear, determinístico e invariante no tempo, a sua representação em espaço de estados discreta é dada por:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k) \end{aligned} \tag{35}$$

onde $\mathbf{x} \in \mathbb{R}^{n_s}$ representa o vetor de variáveis de estado, $\mathbf{u} \in \mathbb{R}^{n_i}$ é o vetor de sinais de controle, $\mathbf{y} \in \mathbb{R}^{n_o}$ é o vetor de saídas, $\mathbf{A} \in \mathbb{R}^{n_s\times n_s}$ é a matriz de estados, $\mathbf{B} \in \mathbb{R}^{n_s\times n_i}$ é a matriz de entrada e $\mathbf{C} \in \mathbb{R}^{n_s\times n_i}$ é a matriz de saída.

A partir da representação em espaço de estados de um sistema genérico (35) e objetivos de controle, pode-se obter uma formulação MPC. Uma forma comum e presente em trabalhos como Patrinos e Bemporad (2014), Wang e Boyd (2010) e

Ferreau *et al.* (2017) é definida por:

$$\begin{aligned}
\min_{\mathbf{u},\mathbf{x}} \quad & \sum_{j=0}^{N-1} \ell(\mathbf{x}(j),\mathbf{u}(j)) + \ell_f(\mathbf{x}(N)) \\
\text{s.a.} \quad & \mathbf{x}(0) = \mathbf{x}(k); \\
& \mathbf{x}(j+1) = \mathbf{A}\mathbf{x}(j) + \mathbf{B}\mathbf{u}(j), \quad j = 0,..,N-1; \\
& \bar{\mathbf{R}}_x\mathbf{x}(j) + \bar{\mathbf{R}}_u\mathbf{u}(j) \leq \bar{\mathbf{r}}_{xu}, \quad j = 0,..,N-1; \\
& \bar{\mathbf{R}}_N\mathbf{x}(N) \leq \bar{\mathbf{r}}_N.
\end{aligned} \tag{36}$$

Diferentemente do GPC e DMC, no ssMPC, tipicamente, o horizonte de predição é igual ao horizonte de controle, ou seja, $N = N_u$. A função custo quadrática é representada por:

$$\ell(\mathbf{x},\mathbf{u}) = \frac{1}{2}\begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix}^T \begin{bmatrix}\mathbf{Q} & \mathbf{S}^T\\ \mathbf{S} & \mathbf{R}\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix} \tag{37}$$

onde $\mathbf{Q} = \mathbf{Q}^T \in \mathbb{R}^{n_s \times n_s}$, $\mathbf{R} = \mathbf{R}^T \in \mathbb{R}^{n_i \times n_i}$, $\mathbf{S} \in \mathbb{R}^{n_s \times n_i}$ são os parâmetros de ponderação dos estados, das ações de controle e do produto cruzado, respectivamente. A matriz de ponderações deve ser positiva semidefinida, ou seja:

$$\begin{bmatrix}\mathbf{Q} & \mathbf{S}^T\\ \mathbf{S} & \mathbf{R}\end{bmatrix} \succeq \mathbf{0}.$$

A primeira restrição de igualdade, $\mathbf{x}(0) = \mathbf{x}(k)$, define o estado no tempo discreto k como a condição inicial para as predições. A segunda restrição de igualdade, $\mathbf{x}(j+1) = \mathbf{A}\mathbf{x}(j) + \mathbf{B}\mathbf{u}(j)$, impõe a dinâmica modelada do sistema. Já a restrição de desigualdade, $\bar{\mathbf{R}}_x\mathbf{x}(j) + \bar{\mathbf{R}}_u\mathbf{u}(j) \leq \bar{\mathbf{r}}_{xu}$ com $\bar{\mathbf{R}}_x \in \mathbb{R}^{n_{rin} \times n_s}$, $\bar{\mathbf{R}}_u \in \mathbb{R}^{n_{rin} \times n_u}$ e $\bar{\mathbf{r}}_{xu} \in \mathbb{R}^{n_{rin}}$, representa as restrições nos estados e nos sinais de controle a partir de um conjunto de inequações lineares.

O segundo termo da função custo, $\ell_f : \mathbb{R}^{n_s} \to \mathbb{R}$ é chamado de custo terminal e é utilizado para obter-se garantia de estabilidade no caso nominal. Nesse contexto, o custo terminal pode ser dada por:

$$\ell_f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{Q}_N\mathbf{x} + \mathbf{q}_N^T\mathbf{x} \tag{38}$$

com $\mathbf{Q}_N \succeq \mathbf{0}$.

A última restrição de (36) aparece como outra forma de garantir a estabilidade do caso nominal e é conhecida como restrição terminal. Para maiores detalhes sobre a utilização do custo terminal e da restrição terminal, recomenda-se a leitura de Mayne *et al.* (2000).

Existem algumas abordagens para a reformulação do MPC em espaço de estados para a forma padrão de QP. Em Wang e Boyd (2010) é apresentada uma abordagem que mantém a característica de esparsidade das matrizes que podem ser

exploradas no cômputo do problema de otimização. Nessa abordagem é redefinida a variável de decisão, sendo composta pelos estados e entradas, da seguinte forma:

$$\mathbf{z}_{ss} = \begin{bmatrix} \mathbf{u}(k) \\ \mathbf{x}(k+1) \\ \vdots \\ \mathbf{u}(k+N-1) \\ \mathbf{x}(k+N) \end{bmatrix}_{N(n_u+n_s)},$$

com $\mathbf{z}_{ss} \in \mathbb{R}^{N(n_u+n_s)}$. Dessa forma, mantendo as restrições de igualdade, é obtido o seguinte problema de programação quadrática:

$$\begin{aligned} \min_{\mathbf{z}_{ss}} \quad & \frac{1}{2}\mathbf{z}_{ss}^T \mathbf{H} \mathbf{z}_{ss} + \mathbf{b}^T \mathbf{z}_{ss} \\ \text{s.a.} \quad & \bar{\mathbf{R}}\mathbf{z}_{ss} \leq \bar{\mathbf{r}}; \\ & \bar{\mathbf{M}}\mathbf{z}_{ss} = \bar{\mathbf{m}}, \end{aligned} \tag{39}$$

onde $\mathbf{H} \in \mathbb{R}^{N(n_u+n_s)\times N(n_u+n_s)}$, $\mathbf{b} \in \mathbb{R}^{N(n_u+n_s)}$, $\bar{\mathbf{R}} \in \mathbb{R}^{Nn_{rin}\times N(n_u+n_s)}$, $\bar{\mathbf{r}} \in \mathbb{R}^{Nn_{rin}}$, $\bar{\mathbf{M}} \in \mathbb{R}^{Nn_{req}\times N(n_u+n_s)}$ e $\bar{\mathbf{m}} \in \mathbb{R}^{Nn_{req}}$ são dados por:

$$\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \mathbf{S} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}^T & \mathbf{R} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{Q} & \mathbf{S} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{S}^T & \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{Q}_N \end{bmatrix}; \quad \mathbf{b} = \begin{bmatrix} \mathbf{r} + \mathbf{S}^T\mathbf{x}(k) \\ \mathbf{q} \\ \mathbf{r} \\ \vdots \\ \mathbf{q} \\ \mathbf{r} \\ \mathbf{q}_N \end{bmatrix};$$

$$\bar{\mathbf{R}} = \begin{bmatrix} \bar{\mathbf{R}}_u & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{R}}_x & \bar{\mathbf{R}}_u & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \bar{\mathbf{R}}_x & \bar{\mathbf{R}}_u & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \bar{\mathbf{R}}_N \end{bmatrix}; \quad \bar{\mathbf{r}} = \begin{bmatrix} \bar{\mathbf{r}}_{xu} + \bar{\mathbf{R}}_x\mathbf{x}(k) \\ \bar{\mathbf{r}}_{xu} \\ \vdots \\ \bar{\mathbf{r}}_{xu} \\ \bar{\mathbf{r}}_N \end{bmatrix};$$

$$\bar{\mathbf{M}} = \begin{bmatrix} -\mathbf{B} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{A} & -\mathbf{B} & \mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{A} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & -\mathbf{A} & -\mathbf{B} & \mathbf{I} \end{bmatrix}; \quad \bar{\mathbf{m}} = \begin{bmatrix} \mathbf{A}\mathbf{x}(k) \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}.$$

Pode-se perceber que essa abordagem é bastante direta na montagem das matrizes e com boas características de esparsidade que favorecem o cômputo rápido em alguns algoritmos de otimização. Entretanto, o ônus dessa abordagem está associado com as elevadas dimensões das matrizes e vetores. Para se ter uma base numérica, um sistemas simples com 2 entradas, 4 estados e um horizonte de predição de 20 amostras gera um QP com 120 variáveis de decisão e 328 restrições, considerando restrições de limites inferior e superior nas variáveis de decisão (WANG; BOYD, 2010).

Uma forma de reduzir o tamanho do QP é substituir explicitamente as restrições de igualdade na função custo, deixando assim apenas a ação de controle como variável de decisão. Nesse caso, as matrizes obtidas serão densas. As predições em espaço de estados podem ser dadas por:

$$\begin{bmatrix} \mathbf{x}(k+1|k) \\ \mathbf{x}(k+2|k) \\ \vdots \\ \mathbf{x}(k+N-1|k) \end{bmatrix} = \begin{bmatrix} \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^N \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} \mathbf{B} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{AB} & \mathbf{B} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \cdots & \mathbf{B} \end{bmatrix} \begin{bmatrix} \mathbf{u}(k|k) \\ \mathbf{u}(k+1|k) \\ \vdots \\ \mathbf{u}(k+N-1|k) \end{bmatrix} \tag{40}$$

ou de forma compacta:

$$\hat{\mathbf{x}} = \hat{\mathbf{A}}\mathbf{x}(k) + \hat{\mathbf{B}}\hat{\mathbf{u}}. \tag{41}$$

Substituindo a predição de (41) na função objetivo, para o caso comum de $\mathbf{S} = \mathbf{S}^T = \mathbf{0}$ e sem a restrição terminal, obtém-se o QP na forma padrão:

$$\begin{aligned} \min_{\mathbf{u}} \quad & \frac{1}{2}\mathbf{u}^T\mathbf{H}\mathbf{u} + \mathbf{b}^T\mathbf{u} \\ \text{s.a.} \quad & \bar{\mathbf{R}}\mathbf{u} \leq \bar{\mathbf{r}}, \end{aligned} \tag{42}$$

com

$$\mathbf{H} = \hat{\mathbf{R}} + \hat{\mathbf{B}}^T\hat{\mathbf{Q}}\hat{\mathbf{B}},$$
$$\mathbf{b}^T = \mathbf{x}(k)^T\hat{\mathbf{A}}^T\hat{\mathbf{Q}}\hat{\mathbf{B}},$$

onde $\hat{\mathbf{Q}}$ e $\hat{\mathbf{R}}$ são matrizes bloco-diagonais cujos elementos são as matrizes de ponderação $\mathbf{Q}$ e $\mathbf{R}$, respectivamente.

Ainda pode ser citada uma variação que estende a definição de QP para um QP multiparamétrico (mp-QP, do inglês *multi-parametric quadratic programming*) e está presente em trabalhos de MPC embarcado de cômputo rápido, como Cairano *et al.* (2013) e Bemporad *et al.* (2002). Nesse caso do mp-QP, o gradiente e os vetores de restrições podem ser dependentes de forma afim de um dado parâmetro. No caso

do MPC, as variáveis de estado são utilizadas como um parâmetro e o mp-QP é apresentado da seguinte forma:

$$\begin{aligned} \min_{\mathbf{u}} \quad & \frac{1}{2}\mathbf{u}^T\mathbf{H}\mathbf{u} + \mathbf{x}^T(k)\mathbf{b}^T\mathbf{u} \\ \text{s.a.} \quad & \bar{\mathbf{R}}\mathbf{u} \leq \bar{\mathbf{r}} + \bar{\mathbf{r}}_S\mathbf{x}(k). \end{aligned} \tag{43}$$

Essa abordagem permite a utilização de outras ferramentas de otimização e serve como base para algumas abordagens do tipo MPC explícito, na qual a otimização é feita offline.

Vale citar que as formulações por variáveis de estado apresentadas levam em consideração apenas o caso de regulação e existem ainda formulações para a inclusão de referências, ação integral, variáveis estocásticas, entre outras características desejadas de controle (MACIEJOWSKI, 2002; WANG, 2009). Outra premissa assumida é que todos os estados do sistema estão disponíveis e, em caso negativo, pode ser utilizado um observador de estados associado à solução de controle.

2.5 MPC EMBARCADO

Para as soluções de hardware, em geral, existem pesquisas de implementação de MPC de cômputo rápido embarcado em processadores e em FPGAs (PEYRL *et al.*, 2014). As principais soluções com FPGA propõem a implementação do otimizador diretamente na lógica com emprego de alguma linguagem de descrição de *hardware*, como o Verilog e o VHDL. Entretanto, Os FPGAs possuem muitas limitações de recursos. Em Peyrl *et al.* (2014) foi implementado um algoritmo de otimização com o método de gradiente descendente paralelo em um FPGA, um processador de um núcleo e um de oito núcleos. O trabalho conclui que, comparando o mesmo algoritmo, o FPGA é duas ordens de grandeza mais rápido que o implementado em um processador de um núcleo e que, devido ao tempo de comunicação entre os núcleos, o FPGA ainda leva vantagem em relação ao de 8 núcleos. Em Wills *et al.* (2011) é implementado um algoritmo de otimização de ponto interior e consegue-se controlar um sistema de vibração com um horizonte de controle de 12 amostras e restrições de saturação do controle em todo o horizonte, em 30 μs. Já em Yang *et al.* (2012), que implementam um MPC em FPGA através de um otimizador por conjuntos ativos, foi testado o controle de posição de um servomotor com horizonte de controle de 10 unidades em tempo de 20 μs. Os cálculos dos produtos internos entre vetores nesse trabalho foram feitos em paralelo, aproveitando essa característica do FPGA para reduzir drasticamente o tempo de cômputo. Outros exemplos de aplicação recentes podem ser vistos em Gulbudak e Santi (2016), que apresenta a implementação de um MPC para conversores de potência em FPGA, e Hartley e Maciejowski (2015) que discutem a aplicação de MPC rápido em FPGA para o acoplamento de espaçonaves em órbitas elípticas. Uma implementação

DMC em um FPGA foi apresentada em Wojtulewicz e Ławryńczuk (2018), entretanto ela apenas trata o caso irrestrito, o qual não necessita que um QP seja resolvido de forma online.

2.6 COMENTÁRIOS FINAIS

Neste capítulo foram apresentadas definições, revisão da literatura e desenvolvimentos necessários para formar a base de entendimento da pesquisa. Foram revisadas as formulações do GPC e do DMC e definidos algoritmos para os cálculos da ação de controle que são utilizados ao longo do documento.

3 TÉCNICAS DE OTIMIZAÇÃO PARA MPC EMBARCADO

Neste capítulo são apresentados conceitos importantes, do estado da arte, pertinentes para o campo de estudo da pesquisa. Apesar de a pesquisa ser voltada para leitores da área de controle, são tratados, com certa profundidade, conceitos e métodos da área de otimização. Dessa forma, são apresentadas no Apêndice A definições formais dos principais termos da área de otimização que são utilizados neste trabalho. As definições estão concentradas com o intuito de evitar repetições devido às referências em diversos pontos distintos do texto e para facilitar uma eventual consulta durante a leitura.

3.1 MPC EXPLÍCITO

A linha mais popular do MPC explícito foi apresentada originalmente em Bemporad *et al.* (2002) e se baseia no fato de a solução do problema de otimização ser uma função afim por partes contínua. Isso permite que os cálculos do problema de otimização sejam realizados offline. Nessa técnica, o problema de otimização é reformulado como um problema de programação quadrática multiparamétrica offline. Os resultados da otimização são armazenados na memória através de estruturas como as *lookup tables*. Em Johansen *et al.* (2006) são apresentados detalhes para a implementação do MPC explícito em um FPGA padrão de 20 000 portas, obtendo um tempo de computação da ordem de um microssegundo. Esse tipo de implementação de MPC com restrições possibilita aplicações industriais de pequena escala caracterizadas por rápidos períodos de amostragem e baixo custo de produção, como máquinas, equipamentos mecatrônicos, sistemas microeletrônicos (MEMS), eletrônica de potência e acústica. Entretanto, esse tipo de solução geralmente se aplica em sistemas de pequenas dimensões devido a limitações de memória (CAI *et al.*, 2014). A literatura apresenta diferentes definições para problemas de pequeno porte, mas em linha geral pode-se definir como um sistema com menos de 5 estados, 3 entradas e 12 restrições (WANG; BOYD, 2010).

3.2 MÉTODO DE CONJUNTO ATIVO

O método de conjunto ativo (AS, do inglês *Active Set*) é considerado o método mais antigo para a solução de QP e foi apresentado em Dantzig (1963). Foi desenvolvido a partir de uma variação para QP do método *simplex*. A ideia principal é adaptar o problema, avaliando quais das restrições de desigualdade estão ativas, e formar um conjunto de trabalho. A partir desse conjunto, o QP é resolvido apenas com restrições de igualdade, problema esse que é mais simples de ser resolvido. Em geral, para um conjunto não muito grande de restrições, o AS tem uma convergência muito rápida e

com boa precisão na solução de QPs (FERREAU *et al.*, 2017). Historicamente esse método sofre, do ponto de vista teórico, com o cálculo da garantia de pior caso do número de iterações. Esse foi um dos principais motivadores para o grande desenvolvimento das técnicas conhecidas por ponto interior (IP) (FERREAU *et al.*, 2017). Entretanto, em um trabalho recente, Cimini e Bemporad (2017) publicaram uma forma exata de certificação de complexidade para a versão Dual do AS. Isso mostra que a pesquisa nesse método ainda está ativa e pode produzir boas soluções.

Bemporad (2016) propôs um novo método de AS, para solucionar QPs resultantes de MPCs embarcados, através da reformulação do problema. A ideia principal é reformular o QP como um problema de menor distância (LDP, do inglês *Least Distance Problem*) baseado em mínimos quadrados não negativos (NNLS, do inglês *Non-Negative Least Squares*). O autor argumenta que essa abordagem é mais simples de codificar e mais rápida para computar. Alguns problemas de robustez numérica foram resolvidos em Bemporad (2018).

3.3 MÉTODO DE PONTO INTERIOR

Os métodos de ponto interior se baseiam na reformulação das restrições de desigualdades, que são realocadas na função objetivo. Em geral, isso é feito através de uma função de penalização logarítmica que é redimensionada a cada iteração (BOYD; VANDENBERGHE, 2004). Após a reformulação, o QP é então resolvido pelo método de Newton e suas variantes. A utilização do IP em otimização embarcada foi apresentada em Rao *et al.* (1998). Em Ling *et al.* (2006) é implementado um IP em um FPGA, mostrando a viabilidade do MPC embarcado com essa técnica. Em Wang e Boyd (2010) é apresentada uma variação do IP capaz de computar 100 vezes mais rápido a ação de controle do que com um otimizador genérico para o mesmo problema. O método de otimização abordado é baseado nas técnicas de ponto interior, mais especificamente, o método de barreira logarítmica com passo de Newton com inicialização infactível. A partir dessa técnica, são elencadas 3 variações importantes para acelerar a computação da ação de controle. A primeira, e mais importante, é a exploração da estrutura das matrizes do QP de modo a se obter uma forma eficiente de inversão baseado na decomposição de Cholesky para matrizes bloco-tridiagonais. A segunda é a forma da inclusão de inicialização a quente (*warm-starting*). Já a terceira se baseia na antecipação da parada do algoritmo com uma solução subótima em poucos passos do algoritmo.

A seguir é apresentado um algoritmo particular de ponto interior, bastante popular, primeiramente proposto em Fiacco e McCormick (1968) e posteriormente republicado em Fiacco e McCormick (1990), usualmente chamado de método de barreira.

3.3.1 Método de barreira

Os métodos de ponto interior foram desenvolvidos para resolver problemas convexos da seguinte forma:

$$\begin{aligned} \min_{\mathbf{u}} \quad & h(\mathbf{u}) \\ \text{s.a.} \quad & \bar{r}_i(\mathbf{u}) \leq 0, \quad i = 1,...,n_{rin} \\ & \bar{\mathbf{M}}\mathbf{u} = \bar{\mathbf{m}} \end{aligned} \tag{44}$$

com $h : \mathbb{R}^n \to \mathbb{R}$ sendo uma função convexa contínua duplamente diferenciável, $\mathbf{u} \in \mathbb{R}^n$, $\bar{\mathbf{M}} \in \mathbb{R}^{n_{req} \times n}$, $\bar{\mathbf{m}} \in \mathbb{R}^{n_{req}}$, n é a quantidade de graus de liberdade e n_{rin} e n_{req} são as quantidades de restrições de desigualdade e igualdade, respectivamente.

Os métodos IP resolvem problemas do tipo (44) a partir da aplicação do método de Newton para uma sequência de subproblemas com restrições de igualdade (BOYD; VANDENBERGHE, 2004). Para adaptar o problema original, um algoritmo particular bastante utilizado é o método de barreira.

O método de barreira tem como objetivo aproximar um problema de otimização com restrições de desigualdade de um problema com apenas restrições de igualdade nos quais o método de Newton pode ser aplicado. De forma genérica, pode-se reescrever o problema (44) com as restrições de desigualdade definidas implicitamente na função objetivo:

$$\begin{aligned} \min_{\mathbf{u}} \quad & h(\mathbf{u}) + \sum_{i=0}^{n_{rin}} I\_(\bar{r}_i(\mathbf{u})) \\ \text{s.a.} \quad & \bar{\mathbf{M}}\mathbf{u} = \bar{\mathbf{m}} \end{aligned} \tag{45}$$

onde $I\_ : \mathbb{R} \to \mathbb{R}$ é uma função indicadora para reais não-positivos definida como:

$$I\_(\bar{u}) = \begin{cases} 0, & \text{se } \bar{u} \leq 0 \\ \infty, & \text{se } \bar{u} > 0 \end{cases}. \tag{46}$$

A formulação (45) não apresenta restrições de desigualdade, mas, em geral, a sua função objetivo não é diferenciável e o método de Newton não pode ser aplicado. De modo a fazer uma aproximação da função indicadora, pode-se utilizar a função logarítmica (BOYD; VANDENBERGHE, 2004):

$$\hat{I}\_ = -(1/t)\log(-\bar{u}), \quad \mathbf{dom}\ \hat{I}\_ = -\mathbb{R}_+,$$

onde $t > 0$ é um parâmetro que ajusta a exatidão da aproximação.

Substituindo $\hat{I}\_$ por $I\_$ em (45) resulta o problema aproximado:

$$\begin{aligned} \min_{\mathbf{u}} \quad & h(\mathbf{u}) + \sum_{i=0}^{n_{rin}} -(1/t)\log(-\bar{r}_i(\mathbf{u})) \\ \text{s.a.} \quad & \bar{\mathbf{M}}\mathbf{u} = \bar{\mathbf{m}}. \end{aligned} \tag{47}$$

A função objetivo de (47) é convexa, uma vez que $-(1/t)\log(-\bar{u})$ é convexa e diferenciável, logo o algoritmo de Newton se aplica. A função

$$\eta = \sum_{i=0}^{n_{rin}} -\log(-\bar{r}_i(u)),$$

com $\mathbf{dom}\ \eta = u \in \mathbb{R}^n \mid \bar{r}_i(\mathbf{u}) < 0,\ i = 1,...,n_{rin}$, é chamada de barreira logarítmica para o problema (47).

3.3.2 Caminho central

Outro conceito importante para o método de barreira formulado em (47) é o do caminho central. Simplificando a notação de (47) e multiplicando por t, obtemos:

$$\begin{aligned} \min_{\mathbf{u}} \quad & th(\mathbf{u}) + \eta(\mathbf{u}) \\ \text{s.a.} \quad & \bar{\mathbf{M}}\mathbf{u} = \bar{\mathbf{m}}, \end{aligned} \tag{48}$$

o qual tem os mesmos mínimos. Assumindo que o problema (48) pode ser resolvido pelo método de Newton e, em particular, que tem solução única para cada $t > 0$, define-se $\mathbf{u}^*(t)$ como a sua solução. O caminho central associado ao problema (44) é definido como o conjunto de pontos $\mathbf{u}^*(t)$, com $t > 0$, os quais são chamados de pontos centrais (BOYD; VANDENBERGHE, 2004). Os pontos centrais são caracterizados por atenderem as seguintes condições necessárias e suficientes: $\mathbf{u}^*(t)$ é estritamente factível se e somente se satisfaz

$$\bar{\mathbf{M}}\mathbf{u}^*(t) = \bar{\mathbf{m}}, \quad \bar{r}_i(\mathbf{x}^*) \leq 0, \quad i = 1,...,n_{rin},$$

e existe um $\boldsymbol{\nu} \in \mathbb{R}^{n_{req}}$ tal que

$$t\nabla h(\mathbf{u}^*(t)) + \nabla\eta(\mathbf{u}^*(t)) + \bar{\mathbf{M}}^T\boldsymbol{\nu} = 0.$$

$$t\nabla h(\mathbf{u}^*(t)) + \sum_{i=0}^{n_{rin}} \frac{1}{-\bar{r}_i(\mathbf{u}^*)}\nabla h_i(\mathbf{u}) + \bar{\mathbf{M}}^T\boldsymbol{\nu} = 0. \tag{49}$$

A partir da condição de otimalidade (49), pode-se derivar uma importante propriedade do caminho central: todo ponto central implica em um ponto dual factível e consequentemente um limite inferior para valor ótimo primal p^* (BOYD; VANDENBERGHE, 2004). Para se verificar essa propriedade pode-se definir:

$$\lambda^*(t) = -\frac{1}{t\bar{r}_i(u^*(t))},\ i = 1,...,n_{rin}. \quad \nu^*(t) = \nu/t, \tag{50}$$

e expressar a condição de otimalidade (49) como:

$$t\nabla h(\mathbf{u}^*(t)) + \sum_{i=0}^{n_{rin}} \lambda^*(t)\nabla\bar{r}_i(\mathbf{u}) + \bar{\mathbf{M}}^T\boldsymbol{\nu}^*(t) = 0. \tag{51}$$

Pode-se verificar que $u^*(t)$ minimiza o Lagrangiano

$$L(u,\lambda,\nu) = h(\mathbf{u}) + \sum_{i=0}^{n_{rin}} \lambda_i \bar{r}_i(\mathbf{u}) + \boldsymbol{\nu}^T(\bar{\mathbf{M}}\mathbf{u} - \bar{\mathbf{m}}), \tag{52}$$

para $\lambda = \lambda^*$ e $\nu = \nu^*$, o que significa que $\lambda^*(t)$ e $\nu^*(t)$ formam um par dual factível (BOYD; VANDENBERGHE, 2004).

Portanto, a função dual $h_d(\lambda^*(t),\nu^*(t))$ é finita e

$$\begin{aligned} h_d(\lambda^*(t),\nu^*(t)) &= h(\mathbf{u}^*(t)) + \sum_{i=0}^{n_{rin}} \lambda_i^*(t)\bar{r}_i(\mathbf{u}^*(t)) + \boldsymbol{\nu}^*(t)^T(\bar{\mathbf{M}}\mathbf{u}^*(t) - \bar{\mathbf{m}}) \\ &= h(\mathbf{u}^*(t)) - n_{rin}/t. \end{aligned}$$

Nesse caso particular, o resíduo dual associado com $\mathbf{u}^*(t)$ e o par factível $\lambda^*(t)$ e $\nu^*(t)$ é n_{rin}/t. Como uma importante consequência, tem-se que:

$$h(\mathbf{u}^*(t)) - p^* \leq n_{rin}/t,$$

ou seja, $\mathbf{u}^*(t)$ é no máximo n_{rin}/t-subótimo. Este resultado confirma a ideia intuitiva de que $\mathbf{u}^*(t)$ converge para um ponto ótimo quando $t \to \infty$.

3.3.3 Algoritmo

A partir dos conceitos apresentados pode-se formular o algoritmo do método barreira. Uma versão simplificada pode ser vista no Algoritmo 3 e se baseia na resolução de uma sequência de problemas de minimização irrestritos, utilizando o último ponto encontrado como o ponto de início da próxima iteração. A cada iteração o valor de t é incrementado, até que $t \geq n_{rin}/\epsilon$, o que garante que seja obtida uma solução ϵ-subótima do problema original.

Algoritmo 3 Método de barreira

Entrada: $\mathbf{u}^0$ estritamente factível, $t = t^0 > 0$, $\mu > 1$, tolerância $\epsilon > 0$
Saída: $\mathbf{u}$
enquanto $n_{rin}/t > \epsilon$ **faça**
  Centralização: computa $\mathbf{u}^*(t)$ minimizando $th + \eta$, sujeito a $\bar{\mathbf{M}}\mathbf{u} = \bar{\mathbf{m}}$ começando em $\mathbf{u}^0$;
  Atualização: $\mathbf{u} \leftarrow \mathbf{u}^*(t)$;
  Incremento de t: $t \leftarrow \mu t$;

Para o passo de centralização, apesar de qualquer método de minimização linear com restrições de igualdade poder ser utilizado, em geral, utiliza-se o método de Newton por conta da rápida convergência (BOYD; VANDENBERGHE, 2004).

3.3.4 Sintonia

Em Boyd e Vandenberghe (2004) são apresentadas várias características práticas para a sintonia do método de barreira logarítmica. A primeira delas é quanto à exatidão do passo de centralização. Os autores defendem que computar $\mathbf{u}^*(t)$ no passo de centralização de forma exata é desnecessário, uma vez que o caminho central tem pouca relevância no direcionamento da solução para o problema original quando $t \to \infty$. A centralização inexata ainda continuará resultando em uma sequência de pontos $\mathbf{u}^{(k)}$ que convergem para um ponto ótimo. Dessa forma, devido à rápida convergência do método de Newton, o passo de centralização pode ser executado com apenas algumas poucas iterações.

Para a escolha da taxa de atualização de t, denotada por μ, é necessário ponderar entre o número de iterações na centralização, chamadas de iterações internas, ou de iterações no método propriamente dito, chamadas de iterações externas. Se μ for escolhido pequeno (próximo de 1), então a cada iteração externa t é incrementado por um fator pequeno. Como resultado, o ponto inicial para o processo de Newton é um ponto mais próximo da solução e o número de passos de Newton é pequeno. Por outro lado, o número de iterações externas é grande, uma vez que o resíduo é também reduzido por valores pequenos. Na prática, para uma grande faixa de valores entre 3 e 100, aproximadamente, os dois efeitos praticamente se cancelam e o número de passos de Newton fica aproximadamente constante. Valores usuais de μ estão na faixa entre 10 e 20 (BOYD; VANDENBERGHE, 2004).

O último parâmetro é o valor inicial de t. A escolha de t_0 grande implica uma primeira iteração externa com um grande número de iterações internas. Já para a escolha de um t_0 pequeno, o algoritmo irá requerer uma quantidade extra de iterações externas. Portanto, já que n_{rin}/t_0 é o resíduo dual que resultará do primeiro passo de centralização, uma escolha razoável de t_0 é tal que n_{rin}/t_0 seja aproximadamente da mesma ordem de grandeza da tolerância de resíduo dual desejada ou μ vezes essa quantidade (BOYD; VANDENBERGHE, 2004).

3.3.5 Convergência

A análise de convergência para o método de barreira é direto. Assumindo que $th + \eta$ pode ser minimizado pelo método de Newton para $t = t_0, \mu t_0, \mu^2 t_0, ...$, o resíduo dual depois do passo inicial e k passos adicionais de centralização é $n_{rin}/(\mu^k t_0)$. Dessa forma, a exatidão desejada ϵ é obtida após exatamente

$$\left\lceil \frac{\log(n_{rin}/(\epsilon t_0))}{\log \mu} \right\rceil \tag{53}$$

passos de centralização mais o passo inicial (BOYD; VANDENBERGHE, 2004).

3.4 MÉTODO DE PROJEÇÃO DE GRADIENTE

O método de projeção de gradiente (GPM, do inglês *Gradient Projection Method*) foi originalmente proposto por Goldstein (1964) e Levitin e Polyak (1966). O GPM é similar ao método de gradiente descendente e pode ser aplicado para problemas de otimização com restrições. O algoritmo computa uma iteração para encontrar a solução do problema convexo sem restrições e, na sequência, faz uma projeção desse passo no conjunto de restrições ativas. Uma versão acelerada foi proposta por Nesterov (1983) com taxa de convergência $O(1/k^2)$ e serve como base para a maioria das aplicações de GPM propostos para a área de controle. Porém, como a projeção pode ser complexa no caso de restrições de saída ou politópicas, algumas variações do método foram propostas. Uma variação específica para a utilização de MPC embarcado foi proposta por Patrinos e Bemporad (2014), que utilizam a forma dual do problema e propõem formas de acelerar o cômputo da otimização baseadas no algoritmo de Nesterov. Essa variação foi denominada Gradiente Projetado Dual Acelerado (GPAD, do inglês *Accelerated Dual Gradient Projection*). Os autores defendem que o GPAD é adequado para MPC embarcado eficiente, pois é simples e fácil de codificar, permite a estimação do número de iterações para, dada uma precisão desejada, computar o valor ótimo e o custo computacional do método cresce linearmente com o horizonte de controle. Em Richter *et al.* (2012) é investigada a utilização do GPM em MPC e dada ênfase na certificação da complexidade computacional requerida pelo mesmo.

A seguir são apresentadas a formulação e alguns conceitos importantes relacionados ao GPM e suas variações úteis para o MPC.

3.4.1 Formulação

O método de projeção de gradiente foi desenvolvido para resolver problemas de otimização da seguinte forma:

$$\begin{aligned} \min_{\mathbf{u}} \quad & h(\mathbf{u}) \\ \text{s.a.} \quad & \mathbf{u} \in \Omega \end{aligned} \tag{54}$$

com $h : \mathbb{R}^n \to \mathbb{R}$ sendo uma função convexa contínua diferenciável (Definição A.2), $\mathbf{u} \in \mathbb{R}^n$ e $\Omega \subset \mathbb{R}^n$ sendo um conjunto convexo fechado (Definição A.1) e não vazio. Nota-se que essa representação de restrições é mais limitada que a representação com desigualdades apresentadas anteriormente. Essa limitação se dá devido ao passo de projeção utilizado no GPM. Uma abordagem pela formulação dual que amplia os tipos de restrições modeladas é apresentada na Seção 3.4.4.

O passo principal do algoritmo do GPM é dado por:

$$\mathbf{u}_{j+1} = P(\mathbf{u}_j - l_j \nabla h(\mathbf{u}_j)),$$

onde P é a função projeção ortogonal (Definição A.14), $l_j > 0$ é o tamanho do passo que pode ser calculado com algum método de busca linear, e $-\nabla h$ é o anti-gradiente da função h, que representa a direção de descenso.

O tamanho do passo $l_j > 0$ deve garantir que o algoritmo iterativo forme uma chamada sequência de relaxação tal que:

$$h(\mathbf{u}_{j+1}) \leq h(\mathbf{u}_j), \quad \forall j \geq 0. \tag{55}$$

Várias regras para a definição do comprimento do passo l_j podem ser encontradas na literatura, como o comprimento de passo fixo, a regra de Armijo e a busca linear retroativa (TRUONG; NGUYEN, 2020). A regra mais simples é a de passo fixo $l_j = l > 0$, $\forall j \geq 1$ e é geralmente utilizada, uma vez que a função h é convexa e foi provado em Nesterov (2014) ter a mesma taxa de convergência das outras regras para este caso específico.

3.4.2 Convergência

Uma das condições para a análise de convergência do GPM aplicado ao MPC é que o gradiente da função seja Lipschitz contínuo em um conjunto Ω (Definição A.4), ou seja, exista uma constante Lipschitz $L \geq 0$ tal que (RICHTER *et al.*, 2009):

$$\|\nabla h(\mathbf{u}) - \nabla h(\bar{\mathbf{u}})\| \leq L\|\mathbf{u} - \bar{\mathbf{u}}\|, \quad \forall\, \mathbf{u}, \bar{\mathbf{u}} \in \Omega. \tag{56}$$

A partir da equação (56) pode-se definir um limite superior para h dado por(RICHTER *et al.*, 2009):

$$h(\mathbf{u}) \leq h(\bar{\mathbf{u}}) + \nabla h(\bar{\mathbf{u}})^T(\mathbf{u} - \bar{\mathbf{u}}) + \frac{L}{2}\|\mathbf{u} - \bar{\mathbf{u}}\|^2, \quad \forall\, \mathbf{u}, \bar{\mathbf{u}} \in \Omega. \tag{57}$$

Pode-se observar que o lado direito da desigualdade (57) é uma função convexa quadrática de $\bar{\mathbf{u}}$ (para um $\mathbf{u}$ fixo). Portanto, pode-se obter a minimização através da igualdade do gradiente da função a zero e obter-se o mínimo com $\bar{\mathbf{u}} = \mathbf{u} - 1/L(\nabla h(\mathbf{u}))$. Dessa forma, pode-se obter um passo factível na direção de descenso com o passo fixo $1/L$, ou seja, o passo do método pode ser determinado como:

$$\mathbf{u}_{j+1} = P(\mathbf{u}_j - \frac{1}{L}\nabla h(\mathbf{u}_j)). \tag{58}$$

Para encontrar a constante Lipschitz L pode-se utilizar o maior autovalor da Hessiana de h, ou seja, $L = \lambda_{max}(\nabla^2 h(u))$ (RICHTER *et al.*, 2009). Outra abordagem equivalente, proposta em Patrinos e Bemporad (2014), é a utilização da norma espectral ou da norma Frobenius da Hessiana para a definição de L:

$$L = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}|h_{ij}|^2}, \tag{59}$$

onde n é a dimensão da matriz Hessiana e h_{ij} é o elemento da matriz correspondente à linha i e coluna j.

A partir das definições apresentadas, um algoritmo do GPM pode ser visto no Algoritmo 4.

Algoritmo 4 Método de gradiente projetado

Entrada: h, $\mathbf{u}_0$ estritamente factível, L, tolerância $\epsilon > 0$

Saída: $\mathbf{u}$

$j \leftarrow 0$;

enquanto $|\mathbf{u}_j - \mathbf{u}_{j-1}| > \epsilon$ **faça**

Atualização: $\mathbf{u}_j \leftarrow P(\mathbf{u}_j - \frac{1}{L}\nabla h(\mathbf{u}_j))$;

Incremento: $j \leftarrow j + 1$;

3.4.3 Método de gradiente projetado acelerado

A versão acelerada do GPM (AGPM, do inglês *Accelerated Gradient Projection Method*) foi proposta por Nesterov (1983) e vem sendo amplamente utilizada no contexto de MPC de cômputo rápido. O ponto principal para a aceleração do método é a não observância da propriedade da sequência de relaxação definida em (55) e a sua substituição pela chamada sequência estimada. Dessa forma, é inserida uma operação *a priori* no cálculo de $\mathbf{u}$:

$$\mathbf{w}_j = \mathbf{u}_j + \theta_j(\theta_{j-1}^{-1} - 1)(\mathbf{u}_j - \mathbf{u}_{j-1}), \tag{60}$$

$$\mathbf{u}_{j+1} = P(\mathbf{w}_j - \frac{1}{L}\nabla h(\mathbf{w}_j)), \tag{61}$$

com $\theta_{j+1} = (\sqrt{(\theta_j)^4 + 4(\theta_j)^2} - (\theta_j)^2)/2$.

Como o cômputo de θ apresenta operações algébricas não desejadas para sistemas embarcados e como foi demonstrada a convergência do AGPM para quaisquer escolhas de θ tal que:

$$\frac{1 - \theta_{j+1}}{(\theta_{j+1})^2} \leq \frac{1}{(\theta_j)^2}, \tag{62}$$

uma escolha possível que atende (62) é $\theta_j = 2/(j+2)$ (TSENG, 2008). Substituindo $\theta_j = 2/(j+2)$ em (60) obtém-se uma equação mais simples:

$$\theta_j(\theta_{j-1}^{-1} - 1) = \frac{j-1}{j+2},$$

e se definirmos uma variável $\beta = \frac{j-1}{j+2}$, a equação (60) torna-se:

$$\mathbf{w}_j = \mathbf{u}_j + \beta(\mathbf{u}_j - \mathbf{u}_{j-1}). \tag{63}$$

Portanto, o AGPM pode ser representado pelo Algoritmo 5.

Algoritmo 5 Método de gradiente projetado acelerado

Entrada: h, $\mathbf{u}_0$ estritamente factível, L, tolerância $\epsilon > 0$
Saída: $\mathbf{u}$
$j \leftarrow 0$;
enquanto $|\mathbf{u}_j - \mathbf{u}_{j-1}| > \epsilon$ **faça**
  se $j = 0$ **então**
    $\beta \leftarrow 0$
  senão
    $\beta \leftarrow \frac{j-1}{j+2}$;
  $\mathbf{w}_j \leftarrow \mathbf{u}_j + \beta(\mathbf{u}_j - \mathbf{u}_{j-1})$;
  $\mathbf{u}_j \leftarrow P(\mathbf{w}_j - \frac{1}{L}\nabla h(\mathbf{w}_j))$;
  $j \leftarrow j + 1$;

3.4.4 Formulação Dual

Como o cômputo da projeção do vetor no conjunto Ω pode ser complicado para alguns tipos de restrições do problema de otimização, uma solução é a abordagem através do problema dual. O ótimo primal pode ser encontrado através da solução do problema dual, que é dado por:

$$\begin{aligned} \min_{\boldsymbol{\psi}} \quad & -h_d(\boldsymbol{\psi}) \\ \text{s.a.} \quad & \boldsymbol{\psi} \geq \mathbf{0}, \end{aligned} \tag{64}$$

com $h_d : \mathbb{R}^n \rightarrow \mathbb{R}$ sendo a função dual associada a h (Definição A.10) e $\boldsymbol{\psi} \in \mathbb{R}^n$ o vetor de multiplicadores de Lagrange associados às restrições de desigualdade. Dessa forma, uma vez obtidos os multiplicadores de Lagrange ótimos ψ^*, assumindo que a dualidade forte se aplica (Definição A.12), o ótimo primal pode ser obtido por:

$$\mathbf{u}^* = \text{argmin}(\mathcal{L}(\mathbf{u}, \boldsymbol{\psi}^*)). \tag{65}$$

Dessa forma, pode-se perceber que as restrições associadas a (64) são apenas $\boldsymbol{\psi} > \mathbf{0}$ e portanto a projeção se apresenta como uma saturação de ψ para valores positivos, o que pode ser representado matematicamente como:

$$P(\boldsymbol{\psi}) = \max(\boldsymbol{\psi}, \mathbf{0}). \tag{66}$$

O algoritmo do método do gradiente projetado dual pode então ser representado pelo Algoritmo 6.

3.5 MÉTODOS DE OPERADOR DE PARTIÇÃO

A classe de métodos de operador de partição (OSM, do inglês *Operator Split Method*) baseia-se na ideia de que, a partir de uma função objetivo separável, o problema de otimização pode ser separado em vários problemas mais simples. Dessa

Algoritmo 6 Método de gradiente projetado acelerado dual

Entrada: h_d, $\psi_0 = \psi_{-1} = 0$, L, tolerância $\epsilon > 0$
Saída: $\mathbf{u}$
$j \leftarrow 0$;
enquanto $|\psi_j - \psi_{j-1}| > \epsilon$ **faça**
  se $j = 0$ **então**
    $\beta \leftarrow 0$
  senão
    $\beta \leftarrow \frac{j-1}{j+2}$;
  $\mathbf{w}_j \leftarrow \psi_j + \beta(\psi_j - \psi_{j-1})$;
  $\psi_j \leftarrow \max(\mathbf{w}_j - \frac{1}{L}\nabla f(\mathbf{w}_j), 0)$;
  $j \leftarrow j + 1$;
$\mathbf{u} \leftarrow \text{argmin}(\mathcal{L}(\mathbf{u}, \psi_{j+1}))$;

forma, esses problemas simples podem ser resolvidos paralelamente. Para resolver cada um dos problemas reduzidos, existem técnicas que podem utilizar a formulação primal, dual ou até primal-dual. Os métodos que se destacam são o Método dos Multiplicadores com Direções Alternadas (ADMM, do inglês *Alternating Direction Method of Multipliers*), o Algoritmo de Minimização Alternada (AMA, do inglês *Alternating Minimization Algorithm*) e o Algoritmo Primal-Dual (PDA, do inglês *Primal-Dual Algorithm*) (STATHOPOULOS *et al.*, 2016).

O ADMM é um algoritmo simples, mas poderoso, que se adapta muito bem a problemas de otimização convexa distribuída. Ele se apresenta na forma de um algoritmo de decomposição/coordenação, no qual as soluções de pequenos subproblemas locais são coordenadas para encontrar uma solução para um problema global de grande escala (O'DONOGHUE *et al.*, 2013). ADMM pode ser visto como uma tentativa de colher ambos os benefícios da decomposição dual e dos métodos de Lagrangiano aumentado para otimização com restrições (BOYD *et al.*, 2011).

Devido à possibilidade de paralelização dos métodos de partição, o ADMM se apresenta como uma boa abordagem para as soluções modernas de hardware, como os processadores de múltiplos núcleos e os FPGAs. Em O'Donoghue *et al.* (2013) é apresentada uma forma de implementação de MPC com ADMM denominada *operator splitting for control* (OSC). Essa formulação, em muitos casos, pode não necessitar de operações de divisão, o que facilita a implementação em arquiteturas de aritmética de ponto fixo, como FPGAs. Os pontos fracos do ADMM são a dependência do tipo do problema no desempenho do método e a baixa precisão nos resultados. Em Raghunathan e Cairano (2014) é proposto um mecanismo de detecção de infactibilidade e o seu desempenho é exemplificado em uma aplicação de MPC.

A seguir são apresentadas a formulação e alguns conceitos importantes relacionados ao ADMM e suas variações úteis para o MPC.

3.5.1 Precursores

Pode-se afirmar que os precursores do ADMM são os métodos conhecidos como o método ascendente dual e o método dos multiplicadores (BOYD *et al.*, 2011). O método ascendente dual assume um problema de otimização convexa com restrições de igualdade da seguinte forma:

$$\begin{aligned} \min_{\mathbf{u}} \quad & h(\mathbf{u}) \\ \text{s.a.} \quad & \bar{\mathbf{M}}\mathbf{u} = \bar{\mathbf{m}} \end{aligned} \tag{67}$$

com $h : \mathbb{R}^n \to \mathbb{R}$ convexa, $\mathbf{u} \in \mathbb{R}^n$, $\bar{\mathbf{M}} \in \mathbb{R}^{n_{req} \times n}$, $\bar{\mathbf{m}} \in \mathbb{R}^{n_{req}}$.

A partir de (67), o problema dual pode ser obtido através do Lagrangiano e da função dual h_d (Definição A.10) associados. O Lagrangiano é dado por:

$$\mathcal{L}(\mathbf{u}, \boldsymbol{\nu}) = h(\mathbf{u}) + \boldsymbol{\nu}^T(\bar{\mathbf{M}}\mathbf{u} - \bar{\mathbf{m}}),$$

com $\boldsymbol{\nu} \in \mathbb{R}^{n_{req}}$ sendo o vetor de multiplicadores de Lagrange associados às restrições de igualdade.

O problema dual recai em:

$$\begin{aligned} \max_{\boldsymbol{\nu}} \quad & h_d(\boldsymbol{\psi}) \\ \text{s.a.} \quad & \boldsymbol{\nu} \geq \mathbf{0}. \end{aligned} \tag{68}$$

Dessa forma, uma vez obtidos os multiplicadores de Lagrange ótimos $\boldsymbol{\nu}^*$, assumindo que a dualidade forte (Definição A.12) se aplica, o ótimo primal pode ser obtido a partir do ótimo dual:

$$\mathbf{u}^* = \text{argmin}(\mathcal{L}(\mathbf{u}, \boldsymbol{\nu}^*)). \tag{69}$$

O método ascendente dual consiste em iterativamente computar a variável primal e atualizar a variável dual na direção do gradiente em relação a $\boldsymbol{\nu}$ de h_d com um passo de comprimento l, ou seja:

$$\begin{aligned} \mathbf{u}_{j+1} &= \text{argmin}(\mathcal{L}(\mathbf{u}, \boldsymbol{\nu}_j)) \\ \boldsymbol{\nu}_{j+1} &= \boldsymbol{\nu}_j + l\nabla_{\boldsymbol{\nu}} f_d := \boldsymbol{\nu}_j + l(\bar{\mathbf{M}}\mathbf{u}_{j+1} - \bar{\mathbf{m}}). \end{aligned} \tag{70}$$

Como o método ascendente dual necessita de algumas propriedades para a convergência, como a finitude e a convexidade estrita (Definição A.3) da função h, o método de multiplicadores foi desenvolvido para contornar esses requisitos.

O método dos multiplicadores utiliza o Lagrangiano aumentado que para (67) é dado por:

$$\mathcal{L}_\rho(\mathbf{u}, \boldsymbol{\nu}) = h(\mathbf{u}) + \boldsymbol{\nu}^T(\bar{\mathbf{M}}\mathbf{u} - \bar{\mathbf{m}}) + \frac{\rho}{2}\|\bar{\mathbf{M}}\mathbf{u} - \bar{\mathbf{m}}\|^2,$$

onde $\rho > 0$ é chamado de parâmetro de penalização. A vantagem de incluir esse termo com penalização no Lagrangiano é que a função dual obtida é diferenciável sob

condições menos restritas do que na formulação original. O gradiente do Lagrangiano aumentado pode ser obtido da mesma forma que o Lagrangiano normal e, dessa forma, o método dos multiplicadores se apresenta da seguinte forma:

$$\begin{aligned} \mathbf{u}_{j+1} &= \text{argmin}(\mathcal{L}_\rho(\mathbf{u},\nu_j)) \\ \nu_{j+1} &= \nu_j + \rho\nabla_\nu h_d = \nu_j + \rho(\bar{\mathbf{M}}\mathbf{u}_{j+1} - \bar{\mathbf{m}}). \end{aligned} \tag{71}$$

Pode-se perceber que o método dos multiplicadores segue o mesmo padrão do ascendente dual, exceto que o passo de minimização de *x* usa o Lagrangiano aumentado e que o parâmetro de penalização ρ é usado como o comprimento do passo *l*. O método dos multiplicadores converge sob condições mais gerais que o ascendente dual, incluindo casos nos quais *h* tende a infinito ou não é estritamente convexa. A escolha de ρ como comprimento do passo na direção do gradiente pode ser justificada como uma boa escolha para manter pontos factíveis, tanto primal quanto dual. Essa característica pode ser vista se avaliadas as condições de otimalidade Karush-Kuhn-Tucker (KKT) (Teorema A.2) de (67). Supondo *h* convexa, as condições de otimalidade primal e dual são (STATHOPOULOS *et al.*, 2016):

$$\bar{\mathbf{M}}\mathbf{u}^* - \bar{\mathbf{m}} = \mathbf{0}, \quad \nabla h(\mathbf{u}^*) + \bar{\mathbf{M}}^T\nu^* = \mathbf{0},$$

respectivamente. Como, por definição, u_{j+1} minimiza o Lagrangiano aumentado $\mathcal{L}_\rho(\mathbf{u},\nu_j)$, então

$$\begin{aligned} \mathbf{0} &= \nabla_u \mathcal{L}_\rho(\mathbf{u}_{j+1},\nu_j) \\ &= \nabla_u h(\mathbf{u}_{j+1}) + \bar{\mathbf{M}}^T(\nu_j + \rho(\bar{\mathbf{M}}\mathbf{u}_{j+1} - \bar{\mathbf{m}})) \\ &= \nabla_u h(\mathbf{u}_{j+1}) + \bar{\mathbf{M}}^T\nu_{j+1}. \end{aligned} \tag{72}$$

Pode-se perceber a partir de (72) que a escolha de ρ como o comprimento de passo, na atualização da variável dual do método, fornece um par $\{\mathbf{u}_{j+1},\nu_{j+1}\}$ factível dual.

A melhora nas propriedades de convergência do método dos multiplicadores em relação ao ascendente dual implica um ônus na possibilidade de decomposição do mesmo. Quando *h* é separável, o Lagrangiano aumentado não é separável e portanto o passo de minimização de $\mathbf{u}$ não pode ser computado de forma paralela, como ocorre no método ascendente dual (STATHOPOULOS *et al.*, 2016).

3.5.2 Formulação ADMM

O ADMM foi desenvolvido de modo a reunir as propriedades de decomposição do método ascendente dual com as boas propriedades de convergência do método dos multiplicadores (BOYD *et al.*, 2011). Para isso, a variável $\mathbf{u}$ é particionada em duas novas variáveis, o que torna a função objetivo separável. Para entender o ADMM,

pode-se assumir que método resolve problemas da seguinte forma:

$$\begin{aligned} \min_{\mathbf{u},\mathbf{z}} \quad & h(\mathbf{u}) + \zeta(\mathbf{z}) \\ \text{s.a.} \quad & \bar{\mathbf{M}}\mathbf{u} + \bar{\mathbf{D}}\mathbf{z} = \bar{\mathbf{m}} \end{aligned} \tag{73}$$

com $\mathbf{u} \in \mathbb{R}^n$, $\mathbf{z} \in \mathbb{R}^m$, $\bar{\mathbf{M}} \in \mathbb{R}^{n_{req} \times n}$, $\bar{\mathbf{D}} \in \mathbb{R}^{n_{req} \times m}$, $\bar{\mathbf{m}} \in \mathbb{R}^{n_{req}}$ e h e ζ sendo funções convexas.

Para incluir a restrição na função objetivo, a partir de (73), pode ser formado o Lagrangiano aumentado como no método dos multiplicadores:

$$\mathcal{L}_\rho(\mathbf{u},\mathbf{z},\boldsymbol{\nu}) = h(\mathbf{u}) + \zeta(\mathbf{z}) + \boldsymbol{\nu}^T(\bar{\mathbf{M}}\mathbf{u} + \bar{\mathbf{D}}\mathbf{z} - \bar{\mathbf{m}}) + \frac{\rho}{2}\|\bar{\mathbf{M}}\mathbf{u} + \bar{\mathbf{D}}\mathbf{z} - \bar{\mathbf{m}}\|^2.$$

Assumindo que a dualidade forte se aplica, os valores ótimos para o problema primal e dual são os mesmos. Dessa forma, o ótimo do problema primal $\mathbf{u}^*$ pode ser obtido através do ponto ótimo dual $\boldsymbol{\nu}^*$:

$$\mathbf{u}^* = \text{argmin } \mathcal{L}_\rho(\mathbf{u},\boldsymbol{\nu}^*)$$

Portanto o ADMM pode ser computado de forma recursiva, alternando entre as minimizações de $\mathbf{u}$ e $\mathbf{z}$ e atualizando a variável ν da seguinte forma:

$$\mathbf{u}_{j+1} = \text{argmin } \mathcal{L}_\rho(\mathbf{u}_j,\mathbf{z}_j,\boldsymbol{\nu}_j) \tag{74}$$

$$\mathbf{z}_{j+1} = \text{argmin } \mathcal{L}_\rho(\mathbf{u}_{j+1},\mathbf{z}_j,\boldsymbol{\nu}_j) \tag{75}$$

$$\boldsymbol{\nu}_{j+1} = \boldsymbol{\nu}_j + \rho(\bar{\mathbf{M}}\mathbf{u}_{j+1} + \bar{\mathbf{D}}\mathbf{z}_{j+1} - \bar{\mathbf{m}}). \tag{76}$$

3.5.3 Sintonização

O comprimento do passo no ADMM coincide com o parâmetro de penalização ρ do Lagrangiano aumentado e pode ser utilizado como um grau de liberdade para uma convergência mais rápida do algoritmo. Na forma padrão do ADMM, o valor de ρ é mantido constante, entretanto, existem variações do ADMM que propõem diferentes valores de ρ_j a cada iteração (BOYD *et al.*, 2011). As equações de atualização do ADMM sugerem que grandes valores de ρ causam uma grande penalização nas violações da factibilidade primal e tendem a produzir pequenos resíduos primais. Por outro lado, a definição do resíduo dual sugere que pequenos valores de ρ tendem a diminuir o resíduo dual, ao custo de reduzirem a penalização da factibilidade primal, o que pode resultar em um maior resíduo primal. Dessa forma, o valor de ρ deve ser escolhido de forma a reduzir o número de iterações necessárias para o ponto ϵ-ótimo.

3.5.4 Condições de otimalidade

Assumindo h e ζ diferenciáveis, as condições de otimalidade necessárias e suficientes para o ADMM são a factibilidade primal,

$$\bar{\mathbf{M}}\mathbf{u}^* + \bar{\mathbf{D}}\mathbf{z}^* - \bar{\mathbf{m}} = \mathbf{0}, \tag{77}$$

e a factibilidade dual,

$$\begin{aligned} \mathbf{0} &= \nabla h(\mathbf{u}^*) + \bar{\mathbf{M}}^T \psi^*, \\ \mathbf{0} &= \nabla \zeta(\mathbf{z}^*) + \bar{\mathbf{D}}^T \psi^*. \end{aligned} \tag{78}$$

Uma vez que, por definição, $\mathbf{u}_{j+1}$ minimiza o Lagrangiano $\mathcal{L}_\rho(\mathbf{u}_j, \mathbf{z}_j, \boldsymbol{\psi}_j)$ tem-se, a partir de (78), que:

$$\begin{aligned} \mathbf{0} &= \nabla h(\mathbf{u}_{j+1}) + \bar{\mathbf{M}}^T(\boldsymbol{\psi}_j + \rho(\bar{\mathbf{M}}\mathbf{u}_{j+1} + \bar{\mathbf{D}}\mathbf{z}_j - \bar{\mathbf{m}})) \\ &= \nabla h(\mathbf{u}_{j+1}) + \bar{\mathbf{M}}^T(\boldsymbol{\psi}_j + \rho(\bar{\mathbf{M}}\mathbf{u}_{j+1} + \bar{\mathbf{D}}\mathbf{z}_{j+1} - \bar{\mathbf{m}} - \bar{\mathbf{D}}\mathbf{z}_{j+1} + \bar{\mathbf{D}}\mathbf{z}_j)) \\ &= \nabla h(\mathbf{u}_{j+1}) + \bar{\mathbf{M}}^T\boldsymbol{\psi}_{j+1} + \rho\bar{\mathbf{M}}^T\bar{\mathbf{D}}(\mathbf{z}_j - \mathbf{z}_{j+1}) \\ \rho\bar{\mathbf{M}}^T\bar{\mathbf{D}}(\mathbf{z}_{j+1} - \mathbf{z}_j) &= \nabla h(\mathbf{u}_{j+1}) + \bar{\mathbf{M}}^T\boldsymbol{\psi}_{j+1}, \end{aligned}$$

e, portanto, a parcela

$$\mathbf{s}_{j+1} = \rho\bar{\mathbf{M}}^T\bar{\mathbf{D}}(\mathbf{z}_{j+1} - \mathbf{z}_j)$$

pode ser considerada o resíduo dual na iteração $j+1$.

A partir de (77), tem-se que o resíduo primal é dado por:

$$\mathbf{r}_{j+1} = \bar{\mathbf{M}}\mathbf{u}_{j+1} + \bar{\mathbf{D}}\mathbf{z}_{j+1} - \bar{\mathbf{m}}.$$

Dessa forma, como um critério de parada frequentemente utilizado em ADMM, é utilizada a comparação dos resíduos com as tolerâncias primal ϵ_{pri} e dual ϵ_{dual}, de acordo com algum critério de otimalidade desejado, ou seja:

$$\|\mathbf{r}_j\| \leq \epsilon_{pri} \quad \text{e} \quad \|\mathbf{s}_j\| \leq \epsilon_{dual}. \tag{79}$$

3.5.5 Convergência

Uma prova da convergência do Algoritmo ADMM pode ser encontrada em Boyd *et al.* (2011). A partir da prova de convergência do algoritmo, pode-se concluir que as iterações do ADMM, assumindo h e ζ funções próprias, convexas e fechadas e que o Lagrangiano associado tem um ponto de sela, satisfazem o seguinte:

- convergência residual: $\mathbf{r}_j \to 0$ se $j \to \infty$;
- convergência do objetivo: $h(\mathbf{u}_j) + \zeta(\mathbf{z}_j) \to \mathbf{p}^*$ se $j \to \infty$, onde $\mathbf{p}^*$ é o ponto ótimo primal;
- convergência da variável dual: $\boldsymbol{\psi}_j \to \boldsymbol{\psi}^*$ se $j \to \infty$, onde $\boldsymbol{\psi}^*$ é o ponto ótimo dual.

3.5.6 Forma escalonada

O ADMM pode ser representado de uma forma mais compacta e que pode ser mais conveniente para a implementação. Essa forma é chamada de escalonada e pode ser obtida por meio da definição de uma nova variável $\phi_j = (\frac{1}{\rho})\nu_j$, chamada de

variável escalonada dual (BOYD *et al.*, 2011). Dessa forma, o ADMM escalonado pode ser computado recursivamente da seguinte forma:

$$\begin{aligned}
\mathbf{u}_{j+1} &= \text{argmin}\left(h(\mathbf{u}_j) + \frac{\rho}{2}\|\bar{\mathbf{M}}\mathbf{u}_j + \bar{\mathbf{D}}\mathbf{z}_j - \bar{\mathbf{m}} + \phi_j\|_2^2\right), \\
\mathbf{z}_{j+1} &= \text{argmin}\left(\zeta(\mathbf{z}_j) + \frac{\rho}{2}\|\bar{\mathbf{M}}\mathbf{u}_{j+1} + \bar{\mathbf{D}}\mathbf{z}_j - \bar{\mathbf{m}} + \phi_j\|_2^2\right), \\
\phi_{j+1} &= \phi_j + (\bar{\mathbf{M}}\mathbf{u}_{j+1} + \bar{\mathbf{D}}\mathbf{z}_{j+1} - \bar{\mathbf{m}}).
\end{aligned} \tag{80}$$

A partir dos conceitos apresentados, o algoritmo genérico do ADMM para o problema (73) na forma escalonada pode ser visto no Algoritmo 7.

Algoritmo 7 Método dos multiplicadores em direções alternadas (ADMM) na forma escalonada

Entrada: h, ζ, $\bar{\mathbf{M}}$, $\bar{\mathbf{D}}$, $\bar{\mathbf{m}}$, $\mathbf{u}^0$ factível, $\mathbf{z}^0$ factível

Dados: $\rho > 0$, $\boldsymbol{\phi}^0 = \mathbf{0}$, $\epsilon_{pri} > 0$, $\epsilon_{dual} > 0$, j_{max}

Saída: $\mathbf{u}$

$j \leftarrow 0$;

enquanto $j < j_{max}$ **faça**

- $\mathbf{u}_{j+1} \leftarrow \text{argmin}\left(h(\mathbf{u}_j) + \frac{\rho}{2}\|\bar{\mathbf{M}}\mathbf{u}_j + \bar{\mathbf{D}}\mathbf{z}_j - \bar{\mathbf{m}} + \phi_j\|^2\right)$;
- $\mathbf{z}_{j+1} \leftarrow \text{argmin}\left(\zeta(\mathbf{z}_j) + \frac{\rho}{2}\|\bar{\mathbf{M}}\mathbf{u}_{j+1} + \bar{\mathbf{D}}\mathbf{z}_j - \bar{\mathbf{m}} + \phi_j\|^2\right)$;
- $\boldsymbol{\phi}_{j+1} \leftarrow \boldsymbol{\phi}_j + (\bar{\mathbf{M}}\mathbf{u}_{j+1} + \bar{\mathbf{D}}\mathbf{z}_{j+1} - \bar{\mathbf{m}})$;
- **se** $\|\bar{\mathbf{M}}\mathbf{u}_{j+1} + \bar{\mathbf{D}}\mathbf{z}_{j+1} - \bar{\mathbf{m}}\| \leq \epsilon_{pri}$ ***e*** $\|\rho\bar{\mathbf{M}}^T\bar{\mathbf{D}}(\mathbf{z}_{j+1} - \mathbf{z}_j)\| \leq \epsilon_{dual}$ **então**
  - **retorna**;
- $j \leftarrow j + 1$;

3.6 MÉTODO DE PROGRAMAÇÃO QUADRÁTICA SEM PROJEÇÃO

O método conhecido como programação quadrática sem projeção (PFQP, do inglês *Projection Free Quadratic Programming*) foi apresentado em Brand e Chen (2011) e recebeu esse nome devido ao seu cômputo poder ser eficientemente paralelizado e não precisar do passo de projeção. Originalmente, foi utilizado para resolver problemas de mínimos quadrados não negativos (NNLS) advindos da área de processamento de imagem. A ideia básica do PFQP é obter a solução de forma iterativa, na direção do anti-gradiente e com um comprimento de passo que garanta a factibilidade dentro das iterações. Dessa forma, não é necessário o passo de projeção da solução para o domínio das restrições, como no caso do GPM. Em Cairano *et al.* (2013) foi apresentada uma aplicação do PFQP para a formulação paramétrica do MPC em espaço de estados. Como o PFQP não necessita do passo de projeção para o domínio das restrições, o método se torna bastante competitivo em casos nos quais as restrições são complexas.

3.6.1 Formulação PFQP

O problema de NNLS pode ser apresentado da seguinte forma:

$$\mathbf{u}^* = \text{argmin}_{\mathbf{u}\in\mathbb{R}^n_{0+}} \left(\frac{1}{2} \|\mathbf{H}\mathbf{u} - \mathbf{b}\|^2 \right), \tag{81}$$

que pode ser reformulado como o seguinte problema quadrático de minimização:

$$\begin{aligned} \min_{\mathbf{u}} \quad & \frac{1}{2}\mathbf{u}^T\hat{\mathbf{H}}\mathbf{u} + \hat{\mathbf{b}}^T\mathbf{u} \\ \text{s.a.} \quad & \mathbf{u} \geq \mathbf{0}, \end{aligned} \tag{82}$$

com $\hat{\mathbf{H}} = \mathbf{H}^T\mathbf{H}$ e $\hat{\mathbf{b}} = -\mathbf{H}^T\mathbf{b}$.

A ideia principal do PFQP é, a cada passo do método, aproximar-se do valor ótimo a partir da direção do anti-gradiente e com o tamanho do passo escolhido para garantir que se mantenha na região factível.

De modo a computar a solução ótima de (82), o Lagrangiano associado é dado por:

$$\mathcal{L}(\mathbf{u},\boldsymbol{\psi}) = \frac{1}{2}\mathbf{u}^T\hat{\mathbf{H}}\mathbf{u} + \hat{\mathbf{b}}^T\mathbf{u} - \boldsymbol{\psi}^T\mathbf{u}. \tag{83}$$

A partir de (83), a condição de otimalidade de (82), obtida por cálculo variacional:

$$\mathbf{u} \circ \nabla_u \mathcal{L}(\mathbf{u},\boldsymbol{\psi}) = \mathbf{0}, \tag{84}$$

onde $\circ$ representa o produto de Hadamard entre dois vetores.

Aplicando a condição de otimalidade (84) em (83) obtém-se:

$$\mathbf{u} \circ (\hat{\mathbf{H}}\mathbf{u} + \hat{\mathbf{b}} - \boldsymbol{\psi}) = \mathbf{0}. \tag{85}$$

Se for definido $\hat{\mathbf{H}} = \hat{\mathbf{H}}^+ - \hat{\mathbf{H}}^-$ com $\hat{\mathbf{H}}^+ \triangleq \max(\mathbf{0},\hat{\mathbf{H}})$, $\hat{\mathbf{H}}^- \triangleq \max(\mathbf{0}, -\hat{\mathbf{H}})$ e for aplicada a mesma lógica para $\hat{\mathbf{b}}$, obtém-se:

$$\mathbf{u} \circ ((\hat{\mathbf{H}}^+\mathbf{u} + \hat{\mathbf{b}}^+) - (\hat{\mathbf{H}}^-\mathbf{u} + \hat{\mathbf{b}}^- + \boldsymbol{\psi})) = \mathbf{0}, \tag{86}$$

que pode ser apresentado de forma reduzida como:

$$\left[\mathbf{u}_{j+1}\right]_i = \frac{\left[\hat{\mathbf{H}}^-\mathbf{u}_j + \hat{\mathbf{b}}^-\right]_i}{\left[\hat{\mathbf{H}}^+\mathbf{u}_j + \hat{\mathbf{b}}^+\right]_i} \left[\mathbf{u}_j\right]_i.$$

Para garantir a convergência, foi inserido um termo $\phi_{PFQP} \in \mathbb{R}^{n\times n}$

$$\left[\mathbf{u}_{j+1}\right]_i = \frac{\left[(\hat{\mathbf{H}}^- + \phi_{PFQP})\mathbf{u}_j + \hat{\mathbf{b}}^-\right]_i}{\left[(\hat{\mathbf{H}}^+ + \phi_{PFQP})\mathbf{u}_j + \hat{\mathbf{b}}^+\right]_i} \left[\mathbf{u}_j\right]_i. \tag{87}$$

Em Cairano *et al.* (2013) foi provada a garantia de convergência com a escolha de ϕ_{PFQP} como uma matriz diagonal não negativa tal que os elementos da diagonal são dados por $\left[\phi_{PFQP}\right]_{ii} \geq \left[\hat{\mathbf{H}}^- \cdot \mathbf{1}\right]_i$, $\forall i = 1,..,n$.

3.6.2 Aceleração por busca linear

Como visto nos métodos anteriores, geralmente os algoritmos computam as iterações em duas etapas, sendo uma de seleção da direção de descenso e a outra de seleção do comprimento do passo (também chamada de busca linear). O método PFQP original representado na equação (87) realiza em apenas uma etapa. Porém, o PFQP pode ser interpretado como um método de gradiente descendente dinamicamente escalonado, ou seja, com o comprimento do passo recalculado a cada iteração de modo a garantir a factibilidade. Para explicitar essa característica, pode-se somar e subtrair o numerador em (87), com $\phi_{PFQP} = 0$ por simplicidade, e obter:

$$\begin{aligned}
\left[\mathbf{u}_{j+1}\right]_i &= \left(\frac{\left[\hat{\mathbf{H}}^{-}\mathbf{u}_j + \hat{\mathbf{b}}^{-}\right]_i}{\left[\hat{\mathbf{H}}^{+}\mathbf{u}_j + \hat{\mathbf{b}}^{+}\right]_i} + \frac{\left[\hat{\mathbf{H}}^{+}\mathbf{u}_j + \hat{\mathbf{b}}^{+}\right]_i}{\left[\hat{\mathbf{H}}^{+}\mathbf{u}_j + \hat{\mathbf{b}}^{+}\right]_i} - \frac{\left[\hat{\mathbf{H}}^{+}\mathbf{u}_j + \hat{\mathbf{b}}^{+}\right]_i}{\left[\hat{\mathbf{H}}^{+}\mathbf{u}_j + \hat{\mathbf{b}}^{+}\right]_i} \right) \left[\mathbf{u}_j\right]_i, \\
\left[\mathbf{u}_{j+1}\right]_i &= \left(-\frac{\left[\hat{\mathbf{H}}\mathbf{u}_j + \hat{\mathbf{b}}\right]_i}{\left[\hat{\mathbf{H}}^{+}\mathbf{u}_j + \hat{\mathbf{b}}^{+}\right]_i} + 1 \right) \left[\mathbf{u}_j\right]_i, \\
\left[\mathbf{u}_{j+1}\right]_i &= \left[\mathbf{u}_j\right]_i - \frac{\left[\mathbf{u}_j\right]_i}{\left[\hat{\mathbf{H}}^{+}\mathbf{u}_j + \hat{\mathbf{b}}^{+}\right]_i} \left[\hat{\mathbf{H}}\mathbf{u}_j + \hat{\mathbf{b}}\right]_i,
\end{aligned} \tag{88}$$

e, portanto, pode-se representar cada iteração do PFQP como um passo na direção do anti-gradiente com comprimento de passo $\mathbf{L}$, que garante a factibilidade, tal que:

$$\mathbf{u}_{j+1} = \mathbf{u}_j - \mathbf{L}\nabla_u \mathbf{J}. \tag{89}$$

A matriz $\mathbf{L}$, que representa o comprimento do passo, é uma matriz diagonal definida por:

$$[\mathbf{L}]_{ii} = \frac{\left[\mathbf{u}_j\right]_i}{\left[\hat{\mathbf{H}}^{+}\mathbf{u}_j + \hat{\mathbf{b}}^{+}\right]_i} \left[\hat{\mathbf{H}}\mathbf{u}_j + \hat{\mathbf{b}}\right]_i. \tag{90}$$

Nessa representação, fica explícito um ponto fraco do PFQP: a convergência para o ponto ótimo pode se tornar muito lenta se o numerador de $\mathbf{L}$, $\left[\mathbf{u}_j\right]_i$ tiver um valor muito próximo de zero e o valor ótimo da variável $[\mathbf{u}^*]_i$ não. Uma possível solução para essa limitação é mover essas variáveis para pontos distantes de zero.

Em Cairano *et al.* (2013) foi apresentada uma versão acelerada do PFQP com busca linear que propõe uma etapa de cômputo com um comprimento de passo diferente para solucionar essa limitação. O comprimento de passo do PFQP acelerado é dado por:

$$l = \begin{cases} -\dfrac{(\hat{\mathbf{H}}\mathbf{u} + \hat{\mathbf{b}})^T\mathbf{p}}{\mathbf{p}^T\hat{\mathbf{H}}\mathbf{p}} & \text{se } \mathbf{p}^T\hat{\mathbf{H}}\mathbf{p} > 0 \\ 0 & \text{senão} \end{cases}, \tag{91}$$

onde $\mathbf{p}$ representa a direção de descenso (valores negativos do gradiente) e, para a função custo *J* de (82), é dado por:

$$\mathbf{p} = \max(\mathbf{0}, -(\hat{\mathbf{H}}\mathbf{u} + \hat{\mathbf{b}})). \tag{92}$$

Dessa forma, a etapa de cômputo da iteração do PFQP acelerado é dada por:

$$\mathbf{u}_{j+1} = \mathbf{u}_j - l\mathbf{p}. \tag{93}$$

Em Cairano *et al.* (2013), propõe-se combinar as duas etapas e são apresentadas provas de convergência e aceleração do processo de otimização. O passo de aceleração deve ser executado a cada η_{PFQP} iterações do passo principal. A partir de experimentos, os autores indicam que obtêm-se melhores resultados para um η_{PFQP} inteiro entre 10 e 50.

O algoritmo do PFQP com o passo de aceleração pode ser visto no Algoritmo 8.

3.7 COMENTÁRIOS FINAIS

Neste capítulo foram apresentados os fundamentos teóricos e uma revisão bibliográfica dos principais métodos de programação quadrática utilizados no contexto de MPC de cômputo rápido. Os métodos de ponto interior, de projeção de gradiente, de operador de partição e o de programação quadrática paralela foram utilizados no desenvolvimento deste trabalho e os algoritmos resultantes são apesentados nos capítulos seguintes.

Algoritmo 8 Método de programação quadrática sem projeção

Entrada: $\mathbf{H}$, $\mathbf{b}$

Saída: $\mathbf{u}$

Dados: $\psi^0 = 0$, ϕ_{PFQP}, ϵ, η_{PFQP}

início

$\hat{\mathbf{H}} \leftarrow \mathbf{H}^T\mathbf{H}$;

$\hat{\mathbf{H}}^+ \leftarrow \max(\mathbf{0},\hat{\mathbf{H}})$;

$\hat{\mathbf{H}}^- \leftarrow \max(\mathbf{0},-\hat{\mathbf{H}})$;

$\hat{\mathbf{b}} \leftarrow -\mathbf{H}^T\mathbf{b}$;

$\hat{\mathbf{b}}^+ \leftarrow \max(\mathbf{0},\hat{\mathbf{b}})$;

$\hat{\mathbf{b}}^- \leftarrow \max(\mathbf{0},-\hat{\mathbf{b}})$;

$j \leftarrow 0$;

enquanto $|\mathbf{u}_j - \mathbf{u}_{j-1}| \leq \epsilon$ **faça**

se *j é múltiplo de* η_{PFQP} **então**

$\mathbf{p} \leftarrow \max(\mathbf{0},-(\hat{\mathbf{H}}\mathbf{u}+\hat{\mathbf{b}}))$;

se $\mathbf{p}^T\hat{\mathbf{H}}\mathbf{p} > 0$ **então**

$$l \leftarrow -\frac{(\hat{\mathbf{H}}\mathbf{u}+\hat{\mathbf{b}})^T\mathbf{p}}{\mathbf{p}^T\hat{\mathbf{H}}\mathbf{p}}$$

senão

$l \leftarrow 0$;

$\mathbf{u}_{j+1} \leftarrow \mathbf{u}_j + l\mathbf{p}$;

senão

para $i = 1 : n_{rin}$ **faça**

$$\left[\mathbf{u}_{j+1}\right]_i \leftarrow \frac{\left[(\hat{\mathbf{H}}^- + \phi_{PFQP})\mathbf{u}_j + \hat{\mathbf{b}}^-\right]_i}{\left[(\hat{\mathbf{H}}^+ + \phi_{PFQP})\mathbf{u}_j + \hat{\mathbf{b}}^+\right]_i}\left[\mathbf{u}_j\right]_i;$$

$j \leftarrow j + 1$;

fim

4 ARTIGO 1 - GPC DE CÔMPUTO RÁPIDO BASEADO NO MÉTODO PROJEÇÃO DE GRADIENTE ACELERADO DUAL

Nesta capítulo, é apresentada uma tradução do desenvolvimento e resultados do artigo originalmente intitulado *Fast Generalized Predictive Control Based on Accelerated Dual Gradient Projection Method* (PECCIN *et al.*, 2019). O artigo foi apresentado oralmente no *12th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems* (DYCOPS) e publicado nos anais do mesmo. Nesse artigo, foi apresentado um algoritmo de cômputo rápido do GPC, para o caso SISO, com o método de projeção de gradiente acelerado dual. O algoritmo resultante foi chamado GPCGPAD e pode ser utilizado para controlar sistemas com dinâmicas rápidas e em aplicações embarcadas. O método foi primeiro validado em simulação com o software MATLAB e os resultados foram comparados com o otimizador *quadprog*. Um sistema de tamanho reduzido foi também avaliado em um FPGA com o QP computado em microssegundos. As três principais contribuições desse trabalho podem ser destacadas como segue:

- a proposição de um algoritmo GPC eficiente, integrado com o otimizador de QP e customizado para a implementação em hardwares de alta velocidade;
- a validação do algoritmo proposto em relação a um otimizador de uso geral;
- a apresentação de resultados experimentais em um hardware de alta velocidade.

4.1 ALGORITMO GPCGPAD

Nessa seção, um algoritmo para GPC com o método de projeção de gradiente acelerado dual é proposto. Esse algoritmo baseia-se no método GPAD apresentado em Patrinos e Bemporad (2014), o qual aplica o GPAD em uma formulação MPC baseada em espaço de estados.

Considerando o problema de otimização do GPC (16), o primeiro passo é definir a função Lagrangiana associada:

$$\mathcal{L}(\Delta\mathbf{u},\psi) = \frac{1}{2}\Delta\mathbf{u}^T\mathbf{H}\Delta\mathbf{u} + \mathbf{b}^T\Delta\mathbf{u} + \psi^T(\mathbf{R}\Delta\mathbf{u} - \bar{\mathbf{r}}). \tag{94}$$

O problema dual de otimização, apresentado como um problema de minimização, pode ser visto como:

$$\begin{aligned} \min_{\psi} \quad & -h_d(\psi) = \frac{1}{2}\psi^T\mathbf{R}\mathbf{H}^{-1}\mathbf{R}^T\psi + (\mathbf{R}\mathbf{H}^{-1}\mathbf{b} + \bar{\mathbf{r}})^T\psi + \frac{1}{2}\mathbf{b}^T\mathbf{H}^{-1}\mathbf{b} \\ \text{s.a.} \quad & \psi \geq 0, \end{aligned} \tag{95}$$

o qual é um problema de programação quadrática em que se aplica a propriedade da dualidade forte, a partir da condição de Slater, uma vez que há um ponto estritamente factível no problema primal (BOYD; VANDENBERGHE, 2004).

A partir de (58) e (66), o passo principal do GPM aplicado em (95) torna-se:

$$\begin{aligned} \boldsymbol{\psi}_{k+1} &= \max(\boldsymbol{\psi}_k - \frac{1}{L}\nabla h_d(\boldsymbol{\psi}_k), \mathbf{0}) \\ \boldsymbol{\psi}_{k+1} &= \max(\boldsymbol{\psi}_k - \frac{1}{L}(\mathbf{R}\mathbf{H}^{-1}\mathbf{R}^T\boldsymbol{\psi} + \mathbf{R}\mathbf{H}^{-1}\mathbf{b} + \bar{\mathbf{r}}), \mathbf{0}). \end{aligned} \tag{96}$$

A equação (96) pode ser dividida em dois passos:

$$\begin{aligned} \Delta\mathbf{u}_k &= -\mathbf{H}^{-1}(\mathbf{b} + \mathbf{R}^T\boldsymbol{\psi}), \\ \boldsymbol{\psi}_{k+1} &= \max(\boldsymbol{\psi}_k + \frac{1}{L}(\mathbf{R}\Delta\mathbf{u}_k + \bar{\mathbf{r}}), \mathbf{0}). \end{aligned} \tag{97}$$

Para o critério de parada do algoritmo, a factibilidade e a otimalidade primal podem ser utilizadas. Para a factibilidade primal, é possível testar se as restrições estão sendo atendidas dentro de uma tolerância ϵ_f, ou seja:

$$\mathbf{R}\Delta\mathbf{u}_k - \bar{\mathbf{r}} \leq \epsilon_f. \tag{98}$$

Já para o critério de otimalidade, uma propriedade derivada da dualidade forte pode ser utilizada, o que implica que:

$$h(\Delta\mathbf{u}) - p^* \leq h(\Delta\mathbf{u}) - h_d(\boldsymbol{\psi}). \tag{99}$$

Portanto, pra se obter uma solução ϵ_o ótima, é necessário que:

$$h(\Delta\mathbf{u}) - h_d(\boldsymbol{\psi}) \leq \epsilon_o, \tag{100}$$

ou seja, substituindo $h_d(\boldsymbol{\psi})$ da definição (94) e da função custo $h(\Delta\mathbf{u})$ de (16) em (100):

$$-\boldsymbol{\psi}^T(\mathbf{R}\Delta\mathbf{u}_k - \bar{\mathbf{r}}) \leq \epsilon_o. \tag{101}$$

A partir dos desenvolvimentos apresentados, o GPCGPAD pode ser representado no Algoritmo 9.

4.2 SIMULAÇÃO

De modo a avaliar o algoritmo proposto, uma simulação em MATLAB foi implementada e o Algoritmo 9 foi aplicado em uma planta SISO. A escolha de uma planta SISO com um método de controle avançado é justificada como uma opção viável para uma implementação embarcada inicial e, apesar da pequena dimensão, é capaz de avaliar o funcionamento do algoritmo. A planta foi modelada como uma função de transferência discreta:

$$G(z) = \frac{0{,}035z + 0{,}0307}{z^2 - 1{,}6375z + 0{,}6703}$$

Algoritmo 9 Algoritmo GPCGPAD

Entrada: A, B, d, $\mathbf{R}$, $\bar{\mathbf{r}}$

Saída: $u(k)$

Dados: λ, δ, N_u, N_1, N_2, $\boldsymbol{\psi}^0 = \boldsymbol{\psi}^{-1} = 0$, $\boldsymbol{\epsilon}_\mathbf{f}$, $\boldsymbol{\epsilon}_\mathbf{o}$, j_{max}

início

$\tilde{A}(z^{-1}) \leftarrow (1 - z^{-1})A(z^{-1})$;

para $i = N_1 : N_2$ **faça**

$g_i \leftarrow z(1 - \tilde{A}(z^{-1}))g_{i-1} + B(z^{-1})\bar{u}_i$;

$\mathbf{H} \leftarrow 2(\mathbf{G}^T\mathbf{G} + \lambda\mathbf{I})$;

$\breve{\mathbf{H}} \leftarrow \mathbf{H}^{-1}$;

$L \leftarrow \sqrt{\sum_{i=1}^{n_{rin}} \sum_{j=1}^{n_{rin}} \left|[\mathbf{R}\breve{\mathbf{H}}\mathbf{R}^T]_{ij}\right|^2}$;

$k \leftarrow 0$;

enquanto *modo automático* **faça**

se *período de amostragem* **então**

Obtém $y(k)$ e a referência $w(k)$;

$f_0 \leftarrow y(k)$;

para $i = 1 : N_2$ **faça**

$f_i \leftarrow z(1 - \tilde{A}(z^{-1}))f_{i-1} + B(z^{-1}\Delta u(k - d + i - 1))$;

$\mathbf{b}^T \leftarrow 2(\mathbf{f}_{N_1:N_2} - \mathbf{w}_{N_1:N_2})^T\mathbf{G}$;

enquanto $j < j_{max}$ **faça**

se $j = 0$ **então**

$\beta \leftarrow 0$

senão

$\beta \leftarrow \frac{j-1}{j+2}$;

$\mathbf{w} \leftarrow \boldsymbol{\psi}_j + \beta(\boldsymbol{\psi}_j - \boldsymbol{\psi}_{j-1})$;

$\Delta\mathbf{u} \leftarrow -\breve{\mathbf{H}}(\mathbf{R}^T\mathbf{w} + \mathbf{b})$;

$\mathbf{s} \leftarrow (1/L)(\mathbf{R}\Delta\mathbf{u} - \bar{\mathbf{r}})$;

se $\mathbf{s}_i \leq \epsilon_f/L,\ \forall i = 1,..,n_{rin}$ **então**

se $-\mathbf{w}^T\mathbf{s} < \epsilon_o/L$ **então**

retorna;

$\boldsymbol{\psi}_{j+1} \leftarrow \max(\mathbf{w} + \mathbf{s}, 0)$;

$j \leftarrow j + 1$;

$u(k) \leftarrow u(k-1) + \Delta u(k)$;

$k \leftarrow k + 1$;

fim

com o horizonte de controle $N_u = 5$, o horizonte de predição definido por $N_1 = 1$ e $N_2 = 20$, a ponderação no esforço de controle foi escolhida como $\lambda = 10$, a ponderação no erro foi escolhida como $\delta = 1$ e referências futuras foram assumidas conhecidas.

Além disso, foram definidas restrições nos incrementos de controle $|\Delta\mathbf{u}(k+j)| \leq 0{,}05$ com $j = 0,...,N_u - 1$. O problema de otimização resultou em uma matriz $\mathbf{H} \in \mathbb{R}^{5\times5}$ com 10 restrições. O GPC foi simulado com o algoritmo proposto e com um otimizador do MATLAB que utiliza o método de ponto interior (*quadprog*). Ambos algoritmos foram sintonizados com as tolerâncias de resíduo dual $\epsilon_o = 0{,}001$. A simulação foi executada por 150 períodos de tempo discreto e três degraus foram usados como referência. Não foi utilizada inicialização a quente (*warm-starting*) em nenhum dos algoritmos.

Os resultados podem ser vistos na Figura 3, onde os sinais de saída da planta e os incrementos de controle aplicados são apresentados. Pode-se perceber que o comportamento dos controladores é equivalente para ambos os algoritmos e os seus esforços de controle estão dentro dos tolerâncias definidas. Quando há a troca no valor de referência, pode-se perceber as restrições ativas e a consequente limitação do incremento de controle em $\pm 0{,}05$. Portanto, a partir dos resultados apresentados na Figura 3, o algoritmo proposto pode ser validado em comparação ao GPC com o otimizador *quadprog*.

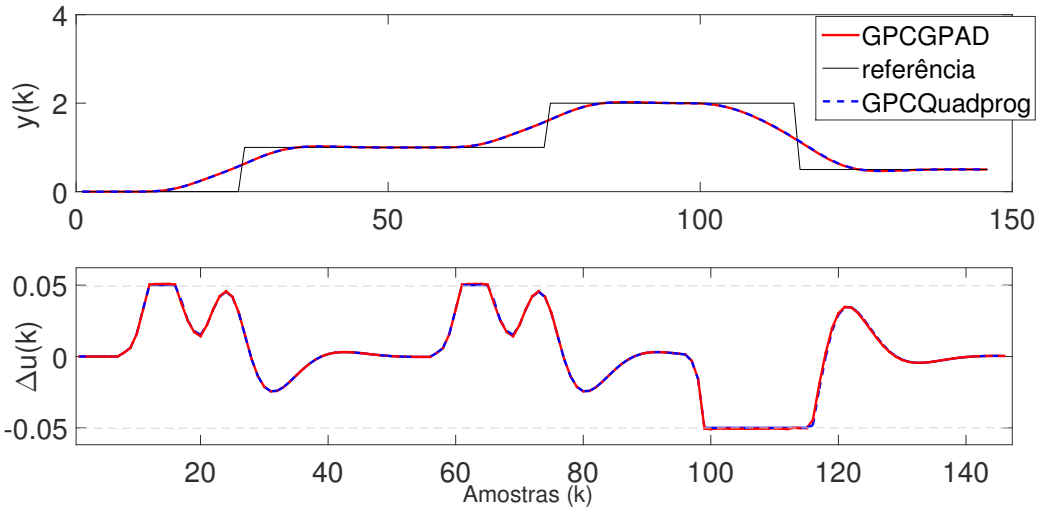


Figura 3 – Comparação da saída da planta e o sinal de incremento de controle entre o GPCGPAD e o *quadprog*.

A comparação de desempenho entre os algoritmos é ilustrada na Figura 4, onde o tempo de execução e o número de iterações de cada algoritmo são apresentados. O tempo de execução foi definido como o tempo entre o início do laço do otimizador até a saída do laço com a ação de controle obtida e o número de iterações como a quantidade de repetições do mesmo laço. É evidente que o tempo de execução é menor para o GPAD do que para o *quadprog* por mais de uma ordem de magnitude. Apesar de o GPAD necessitar de um grande número de iterações para convergir em

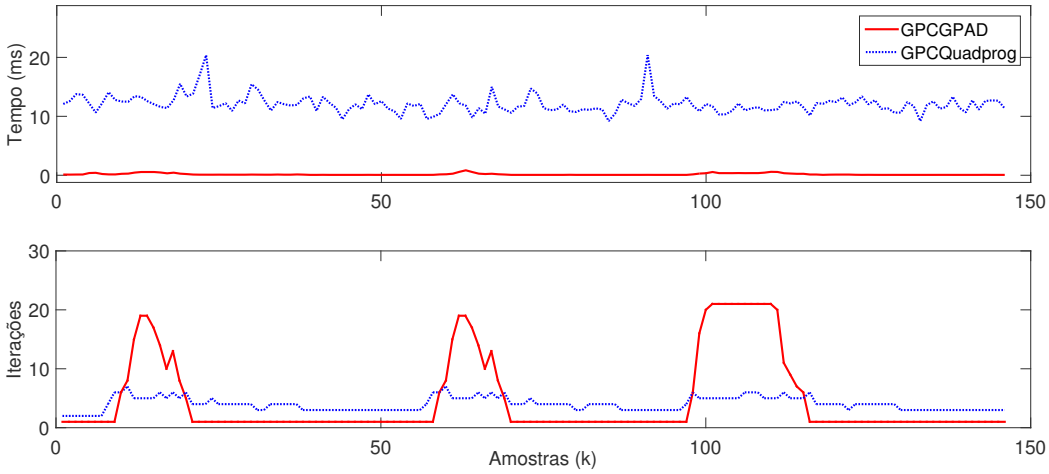


Figura 4 – Comparação do tempo de execução e do número de iterações entre o GPCGPAD e o *quadprog*.

relação ao *quadprog* durante as trocas de referência, ele ainda apresenta tempos pequenos de computação, uma vez que cada iteração é muito enxuta do ponto de vista computacional.

A Tabela 1 apresenta uma comparação dos tempos de execução médios e máximos, obtidos a partir de 10 simulações com cada método. Com os dados do tempo de execução coletados, os piores casos de tempo de execução (WCET, do inglês *Worst Case Execution Time*) do GPAD e do *quadprog* foram de 0,68 ms e 15,95 ms, respectivamente.

Tabela 1 – Comparação dos tempos de execução entre o GPCGPAD e o GPCquadprog.

	GPCGPAD	GPCquadprog
Tempo médio	0,13 ms	8,20 ms
WCET	0,68 ms	15,95 ms

4.3 IMPLEMENTAÇÃO EM FPGA

Para se avaliar o tempo de computação do algoritmo proposto embarcado em um hardware de alto desempenho, o Algoritmo 9 foi implementado em um FPGA. O FPGA utilizado foi um Altera, da família MAX 10, número 10M50DAF484C7G. A família MAX 10 apresenta conversores analógico/digitais embarcados e 50 000 elementos lógicos programáveis. O Algoritmo 9 foi implementado com a linguagem de descrição de hardware chamada Verilog. Para a representação numérica, uma aritmética de ponto fixo de 32 bits foi utilizada, com 16 bits para a representação dos decimais. Uma

máquina de estados foi utilizada para sincronizar o algoritmo e todas as operações matriciais foram implementadas com computação paralela.

Os resultados obtidos para o tempo de computação da otimização embarcada no FPGA podem ser vistos na Tabela 2. Um sinal de frequência de 50 MHz foi utilizado no FPGA e, devido ao paralelismo, o tempo de execução foi bastante rápido com um valor máximo observado de 11,76 μs.

Tabela 2 – Tempo de execução do GPCGPAD em um FPGA.

Tempo por iteração	WCET (21 iter.)
0,56 μs	11,76 μs

4.4 CONCLUSÃO

Neste capítulo foi proposto um algoritmo GPC de cômputo rápido baseado no método de projeção de gradiente acelerado dual. Uma vez que o método apenas utiliza operações matemáticas básicas a cada iteração, ele permite uma implementação direta em um FPGA. Além disso, o algoritmo proposto e o otimizador *quadprog* apresentaram resultados semelhantes na simulação em MATLAB, em termos do sinal resultante de controle, entretanto o método proposto foi pelo menos uma ordem de magnitude mais rápido. A implementação em FPGA mostrou-se bastante rápida, com um tempo de computação de 11,76 μs para o exemplo apresentado. Trabalhos futuros envolvem a extensão para os sistemas MIMO e a implementação em computação paralela do otimizador para diferentes pontos de inicialização.

5 ARTIGO 2 - GPC DE CÔMPUTO RÁPIDO COM ADMM EMBARCADO EM UM FPGA

Nesse capítulo, são apresentados os desenvolvimentos e resultados do artigo originalmente intitulado *Fast Constrained Generalized Predictive Control with ADMM Embedded in an FPGA* (PECCIN *et al.*, 2020b). O artigo foi publicado em uma edição especial de sistemas embarcados no periódico *IEEE Latin America Transactions*. Uma versão curta do GPC com ADMM foi também apresentada oralmente no Congresso Brasileiro de Automática (CBA) em 2018 e publicada nos anais do mesmo (PECCIN *et al.*, 2018). Nesse artigo, destaca-se a apresentação de um algoritmo de cômputo rápido do GPC, para o caso SISO, com o método dos multiplicadores em direções alternadas. Algumas questões de implementação em computação paralela são também discutidas de modo a acelerar o tempo requerido para as operações. A implementação em um FPGA provou ser bastante rápida, com um pior caso de tempo de execução observado de 11,54 μs para o exemplo apresentado. Esses resultados contribuem para a aplicação do GPC embarcado em processos que são tipicamente controlados por controladores clássicos por causa das suas dinâmicas rápidas. As principais contribuições desse artigo são:

- a proposição de um algoritmo eficiente para GPC, integrado com o otimizador de QP e dedicado para a implementação embarcada em um hardware paralelo de alta velocidade;
- a proposição de uma arquitetura de implementação para FPGA;
- a discussão dos detalhes de implementação embarcada que tornam a solução adequada para sistemas com dinâmicas rápidas;
- os resultados experimentais em um *hardware* paralelo de alta velocidade.

5.1 ALGORITMO GPCADMM

A partir do QP do GPC, definido em (16), o primeiro passo proposto é representar as restrições por meio de uma função indicadora $I\_(\mathbf{x})$. Pode-se definir $h(\Delta\mathbf{u}) = \frac{1}{2}\Delta\mathbf{u}^T\mathbf{H}\Delta\mathbf{u} + \mathbf{b}^T\Delta\mathbf{u}$ e reescrever o problema rearranjando as restrições com uso da função indicadora $I\_(x)$ na função objetivo, resultando em:

$$\min_{\Delta\mathbf{u}} \quad h(\Delta\mathbf{u}) + I\_(\bar{\mathbf{R}}\Delta\mathbf{u} - \bar{\mathbf{r}}). \tag{102}$$

Para se obter a forma padrão do ADMM, apresentada em (73), pode-se definir uma nova variável $\mathbf{z} = \bar{\mathbf{R}}\Delta\mathbf{u} - \bar{\mathbf{r}}$ para a segunda equação e inserir uma restrição de

igualdade para manter o acoplamento entre as variáveis:

$$\begin{aligned} \min_{\Delta\mathbf{u},\mathbf{z}} \quad & h(\Delta\mathbf{u}) + I\_(\mathbf{z}) \\ \text{s.a.} \quad & \bar{\mathbf{R}}\Delta\mathbf{u} - \mathbf{z} = \bar{\mathbf{r}}. \end{aligned} \tag{103}$$

A partir da forma padrão de (103), o Lagrangiano aumentado $\mathcal{L}_\rho$ pode ser definido como:

$$\mathcal{L}_\rho(\Delta\mathbf{u},\mathbf{z},\boldsymbol{\nu}) = h(\Delta\mathbf{u}) + I\_(\mathbf{z}) + \boldsymbol{\nu}^T(\bar{\mathbf{R}}\Delta\mathbf{u} - \mathbf{z} - \bar{\mathbf{r}}) + \frac{\rho}{2}\|\bar{\mathbf{R}}\Delta\mathbf{u} - \mathbf{z} - \bar{\mathbf{r}}\|^2.$$

Com o Lagrangiano aumentado e os passos básicos do ADMM escalonado descritos em (80), o GPCADMM pode ser computado como:

$$\Delta\mathbf{u}_{j+1} = \text{argmin}\left(f(\Delta\mathbf{u}_j) + \frac{\rho}{2}\|\bar{\mathbf{R}}\Delta\mathbf{u} - \mathbf{z}_j - \bar{\mathbf{r}} + \boldsymbol{\phi}_j\|^2\right), \tag{104}$$

$$\mathbf{z}_{j+1} = \text{argmin}\left(I\_(\mathbf{z}) + \frac{\rho}{2}\|\bar{\mathbf{R}}\Delta\mathbf{u}_{j+1} - \mathbf{z}_j - \bar{\mathbf{r}} + \boldsymbol{\phi}_j\|^2\right), \tag{105}$$

$$\boldsymbol{\phi}_{j+1} = \boldsymbol{\phi}_j + (\bar{\mathbf{R}}\Delta\mathbf{u}_{j+1} - \mathbf{z}_{j+1} - \bar{\mathbf{r}}). \tag{106}$$

Para computar o passo da equação (104), pode-se abrir o termo da norma e agrupar da seguinte forma:

$$\Delta\mathbf{u}_{j+1} = \text{argmin}\left(\frac{1}{2}\Delta\mathbf{u}^T(\mathbf{H} + \bar{\mathbf{R}}^T\bar{\mathbf{R}})\Delta\mathbf{u} + (\mathbf{b}^T + \bar{\mathbf{R}}^T(\boldsymbol{\phi} - \mathbf{z} - \bar{\mathbf{r}}))\Delta\mathbf{u}\right).$$

Portanto, se forem definidos $\tilde{\mathbf{H}} = \mathbf{H} + \bar{\mathbf{R}}^T\bar{\mathbf{R}}$ e $\tilde{\mathbf{b}} = \mathbf{b}^T + \bar{\mathbf{R}}^T(\boldsymbol{\phi} - \mathbf{z} - \bar{\mathbf{r}})$ recai-se num problema padrão de programação quadrática sem restrições:

$$\min_{\Delta\mathbf{u}} \quad \frac{1}{2}\Delta\mathbf{u}^T\tilde{\mathbf{H}}\Delta\mathbf{u} + \tilde{\mathbf{b}}^T\Delta\mathbf{u}, \tag{107}$$

que possui solução analítica:

$$\Delta\mathbf{u}_{j+1} = -\tilde{\mathbf{H}}^{-1}\tilde{\mathbf{b}}.$$

Como em um problema típico de GPC a matriz $\tilde{\mathbf{H}}$ pode ser computada offline, então a sua inversa pode ser armazenada *offline* e a solução desse passo pode ser computada apenas com uma multiplicação matricial.

Para computar o passo da equação (105), pode-se verificar que ela pode ser reescrita com emprego do operador de proximidade (STATHOPOULOS *et al.*, 2016) da seguinte forma:

$$\mathbf{z}_{j+1} = \text{prox}_{I\_,\rho}(\bar{\mathbf{R}}\Delta\mathbf{u}_{j+1} - \bar{\mathbf{r}} + \boldsymbol{\phi}_j). \tag{108}$$

Dado que $I\_(\mathbf{z})$ é uma função indicadora do ortante negativo $\mathbb{R}^n_-$, o qual é um conjunto convexo fechado e não vazio, o operador de proximidade torna-se uma projeção em $\mathbb{R}^n_-$ (PARIKH; BOYD, 2014). Dessa forma, a solução de (108) pode ser obtida a partir da seleção dos elementos negativos de cada elemento ou zero:

$$\mathbf{z}_{j+1} = \min(\bar{\mathbf{R}}\Delta\mathbf{u}_{j+1} - \bar{\mathbf{r}} + \boldsymbol{\phi}_j, \mathbf{0}).$$

Os resíduos das condições de otimalidade podem ser usados como critério de parada. O resíduo primal $\mathbf{r}$ e o resíduo dual $\mathbf{s}$ são dados por:

$$\begin{aligned} \mathbf{r}_{j+1} &= \bar{\mathbf{R}}\Delta\mathbf{u}_{j+1} - \mathbf{z}_{j+1} - \bar{\mathbf{r}}, \\ \mathbf{s}_{j+1} &= \rho\bar{\mathbf{R}}^T(\mathbf{z}_{j+1} - \mathbf{z}_j). \end{aligned} \tag{109}$$

Portanto, um critério de parada razoável é que os resíduos devem ser menores que um valor de tolerância desejado:

$$\|\mathbf{r}_j\| \leq \epsilon_p \quad \text{e} \quad \|\mathbf{s}_j\| \leq \epsilon_d. \tag{110}$$

A partir dos desenvolvimentos apresentados, pode-se resumir o GPCADMM, para o caso SISO, no Algoritmo 10. A generalização do algoritmo para o caso MIMO é direta, como foi apresentada na Seção 2.2.4.

5.2 DETALHES DA IMPLEMENTAÇÃO EMBARCADA

Esta seção descreve os detalhes da implementação para possibilitar embarcar o algoritmo em arquitetura paralela, os quais contribuem para um ganho de velocidade no tempo de computação e pouco uso de memória.

5.2.1 Aritmética de ponto fixo

A representação numérica geralmente é feita escolhendo-se entre a aritmética de ponto fixo e a aritmética de ponto flutuante. A aritmética de ponto fixo tem vantagens para sistemas embarcados, tais como introduzir um menor atraso computacional e ocupar menos espaço de memória em relação à de ponto flutuante. O pequeno atraso nas operações de aritmética de ponto fixo permite que as implementações atinjam tempos de execução menores que os que seriam alcançados com uma representação em ponto flutuante. Por outro lado, a aritmética de ponto fixo possui uma faixa dinâmica mais limitada. Dessa forma, a opção pela aritmética de ponto fixo requer uma análise numérica cuidadosa para a determinação do número de bits para representar a parte inteira e a parte fracionária tais que *overflows* e o acúmulo de erros de arredondamento mantenham-se suficientemente pequenos (PEYRL *et al.*, 2014).

A implementação da aritmética de ponto fixo é adequada para a utilização no algoritmo GPCADMM graças a algumas características. A primeira delas é que, em aplicações de controle, o modelo da planta e suas variáveis de entrada e saída podem ser normalizados. Além disso, as limitações físicas como a resolução dos transdutores e conversores analógico-digital (A/D) e os limites dos atuadores fazem com que a faixa dinâmica necessária seja pequena e, nesse caso, uma representação com poucos bits é suficiente. A segunda característica é que o algoritmo proposto não requer operações de divisão e, dessa forma, o problema de erro relativo grande, comum na representação de números pequenos em aritmética de ponto fixo, não é amplificado.

Algoritmo 10 GPCADMM

Entrada: A, B, d, $\bar{\mathbf{R}}$, $\bar{\mathbf{r}}$

Saída: $u(k)$

Dados: q_λ, q_δ, N_u, N_1, N_2, $\rho > 0$, $\mathbf{\Phi}_0 = \mathbf{0}$, ϵ_p, ϵ_d, $\Delta\mathbf{u}(-1)$, j_{max}

início

$\tilde{A}(z^{-1}) \leftarrow (1 - z^{-1})A(z^{-1})$;

para $i = 1 : N_2$ **faça**

$g_i \leftarrow z(1 - A(z^{-1}))g_{i-1} + B(z^{-1})\bar{u}_i$;

para $i = 1 : N_u$ **faça**

$\mathbf{G}_{i:N,i} \leftarrow \mathbf{g}_{N_1:N_2+1-i}$;

$\mathbf{H} \leftarrow 2(\mathbf{G}^T\mathbf{G} + q_\lambda\mathbf{I})$;

$\tilde{\mathbf{H}} \leftarrow \mathbf{H} + \bar{\mathbf{R}}^T\rho\bar{\mathbf{R}}$;

$\breve{\mathbf{H}} \leftarrow \tilde{\mathbf{H}}^{-1}$;

$k \leftarrow 0$;

enquanto *modo automático* **faça**

se *tempo de amostragem* **então**

Adquire saída $y(k)$ e referências futuras;

$f_0 \leftarrow y(k)$;

para $i = 1 : N_2$ **faça**

$f_i \leftarrow z(1 - \tilde{A}(z^{-1}))f_{i-1} + B(z^{-1})\Delta u(k - d + i - 1)$;

$\mathbf{b}^T \leftarrow 2(\mathbf{f}_{N_1:N_2} - \mathbf{w}_{N_1:N_2})^T\mathbf{G}$;

$\mathbf{z}_0 \leftarrow \bar{\mathbf{R}}\Delta\mathbf{u}(k-1) - \bar{\mathbf{r}}$;

$j \leftarrow 0$;

enquanto $j < j_{max}$ **faça**

$\tilde{\mathbf{b}} \leftarrow \mathbf{b}^T + \rho\bar{\mathbf{R}}^T(\mathbf{\Phi}_j - \mathbf{z}_j - \bar{\mathbf{r}})$;

$\Delta\mathbf{u} \leftarrow -\breve{\mathbf{H}}\tilde{\mathbf{b}}$;

$\mathbf{z}_{j+1} \leftarrow \min(\bar{\mathbf{R}}\Delta\mathbf{u} - \bar{\mathbf{r}} + \mathbf{\Phi}_j, \mathbf{0})$;

$\mathbf{\Phi}_{j+1} \leftarrow \mathbf{\Phi}_j + (\bar{\mathbf{R}}\Delta\mathbf{u} - \mathbf{z}_{j+1} - \bar{\mathbf{r}})$;

se $\|\bar{\mathbf{R}}\Delta\mathbf{u} - \mathbf{z}_{j+1} - \bar{\mathbf{r}}\|_2 \leq \epsilon_p$ ***e*** $\|\rho\bar{\mathbf{R}}^T(\mathbf{z}_{j+1} - \mathbf{z}_j)\|_2 \leq \epsilon_d$ **então**

retorna;

$j \leftarrow j + 1$;

$\Delta u(k) \leftarrow [1 \quad 0 \quad \cdots \quad 0]_{1\times N_u}\Delta\mathbf{u}$;

$u(k) \leftarrow u(k-1) + \Delta u(k)$;

$k \leftarrow k + 1$;

fim

5.2.2 Multiplicação matricial paralela

Uma das operações mais custosas do ponto de vista computacional, no contexto dos algoritmos propostos, é a multiplicação matricial. Algoritmos clássicos de multiplicação de matrizes densas, para um processador simples, são computados dentro de três laços de repetição aninhados, com n repetições cada. Tais algoritmos possuem uma complexidade $O(n^3)$, o que representa um crescimento acentuado do número de operações com a dimensão da matriz (QASIM *et al.*, 2009).

O uso de computação paralela pode diminuir consideravelmente o tempo necessário para essa operação. Existem formas bastante sofisticadas de implementação paralela, como a utilização da arquitetura sistólica, que se propõe a otimizar três métricas, a velocidade, a área ocupada e o consumo de energia (KUMM *et al.*, 2017), (XIE *et al.*, 2017). Neste trabalho é proposta uma forma de implementação mais simples, focada na métrica de velocidade.

A ideia geral é agrupar as operações independentes, executá-las de forma paralela e repassar os resultados intermediários de forma sequencial por meio de registradores. Um esquemático da proposta de arquitetura pode ser visto na Figura 5. Neste exemplo, é apresentado o cômputo de um elemento de uma matriz de ordem $n = 10$ com um bloco somador de 10 elementos. Como cada elemento pode também ser computado em paralelo, a multiplicação entre duas matrizes 10×10 apresenta uma latência de 2 pulsos de *clock* (para uma aritmética de ponto fixo). O algoritmo paralelo proposto apresenta uma propriedade de complexidade dada por $O(\lceil \log_{10}(n) \rceil + 1)$. Para comparação, uma multiplicação entre duas matrizes 10×10 no algoritmo paralelo necessita de 2 pulsos de *clock*, enquanto o algoritmo sequencial requer 1 000 pulsos.

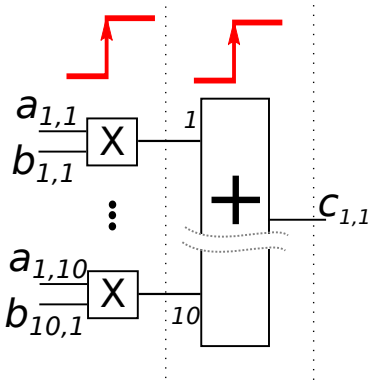


Figura 5 – Exemplo de multiplicação matricial paralela entre duas matrizes 10×10 em apenas 2 pulsos de *clock*.

5.2.3 Arquitetura de implementação do ADMM

De modo a implementar o algoritmo do ADMM em arquitetura paralela, foi proposta uma solução a partir de uma máquina de estados finita. A solução foi centrada

no controle sequencial das ações através da máquina de estados, de modo que os valores intermediários de cômputo fossem coordenados enquanto as operações de soma, subtração e multiplicação matriciais executam paralelamente. Na Figura 6 são representadas de forma esquemática a implementação em máquina de estados e as operações correspondentes. No estado 11, o critério de parada é verificado e, em caso positivo, após o passo 12, o processo é então terminado, sinalizando o incremento de controle encontrado. Dessa forma, pode-se fazer uma estimativa do tempo de execução do algoritmo a partir da seguinte equação:

$$t = T_c(2 + n_{iter}(8 + (\lceil \log_{10}(N_u) \rceil + 1) + 3(\lceil \log_{10}(n_{rin}) \rceil + 1))),$$

onde T_c é o período de *clock* utilizado pelo *hardware* e n_{iter} é o número de iterações necessários pelo algoritmo atingir o critério de parada.

Para ilustrar a utilização, toma-se um cenário com um horizonte de controle $N_u = 5$ e com restrições de limite superior e inferior no incremento de controle $n_{rin} = 10$. Neste cenário, obtém-se um total de 18 pulsos de *clock* para uma iteração completa. Para um *hardware* com frequência de *clock* de 50 MHz, o período é $T_c = 0{,}02\,\mu$s e, portanto, o tempo total de uma iteração é de 0,36 μs.

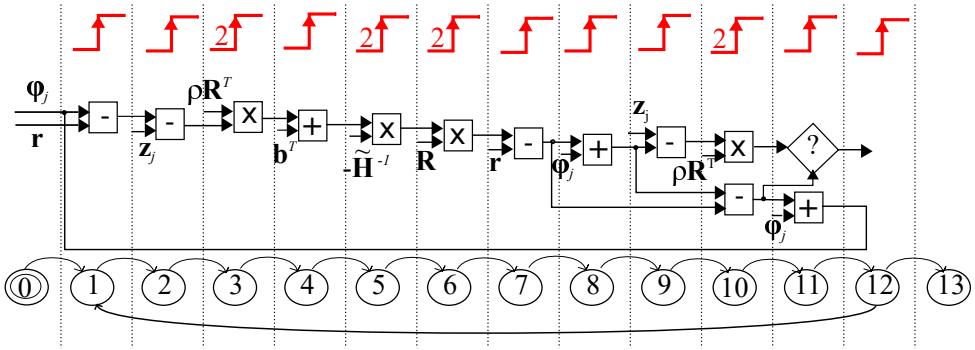


Figura 6 – Arquitetura de implementação do GPCADMM com máquina de estados finita.

5.3 IMPLEMENTAÇÃO EM FPGA

Para se avaliar o tempo de cômputo do algoritmo embarcado, o Algoritmo 10 foi implementado em um FPGA. O FPGA utilizado foi um Altera MAX10 modelo 10M50DAF484C7G. A família MAX 10 apresenta conversores A/D embarcados no FPGA e 50 000 elementos lógicos programáveis. O sinal de *clock* utilizado foi de 50 MHz na placa de desenvolvimento DE10-Lite. Essa é uma solução de baixo custo e com uma velocidade modesta se comparada com FPGAs de ponta, como o Stratix IV, que pode ser utilizado com sinais de *clock* de 600 MHz (ALTERA, 2016). O algoritmo foi implementado em Verilog. Para a representação numérica, foi implementada uma aritmética de ponto fixo com uma palavra de 16 bits, com 10 bits para os decimais.

Máquinas de estados foram utilizadas para sincronizar os módulos e todas as operações de multiplicação matricial foram implementadas para execução em paralelo. Um esquemático da arquitetura implementada pode ser visto na Figura 7. Os parâmetros de configuração do controlador são passados por meio de comunicação externa (USB-Blaster) com o processador NIOS II, em sua versão econômica, implementado no FPGA. A comunicação do software embarcado no NIOS II com o bloco de lógica GPC foi feito através do barramento Avalon®. O bloco GPC instancia os blocos ADMM e Resposta livre. As bibliotecas de multiplicação matricial e as operações básicas com aritmética de ponto fixo também foram implementadas diretamente em lógica customizada. A implementação foi compilada no software Intel Quartus Prime 17.1.0 e resultou em uma ocupação de 30 662 elementos lógicos (62%), 7 691 registradores, 49,125 KB de memória (24%) e 40 multiplicadores embarcados (14%). Por se tratar de um FPGA de baixo custo, a ocupação de 62% pode ser considerada boa e com margem para o aumento no tamanho dos horizontes se necessário. Os requisitos temporais foram obtidos com valores positivos de folga, sendo os piores casos, com folgas menores, de 1,034 ns de *setup*, 0,074 ns de *hold*, 15,394 ns de *recovery* e 0,406 ns de *removal*.

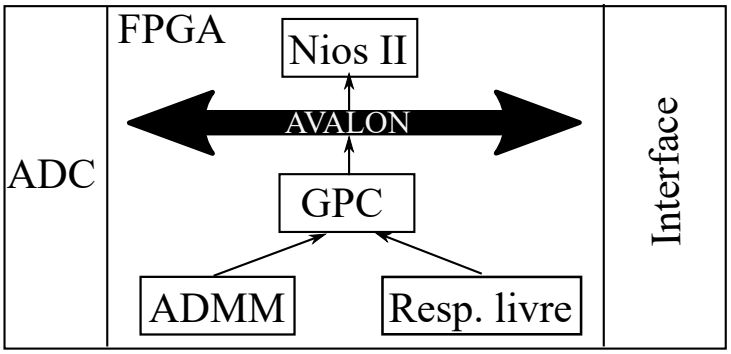


Figura 7 – Arquitetura de implementação em FPGA.

O estudo de caso utilizado foi o mesmo de Peccin *et al.* (2018), cuja planta foi modelada com uma função de transferência discreta de segunda ordem:

$$G(z) = \frac{0{,}035z + 0{,}0307}{z^2 - 1{,}6375z + 0{,}6703}, \tag{111}$$

com um horizonte de controle $N_u = 5$, horizontes de predição $N_1 = 1$ e $N_2 = 20$, ponderação no esforço de controle $\lambda = 10$, ponderação no erro $\delta = 1$ e referências futuras conhecidas $w(t+j)$, $j = N_1,...,N_2$.

Foram também definidas restrições para os incrementos de controle $|\Delta \mathbf{u}(k+j)| \leq 0{,}05$ com $j = 0,...,N_u - 1$. O problema de otimização resulta em uma matriz $\mathbf{H} \in \mathbb{R}^{5\times5}$ e 10 restrições. O GPCADMM foi sintonizado com $\rho = 50$ e com as tolerâncias dos resíduos primal e dual $\epsilon_p = \epsilon_d = \sqrt{N_u}0{,}001$.

O estudo de caso foi primeiramente simulado em MATLAB de modo a se obter o pior caso no tempo de execução. A simulação foi realizada por 150 amostras discretas

e com três trocas de referência do tipo degrau. Não foi utilizado *warm-start* para se avaliar o WCET. As simulações foram feitas em um computador com processador Core i3 de 2,40 GHz e 12 GB RAM. Os tempos de execução do algoritmo foram medidos através do comando *tic-toc* do MATLAB.

Os comportamento do sistema de controle pode ser visto na Figura 8, na qual são apresentados a saída da planta e o incremento de controle aplicado. Nas trocas de referência, pode-se perceber que as restrições estão ativas limitando o incremento de controle em $\pm$0,05.

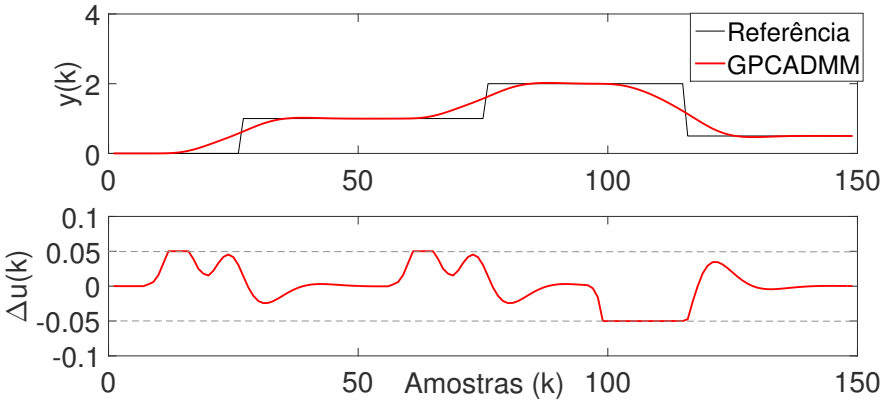


Figura 8 – Resultado da simulação com a saída da planta e o sinal de incremento de controle para o algoritmo GPCADMM proposto.

O desempenho do algoritmo é ilustrado na Figura 9, na qual o tempo de execução e o número de iterações são apresentados. O tempo de execução foi definido como o tempo entre o início do laço do otimizador até a saída do laço com a ação de controle obtida e o número de iterações como a quantidade de repetições do mesmo laço. Apesar de o GPCADMM apresentar um aumento no número de iterações durante as trocas de referência, cada iteração é pouco custosa do ponto de vista computacional. Essa é uma característica desejada para o algoritmo obter um bom desempenho ao ser embarcado.

A Tabela 3 apresenta os tempos de execução médio e máximo, obtidos a partir de 10 simulações. Dos dados de tempo de execução coletados, o tempo máximo observado pelo GPCADMM, com 33 iterações, foi de 1,30 ms.

Tabela 3 – Apresentação dos tempos de cômputo médio e máximo do GPCADMM.

GPCADMM	
Tempo médio	0,66 ms
Tempo Máximo (iter.)	1,30 ms (33)

O pior caso observado na simulação foi replicado na implementação embarcada em FPGA. O tempo gasto por iteração em cada módulo foi de 0,14 μs para o GPC,

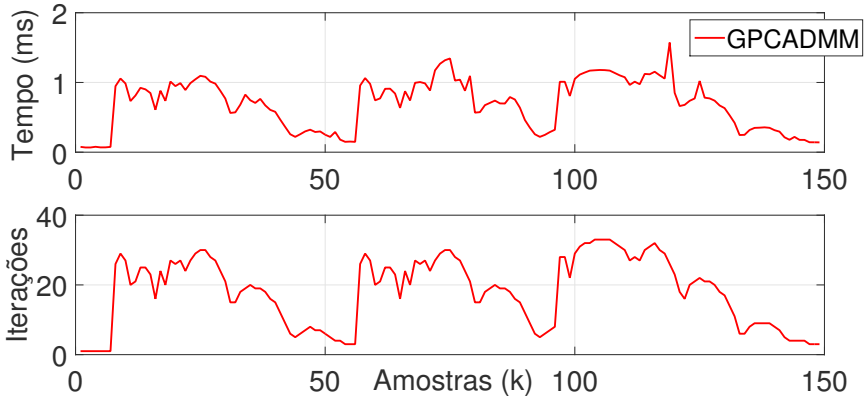


Figura 9 – Resultado da simulação com o tempo de execução e o número de iterações do algoritmo GPCADMM proposto.

0,36 μs para o ADMM e 0,80 μs para resposta livre, totalizando 1,30 μs para 1 iteração completa do GPCADMM. Os resultados obtidos para o pior caso, com 33 iterações do ADMM, podem ser vistos na Tabela 4. Devido ao paralelismo implementado, pode-se perceber que o tempo de execução é bastante rápido, com valor máximo de 11,54 μs.

Esse resultado pode ser considerado competitivo com outras soluções para MPC. Em Peccin *et al.* (2019), um algoritmo GPC com o método de projeção de gradiente acelerado dual (GPAD, do inglês *Accelerated Dual Gradient Projection*) foi testado com a mesma planta de exemplo, apresentada em (111), e com o mesmo FPGA. O WCET observado nesse trabalho foi de 11,76 μs com 21 iterações do algoritmo de otimização. Percebe-se que a arquitetura GPCADMM proposta foi mais rápida, mesmo com um número maior de iterações. Esse desempenho pode ser explicado, em grande parte, graças ao ganho obtido com a arquitetura de multiplicação matricial, implementada com o somador de 10 entradas. Vale ressaltar que os métodos acelerados de primeira ordem, como o GPAD, sofrem de acumulação de erro quando aplicados com aritmética de ponto fixo (DEVOLDER *et al.*, 2014). Um estudo sobre a representação numérica de ponto fixo adequada para o ADMM pode ser vista em Jerez *et al.* (2014). Já para as formulações em espaço de estados, em Wills *et al.* (2011), um algoritmo MPC com o método de ponto interior foi aplicado em uma planta de controle de vibração em uma haste flexível. A planta foi modelada com 14 estados e o horizonte de predição de 14 amostras. O tempo de cômputo em um FPGA com 70 MHz e aritmética de ponto flutuante foi de 30,0 μs. Em Yang *et al.* (2012), um MPC com o método de conjunto ativo foi aplicado em um servomotor modelado com 2 estados. O horizonte de controle utilizado foi de 3 amostras e o tempo de cômputo em um FPGA com *clock* de 100 MHz foi de 20,0 μs.

Tabela 4 – Tempos de execução do GPCADMM em FPGA.

	Tempo máximo com 33 iterações
GPC	0,14 μs
ADMM	10,60 μs
Resp. livre	0,80 μs
GPCADMM	11,54 μs

5.4 CONCLUSÃO

Neste trabalho foi apresentada uma implementação embarcada de um algoritmo GPC de cômputo rápido baseado em ADMM. Como o algoritmo proposto utiliza apenas operações básicas em cada iteração, a implementação embarcada em FPGA se torna bastante adequada. O GPCADMM embarcado em FPGA mostrou-se bastante rápido para o estudo de caso apresentado, validando a arquitetura proposta e as acelerações propostas nos cômputos das operações matemáticas. Os resultados obtidos contribuem para a viabilidade da utilização do GPC em processos que tipicamente são controlados por PIDs por conta da dinâmica rápida, trazendo assim as vantagens inerentes ao método.

6 ARTIGO 3 - ALGORITMOS RÁPIDOS PARA O CONTROLE PREDITIVO GENERALIZADO COM OTIMIZAÇÃO ONLINE

Neste capítulo, são apresentados os desenvolvimentos e resultados traduzidos do artigo originalmente intitulado *Fast Algorithms for Constrained Generalized Predictive Control with On-line Optimization* (PECCIN *et al.*, 2021a). O artigo foi publicado no periódico *IET Control Theory & Applications*. Nesse artigo são propostos algoritmos de cômputo rápido para o controle preditivo generalizado com restrições, baseados em dois métodos de primeira ordem chamados método de projeção de gradiente acelerado dual (GPAD) e algoritmo rápido de minimização alternada (FAMA). Os algoritmos resultantes foram nomeados como GPCGPAD e GPCFAMA e eles podem ser utilizados em sistemas de controle com dinâmicas rápidas. Além disso, eles são adequados para aplicações embarcadas. Uma vantagem dos algoritmos propostos em relação à abordagens baseadas em espaço de estados com métodos de primeira ordem, tais como as apresentadas em Cairano *et al.* (2013)-Patrinos e Bemporad (2014), é a necessidade de um número reduzido de sensores e variáveis utilizadas pelo modelo de predição do GPC. O GPCGPAD foi apresentado previamente em Peccin *et al.* (2019), e neste capítulo o resultado é estendido para processos com múltiplas entradas e múltiplas saídas (MIMO, do inglês *Multi-Input Multi-Output*). Como a formulação utilizada na abordagem proposta é baseada no modelo de funções de transferência, a extensão para o caso MIMO não é tão direto como no caso dos modelos em espaço de estados. Além disso, a generalização proposta neste capítulo permite considerar um horizonte de predição diferente para cada variável de processo e um horizonte de controle diferente para cada variável manipulada. São também apresentadas análises de limites máximos teóricos no número de iterações dos algoritmos, alguns detalhes importantes de implementação e um método de precondicionamento para acelerar o cômputo. Uma comparação com novo algoritmo GPCFAMA é apresentado, o que também é uma contribuição original deste trabalho. Os métodos foram primeiramente validados por simulação e seus resultados são comparados com os resultados de otimizadores comerciais de uso geral. Um inversor de frequência trifásico com filtro LCL foi utilizado como estudo de caso. Alguns trabalhos recentes para o GPC de cômputo rápido aplicados em inversores de frequência são apresentados em Judewicz *et al.* (2016) e em Judewicz *et al.* (2018), entretanto, neles são tratados apenas o caso sem restrições. Alguns exemplos de MPC utilizados para o controle de conversores de potência trifásicos são apresentados em Maccari Jr *et al.* (2020), no qual um MPC robusto sem restrições e com controladores ressonantes é utilizado para se obter uma rejeição de harmônicos e em Guzman *et al.* (2019), no qual um MPC sem restrições com modelo reduzido, integrador e um termo de retroalimentação é introduzido para minimizar a carga harmônica nas correntes da rede. Os algoritmos propostos foram também avaliados em um FPGA e o programa quadrático foi computado em microssegundos. As

cinco principais contribuições desse trabalho são:

- a proposição de algoritmos GPC com restrições, baseados no GPAD e FAMA, e adequados para implementação embarcada com otimização online;
- uma comparação das estratégias propostas com outras abordagens MPC e resolvedores de uso geral;
- análise determinística do WCET para as implementações em um FPGA baseados nos limites máximos teóricos para os algoritmos propostos;
- validação com um estudo de caso baseado em um inversor de frequência trifásico com filtro LCL e conectado à rede, o qual representa um importante papel no cenário de aplicações de energias renováveis;
- discussão de questões de implementação tais como a multiplicação matricial paralela e um precondicionamento que permite a redução do tempo de computação para a arquitetura do FPGA considerado neste estudo.

6.1 ALGORITMOS PROPOSTOS

Um sistema MIMO com n_c entradas e n_o saídas pode ser representado pelo modelo autorregressivo com média móvel, integrador e entrada controlada (CARIMA), como segue:

$$\mathbf{A}\left(z^{-1}\right)\mathbf{y}(k) = \mathbf{D}\left(z^{-1}\right)\mathbf{B}\left(z^{-1}\right)\mathbf{u}(k-1) + \mathbf{C}\left(z^{-1}\right)\frac{e(k)}{\Delta},$$

onde $\mathbf{y} \in \mathbb{R}^{n_o}$ é um vetor das saídas do sistema, $\mathbf{u} \in \mathbb{R}^{n_c}$ é um vetor com as entradas do sistema, $\mathbf{A}(z^{-1})$ é uma matriz polinomial de dimensões $n_o \times n_o$, a matriz polinomial $\mathbf{B}(z^{-1})$ tem dimensões $n_o \times n_c$ e a matriz polinomial $\mathbf{C}(z^{-1})$ tem dimensões $n_o \times n_o$. $\mathbf{D}(z^{-1})$ é uma matriz diagonal de dimensões $n_o \times n_o$ com cada elemento representando o menor tempo morto entre as entradas relacionadas às correspondentes saídas, $e(k)$ é um ruído branco de média nula e $\Delta = 1 - z^{-1}$ é utilizado para melhorar a modelagem de perturbações não estacionárias.

A função custo típica de um GPC MIMO é dada por:

$$J_{GPC} = \sum_{p=1}^{n_o} \sum_{h=N_1^{(p)}}^{N_2^{(p)}} \left[\hat{y}^{(p)}(k+h|k) - w^{(p)}(k+h)\right]^2 q_\delta^{(p)} + \sum_{l=1}^{n_c} \sum_{h=1}^{N_u^{(l)}} \left[\Delta u^{(l)}(k+h-1)\right]^2 q_\lambda^{(l)}, \tag{112}$$

onde $\hat{y}^{(p)}(k+h|k)$ representa a predição da saída p do sistema no instante de tempo discreto futuro $k+h$, determinado em k, $w^{(p)}(k+h)$ é a trajetória de referência em $k+h$, $\Delta u^{(l)}$ é o incremento de controle para a entrada l. $N^{(p)} = N_2^{(p)} - N_1^{(p)} + 1$ define o

horizonte de predição para a saída p, onde $N_1^{(p)}$ é a amostra de horizonte inicial e $N_2^{(p)}$ é a amostra final do horizonte. $N_u^{(l)}$ é o horizonte de controle para a entrada l. Assumindo ponderações constantes sobre os horizontes, $q_\delta^{(p)}$ e $q_\lambda^{(p)}$ são as ponderações nos erros e nos esforços de controle, os quais podem ser agrupados nas matrizes bloco diagonais positivas definidas $\mathbf{Q}_\delta^{(p)}$ e $\mathbf{Q}_\lambda^{(l)}$, respectivamente.

Se o vetor de incrementos de controle for definido como:

$$\Delta\mathbf{u} = [\Delta\mathbf{u}^{(1)}\ \Delta\mathbf{u}^{(2)}\ ...\ \Delta\mathbf{u}^{(n_c)}]^T, \quad N_u = \sum_{l=1}^{n_c} N_u^{(l)},$$

o QP típico do GPC pode ser representado por:

$$\begin{aligned} \min_{\Delta\mathbf{u}} \quad & h(\Delta\mathbf{u}) = \frac{1}{2}\Delta\mathbf{u}^T\mathbf{H}\Delta\mathbf{u} + \mathbf{b}^T\Delta\mathbf{u} \\ \text{s.a.} \quad & \mathbf{R}\Delta\mathbf{u} \leq \mathbf{r}, \end{aligned} \tag{113}$$

onde $\mathbf{R} \in \mathbb{R}^{n_{rin} \times N_u}$ é a matriz de restrições, $\mathbf{r} \in \mathbb{R}^{n_{rin}}$ é o vetor de restrições e n_{rin} é o número de restrições de desigualdades. A forma resumida de representar as restrições $\mathbf{R}\Delta\mathbf{u} \leq \mathbf{r}$ é genérica e permite a representação de muitos tipos de restrições, tais como de magnitude, de taxa de variação e de sobressinal para ambas as variáveis de entrada e saída (CAMACHO; BORDONS, 2004). Os detalhes para a obtenção de (113) e da formulação GPC para sistemas MIMO podem ser vistas em Peccin *et al.* (2020b) e em Normey-Rico e Camacho (2000).

O conjunto de QPs obtidos a partir do GPC e definidos em (113), considerando todas as iterações de controle, pode ser reformulado como um problema de otimização paramétrica de apenas um parâmetro com a troca de variáveis apropriada. Essa representação é utilizada para se obter os limites de subotimalidade no número de iterações para os algoritmos propostos. Considerando um conjunto de referências constante $\mathbf{w}$, o QP do GPC, para todas as iterações de controle, pode ser visto como o problema de otimização paramétrica:

$$\begin{aligned} \mathcal{F}^*(\mathbf{e}) = \min_{\Delta\mathbf{u}} \quad & \frac{1}{2}\Delta\mathbf{u}^T\mathbf{H}\Delta\mathbf{u} + 2(\mathbf{E}\mathbf{e} - \mathbf{w})^T\mathbf{Q}_\delta\mathbf{G}\Delta\mathbf{u} \\ \text{s.a.} \quad & \mathbf{R}\Delta\mathbf{u} \leq \bar{\mathbf{r}} + \bar{\mathbf{E}}\mathbf{e}, \end{aligned} \tag{114}$$

onde a variável de realimentação $\mathbf{e}$, como uma forma de representar o estado do sistema, é composta das saídas presentes e uma série finita de entradas e saídas passadas dada por $\mathbf{e} = [y(k)\ \cdots\ y(k-n_a)\ \Delta u(k-1)\cdots\ \Delta u(k-1-n_b)]^T$. Comparando com (113), os vetores $\mathbf{b}$ e $\mathbf{r}$, os quais são funções de $\mathbf{e}$, agora são representados como $\mathbf{b}^T = 2(\mathbf{E}\mathbf{e} - \mathbf{w})^T\mathbf{Q}_\delta\mathbf{G}$ e $\mathbf{r} = \bar{\mathbf{r}} + \bar{\mathbf{E}}\mathbf{e}$. A matriz $\mathbf{E}$ pode ser obtida a partir da matriz de coeficientes das matrizes polinomiais $\mathbf{F}(z^{-1})$ e $\mathbf{S}(z^{-1})$ como $\mathbf{E} = [\mathbf{F}\ \mathbf{S}]$. As matrizes polinomiais $\mathbf{F}(z^{-1})$ e $\mathbf{S}(z^{-1})$ podem ser obtidas da solução de uma equação Diofantina como explicado em Camacho e Bordons (2004, p. 49). A matriz $\bar{\mathbf{E}}$ é utilizada

quando há restrições que dependem do estado do sistema, tais como as restrições nos sinais de controle ou de saída. Se apenas restrições nos incrementos de controle são consideradas, então $\bar{\mathbf{E}}$ é formada como uma matriz de zeros. É importante notar que as matrizes e os vetores que definem as restrições, custos e dinâmicas em (114) não variam durante a execução do controlador GPC e podem ser pré-computados de modo offline.

6.1.1 Algoritmo GPCGPAD

O método de gradiente projetado é similar ao método de gradiente descendente e pode ser utilizado para resolver problemas de otimização convexos com restrições. O método GPAD, apresentado em Patrinos e Bemporad (2014), aplica a forma acelerada dual do método de gradiente projetado.

Considerando o problema de otimização do GPC como apresentado em (113), o primeiro passo para se obter a forma dual é definir o Lagrangiano associado:

$$\mathcal{L}(\Delta\mathbf{u},\psi) = \frac{1}{2}\Delta\mathbf{u}^T\mathbf{H}\Delta\mathbf{u} + \mathbf{b}^T\Delta\mathbf{u} + \psi^T(\mathbf{R}\Delta\mathbf{u} - \mathbf{r}), \tag{115}$$

onde $\psi \in \mathbb{R}^n$ é o vetor de multiplicadores de Lagrange associados às restrições de desigualdade.

O problema de otimização dual, apresentado como uma minimização, pode ser dado como:

$$\begin{aligned} \min_{\psi} \quad & -h_d(\psi) = \frac{1}{2}\psi^T\mathbf{R}\mathbf{H}^{-1}\mathbf{R}^T\psi + \left(\mathbf{R}\mathbf{H}^{-1}\mathbf{b} + \mathbf{r}\right)^T\psi + \frac{1}{2}\mathbf{b}^T\mathbf{H}^{-1}\mathbf{b} \\ \text{s.a.} \quad & \psi \geq \mathbf{0}, \end{aligned} \tag{116}$$

o qual é um QP com dualidade forte, a partir da condição de Slater, uma vez que há um ponto estritamente factível no problema primal (BOYD; VANDENBERGHE, 2004, Sec. 5.2.3).

O passo principal do GPM aplicado em (116) se torna:

$$\psi_{j+1} = P\left(\psi_j - \frac{1}{L}\nabla h_d(\psi_j)\right), \tag{117}$$

onde j é o índice de iteração, P é uma projeção ortogonal e $L \geq 0$ é uma constante Lipschitz, a qual garante a convergência com um passo factível na direção de descenso (PATRINOS; BEMPORAD, 2014).

Uma vez que as restrições associadas com (116) são apenas $\psi \geq 0$, a projeção é equivalente à saturação de ψ para garantir valores positivos (PATRINOS; BEMPORAD, 2014), a qual pode ser representada matematicamente como:

$$\mathrm{P}(\psi) = \max(\psi, 0). \tag{118}$$

Portanto, o problema em (117) pode ser reescrito como:

$$\boldsymbol{\psi}_{j+1} = \max\left(\boldsymbol{\psi}_j - \frac{1}{L}\left(\mathbf{R}\mathbf{H}^{-1}\mathbf{R}^T\boldsymbol{\psi}_j + \mathbf{R}\mathbf{H}^{-1}\mathbf{b} + \mathbf{r}\right), \mathbf{0}\right), \tag{119}$$

o qual pode ser dividido em dois passos:

$$\begin{aligned} \Delta\mathbf{u}_j &= -\mathbf{H}^{-1}\left(\mathbf{b} + \mathbf{R}^T\boldsymbol{\psi}_j\right), \\ \boldsymbol{\psi}_{j+1} &= \max\left(\boldsymbol{\psi}_j + \frac{1}{L}\left(\mathbf{R}\Delta\mathbf{u}_j + \mathbf{r}\right), \mathbf{0}\right). \end{aligned} \tag{120}$$

O passo de aceleração foi proposto por Nesterov em Nesterov (1983) e tem sido amplamente utilizado no contexto de MPC de cômputo rápido. O ponto principal para a aceleração do método é a não observância da propriedade de sequência de relaxamento e sua substituição pela chamada sequência estimada. Dessa forma, uma operação a priori é inserida no cálculo:

$$\mathbf{w} = \boldsymbol{\psi}_j + \beta(\boldsymbol{\psi}_j - \boldsymbol{\psi}_{j-1}), \tag{121}$$

onde β é definido como $\beta = \frac{j-1}{j+2}$.

Como critério de parada do algoritmo, a factibilidade e a otimalidade primal podem ser utilizadas. Para a factibilidade primal, é possível testar se as restrições são atendidas dentro da tolerância ϵ_f, ou seja:

$$\mathbf{R}\Delta\mathbf{u}_j - \mathbf{r} \leq \epsilon_f. \tag{122}$$

Para o critério de otimalidade primal, uma propriedade derivada da dualidade forte pode ser utilizada, o que implica:

$$h(\Delta\mathbf{u}) - h(\Delta\mathbf{u}^*) \leq h(\Delta\mathbf{u}) - h_d(\boldsymbol{\psi}). \tag{123}$$

Portanto, uma solução é considerada ϵ_o-ótima se:

$$h(\Delta\mathbf{u}) - h_d(\boldsymbol{\psi}) \leq \epsilon_o, \tag{124}$$

e, então, substituindo a função $h(\Delta\mathbf{u})$, definida em (113), e a função $h_d(\boldsymbol{\psi})$, definida em (116), a solução será ϵ_o-ótima se:

$$-\boldsymbol{\psi}^T(\mathbf{R}\Delta\mathbf{u}_j - \mathbf{r}) \leq \epsilon_o. \tag{125}$$

A partir dos desenvolvimentos apresentados e do algoritmo recursivo para a obtenção das matrizes do GPC detalhadas em Peccin *et al.* (2020b), o GPCGPAD pode ser resumido no Algoritmo 11, onde j_{max} é o número máximo de iterações e $\bar{u}_i$ é um sinal de degrau unitário com $\bar{u}_i = 1$ para $i \geq 0$ e $\bar{u}_i = 0$ para $i < 0$.

Os limites de subotimalidade no número de iterações para o GPAD foram provados em Patrinos e Bemporad (2014). Se eles forem aplicados para o problema (114),

Algoritmo 11 GPCGPAD

Entrada: $A^{(1)}$, ..., $A^{(n_o)}$, $B^{(1,1)}$, ..., $B^{(n_c,n_o)}$, $\mathbf{D}$, $\mathbf{R}$, $\mathbf{r}$

Saída: $u^{(1)}(k)$, ..., $u^{(n_c)}(k)$

Dados: n_o, n_c, $\mathbf{Q}_\lambda$, $\mathbf{Q}_\delta$, $N_u^{(1)}$, ..., $N_u^{(n_c)}$, $N_1^{(1)}$, ..., $N_1^{(n_o)}$, $N_2^{(1)}$, ..., $N_2^{(n_o)}$, $\boldsymbol{\psi}^0 = \boldsymbol{\psi}^{-1} = \mathbf{0}$, $\boldsymbol{\epsilon}_\mathbf{f}$, $\boldsymbol{\epsilon}_\mathbf{o}$, $j_{\max}$

início

  para $p = 1 : n_o$ **faça**

    $\tilde{A}^{(p)}\left(z^{-1}\right) \leftarrow (1 - z^{-1})A^{(p)}\left(z^{-1}\right)$;

    para $l = 1 : n_c$ **faça**

      para $i = 1 : N_2^{(p)}$ **faça**

        $g_i^{(p,l)} \leftarrow z\left(1 - A^{(p)}(z^{-1})\right)g_{i-1}^{(p,l)} + B^{(p,l)}\left(z^{-1}\right)\bar{u}_i$;

      para $i = 1 : N_u^{(l)}$ **faça**

        $\mathbf{G}^{(p,l)}_{i:N_2^{(p)}-N_1^{(p)}+1,i} \leftarrow \mathbf{g}^{(p,l)}_{N_1^{(p)}:N_2(p)+1-i}$;

  $\mathbf{H} \leftarrow 2(\mathbf{G}^T\mathbf{Q}_\delta\mathbf{G} + \mathbf{Q}_\lambda)$;

  $\breve{\mathbf{H}} \leftarrow \mathbf{H}^{-1}$;

  $L \leftarrow \sqrt{\sum_{i=1}^{n_{rin}}\sum_{j=1}^{n_{rin}}\left|[\mathbf{R}\breve{\mathbf{H}}\mathbf{R}^T]_{ij}\right|^2}$;

  $k \leftarrow 0$;

  enquanto *modo automático* **faça**

    se *período de amostragem* **então**

      para $p = 1 : n_o$ **faça**

        Obtém as saídas $y^{(p)}(k)$ e referências $w^{(p)}(k)$;

        $f_0^{(p)} \leftarrow y^{(p)}(k)$;

        para $i = 1 : N_2^{(p)}$ **faça**

          $f_i^{(p)} \leftarrow z\left(1 - \tilde{A}^{(p)}\left(z^{-1}\right)\right)f_{i-1}^{(p)} + \sum_{l=1}^{n_c} B^{(p,l)}\left(z^{-1}\right)\Delta u^{(l)}(k - \mathbf{D}^{(p)} + i - 1)$;

      $\mathbf{f} \leftarrow [\mathbf{f}^{(1)}_{N_1^{(1)}:N_2^{(1)}} \ldots \mathbf{f}^{(n_o)}_{N_1^{(n_o)}:N_2^{(n_o)}}]^T$;

      $\mathbf{b}^T \leftarrow 2(\mathbf{f} - \mathbf{w})^T\mathbf{Q}_\delta\mathbf{G}$;

      enquanto $j < j_{\max}$ **faça**

        se $j = 0$ **então**

          $\beta \leftarrow 0$

        senão

          $\beta \leftarrow \frac{j-1}{j+2}$;

        $\mathbf{w} \leftarrow \boldsymbol{\psi}_j + \beta(\boldsymbol{\psi}_j - \boldsymbol{\psi}_{j-1})$;

        $\Delta\mathbf{u} \leftarrow -\breve{\mathbf{H}}(\mathbf{R}^T\mathbf{w} + \mathbf{b})$;

        $\mathbf{s} \leftarrow (1/L)(\mathbf{R}\Delta\mathbf{u} - \mathbf{r})$;

        se $\mathbf{s}_i \leq \epsilon_f/L,\ \forall i = 1,..,n_{rin}$ **então**

          se $-\mathbf{w}^T\mathbf{s} < \epsilon_o/L$ **então**

            retorna;

        $\boldsymbol{\psi}_{j+1} \leftarrow \max(\mathbf{w} + \mathbf{s}, \mathbf{0})$;

        $j \leftarrow j + 1$;

      $i = 1$;

      para $l = 1 : n_c$ **faça**

        $\Delta u^{(l)}(k) \leftarrow \Delta\mathbf{u}_i$;

        $i \leftarrow i + N_u^{(l)}$;

        $u^{(l)}(k) \leftarrow u^{(l)}(k-1) + \Delta u^{(l)}(k)$;

      $k \leftarrow k + 1$;

fim

considerando $\psi_0 = 0$, o limite máximo para o número de iterações pode ser escrito como:

$$N_f(\mathbf{e},\epsilon_f) = \left\lceil \sqrt{\frac{8L\|\psi^*(\mathbf{e})\|}{\epsilon_f}} \right\rceil - 2, \tag{126}$$

que garante que a solução primal média $\Delta\mathbf{u}_j$ seja factível e dentro de uma faixa de tolerância ϵ_f. Outro limite superior é:

$$N_o(\mathbf{e},\epsilon_o) = \left\lceil \sqrt{\frac{2L}{\epsilon_o}}\|\psi^*(\mathbf{e})\| \right\rceil - 2, \tag{127}$$

o qual garante que a solução primal média $\Delta\mathbf{u}_j$ seja ϵ_o-ótima.

A partir de (126) e (127) pode-se definir um limite máximo no número de iterações $N_j(\epsilon_f,\epsilon_o)$ como:

$$N_j(\epsilon_f,\epsilon_o) \geq \sup_{e\in\mathcal{E}} \max\{N_f(\mathbf{e},\epsilon_f),N_o(\mathbf{e},\epsilon_o)\}. \tag{128}$$

É importante perceber que, dado um problema do GPC, todas as quantidades para se obter o máximo número de iterações são conhecidas, exceto $\psi^*(\mathbf{e})$. O limite máximo no número de iterações para um dado poliedro de condições passadas, $\mathcal{E}$, depende do valor de $\mathbf{e}$ o qual produz a maior solução ótima dual $\psi^*(\mathbf{e})$. De modo a encontrar um limite máximo para a solução dual $\Delta_\psi(\mathcal{E})$, onde:

$$\|\psi^*(\mathbf{e})\|_1 \leq \Delta_\psi(\mathcal{E}), \tag{129}$$

é possível resolver (BEMPORAD; PATRINOS, 2012):

$$\begin{aligned} \Delta_\psi(\mathcal{E}) = &\max_{\psi,\mathbf{e}} \quad \sum_{i=1}^{n_{rin}} \psi_i \\ &\text{s.a.} \quad \mathbf{e} \in \mathcal{E}, \\ &\qquad \psi = \max\left(\psi_k - \frac{1}{L}\nabla h_d(\psi_k), 0\right). \end{aligned} \tag{130}$$

Como apresentado em Bemporad e Patrinos (2012), (130) pode ser reformulada como um problema de programação linear inteira mista (MILP, do inglês *Mixed Integer Linear Programming*) com a introdução de um vetor binário tal como $[\delta_i = 1] \leftrightarrow [\psi_k - \frac{1}{L}\nabla h_d(\psi_k) > 0]$. O MILP resultante para a obtenção do limite máximo para $\|\psi^*(\mathbf{e})\|$ do algoritmo proposto é dado por:

$$\begin{aligned} \min_{\psi,\mathbf{e}} \quad & \sum_{i=1}^{n_{rin}} -\psi_i \\ \text{s.a.} \quad & \psi_i \geq 0, \\ & \psi_i \leq \lambda_i^+ \delta_i, \\ & \frac{1}{L}\tilde{\mathbf{H}}_i\psi \leq -\frac{1}{L}(\tilde{\mathbf{b}}\mathbf{e} + \mathbf{r}) + \mu_i^+(1-\delta_i), \\ & \frac{1}{L}\tilde{\mathbf{H}}_i\psi \geq -\frac{1}{L}(\tilde{\mathbf{b}}\mathbf{e} + \mathbf{r}) + \mu_i^-(1-\delta_i). \end{aligned} \tag{131}$$

6.1.2 Algoritmo GPCFAMA

O método do algoritmo rápido de minimização alternada (FAMA) foi proposto por Goldstein *et al.* (2014) e foi originalmente desenvolvido para resolver problemas de otimização do tipo:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{z}} \quad & h(\mathbf{x}) + g(\mathbf{z}) \\ \text{s.a.} \quad & \bar{\mathbf{C}}\mathbf{x} + \bar{\mathbf{D}}\mathbf{z} = \bar{\mathbf{c}}, \end{aligned} \tag{132}$$

com $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{z} \in \mathbb{R}^m$, $\bar{\mathbf{C}} \in \mathbb{R}^{n_{req} \times n}$, $\bar{\mathbf{D}} \in \mathbb{R}^{n_{req} \times m}$, $\bar{\mathbf{c}} \in \mathbb{R}^{n_{req}}$. Assume-se a função h convexa suave e a função g convexa.

A aceleração do algoritmo é obtida a partir da sobrerelaxação do vetor de multiplicadores de Lagrange depois de cada iteração. O FAMA pode ser visto como uma formulação acelerada equivalente a aplicar o algoritmo rápido iterativo com limite de redução (FISTA, do inglês *Fast Iterative Shrinkage-thresholding Algorithm*) no problema dual (GOLDSTEIN *et al.*, 2014). Portanto, a partir da aplicação da aceleração de Nesterov na variável dual $\boldsymbol{\nu}$, o FAMA é obtido e seus passos podem ser apresentados como:

$$\begin{aligned} \mathbf{x}_{j+1} &= \text{argmin } \mathcal{L}(\mathbf{x}_j, \mathbf{z}_j, \hat{\boldsymbol{\nu}}_j) \\ \mathbf{z}_{j+1} &= \text{argmin } \mathcal{L}_\rho(\mathbf{x}_{j+1}, \mathbf{z}_j, \hat{\boldsymbol{\nu}}_j) \\ \boldsymbol{\nu}_{j+1} &= \boldsymbol{\nu}_j + \rho\left(\bar{\mathbf{C}}\mathbf{x}_{j+1} + \bar{\mathbf{D}}\mathbf{z}_{j+1} - \bar{\mathbf{c}}\right) \\ \theta_{k+1} &= \left(1 + \sqrt{4(\theta_k)^2 + 1}\right)/2 \\ \hat{\boldsymbol{\nu}}_{j+1} &= \boldsymbol{\nu}_{j+1} + (\theta_k - 1)(\boldsymbol{\nu}_{j+1} - \boldsymbol{\nu}_j)/\theta_{k+1}. \end{aligned} \tag{133}$$

De modo a escrever o problema de otimização (113) na forma de (132) para que os resultados do FAMA possam ser utilizados, o primeiro passo proposto é representar as restrições de desigualdade utilizando uma função indicadora $I\_(x)$. Definindo $h(\Delta\mathbf{u}) = \frac{1}{2}\Delta\mathbf{u}^T\mathbf{H}\Delta\mathbf{u} + \mathbf{b}^T\Delta\mathbf{u}$, (113) pode ser reescrita como:

$$\min_{\Delta\mathbf{u}} \quad h(\Delta\mathbf{u}) + I\_(\mathbf{R}\Delta\mathbf{u} - \mathbf{r}). \tag{134}$$

Com a definição de uma nova variável $\mathbf{z} = \mathbf{R}\Delta\mathbf{u} - \mathbf{r}$ e uma nova restrição de igualdade para garantir o acoplamento entre as variáveis, tem-se:

$$\begin{aligned} \min_{\Delta\mathbf{u},\mathbf{z}} \quad & h(\Delta\mathbf{u}) + I\_(\mathbf{z}) \\ \text{s.a.} \quad & \mathbf{R}\Delta\mathbf{u} - \mathbf{z} = \mathbf{r}. \end{aligned} \tag{135}$$

A partir do problema padrão (135), o Lagrangiano $\mathcal{L}$ e o Lagrangiano aumentado $\mathcal{L}_\rho$ podem ser definidos como:

$$\mathcal{L}(\Delta\mathbf{u}, \mathbf{z}, \boldsymbol{\nu}) = h(\Delta\mathbf{u}) + I\_(\mathbf{z}) + \boldsymbol{\nu}^T(\mathbf{R}\Delta\mathbf{u} - \mathbf{z} - \mathbf{r})$$

$$\mathcal{L}_\rho(\Delta\mathbf{u}, \mathbf{z}, \boldsymbol{\nu}) = h(\Delta\mathbf{u}) + I\_(\mathbf{z}) + \boldsymbol{\nu}^T(\mathbf{R}\Delta\mathbf{u} - \mathbf{z} - \mathbf{r}) + \frac{\rho}{2}\|\mathbf{R}\Delta\mathbf{u} - \mathbf{z} - \mathbf{r}\|^2.$$

Com o Lagrangiano e os passos básicos do FAMA (133), GPCFAMA pode ser computado recursivamente como:

$$\Delta\mathbf{u}_{j+1} = \text{argmin}\left(\Delta\mathbf{u}_j^T\mathbf{H}\Delta\mathbf{u}_j + (\mathbf{b}^T + \hat{\nu}_j^T\mathbf{R})\Delta\mathbf{u}_j\right) \tag{136}$$

$$\mathbf{z}_{j+1} = \text{argmin}\left(I\_(\mathbf{z}) - \hat{\nu}_j^T\mathbf{z}_j + \frac{\rho}{2}\|\mathbf{R}\Delta\mathbf{u}_{j+1} - \mathbf{z}_j - \mathbf{r}\|^2\right) \tag{137}$$

$$\boldsymbol{\nu}_{j+1} = \boldsymbol{\nu}_j + \rho(\mathbf{R}\Delta\mathbf{u} - \mathbf{z} - \mathbf{r}) \tag{138}$$

$$\theta_{k+1} = (1 + \sqrt{4(\theta_k)^2 + 1})/2 \tag{139}$$

$$\hat{\nu}_{j+1} = \boldsymbol{\nu}_{j+1} + (\theta_k - 1)(\boldsymbol{\nu}_{j+1} - \boldsymbol{\nu}_j)/\theta_{k+1}. \tag{140}$$

O passo (136) pode ser obtido analiticamente de forma direta:

$$\Delta\mathbf{u}_{j+1} = -\mathbf{H}^{-1}(\mathbf{b} + \mathbf{R}^T\hat{\nu}_j).$$

Em um problema GPC típico, a Hessiana $\mathbf{H}$ pode ser computada de modo offline e a sua inversa pode ser armazenada na memória. Portanto, esse passo é obtido com uma simples multiplicação matriz vetor.

Para computar o passo (137), percebe-se que este tipo de problema pode ser reescrito como um operador proximal (STATHOPOULOS *et al.*, 2016). O operador proximal de uma função escalada $\rho h(x)$ é definido como:

$$\text{prox}_{\rho h}(x) = \text{argmin}_z\left(h(z) + \frac{1}{2\rho}\|z - x\|^2\right). \tag{141}$$

Portanto, aplicando a definição de operador proximal na função indicadora $I\_(z)$ e com um passo de tamanho ρ, o cômputo de (137) pode ser representado como:

$$\mathbf{z}_{j+1} = \text{prox}_{\rho I\_}(\mathbf{R}\Delta\mathbf{u}_{j+1} - \mathbf{r} + (1/\rho)\hat{\nu}_j). \tag{142}$$

Dado que $I\_(\mathbf{z})$ é uma função indicadora no ortante negativo $\mathbb{R}^n_-$, o qual é um conjunto convexo fechado e não vazio, o operador proximal se torna uma projeção sobre $\mathbb{R}^n_-$ (COMBETTES; PESQUET, 2011). Portanto, (142) pode ser obtido com a seleção da parte negativa de cada elemento ou zero:

$$\mathbf{z}_{j+1} = \min(\mathbf{R}\Delta\mathbf{u}_{j+1} - \mathbf{r} + (1/\rho)\hat{\nu}_j, \mathbf{0}).$$

Os residuais das condições de otimalidade podem ser utilizados como critério de parada. O resíduo primal é dado por:

$$r_{j+1} = \mathbf{R}\Delta\mathbf{u}_{j+1} - \mathbf{z}_{j+1} - \mathbf{r},$$

e o resíduo dual é escrito como:

$$s_{j+1} = \rho\mathbf{R}^T(\mathbf{z}_{j+1} - \mathbf{z}_j).$$

Então, um critério de parada razoável é que os resíduos primal e dual devem ser menores do que uma dada tolerância desejada:

$$\|\mathbf{r}_j\| \leq \epsilon_p \quad \text{e} \quad \|\mathbf{s}_j\| \leq \epsilon_d. \tag{143}$$

Para garantir a convergência no objetivo dual com taxa $O(1/k^2)$, o parâmetro ρ deve ser $\rho < \sigma_H/\lambda_{\max}(\mathbf{R}^T\mathbf{R})$ com σ_H sendo a constante de convexidade de $\mathbf{H}$ e $\lambda_{\max}$ o maior autovalor da matriz (GOLDSTEIN *et al.*, 2014). Assumindo que $\mathbf{H}$ é fortemente convexa, ρ pode ser obtida como $\rho = \lambda_{\min}(\mathbf{H})/\lambda_{\max}(\mathbf{R}^T\mathbf{R})$.

A partir dos desenvolvimentos apresentados e do algoritmo recursivo para se obter as matrizes do GPC detalhados em Peccin *et al.* (2020b), GPCFAMA pode ser resumido no Algoritmo 12.

A taxa de convergência para o FAMA foi provada em Goldstein *et al.* (2014) e é dada por $O(1/k^2)$ para o custo dual $f_d(\psi^*) - f_d(\psi_j)$. A partir do Teorema 5 em Goldstein *et al.* (2014) o limite de subotimalidade para o FAMA o qual provê um limite superior na função custo dual $N_d(\mathbf{e},\epsilon_d)$, para $\nu_0 = \mathbf{0}$, é

$$N_d(\mathbf{e},\epsilon_d) = \left\lceil \sqrt{\frac{2\lambda_{\max}(\mathbf{R}^T\mathbf{R})}{\lambda_{\min}(\mathbf{H})\epsilon_d}}\|\nu^*(\mathbf{e})\| \right\rceil - 1, \tag{144}$$

o que garante que a solução $\boldsymbol{\nu}_j$ é ótima dentro de uma faixa de valores desejada ϵ_d.

No Teorema 5.3 de Pu *et al.* (2017), a taxa de convergência para a distância Euclidiana quadrática na variável primal $\|\Delta\mathbf{u}_j - \Delta\mathbf{u}^*\|^2$ foi apresentada como $O(1/k^2)$. Portanto, outro limite superior no número de iterações que pode ser utilizado é $N_p(\mathbf{e},\epsilon_p)$, definido como:

$$N_p(\mathbf{e},\epsilon_p) = \left\lceil \frac{2\sqrt{\lambda_{\max}(\mathbf{R}^T\mathbf{R})}\|\nu^*(\mathbf{e})\|}{\lambda_{\min}(\mathbf{H})\epsilon_p} \right\rceil, \tag{145}$$

o que garante que a distância Euclidiana quadrática na variável primal é ótima dentro de uma faixa de valores desejada ϵ_p.

O limite superior de iterações depende de $\mathbf{e}$ e da solução ótima dual $\nu^*(\mathbf{e})$. De modo a encontrar um limite superior de iterações para o pior caso de $\nu^*(\mathbf{e})$ com $\mathbf{e}$ em um dado poliedro $\mathcal{E}$, duas abordagens são propostas em Pu *et al.* (2017). A primeira delas utiliza a soma de relaxações quadradas e a segunda uma estimação baseada em amostras. A abordagem de estimação baseada em amostras pode ser facilmente aplicada para problemas de todas as dimensões. O procedimento é escolher um parâmetro de confiança $\beta \in [0,1]$, um parâmetro de nível $\epsilon \in [0,1]$ e uma quantidade N_s satisfazendo $N_s \geq \frac{1}{\epsilon\beta} - 1$. O próximo passo é selecionar aleatoriamente N_s amostras $\{\mathbf{e}_1, ..., \mathbf{e}_{N_s}\} \in \mathcal{E}$ e rodar o Algoritmo 12 para calcular os correspondentes $\{\|\nu^*(\mathbf{e}_1)\|,...,\|\nu^*(\mathbf{e}_{N_s})\|\}$. Finalmente, é possível computar $\max_{1\leq i\leq N_s}\{\|\nu^*(\mathbf{e}_i)\|\}$, o qual é um limite superior para encontrar uma solução ótima com nível ϵ de factibilidade para $\nu^*(\mathbf{e})$.

Algoritmo 12 GPCFAMA

Entrada: $A^{(1)}$, ..., $A^{(n_o)}$, $B^{(1,1)}$, ..., $B^{(n_c,n_o)}$, $\mathbf{D}$, $\mathbf{R}$, $\mathbf{r}$

Saída: $u^{(1)}(k)$, ..., $u^{(n_c)}(k)$

Dados: n_o, n_c, $\mathbf{Q}_\lambda$, $\mathbf{Q}_\delta$, $N_u^{(1)}$, ..., $N_u^{(n_c)}$, $N_1^{(1)}$, ..., $N_1^{(n_o)}$, $N_2^{(1)}$, ..., $N_2^{(n_o)}$, $\rho = \lambda_{\min}(\mathbf{H})/\lambda_{\max}(\mathbf{R}^T\mathbf{R})$, $\theta_0 = 1$, $\theta_1 = (1+\sqrt{5})/2$, $\epsilon_p > 0$, $\epsilon_d > 0$, $j_{\max}$

início

  para $p = 1 : n_o$ **faça**

    $\tilde{A}^{(p)}\left(z^{-1}\right) \leftarrow (1-z^{-1})A^{(p)}\left(z^{-1}\right)$;

    para $l = 1 : n_c$ **faça**

      para $i = 1 : N_2^{(p)}$ **faça**

        $g_i^{(p,l)} \leftarrow z\left(1 - A^{(p)}(z^{-1})\right)g_{i-1}^{(p,l)} + B^{(p,l)}\left(z^{-1}\right)\bar{u}_i$;

      para $i = 1 : N_u^{(l)}$ **faça**

        $\mathbf{G}^{(p,l)}_{i:N_2^{(p)}-N_1^{(p)}+1,i} \leftarrow \mathbf{g}^{(p,l)}_{N_1^{(p)}:N_2(p)+1-i}$;

  $\mathbf{H} \leftarrow 2(\mathbf{G}^T\mathbf{Q}_\delta\mathbf{G} + \mathbf{Q}_\lambda)$;

  $\tilde{\mathbf{H}} \leftarrow \mathbf{H} + \mathbf{R}^T\rho\mathbf{R}$;

  $\breve{\mathbf{H}} \leftarrow \tilde{\mathbf{H}}^{-1}$;

  $k \leftarrow 0$;

  enquanto *modo automático* **faça**

    se *período de amostragem* **então**

      para $p = 1 : n_o$ **faça**

        Obtém as saídas$y^{(p)}(k)$ e referências $w^{(p)}(k)$;

        $f_0^{(p)} \leftarrow y^{(p)}(k)$;

        para $i = 1 : N_2^{(p)}$ **faça**

          $f_i^{(p)} \leftarrow z\left(1 - \tilde{A}^{(p)}\left(z^{-1}\right)\right)f_{i-1}^{(p)} + \sum_{l=1}^{n_c} B^{(p,l)}\left(z^{-1}\right)\Delta u^{(l)}(k - \mathbf{D}^{(p)} + i - 1)$;

      $\mathbf{f} \leftarrow [\mathbf{f}^{(1)}_{N_1^{(p)}:N_2^{(p)}} \ldots \mathbf{f}^{(n_o)}_{N_1^{(p)}:N_2^{(p)}}]^T$;

      $\mathbf{b}^T \leftarrow 2(\mathbf{f} - \mathbf{w})^T\mathbf{Q}_\delta\mathbf{G}$;

      $\mathbf{z}_0 \leftarrow \mathbf{R}\Delta\mathbf{u}_0 - \mathbf{r}$;

      $j \leftarrow 0$;

      enquanto $j < j_{\max}$ **faça**

        $\Delta\mathbf{u}_{j+1} \leftarrow -\breve{\mathbf{H}}(\mathbf{b} + \mathbf{R}^T\hat{\boldsymbol{\nu}}_j)$;

        $\mathbf{z}_{j+1} \leftarrow \min(\mathbf{R}\Delta\mathbf{u}_{j+1} - \mathbf{r} + (1/\rho)\hat{\boldsymbol{\nu}}_j, \mathbf{0})$;

        $\boldsymbol{\nu}_{j+1} \leftarrow \boldsymbol{\nu}_j + \rho(\mathbf{R}\Delta\mathbf{u} - \mathbf{z} - \mathbf{r})$;

        se $\|\mathbf{R}\Delta\mathbf{u} - \mathbf{z}_{j+1} - \mathbf{r}\|_2 \leq \epsilon_p$ ***e*** $\|\rho\mathbf{R}^T(\mathbf{z}_{j+1} - \mathbf{z}_j)\|_2 \leq \epsilon_d$ **então**

          retorna;

        $\theta_{k+1} \leftarrow (1 + \sqrt{4(\theta_k)^2 + 1})/2$;

        $\hat{\boldsymbol{\nu}}_{j+1} \leftarrow \boldsymbol{\nu}_{j+1} + (\theta_k - 1)(\boldsymbol{\nu}_{j+1} - \boldsymbol{\nu}_j)/\theta_{k+1}$;

        $j \leftarrow j + 1$;

      $i \leftarrow 1$;

      para $l = 1 : n_c$ **faça**

        $\Delta u^{(l)}(k) \leftarrow \Delta\mathbf{u}_i$;

        $i \leftarrow i + N_u^{(l)}$;

        $u^{(l)}(k) \leftarrow u^{(l)}(k-1) + \Delta u^{(l)}(k)$;

      $k \leftarrow k + 1$;

fim

6.2 QUESTÕES DE IMPLEMENTAÇÃO

Esta seção descreve alguns detalhes de implementação para os algoritmos propostos, os quais podem ser utilizados para acelerar o tempo de cômputo.

6.2.1 Multiplicação matricial paralela

Uma das operações mais custosas no tempo de computação dos algoritmos propostos é a multiplicação de matrizes. Algoritmos simples, para matrizes densas e apenas um processador, são executados em três laços aninhados. A utilização de computação paralela pode melhorar significativamente o tempo requerido para executar essa operação. Neste trabalho é utilizada uma abordagem apresentada em Peccin *et al.* (2020b) (Seção 5.2.2).

6.2.2 Arquitetura de implementação GPCGPAD

Uma máquina de estados finita foi proposta para implementar o algoritmo GPCGPAD em uma arquitetura paralela. A solução foi centrada no controle sequencial de ações em uma máquina de estados, dessa forma os resultados intermediários da computação são coordenados enquanto as operações de adição, subtração e multiplicação matricial executam em paralelo. Na Figura 10 a implementação da máquina de estados e as operações correspondentes são representadas esquematicamente. Pode-se perceber que as operações de multiplicação requerem 2 pulsos de relógio, assumindo todas as matrizes de dimensão 10×10, enquanto as somas e subtrações são executadas em um único pulso. No estado 11, o critério de parada é checado e, se for satisfeito, depois do passo 12, o processo é então terminado, sinalizando que o sinal de incremento de controle foi encontrado. Nos estados 2 e 9, a multiplicação é feita em apenas um pulso de relógio, uma vez que são multiplicações de vetores por escalares. Portanto, é possível estimar o tempo de execução do algoritmo como:

$$t = T_{clock}(2 + n_{iter}(8 + 2(\lceil \log_{10}(N_u) \rceil + 1) + 2(\lceil \log_{10}(n_{rin}) \rceil + 1)), \tag{146}$$

onde T_{clock} é o período de relógio utilizado pelo hardware, n_{iter} é o número de iterações requeridas pelo algoritmo para encontrar o critério de parada, N_u é o horizonte de controle e n_{rin} é o número de restrições de desigualdade.

6.2.3 Arquitetura de implementação GPCFAMA

A implementação paralela do GPCFAMA utiliza a mesma estratégia do GPCGPAD e pode ser vista na Figura 11. Nesse caso, três multiplicações matriciais dependendo de n_{rin} são requiridas e apenas uma dependentes de N_u. No estado 12, o critério de parada é verificado e, se atendido, o processo é terminado e o sinal de incremento

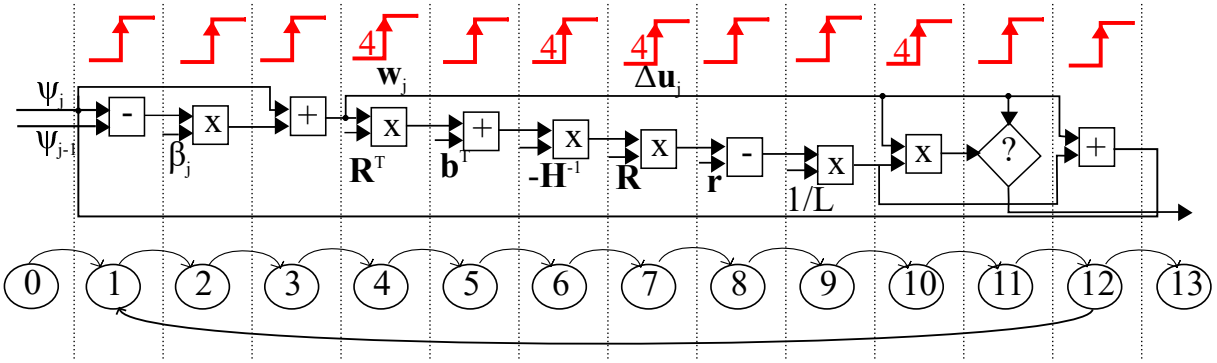


Figura 10 – Arquitetura de implementação do GPCGPAD em uma máquina de estados finitos para um FPGA.

de controle é informado. É possível estimar o tempo de execução do algoritmo como:

$$t = T_{clock}(2 + n_{iter}(8 + (\lceil \log_{10}(N_u) \rceil + 1) + 3(\lceil \log_{10}(n_{rin}) \rceil + 1))), \tag{147}$$

com as mesmas definições de (146).

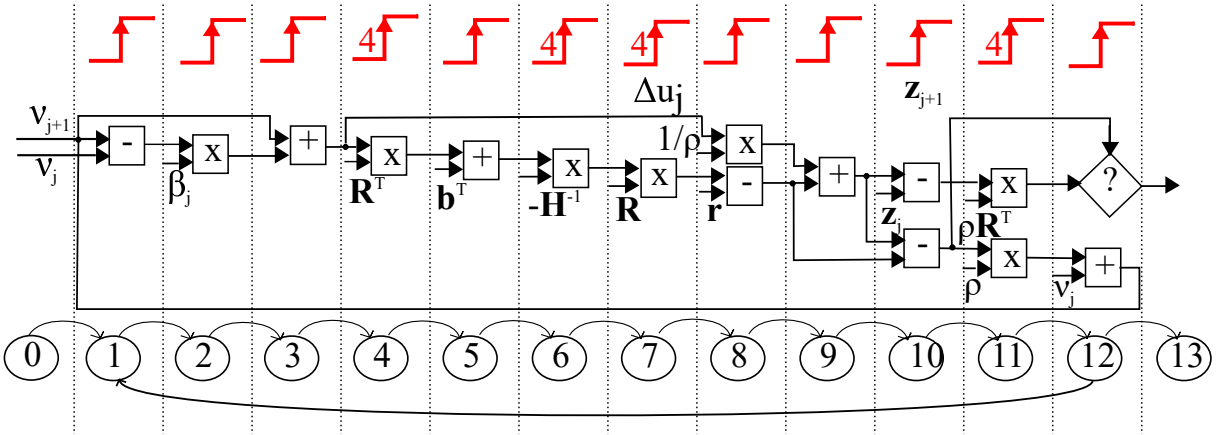


Figura 11 – Arquitetura de implementação GPCFAMA em uma máquina de estados finitos para um FPGA.

6.2.4 Precondicionamento

Um fato bastante conhecido para os métodos de primeira ordem é que o precondicionamento pode oferecer uma aceleração significativa na taxa de convergência (PU *et al.*, 2017; PATRINOS; BEMPORAD, 2014). Em Pu *et al.* (2017), um precondicionamento ótimo é proposto para o FAMA para resolver problemas MPC com restrições politópicas. A ideia principal é aumentar o tamanho do passo do algoritmo a partir da diminuição de $\lambda_{\max}(\mathbf{R}^T\mathbf{R})$. Neste trabalho, uma abordagem simples de precondicionamento é apresentada. O primeiro passo é obter a decomposição Cholesky da

Hessiana $\mathbf{H} = \mathbf{K}\mathbf{K}^T$, a qual permite que (16) seja reescrita como:

$$\begin{aligned} \min_{\Delta\mathbf{u}} \quad & \frac{1}{2}\Delta\mathbf{u}^T\mathbf{K}\mathbf{K}^T\Delta\mathbf{u} + \mathbf{b}^T\Delta\mathbf{u} \\ \text{s.a.} \quad & \mathbf{R}\Delta\mathbf{u} \leq \mathbf{r}. \end{aligned} \tag{148}$$

A partir de uma substituição de variáveis, $\Delta\acute{\mathbf{u}} = \mathbf{K}^T\Delta\mathbf{u}$, um problema remodelado é obtido:

$$\begin{aligned} \min_{\Delta\acute{\mathbf{u}}} \quad & \frac{1}{2}\Delta\acute{\mathbf{u}}^T\Delta\acute{\mathbf{u}} + \mathbf{b}^T\mathbf{K}^T\Delta\acute{\mathbf{u}} \\ \text{s.a.} \quad & \mathbf{R}\mathbf{K}^T\Delta\acute{\mathbf{u}} \leq \mathbf{r}. \end{aligned} \tag{149}$$

Apesar de esse precondicionamento não ser ótimo, ele aumenta o tamanho do passo para alguns casos, diminuindo o número de iterações. Além disso, a Hessiana se torna uma matriz identidade, portanto as multiplicações com $\mathbf{H}$ nos algoritmos (passo 6 na Figura 10 e Figura 11) não precisam ser implementadas, o que se traduz em um decréscimo no tempo de computação. Como um exemplo, em um problema com um horizonte de controle N_u = 10 e com 10 restrições (n_{rin} = 10), o tempo de computação pode ser reduzido em aproximadamente 11% por iteração de acordo com as equações (146) e (147). É importante enfatizar que a decomposição Cholesky pode ser computada offline, uma vez que a matriz $\mathbf{H}$ é constante.

6.3 ESTUDO DE CASO SIMULADO

O estudo de caso considerado neste capítulo é um inversor trifásico com filtro LCL e conectado à rede controlando a potência injetada ou consumida da rede. Esse tipo de sistema tem um papel fundamental nas aplicações de energias renováveis e o circuito esquemático é apresentado em Figura 12. Os controladores PI são amplamente utilizados em coordenadas *dq* rotacionais síncronas para regular os componentes *d* e *q* das correntes. Entretanto, foi reportado na literatura que um controlador PI orientado à tensão, para esse tipo de aplicação, perde sua estabilidade quando a largura de banda do controlador é alta (KUKRER *et al.*, 2019). Alternativamente, uma estratégia de controle proporcional ressonante (PR) modelada em coordenadas estacionárias $\alpha\beta$ é uma solução, mas o desempenho do controlador PR é sensível a variações de frequência na rede (KUKRER *et al.*, 2019). A maioria das aplicações MPC em conversores de potência utiliza o chamado MPC com conjunto de controle finito (FCS-MPC, do inglês *Finite Control Set-MPC*) (LIM *et al.*, 2019; JIN *et al.*, 2019). No FCS-MPC, as possíveis ações de controle são finitas e são representadas pelos estados de chaveamento. Entretanto, o FCS-MPC tem uma frequência de chaveamento variável, o que tem potencial para trazer problemas na qualidade do sinal gerado (MACCARI JR *et al.*, 2020). Além disso, os algoritmos FCS-MPC utilizam métodos de busca para computar a otimização e, portanto, apenas horizontes de predição muito

pequenos podem ser utilizados (geralmente 1 ou 2). Outro método é o MPC com conjunto contínuo de controle (CCS-MPC, do inglês *Continuous Control Set-MPC*), o qual tem uma frequência de chaveamento fixa. O CCS-MPC tem poucas aplicações em controladores de conversores de potência, tais como os apresentados em Guzman *et al.* (2019), Maccari Jr *et al.* (2020) e Cavanini *et al.* (2017), devido ao número de cálculos necessários no problema de otimização com restrições de forma online, o qual é incompatível com períodos de amostragem pequenos. Portanto, este estudo de caso foi escolhido para mostrar como os algoritmos propostos podem ser implementados em tempo real e contribuir para a obtenção de resultados satisfatórios e com um pequeno tempo de computação. As equações dinâmicas do sistema (Figura 12) nas

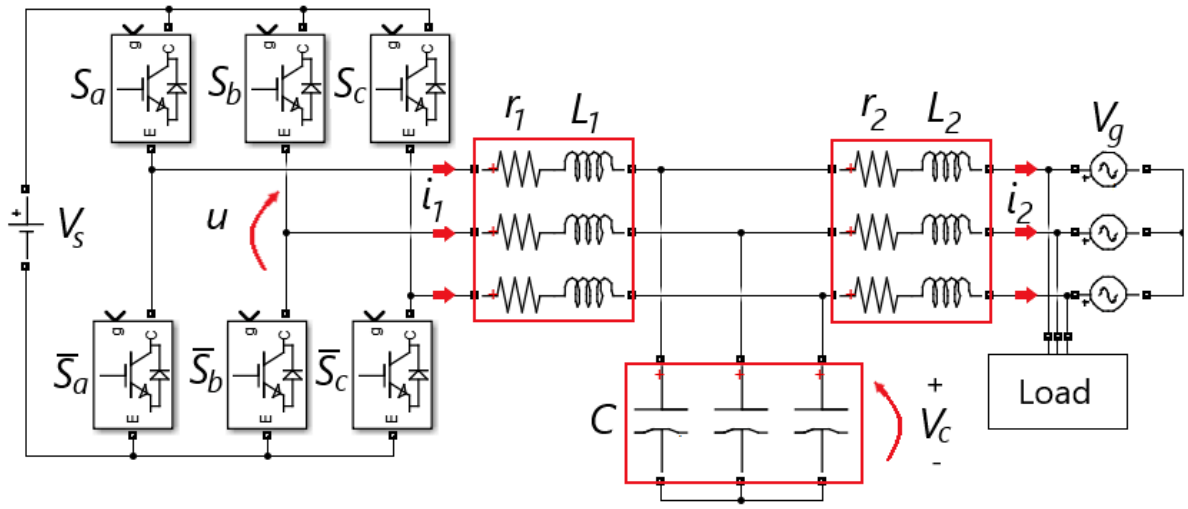


Figura 12 – Inversor trifásico com filtro LCL e conectado à rede.

coordenadas *dq* rotacionais síncronas são:

$$
\begin{aligned}
\frac{di_{1d}}{dt} &= -\frac{r_1}{L_1} i_{1d} - \frac{1}{L_1} v_{cd} + \omega i_{1q} + \frac{1}{L_1} u_d, \\
\frac{dv_{cd}}{dt} &= \frac{1}{C} i_{1d} - \frac{1}{C} i_{2d} + \omega v_{cq}, \\
\frac{di_{2d}}{dt} &= -\frac{r_2}{L_2} i_{1d} + \frac{1}{L_2} v_{cd} + \omega i_{2q} - \frac{1}{L_2} v_{gd}, \\
\frac{di_{1q}}{dt} &= -\frac{r_1}{L_1} i_{1q} - \frac{1}{L_1} v_{cq} - \omega i_{1d} + \frac{1}{L_1} u_q, \\
\frac{dv_{cq}}{dt} &= \frac{1}{C} i_{1q} - \frac{1}{C} i_{2q} - \omega v_{cd}, \\
\frac{di_{2q}}{dt} &= -\frac{r_2}{L_2} i_{1q} + \frac{1}{L_2} v_{cq} - \omega i_{2d} - \frac{1}{L_2} v_{gq},
\end{aligned}
\tag{150}
$$

onde os componentes dos eixos *d* e *q*, $\{i_{1d}, i_{1q}\}$, e $\{i_{2d}, i_{2q}\}$ são as correntes do inversor, e as correntes da rede, respectivamente. Os componentes de tensão no capacitor são $\{v_{cd}, v_{cq}\}$. As tensões na rede são representadas por $\{v_{gd}, v_{gq}\}$ e podem ser assumidas

como perturbações para o problema de controle. Os componentes $\{u_d, u_q\}$ denotam as tensões antes do filtro LCL e são as variáveis manipuladas. A variável ω representa a frequência da rede (em radianos por segundo) e os sinais na porta do inversor são representados por $\{S_a, S_b, S_c, \bar{S}_a, \bar{S}_b, \bar{S}_c\}$.

A partir de (150), a matriz de transferência equivalente em tempo discreto utilizada foi:

$$\mathbf{G}_{dq}(z) = \begin{bmatrix} G_d(z) & G_q(z) \\ -G_q(z) & G_d(z) \end{bmatrix},$$

onde

$$G_d(z) = \frac{0.0129z^5 + 0.0333z^4 - 0.0148z^3 + 0.0150z^2 - 0.0329z - 0.0127}{z^7 - 1.741z^6 + 2.497z^5 - 3.485z^4 + 2.48z^3 - 1.725z^2 + 0.9802z},$$
$$G_q(z) = \frac{0.0004z^5 + 0.0026z^4 - 0.0002z^3 - 0.0002z^2 + 0.0026z + 0.0004}{z^7 - 1.741z^6 + 2.497z^5 - 3.485z^4 + 2.48z^3 - 1.725z^2 + 0.9802z}.$$

As funções de transferência foram obtidas a partir da transformada de Laplace das equações dinâmicas. A discretização foi feita assumindo um modelo de sustentador de ordem zero para as entradas e o período de amostragem foi escolhido como T_s = 100 μs. O período de amostragem é o mesmo período de chaveamento do modulador por largura de pulso (PWM, do inglês *Pulse Width Modulation*). Portanto, um tempo de atraso de uma amostra foi incluída no modelo para compensar o tempo de atraso de computação (JIN *et al.*, 2019).

Uma simulação do sistema de controle de corrente do inversor trifásico foi implementada em Simulink com a biblioteca SimPowerSystems. O sinal de controle computado, convertido em razão cíclica, foi utilizado em uma técnica de PWM baseada em portadora. Além disso, uma ponte de dois níveis com transistores bipolares de porta isolada (IGBTs, do inglês *Insulated Gate Bipolar Transistors*) foi implementada. Um laço travado em fase (PLL, do inglês *Phase Locked Loop*) foi utilizado para gerar as referências de senos e cossenos a partir das tensões da rede. Os parâmetros do circuito simulado são apresentados na Tabela 5, onde os parâmetros do filtro LCL são os mesmos utilizados em Kukrer *et al.* (2019).

A corrente medida na carga foi utilizada como uma referência para o sistema de controle. Portanto, o inversor provê a potência requerida pela carga. Uma carga de 5 kW foi utilizada no começo da simulação e, no instante de tempo t = 0,1 s, outra carga de 5 kW com uma porção reativa de 1 kVAr foi acionada. O tempo simulado foi de 150 ms. O computador utilizado para a simulação tem um processador Intel Core i3 de 2,40 GHz e 12 GB de memória RAM. Os tempos de computação foram medidos através do comando *tic-toc* do MATLAB. Os controladores propostos foram sintonizados com os horizontes de controle $N_u^{(1)} = N_u^{(2)} = N_u = 5$, os horizontes de predição $N_1^{(1)} = N_1^{(2)} = N_1 = 1$ e $N_2^{(1)} = N_2^{(2)} = N_2 = 20$, as ponderações no esforço de controle $\lambda^{(1)} = \lambda^{(2)} = \lambda = 15$, as ponderações no erro $\delta^{(1)} = \delta^{(2)} = \delta = 1$ e referências futuras conhecidas $w^{(1)}(t+j)$ e $w^{(2)}(t+j)$. As ponderações e horizontes

Tabela 5 – Parâmetros do circuito inversor de frequência trifásico com filtro LCL simulado.

Barramento CC	V_s	400 V
Tensão da rede	V_{RMS}	220 V
Frequência da rede	f	60 Hz
Filtro LCL	L_1	1,5 mH
	r_1	0,1 Ω
	L_2	0,5 mH
	r_2	0,05 Ω
	C	5 μF
Frequência Chaveamento PWM		10 kHz
Potência ativa carga constante		5 kW
Potência ativa carga chaveada		5 kW
Potência reativa carga chaveada		1 kVAr

foram definidos com valores iguais pois o fator de importância e as dinâmicas das variáveis são equivalentes.

No referencial *abc*, o sinal de controle pode variar entre $[-V_s, +V_s]$, o qual pode ser facilmente representado como uma restrição linear de desigualdade na forma $\mathbf{R}\Delta\mathbf{u} \leq \mathbf{r}$. Entretanto, em coordenadas *dq*, essa restrição se torna uma desigualdade não linear do tipo: $\sqrt{u_d^2 + u_q^2} \leq \frac{V_s}{\sqrt{3}}$. Para simplificar o cômputo, a seguinte aproximação é proposta:

$$u_d(k+j)u_d(k-1) + u_q(k+j)u_q(k-1) \leq \frac{V_s^2}{3},$$

com $j = 0,...,\frac{N_u}{2} - 1$. Essa é uma aproximação válida se o valor do sinal de controle não muda muito entre uma amostra e outra subsequente. O problema de otimização resulta em uma matriz $\mathbf{H} \in \mathbb{R}^{10\times10}$ e 10 restrições. Os algoritmos propostos foram comparados com o otimizador de ponto interior do MATLAB (Quadprog) e com o otimizador comercial de uso geral chamado Gurobi. Os mesmos parâmetros dos controladores, horizontes e função custo foram utilizados para todos os algoritmos. O otimizador Gurobi foi testado com o algoritmo simplex dual (GurobiSD) e com o método de barreira (GurobiBar) (GUROBI OPTIMIZATION, 2021). O GPCGPAD foi sintonizado para uma tolerância no residual primal $\epsilon_o = 0{,}001$ e tolerância de factibilidade $\epsilon_f = 0{,}001$. O GPCFAMA foi sintonizado com as mesmas tolerâncias residuais primal dadas por $\epsilon_p = \epsilon_d = 0{,}001\sqrt{N_u}$. As tolerâncias para o GUROBI e Quadprog foram também trocadas para 0,001.

Os resultados podem ser vistos na Figura 13, onde a corrente de saída do conversor e o sinal de controle aplicado são apresentados nas coordenadas do referencial *abc*. Na Figura 14 os componentes *dq* das saídas de corrente do conversor e o sinal de controle são apresentados. Pode também ser visto nessa figura que ambas as referên-

cias são seguidas e que as restrições estão ativas limitando o sinal de controle quando há a troca de referências. O comportamento do sistema de controle foi apresentado somente uma vez porque é o mesmo para todos os algoritmos.

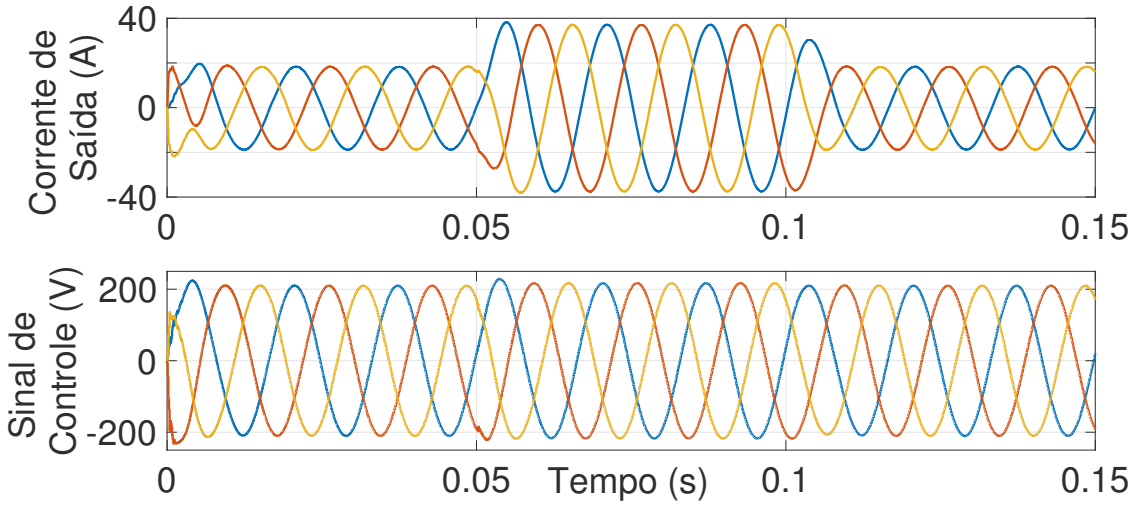
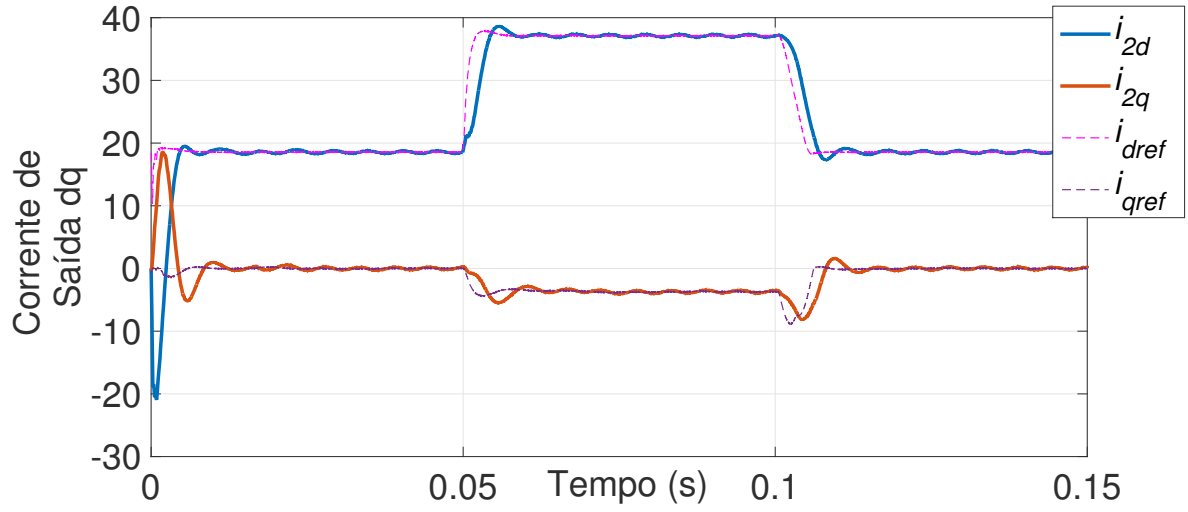


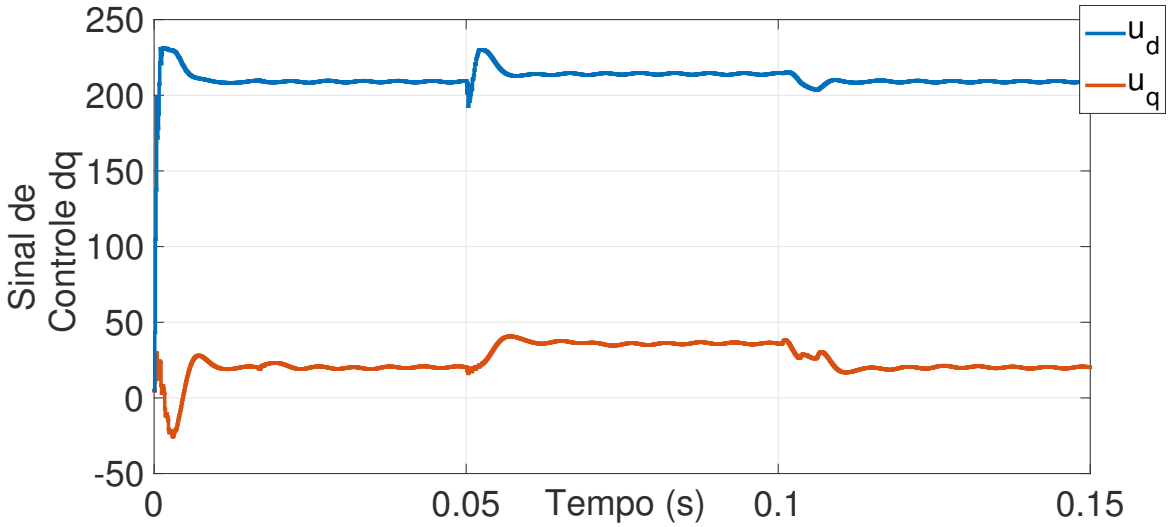
Figura 13 – Correntes de saída do inversor e tensão do sinal de controle nas coordenadas abc.

Uma estratégia MPC sem restrições baseada em espaço de estados (uMPC) (MACCARI JR *et al.*, 2020) e um regulador quadrático linear (LQR, do inglês *Linear Quadratic Regulator*) (MOUTON; GEYER, 2018), as quais são abordagens encontradas na literatura, foram também simuladas. Ambas as estratégias utilizaram modelos em espaço de estados em coordenadas *dq* e o modelo nominal foi aumentado com dois modos integradores para que o sistema em malha fechada pudesse rastrear a referência ao degrau de entrada em cada componente *dq*, resultando em um modelo com 10 estados. Uma técnica *anti-windup* foi utilizada em ambos controladores para evitar problemas de saturação relacionados aos modos integradores. Os perfis dos sinais de controle são apresentados na Figura 15 para o uMPC e na Figura 16 para o LQR. O índice de desempenho utilizado para comparação foi a raiz do erro quadrático médio (RMSE, do inglês *Root Mean Square Error*). O rastreamento de referência, em coordenadas *dq*, para a entrada ao degrau foi analisado e os resultados podem ser vistos na Tabela 6. O RMSE obtido para o GPC pode ser considerado equivalente aos obtidos com o uMPC, com o componente do eixo *d* sendo 21% menor para o GPC e o componente do eixo *q* sendo 13% menor para o uMPC. Além disso, o RMSE obtido a partir do GPC foi 2,19 vezes menor no componente do eixo *d* do que o RMSE do LQR e ambos apresentaram aproximadamente o mesmo valor para o componente do eixo *q*. Adicionalmente aos bons resultados obtidos, o GPC com restrições tem a vantagem de utilizar menos sensores e implicitamente evitar problemas de *windup*.

Uma comparação entre as potências ativa e reativa na saída do inversor e na carga é apresentada na Figura 17. Um rastreamento da potência da carga pode ser



(a) Correntes de saída



(b) Tensões do sinal de controle

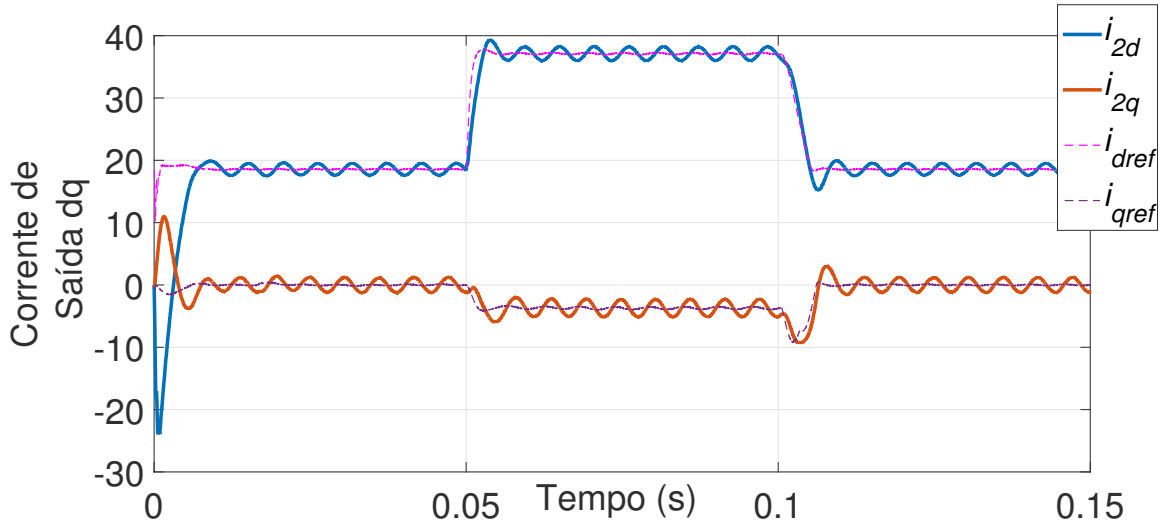
Figura 14 – Correntes de saída do inversor e tensão do sinal de controle nas coordenadas síncronas dq para os algoritmos GPC.

Tabela 6 – RMSE para o rastreamento de referência da corrente no inversor trifásico com filtro LCL.

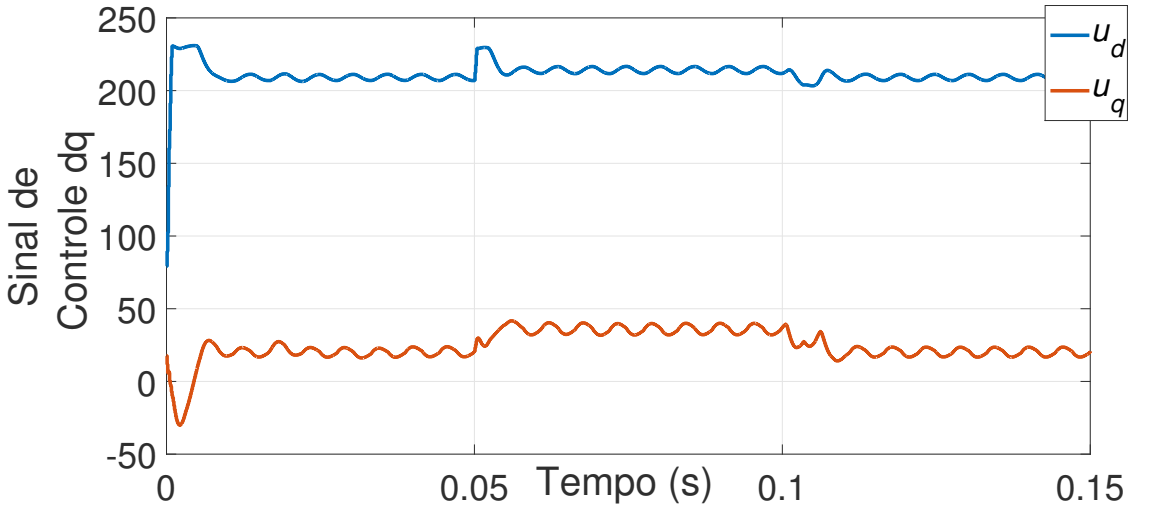
	componente do eixo *d*	componente do eixo *q*
GPC	1,64 A	1,30 A
uMPC	1,99 A	1,15 A
LQR	3,59 A	1,29 A

percebido, o que significa que a potência requerida na carga é disponibilizada pela saída do inversor e não pela rede. Dois blocos de medição PQ do Simulink foram utilizados para se obter as potências na saída do inversor e na carga.

A comparação de desempenho entre os algoritmos é ilustrada na Figura 18.



(a) Correntes de saída

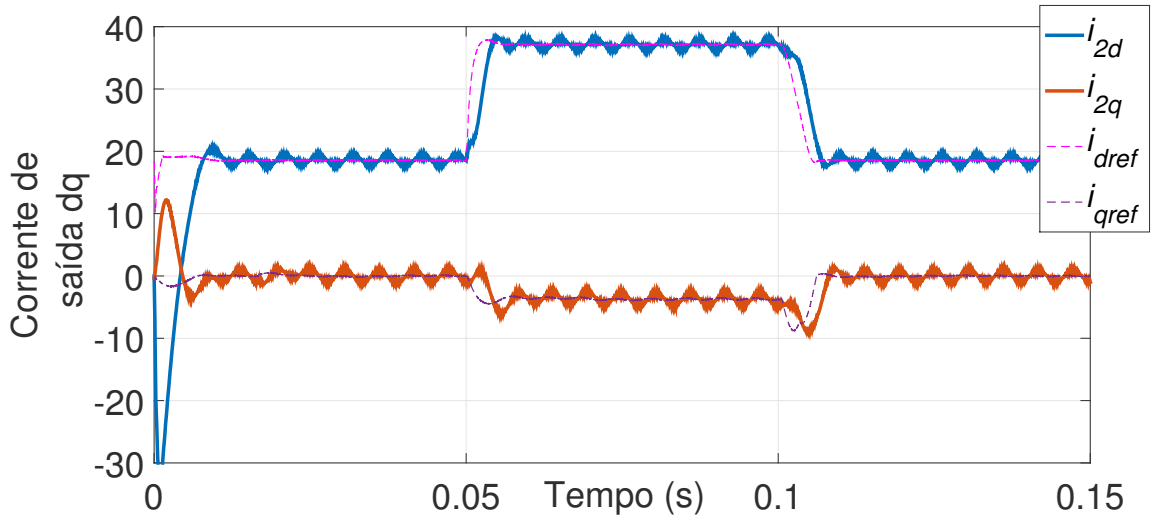


(b) Tensões do sinal de controle

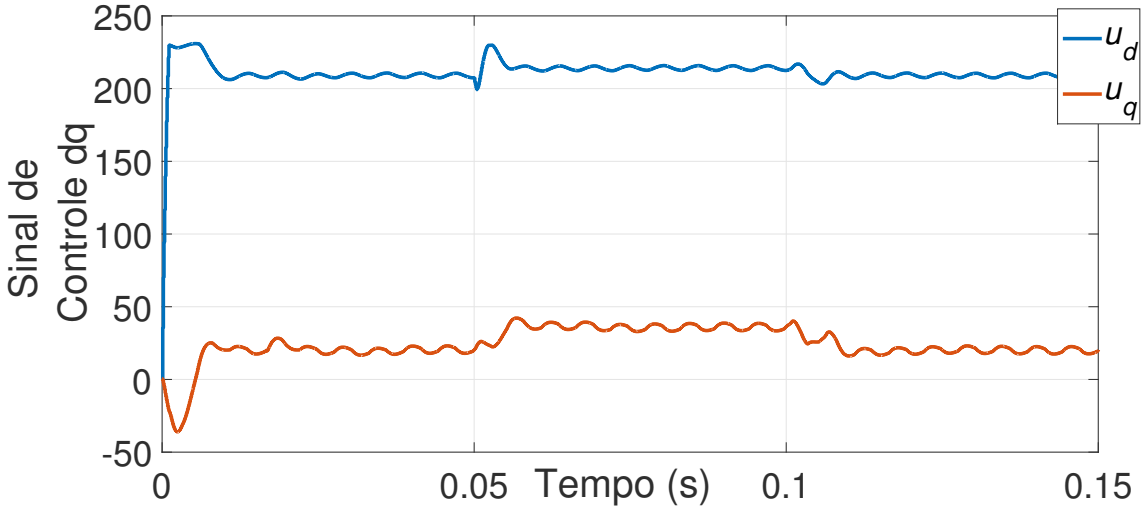
Figura 15 – Correntes de saída do inversor e tensões do sinal de controle nas coordenadas síncronas para algoritmo uMPC.

O tempo de execução de cada algoritmo é apresentado na Figura 18a. É importante mencionar que o tempo só é válido para a comparação pois a simulação não foi feita em tempo real. O tempo de execução é claramente menor para o GPCGPAD e GPCFAMA do que para o Quadprog por mais de uma ordem de magnitude e por mais de duas ordens para o GurobiSD e GurobiBar. Na Figura 18b o perfil do número de iterações é apresentado. Apesar de o GPCGPAD e GPCFAMA requererem um maior número de iterações do que os outros algoritmos durante as trocas de referência, eles ainda apresentam menores tempos de computação, uma vez que cada iteração é menos custosa do ponto de vista computacional.

A tabela 7 apresenta a comparação dos tempos de execução médios e máximos, obtidos a partir de 10 simulações com cada método. Dos dados de tempo de execução



(a) Correntes de saída



(b) Tensões do sinal de controle

Figura 16 – Correntes de saída do inversor e tensões do sinal de controle nas coordenadas síncronas para algoritmo LQR.

coletados, o maior tempo de execução observado para o GPCFAMA, com 76 iterações, foi 5,90 vezes mais rápido do que o Quadprog, 94,69 vezes mais rápido que o GurobiSD e 84,72 vezes mais rápido do que o GurobiBar. Quando comparado com o GPCGPAD, com 240 iterações, o GPCFAMA foi 1,43 vez mais rápido.

A análise de certificação foi feita para se avaliar o quão apertada é a estimativa dos limites de número de iterações e como se comportam com diferentes horizontes de controle e predição. A análise foi feita a partir do estudo de caso em questão. O horizonte de controle variou de $N_u = 5$ até $N_u = 10$ para cada variável de controle e o horizonte de predição foi mantido fixo em $N = 20$ para as variáveis de saída.

Os limites teóricos para o GPCGPAD foram obtidos a partir de (128) e da solução do MILP definido em (131). Os resultados simulados para o GPCGPAD podem ser

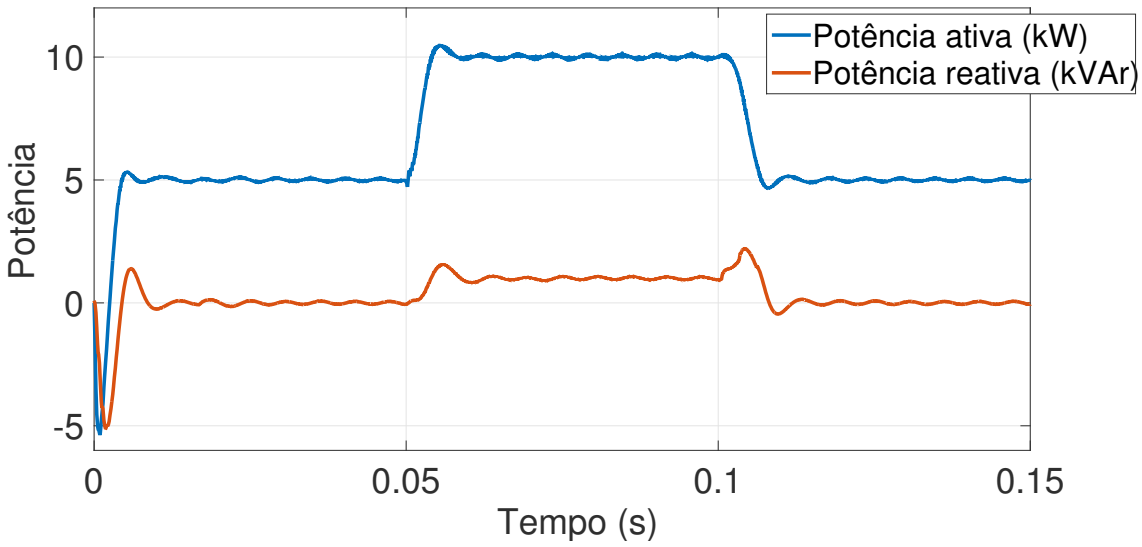


Figura 17 – Potência ativa e reativa na saída do inversor.

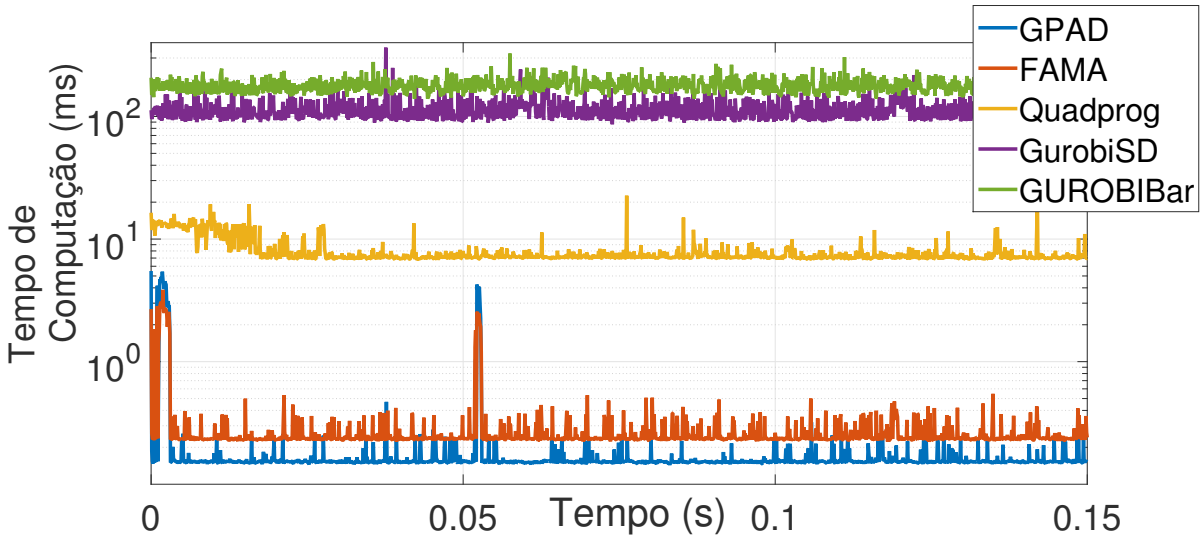
Tabela 7 – Comparação dos tempos de execução dos algoritmos GPC para o inversor trifásico simulado.

	Tempo médio	Tempo Máximo (iter.)
GPAD	0,23 ms	5,49 ms (240)
FAMA	0,30 ms	3,85 ms (76)
Quadprog	7,93 ms	22,70 ms (10)
GurobiSD	112,63 ms	364,56 ms (7)
GurobiBar	178,97 ms	326,18 ms (10)

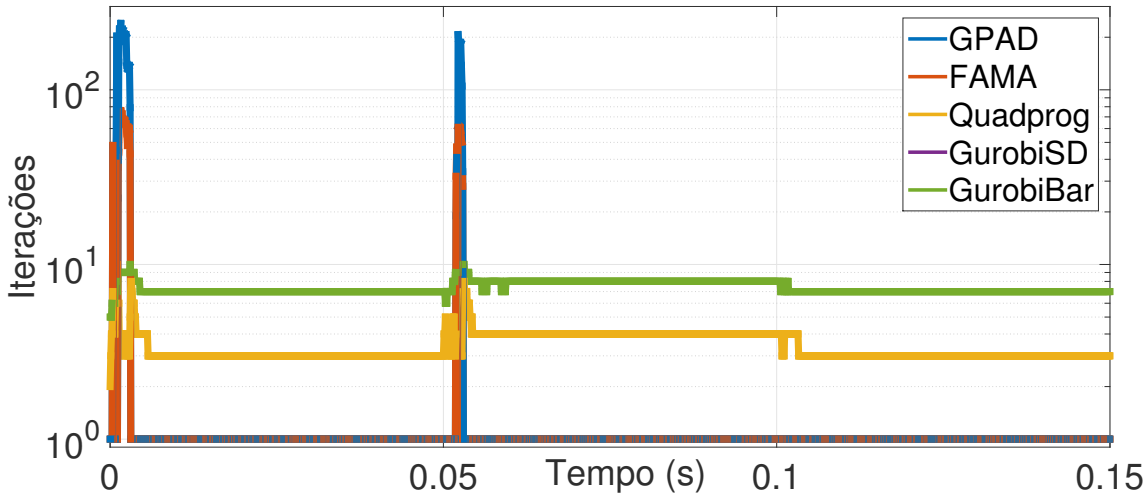
vistos na Figura 19a. Os limites teóricos para o GPCFAMA foram obtidos a partir de (145) e da abordagem de estimação baseada em amostragem com 10 000 amostras. Os resultados experimentais para o GPCFAMA podem ser vistos em Figura 19b. A partir dos ensaios, pode-se verificar que os limites teóricos do número de iterações aumentam com o aumento dos horizontes e são relativamente próximos ao observado. Para o GPCGPAD a diferença foi menor do que uma ordem de grandeza (4,92 vezes) e para o GPCFAMA foi menor de 2 ordens de grandeza (35,36 vezes). Esses limites teóricos são essenciais para se conhecer, *a priori*, o WCET. Com essa informação, é possível determinar se os requisitos de tempo real do sistema de controle podem ser atendidos.

6.4 IMPLEMENTAÇÃO EM FPGA

Os Algoritmos 11 e 12 foram implementados em um FPGA com o intuito de se avaliar o tempo de computação dos mesmos em um hardware de cômputo rápido. O FPGA utilizado foi um Altera, famíla MAX 10, do tipo 10M50DAF484C7G. A família MAX 10 apresenta conversores analógico para digital e 50 000 elementos lógicos pro-



(a) Tempo de execução



(b) Número de iterações

Figura 18 – Comparação de desempenho entre os algoritmos GPC para o inversor trifásico simulado.

gramáveis. A parte de otimização online dos algoritmos foi implementada diretamente na lógica customizada com uma linguagem de descrição de hardware chamada de Verilog. Para a representação numérica, uma aritmética de ponto fixo de 32 bits foi utilizada, com 16 bits para representar os decimais. Uma máquina de estados foi utilizada para sincronizar o algoritmo e as operações matriciais foram implementadas em computação paralela.

O pior caso observado em simulação foi replicado para a implementação em FPGA. Os resultados obtidos para o tempo de computação da otimização podem ser vistos na Tabela 8. Um sinal de relógio de 50 MHz foi utilizado e, devido ao paralelismo implementado, o tempo de computação foi bastante rápido, com um valor máximo para a computação do sinal de controle de 76,84 μs para o GPAD e 24,36 μs para o

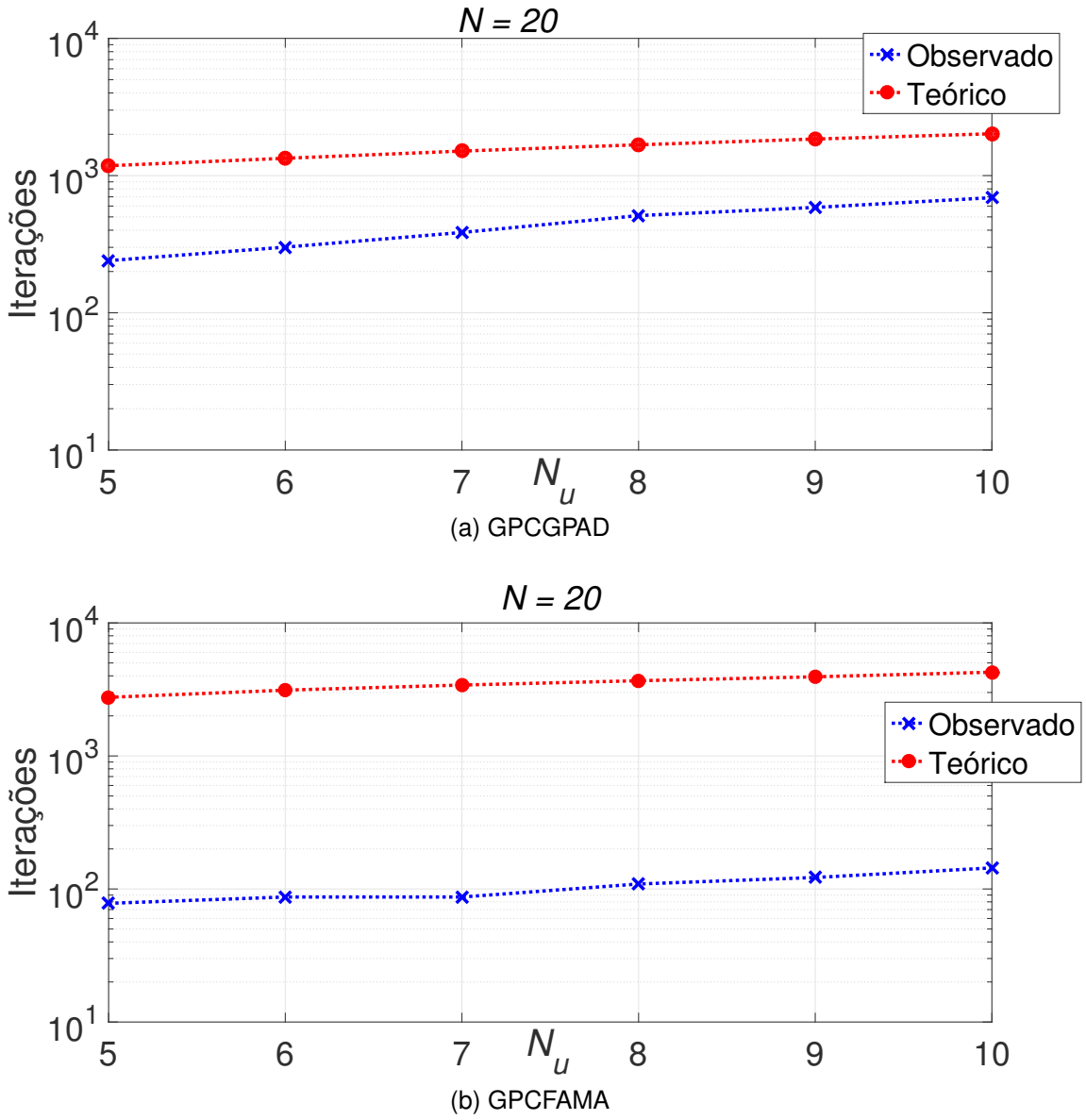


Figura 19 – Análise de certificação para o limite superior do número de iterações dos algoritmos.

FAMA. Os tempos de execução obtidos são consistentes com os valores esperados a partir de (146) e (147). Além disso, os tempos são compatíveis com a frequência de chaveamento do estudo de caso, que é 10 kHz, mesmo se forem acrescidos o tempo de computação do GPC e a resposta livre em computação paralela da mesmo forma que foi apresentado em Peccin *et al.* (2020b) com 0,14 µs e 0,80 µs, respectivamente. Para o caso do GPCFAMA, até mesmo uma frequência de chaveamento de 20 kHz poderia ser utilizada. Por outro lado, WCET teórico para ambos os algoritmos não é compatível com o período de amostragem. É importante mencionar que os limites teóricos do WCET tendem a ser conservadores e, na maioria dos casos, o sistema de controle ainda poderia ser utilizado. Entretanto, se forem exigidas garantias críticas de tempo real, o FPGA poderia ser trocado por outro com uma frequência de relógio

maior.

Tabela 8 – WCET em um FPGA para o GPCGPAD e GPCFAMA no estudo de caso do inversor trifásico.

	Observado	Teórico
GPCGPAD	76,84 μs	330,44 μs
GPCFAMA	24,36 μs	772,58 μs

6.5 CONCLUSÃO

Neste capítulo foram propostos dois algoritmos GPC de cômputo rápido baseados nos métodos gradiente projetado acelerado dual e no método do algoritmo rápido de minimização alternada. Limites superiores teóricos no número de iterações para os algoritmos propostos foram apresentados e eles são bastante importantes para definir os componentes do sistema de controle para atender requisitos rígidos de tempo real. Os algoritmos propostos e os otimizadores comerciais considerados nesse estudo apresentaram resultados similares em termos do sinal de controle calculado, entretanto os métodos propostos são pelo menos duas ordens de grandeza mais rápidos com o mesmo hardware sendo utilizado. Os algoritmos propostos também permitem uma implementação direta em FPGA pois eles utilizam apenas operações matemáticas básicas para a computação online. A implementação em FPGA provou ser bastante rápida para o estudo de caso apresentado e com a frequência de chaveamento de 10 kHz. Além disso, até mesmo uma frequência de 20 kHz pode ser utilizada com o GPCFAMA. Esses resultados contribuem para que o GPC com restrições seja adequado para ser embarcado e obter a ação de controle de forma rápida. Dessa forma, espera-se aumentar a participação dessa abordagem em sistemas de controle de processos com dinâmicas rápidas.

7 ARTIGO 4 - ALGORITMOS GPC DE CÔMPUTO RÁPIDO COM MÉTODOS DE PONTO INTERIOR E PROGRAMAÇÃO QUADRÁTICA SEM PROJEÇÃO

Neste capítulo são apresentados os desenvolvimentos e resultados do artigo intitulado Algoritmos GPC de Cômputo Rápido com Métodos de Ponto Interior e Programação Quadrática Sem Projeção (PECCIN *et al.*, 2020a). O artigo foi apresentado oralmente no XXIII Congresso Brasileiro de Automática (CBA 2020) e publicado nos anais do mesmo. Em geral, os algoritmos de primeira ordem, apesar de rápidos e simples de computar, sofrem com a baixa precisão das soluções apresentadas e com a dificuldade de tratamento de algumas classes de restrições (FERREAU *et al.*, 2017). Portanto, um dos objetivos deste trabalho é apresentar outras opções de algoritmos para GPC de modo a se obter soluções com características distintas e um estudo de comparação desses algoritmos com o GPAD e o ADMM para um problema específico. Um dos algoritmos apresentados baseia-se no método IP de barreira, que é bastante conhecido por resolver um QP em poucas iterações com boa precisão, entretanto ele requer operações algébricas mais complexas, que podem levar a maiores tempos de cômputo por iteração. O outro algoritmo baseia-se no método de programação quadrática sem projeção (PFQP, do inglês *Projection-Free Quadratic Programming*), que não necessita do passo de projeção da solução para o domínio das restrições, como no caso do GPAD (CAIRANO *et al.*, 2013). Dessa forma, o PFQP pode apresentar um desempenho superior para determinados tipos de restrições. Para servir de referência, foi utilizado o otimizador comercial Gurobi. A partir do estudo comparativo, pôde-se verificar um melhor desempenho do algoritmo GPAD e a sensibilidade dos algoritmos ADMM e PFQP em cenários com tolerâncias menores e diferentes tipos de restrições. Ressaltou-se também que, apesar de mais lento, o algoritmo com IP apresentou baixa variação no tempo de cômputo com os cenários propostos. Assim, as principais contribuições deste artigo são:

- a apresentação de dois algoritmos GPC de cômputo rápido baseados no método de barreira e no método PFQP;
- a comparação de desempenho dos métodos propostos com os algoritmos GPCADMM e GPCGPAD por meio de simulação.

7.1 PROPOSTAS DE TÉCNICAS DE OTIMIZAÇÃO

A partir do QP do GPC, apresentado em (16), os algoritmos com os métodos de ponto interior com barreira e programação quadrática sem projeção são apresentados.

7.1.1 Método de ponto interior com barreira

Para incluir as restrições de desigualdade na função objetivo de um problema como (16), pode ser utilizada um função indicadora logarítmica do tipo:

$$\hat{I}_{-} = -(1/t)\log(-\bar{x}), \quad \mathbf{dom}\ \hat{I}_{-} = \mathbb{R}^{-}, \tag{151}$$

onde $t > 0$ é um parâmetro que ajusta a exatidão da aproximação. Aplicando $\hat{I}_{-}$ em (16) obtém-se o problema convexo irrestrito:

$$\min_{\Delta\mathbf{u}} \quad \frac{1}{2}\Delta\mathbf{u}^{T}\mathbf{H}\Delta\mathbf{u} + \mathbf{b}^{T}\Delta\mathbf{u} + \sum_{i=0}^{n_{rin}} -(1/t)\log(-\bar{\mathbf{R}}_{li}\Delta\mathbf{u} + \bar{\mathbf{r}}_{i}), \tag{152}$$

onde $\bar{\mathbf{R}}_{li}$ representa a i-ésima linha da matriz $\bar{\mathbf{R}}$ e $\bar{\mathbf{r}}_{i}$ o i-ésimo elemento do vetor $\bar{\mathbf{r}}$. Dessa forma, o método de barreira se baseia na resolução de uma sequência de problemas de minimização irrestritos, utilizando o último ponto encontrado como o ponto de início da próxima iteração. A cada iteração o valor de t é incrementado, até que $t \geq n_{rin}/\epsilon$, o que garante que seja obtida uma solução ϵ-subótima do problema original (BOYD; VANDENBERGHE, 2004).

Para resolver (152), apesar de qualquer método de minimização linear com restrições de igualdade poder ser utilizado, em geral, utiliza-se o método de Newton que garante uma solução factível a cada iteração (BOYD; VANDENBERGHE, 2004). Esse cômputo é chamado de passo de centralização, uma vez que o caminho central é definido como o conjunto de pontos chamados pontos centrais. Os pontos centrais são caracterizados por atenderem as condições necessárias e suficientes para serem considerados pontos estritamente factíveis. Dessa forma, o passo de Newton é dado por:

$$\Delta\mathbf{x}_{nt} = -\nabla^{2}h(\mathbf{x})^{-1}\nabla h(\mathbf{x}).$$

O gradiente $\mathcal{G}$ e a Hessiana $\mathcal{H}$ da função custo de (152) são dadas por:

$$\mathcal{G} = \mathbf{H}\Delta\mathbf{u} + \mathbf{b} + \sum_{i=0}^{n_{rin}} \frac{(1/t)}{\bar{\mathbf{R}}_{li}\Delta\mathbf{u} - \bar{\mathbf{r}}_{i}}\bar{\mathbf{R}}_{li}^{T}$$

$$\mathcal{H} = \mathbf{H} + \sum_{i=0}^{n_{rin}} \frac{(1/t)}{(\bar{\mathbf{R}}_{li}\Delta\mathbf{u} - \bar{\mathbf{r}}_{i})^{2}}\bar{\mathbf{R}}_{li}^{T}\bar{\mathbf{R}}_{li}.$$

Portanto, o passo de Newton para o problema (152) é:

$$\Delta\mathbf{x}_{nt} = -\mathcal{H}^{-1}\mathcal{G}.$$

O algoritmo de Newton aplicado ao problema (152) pode ser visto no Algoritmo 13.

Ao se avaliar o Algoritmo 13, pode-se constatar que o cômputo mais custoso se refere ao passo de Newton. Pode-se abstrair esse passo como o cômputo de um

Algoritmo 13 Método de Newton

Entrada: $\Delta\mathbf{u}^0$ estritamente factível, t, $\mathbf{H}$, $\mathbf{b}$, $\bar{\mathbf{R}}$, $\bar{\mathbf{r}}$
Saída: $\Delta\mathbf{u}$
Dados: L_{nt}
início
  para $i = 1 : L_{nt}$ **faça**
    Atualiza $\mathcal{G} \leftarrow \mathbf{H}\Delta\mathbf{u} + \mathbf{b} + \sum_{i=0}^{n_{rin}} \frac{(1/t)}{\bar{\mathbf{R}}_{li}\Delta\mathbf{u}-\bar{\mathbf{r}}_i}\bar{\mathbf{R}}_{li}^T$;
    Atualiza $\mathcal{H} \leftarrow \mathbf{H} + \sum_{i=0}^{n_{rin}} \frac{(1/t)}{(\bar{\mathbf{R}}_{li}\Delta\mathbf{u}-\bar{\mathbf{r}}_i)^2}\bar{\mathbf{R}}_{li}^T\bar{\mathbf{R}}_{li}$;
    Computa o passo de Newton: $\Delta\mathbf{x}_{nt} \leftarrow -\mathcal{H}^{-1}\mathcal{G}$;
    Escolhe o tamanho do passo l por busca retroativa linear ;
    Atualiza $\Delta\mathbf{u} \leftarrow \Delta\mathbf{u} + l\Delta\mathbf{x}_{nt}$;
fim

sistema linear do tipo $\mathbf{Ax} = \mathbf{b}$, que tipicamente requer $O(N^3 n_{rin}^3)$ operações em um processador serial (WILLS *et al.*, 2011). Esse tipo de sistema poderia ser resolvido de forma eficiente por meio da decomposição das matrizes e exploração da estrutura da matriz $\mathbf{A}$, como a decomposição de Cholesky. Entretanto, a decomposição só é eficiente se a estrutura de $\mathbf{A}$ for esparsa. Para o caso do GPC, a matriz tipicamente é densa, então não é trivial explorar a estrutura. Dessa forma, optou-se por um método iterativo. O método escolhido foi o do gradiente conjugado (GOLUB; VAN LOAN, 1996). Essa abordagem foi utilizada no contexto de SSMPC e FPGA em Roldao-Lopes *et al.* (2009) e Wills *et al.* (2011). Para entender como o método do gradiente conjugado pode ser utilizado para resolver o problema $\mathbf{Ax} = \mathbf{b}$, pode-se considerar o problema de minimização:

$$\min_{\mathbf{x}} \quad h_{gc}(x) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b}, \tag{153}$$

com $\mathbf{A} \in \mathbb{R}^{n\times n}$ simétrica positiva definida e $\mathbf{b} \in \mathbb{R}^n$. O mínimo de h_{gc} é $-\mathbf{b}^T\mathbf{A}^{-1}\mathbf{b}/2$, com $x^* = \mathbf{A}^{-1}\mathbf{b}$. Dessa forma, minimizar h_{gc} e resolver $\mathbf{Ax} = \mathbf{b}$ são problemas equivalentes. O Algoritmo 14 apresenta o método do gradiente conjugado com busca exata de linha (em inglês, *exact line search*) aplicado ao cômputo do passo de Newton.

A partir dos métodos apresentados, pôde-se chegar ao algoritmo completo do método de barreira proposto para o GPC (GPCIP), que pode ser visto no Algoritmo 15.

As orientações para se escolher os parâmetros do método de barreira podem vistas na Seção 3.3.4. O método de Newton, usado para computar o passo de centralização, foi limitado em L_{nt} iterações. Para o cômputo do passo de Newton foi utilizado o método do gradiente conjugado, limitado em L_{gc} iterações devido à rápida convergência. O tamanho do passo de Newton foi computado utilizando-se o algoritmo de busca retroativa linear, em duas partes. Na primeira, pode-se perceber uma busca por um ponto factível a cada iteração. Na segunda parte, a busca pela minimização da função custo, com o parâmetro ϖ sendo a fração aceitável de decaimento da função custo

Algoritmo 14 Método do Gradiente Conjugado

Entrada: $\Delta \mathbf{x}_{nt}^0$ estritamente factível, $\mathcal{H}$, $\mathcal{G}$
Saída: $\Delta \mathbf{x}_{nt}$
início
  Calcula o resíduo inicial: $\mathbf{r} \leftarrow \mathcal{G} - \mathcal{H}\Delta \mathbf{x}_{nt}^0$;
  $i \leftarrow 0$;
  enquanto $\mathbf{r} \neq 0$ **faça**
    $i \leftarrow i + 1$;
    se $i = 1$ **então**
      Calcula a direção do passo: $\mathbf{p} \leftarrow \mathbf{r}$;
    senão
      Calcula a direção do passo: $\mathbf{p} \leftarrow \mathbf{r} + \frac{\mathbf{r}^T\mathbf{r}}{\mathbf{r}^{-T}\mathbf{r}^-}\mathbf{p}$;
    Calcula o tamanho do passo $\alpha \leftarrow \mathbf{r}^T\mathbf{r} / \left(\mathbf{p}^T\mathcal{H}\mathbf{p}\right)$;
    Atualiza $\Delta \mathbf{x}_{nt} \leftarrow \Delta \mathbf{x}_{nt} + \alpha\mathbf{p}$;
    Guarda valor anterior: $\mathbf{r}^- \leftarrow \mathbf{r}$;
    Atualiza o resíduo: $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathcal{H}\mathbf{p}$;
fim

predita por extrapolamento linear. Em Boyd e Vandenberghe (2004) afirma-se que os valores típicos de ϖ são escolhidos entre 0,01 e 0,3, significando que é aceitável um decaimento da função custo de 1% a 30% da predição baseada na extrapolação linear.

7.1.2 Método programação quadrática sem projeção

O algoritmo proposto é baseado na aplicação do PFQP para a formulação paramétrica do MPC em espaço de estados apresentada em Cairano *et al.* (2013). Como o PFQP não necessita do passo de projeção para o domínio das restrições, o método se torna bastante competitivo em casos nos quais a projeção das restrições é complexa, como restrições de saída ou politópicas. A ideia principal do PFQP é, a cada passo do método, aproximar-se do valor ótimo na direção do anti-gradiente, com o tamanho do passo escolhido para garantir que se mantenha na região factível.

Inicialmente, para adaptar o problema (16) para o formato padrão de mínimos quadrados não negativos, pode ser utilizada a formulação dual de (16):

$$\begin{aligned} \min_{\psi} \quad & \frac{1}{2}\psi^T\hat{\mathbf{H}}\psi + \hat{\mathbf{b}}^T\psi \\ \text{s.a.} \quad & \psi \geq \mathbf{0}, \end{aligned} \tag{154}$$

com $\hat{\mathbf{H}} = \bar{\mathbf{R}}\mathbf{H}^{-1}\bar{\mathbf{R}}^T$ e $\hat{\mathbf{b}} = \bar{\mathbf{R}}\mathbf{H}^{-1}\mathbf{b} + \bar{\mathbf{r}}$.

De modo a computar a solução ótima de (154), o Lagrangiano associado é dado por:

$$\mathcal{L}(\psi, \nu) = \frac{1}{2}\psi^T\hat{\mathbf{H}}\psi + \hat{\mathbf{b}}^T\psi - \nu^T\psi, \tag{155}$$

Algoritmo 15 Controle preditivo generalizado (GPC) com método de barreira

Entrada: A, B, d, $\bar{\mathbf{R}}$, $\bar{\mathbf{r}}$

Saída: $u(k)$

Dados: q_λ, q_δ, N_u, N_1, N_2, $t := t_0 > 0$, $\mu > 1$, $\epsilon > 0$, $\beta \in (0,\ 1)$, $\varpi \in (0,\ 0{,}5)$, L_{gc}, L_{nt}.

início

- $\tilde{A}(z^{-1}) \leftarrow (1 - z^{-1})A(z^{-1})$;
- **para** $i = 0 : N_2$ **faça**
  - $g_i \leftarrow z(1 - A(z^{-1}))g_{i-1} + B(z^{-1})\bar{u}_i$;
- **para** $i = 1 : N_u$ **faça**
  - $\mathbf{G}_{i:N,i} \leftarrow \mathbf{g}_{N_1:N_2+1-i}$;
- $\mathbf{H} \leftarrow 2(\mathbf{G}^T\mathbf{G} + q_\lambda\mathbf{I})$;
- $k \leftarrow 0$;
- **enquanto** *modo automático* **faça**
  - **se** *tempo de amostragem* **então**
    - Adquire saída $y(k)$ e referência $w(k)$;
    - $f_0 \leftarrow y(k)$;
    - **para** $i = 0 : N_2$ **faça**
      - $f_{i+1} \leftarrow z(1 - \tilde{A}(z^{-1}))f_i + B(z^{-1})\Delta u(k - d + i)$;
    - $\mathbf{b}^T \leftarrow 2(\mathbf{f}_{N_1:N_2} - \mathbf{w}_{N_1:N_2})^T\mathbf{G}$;
    - **enquanto** $n_{rin}/t > \epsilon$ **faça**
      - **para cada** $i = 1 : L_{nt}$ **faça**
        - $\mathcal{G} \leftarrow \mathbf{H}\Delta\mathbf{u} + \mathbf{b} + \sum_{i=0}^{n_{rin}} \frac{(1/t)}{\bar{\mathbf{R}}_{li}\Delta\mathbf{u} - \bar{\mathbf{r}}_i} \bar{\mathbf{R}}_{li}^T$;
        - $\mathcal{H} \leftarrow \mathbf{H} + \sum_{i=0}^{n_{rin}} \frac{(1/t)}{(\bar{\mathbf{R}}_{li}\Delta\mathbf{u} - \bar{\mathbf{r}}_i)^2} \bar{\mathbf{R}}_{li}^T\bar{\mathbf{R}}_{li}$;
        - $\mathbf{r} \leftarrow -\mathcal{G} - \mathcal{H}\Delta\mathbf{x}_{nt}$;
        - **para cada** $j = 1 : L_{gc}$ **faça**
          - **se** $j = 1$ **então**
            - $\mathbf{p} \leftarrow \mathbf{r}$;
          - **senão**
            - $\mathbf{p} \leftarrow \mathbf{r} + \frac{\mathbf{r}^T\mathbf{r}}{\mathbf{r}^{-T}\mathbf{r}^-}\mathbf{p}$;
          - $\alpha \leftarrow \mathbf{r}^T\mathbf{r}/\mathbf{p}^T\mathcal{H}\mathbf{p}$;
          - $\Delta\mathbf{x}_{nt} \leftarrow \Delta\mathbf{x}_{nt} + \alpha\mathbf{p}$;
          - $\mathbf{r}^- \leftarrow \mathbf{r}$;
          - $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathcal{H}\mathbf{p}$;
        - **enquanto** $\bar{\mathbf{R}}(\Delta\mathbf{u} + l\Delta\mathbf{x}_{nt}) - \bar{\mathbf{r}} > 0$ **faça**
          - $l \leftarrow \beta l$;
        - **enquanto** $h(\Delta\mathbf{u} + l\Delta\mathbf{x}_{nt}) > h(\Delta\mathbf{u}) + l\varpi\mathcal{G}^T\Delta\mathbf{x}_{nt}$ **faça**
          - $l \leftarrow \beta l$;
        - $\Delta\mathbf{u} \leftarrow \Delta\mathbf{u} + l\Delta\mathbf{x}_{nt}$;
      - $t \leftarrow \mu t$;
    - $\Delta u(k) \leftarrow [1 \quad 0 \quad \cdots \quad 0]_{1\times N_u}\Delta\mathbf{u}$;
    - $u(k) \leftarrow u(k-1) + \Delta u(k)$;
    - $k \leftarrow k + 1$;

fim

com $\boldsymbol{\nu} \in \mathbb{R}^{n_{rin}}$ sendo o vetor de multiplicadores de Lagrange associados às restrições de desigualdade.

A partir de (155), a condição de otimalidade de (154), obtida por cálculo variacional, é dada por:

$$\boldsymbol{\psi} \circ \nabla_{\psi} \mathcal{L}(\boldsymbol{\psi}, \boldsymbol{\nu}) = \mathbf{0}, \tag{156}$$

onde $\circ$ representa o produto de Hadamard entre dois vetores.

Aplicando a condição de otimalidade (156) em (155) obtém-se:

$$\boldsymbol{\psi} \circ (\hat{\mathbf{H}}\boldsymbol{\psi} + \hat{\mathbf{b}} - \boldsymbol{\nu}) = \mathbf{0}. \tag{157}$$

Se for definido $\hat{\mathbf{H}} = \hat{\mathbf{H}}^{+} - \hat{\mathbf{H}}^{-}$ com $\hat{\mathbf{H}}^{+} \triangleq \max(\mathbf{0}, \hat{\mathbf{H}})$, $\hat{\mathbf{H}}^{-} \triangleq \max(\mathbf{0}, -\hat{\mathbf{H}})$ e for aplicada a mesma lógica para $\hat{\mathbf{b}}$, obtém-se:

$$\boldsymbol{\psi} \circ ((\hat{\mathbf{H}}^{+}\boldsymbol{\psi} + \hat{\mathbf{b}}^{+}) - (\hat{\mathbf{H}}^{-}\boldsymbol{\psi} + \hat{\mathbf{b}}^{-} + \boldsymbol{\nu})) = \mathbf{0}, \tag{158}$$

que pode ser apresentado de forma reduzida como:

$$\left[\boldsymbol{\psi}_{j+1}\right]_i = \frac{\left[\hat{\mathbf{H}}^{-}\boldsymbol{\psi}_j + \hat{\mathbf{b}}^{-}\right]_i}{\left[\hat{\mathbf{H}}^{+}\boldsymbol{\psi}_j + \hat{\mathbf{b}}^{+}\right]_i} \left[\boldsymbol{\psi}_j\right]_i. \tag{159}$$

Para garantir a convergência, foi inserido um termo $\phi_{PFQP} \in \mathbb{R}^{n \times n}$

$$\left[\boldsymbol{\psi}_{j+1}\right]_i = \frac{\left[(\hat{\mathbf{H}}^{-} + \phi_{PFQP})\boldsymbol{\psi}_j + \hat{\mathbf{b}}^{-}\right]_i}{\left[(\hat{\mathbf{H}}^{+} + \phi_{PFQP})\boldsymbol{\psi}_j + \hat{\mathbf{b}}^{+}\right]_i} \left[\boldsymbol{\psi}_j\right]_i. \tag{160}$$

Em Cairano *et al.* (2013) foi provada a garantia de convergência com a escolha de ϕ_{PFQP} como uma matriz diagonal não negativa tal que os elementos da diagonal são dados por $\left[\phi_{PFQP}\right]_{ii} \geq \left[\hat{\mathbf{H}}^{-} \cdot \mathbf{1}\right]_i$, $\forall i = 1,..,n$.

A etapa de aceleração do PFQP, aplicada ao problema (154), é dada por:

$$l = \begin{cases} -\dfrac{(\hat{\mathbf{H}}\boldsymbol{\psi} + \hat{\mathbf{b}})^T \mathbf{p}}{\mathbf{p}^T \hat{\mathbf{H}} \mathbf{p}} & \text{se } \mathbf{p}^T \hat{\mathbf{H}} \mathbf{p} > 0 \\ 0 & \text{senão} \end{cases}, \tag{161}$$

e $\mathbf{p}$, que representa a direção de descenso (valores negativos do gradiente), é definido como:

$$\mathbf{p} = \max(\mathbf{0}, -(\hat{\mathbf{H}}\boldsymbol{\psi} + \hat{\mathbf{b}})). \tag{162}$$

O passo de aceleração deve ser executado a cada η_{PFQP} iterações do passo principal.

Para o critério de parada, podem ser utilizadas a factibilidade e a otimalidade primal. No caso do PFQP, a factibilidade primal é dada em termos duais, portanto:

$$-(\hat{\mathbf{b}} + \hat{\mathbf{H}}\boldsymbol{\psi}) < \epsilon_{pri}. \tag{163}$$

Já para o critério de otimalidade primal, utilizando-se a propriedade da dualidade forte, aplicando a (16) e representando em termos duais, tem-se:

$$\psi^T \hat{\mathbf{H}} \psi + \hat{\mathbf{b}}^T \psi \leq \epsilon_{dual}. \tag{164}$$

A partir dos desenvolvimentos apresentados pode-se resumir o GPC com o PFQP (GPCPFQP) no Algoritmo 16.

7.2 COMPARAÇÃO DAS TÉCNICAS

Nesta seção é apresentada uma comparação dos algoritmos propostos com emprego de simulações em MATLAB. Os algoritmos foram aplicados no mesmo estudo de caso apresentados em Peccin *et al.* (2019) e Peccin *et al.* (2020b), cujo processo SISO é modelado pela função de transferência em tempo discreto:

$$G(z) = \frac{0{,}035z + 0{,}0307}{z^2 - 1{,}6375z + 0{,}6703}.$$

Um otimizador comercial chamado Gurobi foi utilizado como base de comparação para os algoritmos. O método escolhido para a solução do QP no otimizador Gurobi foi o de ponto interior com barreira (GUROBI OPTIMIZATION, 2021). A simulação foi realizada por 150 amostras discretas e com três trocas de referência do tipo degrau. Os parâmetros de controle foram escolhidos com horizonte de controle $N_u = 5$, horizontes de predição $N_1 = 1$ e $N_2 = 20$, ponderação no esforço de controle $\lambda = 1$, ponderação no erro $\delta = 1$ e referências futuras $w(t+j)$ conhecidas. Não foi utilizado *warm-start* em nenhum dos algoritmos. O algoritmo GPCIP foi sintonizado com $t_0 = 0{,}1$, $\mu = 8$, $\beta = 0{,}4$, $\varpi = 0{,}02$, $L_{nt} = 3$ e $L_{gc} = 5$. O algoritmo GPCPFQP foi sintonizado com $\eta_{PFQP} = 5$ e o algoritmo GPCADMM com $\rho = 50$. Os tempos de cômputo durante a simulação foram obtidos a partir do comando *tic-toc* do MATLAB. O computador utilizado possui um processador Core i3 de 2,40 GHz e uma memória RAM de 12 GB.

O ensaio foi proposto com três cenários de modo a avaliar o comportamento dos algoritmos em relação ao tempo de cômputo e o número de iterações.

No cenário 1 foi inserida uma restrição no incremento de controle

$$|\Delta \mathbf{u}(k+j)| \leq 0{,}05,$$

com $j = 0,...,N_u - 1$. Dessa forma, o problema de otimização recaiu então em uma matriz $\mathbf{H} \in \mathbb{R}^{5\times 5}$ com 10 restrições. Todos os algoritmos foram sintonizados para uma tolerância dos resíduos de $\pm 10^{-3}$.

No cenário 2 foi utilizada apenas uma restrição na variável de saída

$$0 \leq \bar{\mathbf{y}}(k+j) \leq 2{,}1$$

com $j = 1,...,N-1$ e totalizando 40 restrições.

Algoritmo 16 GPC com PFQP

Entrada: A, B, d, $\bar{\mathbf{R}}$, $\bar{\mathbf{r}}$
Saída: $u(k)$
Dados: q_λ, q_δ, N_u, N_1, N_2, η_{PFQP}, ϵ_{dual}, ϵ_{pri}
início
  $\tilde{A}(z^{-1}) \leftarrow (1-z^{-1})A(z^{-1})$;
  para $i = 0 : N_2$ **faça**
    $g_i \leftarrow z(1-A(z^{-1}))g_{i-1} + B(z^{-1})\bar{u}_i$;
  para $i = 1 : N_u$ **faça**
    $\mathbf{G}_{i:N,i} \leftarrow \mathbf{g}_{N_1:N_2+1-i}$;
  $\mathbf{H} \leftarrow 2(\mathbf{G}^T\mathbf{G} + \lambda\mathbf{I})$;
  $\hat{\mathbf{H}} \leftarrow \bar{\mathbf{R}}\mathbf{H}^{-1}\bar{\mathbf{R}}^T$;
  $\hat{\mathbf{H}}^+ \leftarrow \max(\mathbf{0},\hat{\mathbf{H}})$; $\hat{\mathbf{H}}^- \leftarrow \max(\mathbf{0},-\hat{\mathbf{H}})$;
  $k \leftarrow 0$;
  enquanto *modo automático* **faça**
    se *tempo de amostragem* **então**
      Adquire saída $y(k)$ e referência $w(k)$;
      $f_0 \leftarrow y(k)$;
      para $i = 0 : N_2$ **faça**
        $f_{i+1} \leftarrow z(1-\tilde{A}(z^{-1}))f_i + B(z^{-1})\Delta u(k-d+i)$;
      $\mathbf{b}^T \leftarrow 2(\mathbf{f}_{N_1:N_2} - \mathbf{w}_{N_1:N_2})^T\mathbf{G}$;
      $\hat{\mathbf{b}} \leftarrow \mathbf{b} + \bar{\mathbf{R}}\mathbf{H}^{-1}\mathbf{b}$;
      $\hat{\mathbf{b}}^+ \leftarrow \max(\mathbf{0},\hat{\mathbf{b}})$; $\hat{\mathbf{b}}^- \leftarrow \max(\mathbf{0},-\hat{\mathbf{b}})$;
      $j \leftarrow 0$;
      enquanto $\psi^T\hat{\mathbf{H}}\psi + \hat{\mathbf{b}}^T\psi \geq \epsilon_{dual}$ *ou* $-(\hat{\mathbf{b}} + \hat{\mathbf{H}}\psi) \geq \epsilon_{pri}$ **faça**
        se *j é múltiplo de* η_{PFQP} **então**
          $p \leftarrow \max(\mathbf{0},-(\hat{\mathbf{H}}\psi + \hat{\mathbf{b}}))$;
          se $\mathbf{p}^T\hat{\mathbf{H}}\mathbf{p} > 0$ **então**
            $l \leftarrow -\dfrac{(\hat{\mathbf{H}}\psi + \hat{\mathbf{b}})^T\mathbf{p}}{\mathbf{p}^T\hat{\mathbf{H}}\mathbf{p}}$
          senão
            $l \leftarrow 0$;
          $\psi_{j+1} \leftarrow \psi_j + l\mathbf{p}$;
        senão
          para $i = 1 : n_{rin}$ **faça**
            $\left[\psi_{j+1}\right]_i \leftarrow \dfrac{\left[(\hat{\mathbf{H}}^- + \phi_{PFQP})\psi_j + \hat{\mathbf{b}}^-\right]_i}{\left[(\hat{\mathbf{H}}^+ + \phi_{PFQP})\psi_j + \hat{\mathbf{b}}^+\right]_i}\left[\psi_j\right]_i$;
          $j \leftarrow j + 1$;
      $\Delta\mathbf{u} \leftarrow \mathbf{H}^{-1}(\mathbf{b} + \bar{\mathbf{R}}^T\psi)$;
      $\Delta u(k) \leftarrow [1 \quad 0 \quad \cdots \quad 0]_{1\times N_u}\Delta\mathbf{u}$;
      $u(k) \leftarrow u(k-1) + \Delta u(k)$;
      $k \leftarrow k + 1$;
fim

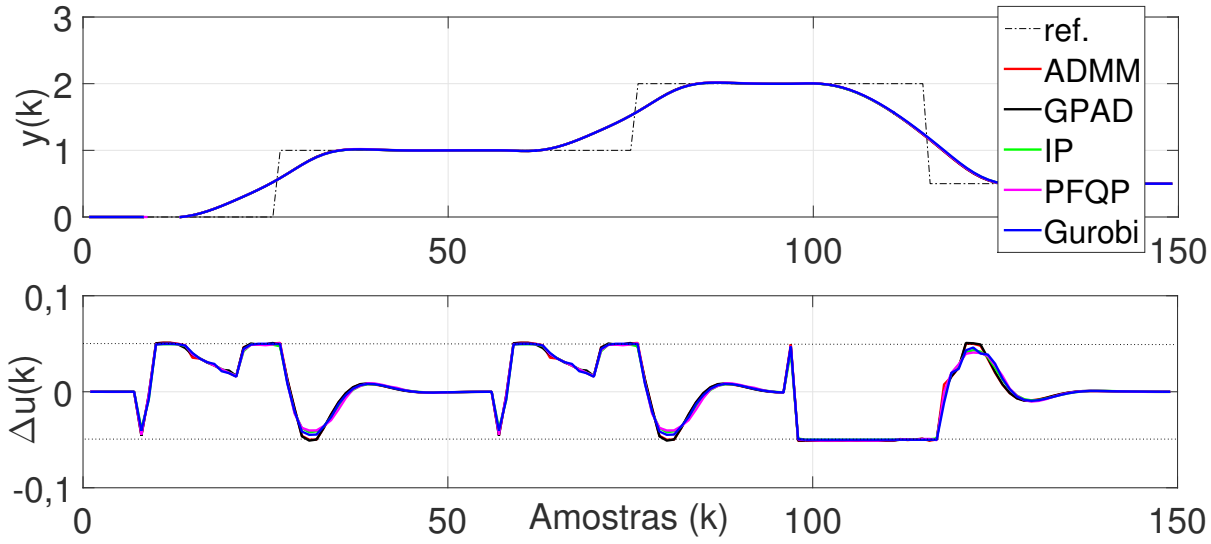
No cenário 3, foram utilizadas ambas as restrições, na variável de controle e na de saída, totalizando 50 restrições. Nesse cenário, as tolerâncias também foram alteradas para $\pm 10^{-6}$.

Os resultados para o cenário 1 podem ser vistos na Figura 20. Os sinais de saída da planta e os incrementos de controle aplicados são apresentados na Figura 20a e verifica-se uma saturação atuando no incremento de controle em $\pm$0,05, evidenciando assim que as restrições estão ativas. Percebe-se na Figura 20b que os tempos de cômputo são menores para ADMM e o GPAD em relação ao PFQP e o IP. A partir do gráfico de iterações, percebe-se uma grande sensibilidade dos algoritmos de primeira ordem quando as restrições estão ativas, refletindo esse comportamento no aumento do número de iterações. Essa característica não aparece nos algoritmos IP e Gurobi. O IP, como esperado, teve um valor fixo de iterações do método de barreira, sem considerar as iterações internas do método de Newton e do gradiente dual. Este cenário é propício para os métodos de primeira ordem que apresentam um melhor desempenho para soluções com tolerâncias maiores.

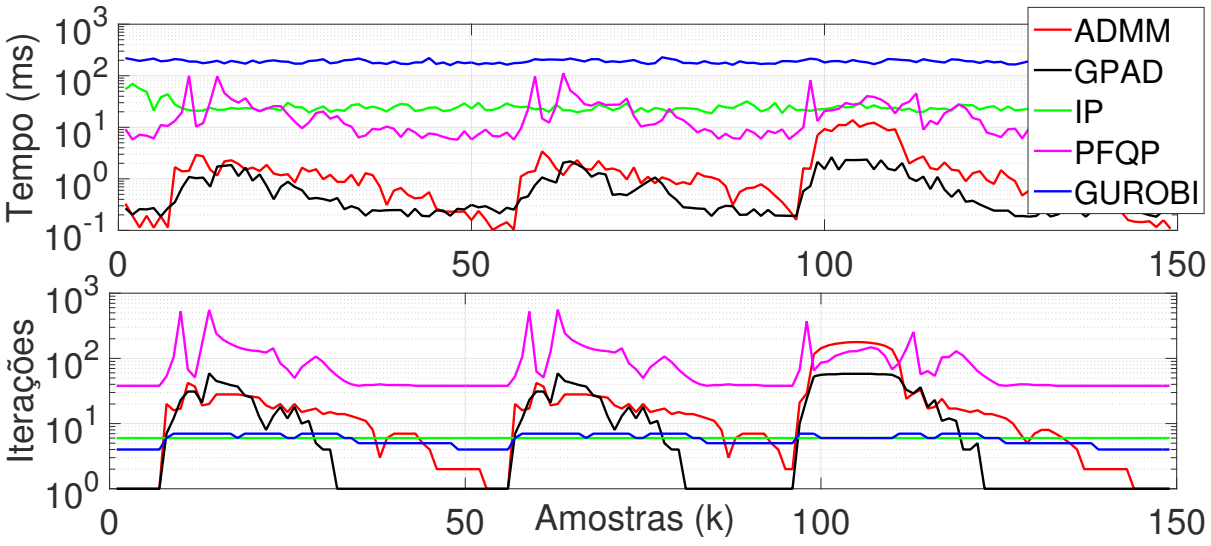
Já no cenário 2, apresentado na Figura 21, com apenas as restrições no sinal de saída, o incremento de controle computado atinge valores mais altos (Figura 21a). Pode-se também perceber uma pequena variação dos valores do sinal de controle computado entre os algoritmos, dentro da faixa de tolerância $\pm 10^{-3}$. Na avaliação de desempenho apresentada na Figura 21b, com um número maior de restrições e um tipo de restrição mais complexa, todos os algoritmos gastaram um tempo de cômputo ligeiramente maior, entretanto o PFQP apresentou uma leve melhora ficando mais próximo do ADMM. O GPAD ainda se apresenta como a solução mais rápida, beneficiando-se da formulação dual que deixa a restrição mais simples antes de ser computada. Os algoritmos IP e Gurobi apresentaram pouca variação entre os cenários 1 e 2.

No terceiro cenário (Figura 22), com ambas as restrições e com a tolerância de $\pm 10^{-6}$, a diferença no sinal de controle entre os algoritmos é imperceptível na escala apresentada no gráfico (Figura 22a). Na avaliação de tempo de cômputo apresentado na Figura 21b, os algoritmos de primeira ordem apresentaram um aumento considerável no número de iterações e consequentemente no tempo de cômputo. No pior caso, o ADMM teve um desempenho muito próximo dos métodos de barreira IP e Gurobi. O algoritmo PFQP teve um desempenho bem inferior ficando até 2 ordens de grandeza mais lento que o Gurobi. Por outro lado, o GPAD ainda se mostrou o algoritmo mais rápido. Neste cenário fica evidenciado que a utilização do método IP se torna competitivo com o aumento da exigência na tolerância da resposta. A vantagem da utilização do PFQP com relação ao aumento de restrições foi encoberto devido à sensibilidade do método ao aumento da exigência da tolerância.

Na Tabela 9 os tempos máximos de cômputo dos algoritmos, em milissegundos,



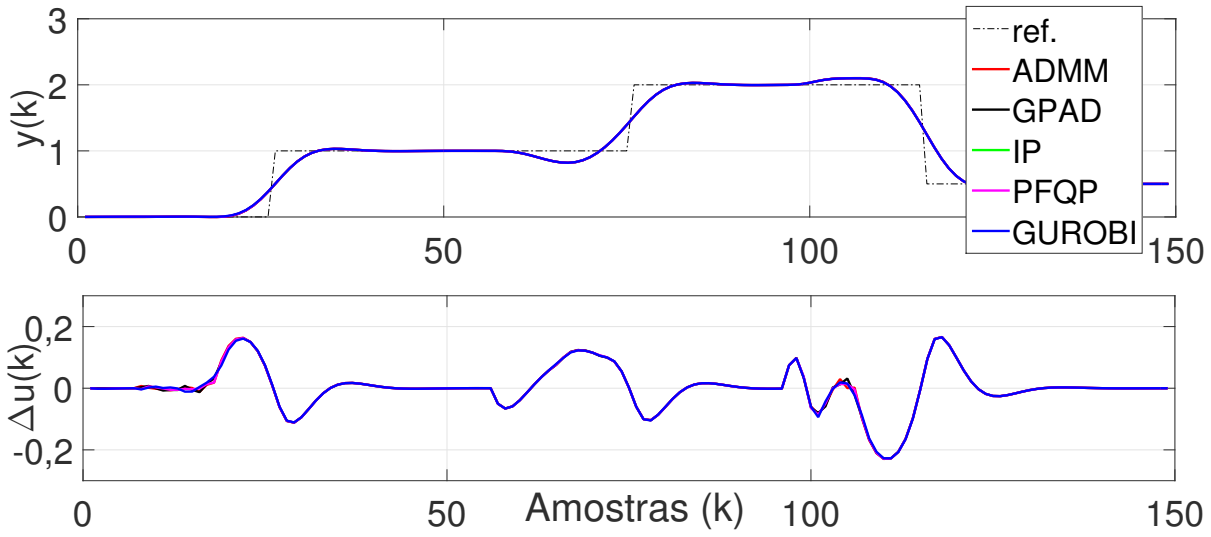
(a) Sinais de saída e incrementos de controle



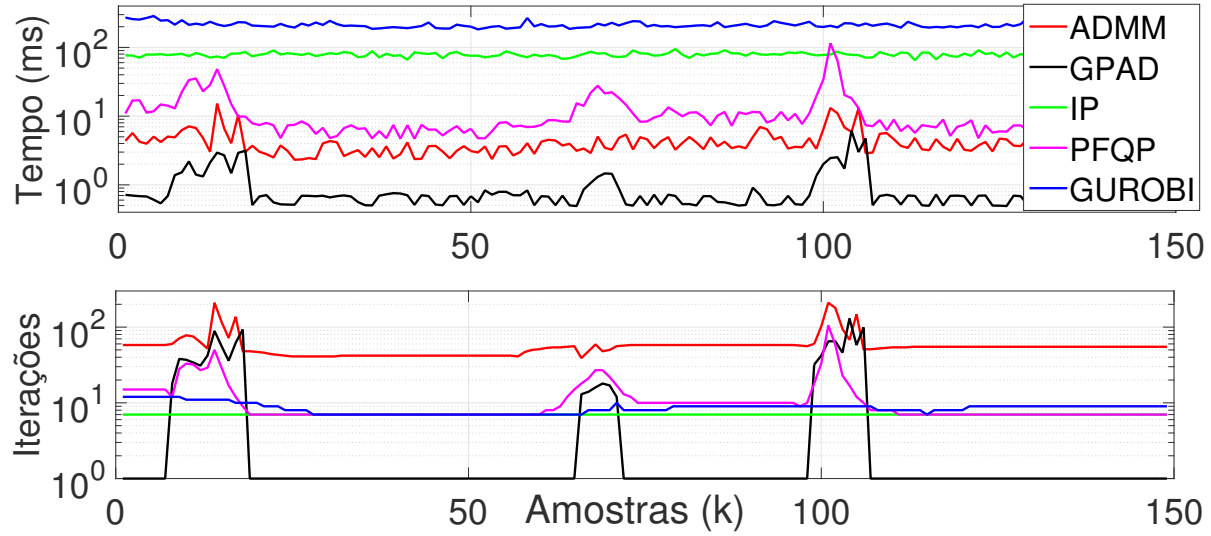
(b) Tempos de cômputo e número de iterações

Figura 20 – Comparação dos algoritmos GPC com ADMM, GPAD, IP, PFQP e Gurobi no Cenário 1.

são apresentados para cada cenário. O algoritmo GPAD foi o mais rápido em todos os cenários e, apesar de ser 8,72 vezes mais lento no terceiro cenário em relação ao primeiro, ainda se mostrou 6,29 vezes mais rápido que o IP. O ADMM e o PFQP apresentaram um tempo de cômputo 41,03 e 97,02 vezes mais lentos entre o cenário 3 e o cenário 1, respectivamente. Esse comportamento evidencia a sensibilidade desses algoritmos em relação às tolerâncias mais apertadas. Por outro lado, os algoritmos IP e Gurobi apresentaram aumentos mais modestos de 2,84 e 1,11 vezes, respectivamente, no terceiro cenário.



(a) Sinais de saída e incrementos de controle



(b) Tempos de cômputo e número de iterações

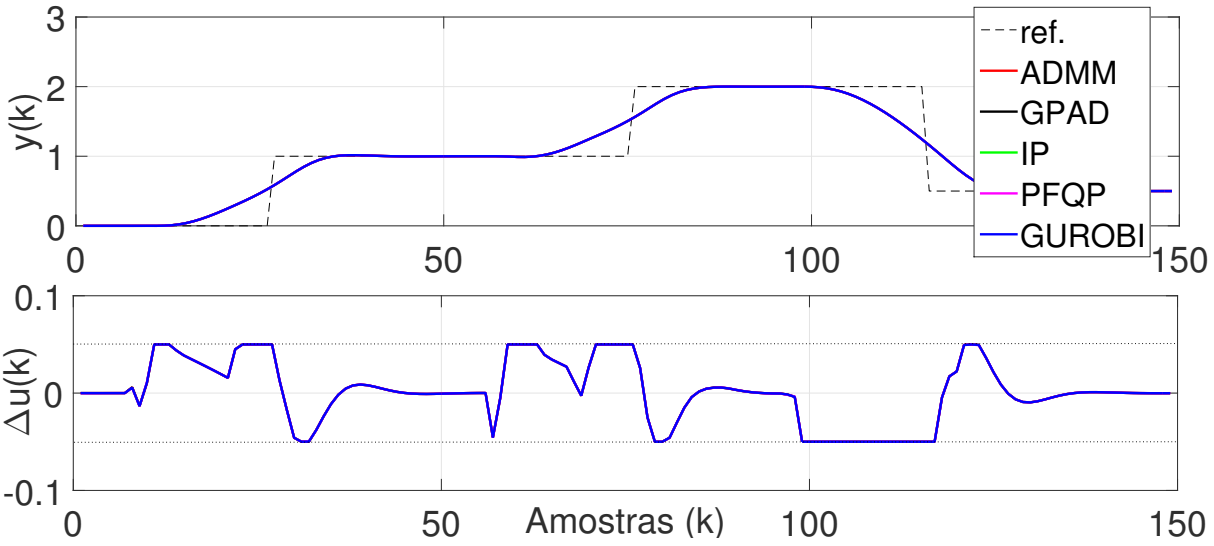
Figura 21 – Comparação dos algoritmos GPC com ADMM, GPAD, IP, PFQP e Gurobi no Cenário 2.

Tabela 9 – Tempos máximos de cômputo em milissegundos dos algoritmos GPC com ADMM, GPAD, IP, PFQP e Gurobi.

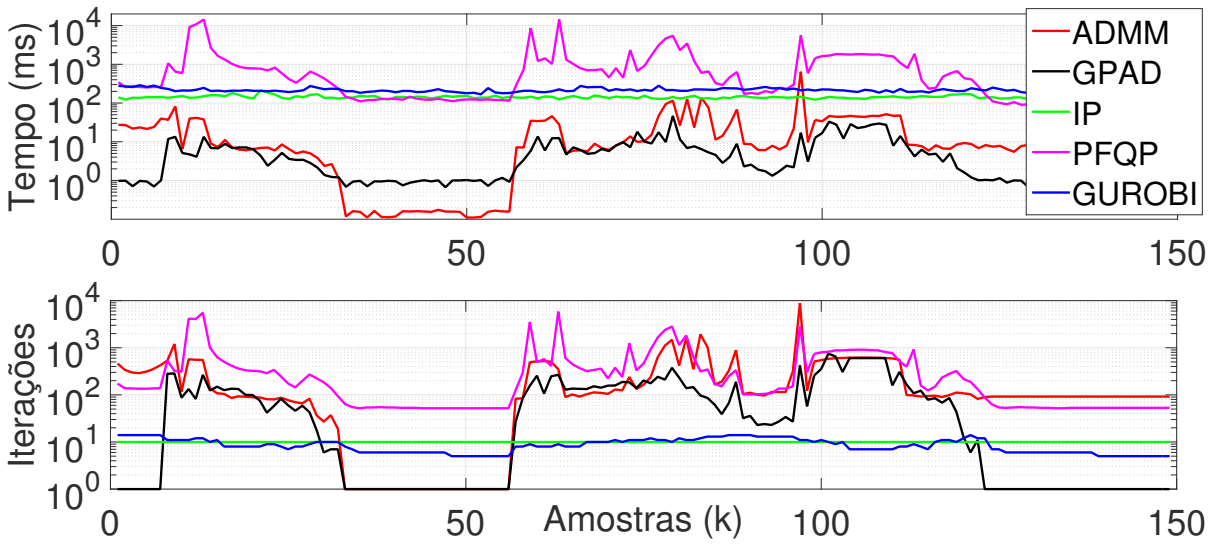
	Cenário 1	Cenário 2	Cenário 3
GPAD	3,31	6,16	28,87
ADMM	15,10	17,08	619,61
IP	63,92	98,00	181,46
PFQP	120,77	102,73	11717,00
Gurobi	242,86	259,31	269,61

7.3 CONCLUSÃO

Foram apresentados algoritmos GPC com os métodos de barreira e PFQP. Uma comparação de desempenho dos algoritmos com o GPCGPAD e GPCADMM foi feita



(a) Sinais de saída e incrementos de controle



(b) Tempos de cômputo e número de iterações

Figura 22 – Comparação dos algoritmos GPC com ADMM, GPAD, IP, PFQP e Gurobi no Cenário 3.

em um estudo de caso simulado em MATLAB. A partir do comportamento dos sinais em simulação, conclui-se que todos os algoritmos poderiam ser soluções plausíveis para o estudo de caso proposto, pois todos apresentaram um comportamento equivalente na saída da planta, mesmo com as tolerâncias mais baixas e uma menor precisão nos algoritmos de primeira ordem. O algoritmo GPCIP proposto, apesar de ser mais lento, torna-se uma boa opção para garantir o determinismo no pior caso do tempo de cômputo com baixa variabilidade nos momentos de restrição ativa. Outra vantagem evidenciada apresenta-se nos casos que requerem uma maior precisão na solução. O algoritmo GPCPFQP proposto apresentou um melhor desempenho com a restrição na variável de saída, entretanto apresentou uma grande sensibilidade ao aumento da exigência na tolerância. Esse algoritmo pode ser uma boa opção em casos de

baixa precisão e restrições complexas. A partir da comparação, pode-se concluir que os algoritmos estudados têm potencial para compor um conjunto de soluções com características distintas para a implementação dos controladores GPC. Dessa forma, o projetista pode escolher o algoritmo de acordo com os requisitos de projeto do controlador e aumentar a aplicação dos controladores preditivos com restrições em plantas com dinâmicas rápidas. No Capítulo 9 é apresentada uma tabela qualitativa para orientar a escolha do algoritmo.

8 ARTIGO 5 - ALGORITMO DE CONTROLE POR MATRIZ DINÂMICA DE CÔMPUTO RÁPIDO COM OTIMIZAÇÃO ONLINE

Neste capítulo são apresentados os desenvolvimentos e resultados traduzidos do artigo originalmente intitulado *Fast Constrained Dynamic Matrix Control Algorithm with Online Optimization* (PECCIN *et al.*, 2021b). O artigo foi submetido no periódico *Computers & Chemical Engineering* e até o momento da publicação desta tese encontra-se em fase de avaliação. O principal objetivo desse artigo é propor um algoritmo DMC de cômputo rápido para o caso com restrições, o qual faz uso do método do gradiente projetado acelerado dual (GPAD) para resolver o QP resultante a cada período de amostragem. O algoritmo resultante é chamado DMCGPAD e pode ser utilizado para controlar sistemas MIMO com dinâmicas rápidas. Além disso, ele é adequado para aplicações embarcadas, uma vez que atende a alguns requisitos para esse tipo de aplicação como a simplicidade para gerar o código e a necessidade de pouco espaço de memória para armazenar os dados que definem o problema de otimização e o código que implementa a solução. A formulação proposta foi avaliada em simulação com um exemplo de uma coluna de destilação e comparada com soluções de MPC explícito. Os resultados mostram que os algoritmos propostos convergem em tempos menores, enquanto apresentam desempenhos equivalentes. Além disso, o DMC foi implementado em um FPGA e o QP resultante foi computado em microssegundos, viabilizando o uso do DMC em processos rápidos. Portanto, as três principais contribuições desse artigo podem ser destacados como segue:

- proposição de um algoritmo eficiente de MIMO DMC com restrições, moldado para requerer apenas recursos computacionais compatíveis com sistemas embarcados;
- comparação do algoritmo proposto com outras abordagens de computação rápida utilizando o exemplo de uma coluna de destilação como referência;
- apresentação de resultados experimentais com os algoritmos embarcados em um FPGA.

8.1 ALGORITMO DMCGPAD

Nesta seção é apresentado um algoritmo para o DMC com o método do gradiente projetado acelerado dual. O GPAD foi proposto por Patrinos e Bemporad (2014) e utiliza a forma dual do QP com o método de aceleração proposto por Nesterov (1983). O GPAD é adequado para o MPC com computação eficiente por ser simples, por ser fácil de codificar e seu custo computacional crescer linearmente com o horizonte de

controle. Além disso, ele permite uma estimativa do número máximo de iterações para se obter o valor ótimo para uma dada precisão desejada.

O passo principal do método de gradiente projetado é dado pela solução do problema sem restrições na direção de descenso seguido pela projeção no conjunto de restrições ativas,

$$\mathbf{x}_{j+1} = P\left(\mathbf{x}_j - l_j \nabla h(\mathbf{x}_j)\right), \tag{165}$$

onde P é uma função de projeção ortogonal, $l_j > 0$ é o tamanho do passo e $-\nabla h$ representa o antigradiente da função h.

O GPAD utiliza a forma dual do problema QP para simplificar a função de projeção. O dual da função (15) é dada por:

$$h_d(\psi) = -\left(\frac{1}{2}\psi^T \bar{\mathbf{R}}\mathbf{H}^{-1}\bar{\mathbf{R}}^T\psi + (\bar{\mathbf{R}}\mathbf{H}^{-1}\mathbf{b} + \bar{\mathbf{r}})^T\psi + \frac{1}{2}\mathbf{b}^T\mathbf{H}^{-1}\mathbf{b}\right), \tag{166}$$

onde $\psi \in \mathbb{R}^{n_{rin}}$ é o vetor de multiplicadores de Lagrange associados às restrições de desigualdade e à função Lagrangiana:

$$\mathcal{L}(\Delta\mathbf{u},\psi) = \frac{1}{2}\Delta\mathbf{u}^T\mathbf{H}\Delta\mathbf{u} + \mathbf{b}^T\Delta\mathbf{u} + \psi^T(\bar{\mathbf{R}}\Delta\mathbf{u} - \bar{\mathbf{r}}).$$

O QP dual de (16) pode ser representado como:

$$\begin{aligned} \min_{\psi} \quad & -h_d(\psi) \\ \text{s.a.} \quad & \psi \geq \mathbf{0}. \end{aligned} \tag{167}$$

O passo principal do GPAD (165) para o problema de QP dual (167) se torna:

$$\psi_{j+1} = \max\left(\psi_j - \frac{1}{L}(\bar{\mathbf{R}}\mathbf{H}^{-1}\bar{\mathbf{R}}^T\psi_j + \bar{\mathbf{R}}\mathbf{H}^{-1}\mathbf{b} + \bar{\mathbf{r}}), \mathbf{0}\right), \tag{168}$$

onde o operador max, definido para ser elemento por elemento, é a função de projeção, $\mathbf{0}$ é um vetor com todos os elementos iguais a zero e dimensões apropriadas e $1/L$ representa o tamanho do passo. L é uma constante Lipschitz e pode ser obtida a partir formação da matriz Hessiana de (166) e do cômputo da sua norma espectral, $L = \|\bar{\mathbf{R}}\mathbf{H}^{-1}\bar{\mathbf{R}}^T\|_2$.

A equação (168) pode ser dividida em dois passos:

$$\begin{aligned} \Delta\mathbf{u}_j &= -\mathbf{H}^{-1}\left(\mathbf{b} + \bar{\mathbf{R}}^T\psi_j\right), \\ \psi_{j+1} &= \max\left(\psi_j + \frac{1}{L}(\bar{\mathbf{R}}\Delta\mathbf{u}_j + \bar{\mathbf{r}}), \mathbf{0}\right). \end{aligned} \tag{169}$$

O passo de aceleração do GPAD, baseado no método de Nesterov, é feito com uma operação *a priori*, a qual é inserida no cômputo de ψ:

$$\mathbf{a} = \psi_j + \beta(\psi_j - \psi_{j-1}), \tag{170}$$

onde β pode ser computado de forma simples, $\beta = \frac{j-1}{j+2}$, e garante a convergência do algoritmo (PATRINOS; BEMPORAD, 2014).

As propriedades de factibilidade e otimalidade primal foram utilizadas como critério de parada dos algoritmos. Para a factibilidade primal, é possível testar se as restrições são atendidas dentro da tolerância desejada ϵ_f, ou seja:

$$\bar{\mathbf{R}}\Delta\mathbf{u}_j - \bar{\mathbf{r}} \leq \epsilon_f. \tag{171}$$

Para o critério de otimalidade primal, uma propriedade derivada da dualidade forte pode ser utilizada, o que implica:

$$h(\Delta\mathbf{u}) - h(\Delta\mathbf{u}^*) \leq h(\Delta\mathbf{u}) - h_d(\boldsymbol{\psi}), \tag{172}$$

onde h é a função custo primal, h_d é a função custo dual e $\Delta\mathbf{u}^*$ representa o valor ótimo de $\Delta\mathbf{u}$. Portanto, para se obter uma solução ϵ_o ótima, uma condição necessária é que:

$$h(\Delta\mathbf{u}) - h_d(\boldsymbol{\psi}) \leq \epsilon_o. \tag{173}$$

Substituindo (15) e (166) em (173) e usando a definição do Lagrangiano, tem-se:

$$-\boldsymbol{\psi}^T(\bar{\mathbf{R}}\Delta\mathbf{u}_j - \bar{\mathbf{r}}) \leq \epsilon_o. \tag{174}$$

A partir dos desenvolvimentos apresentados, O DMCGPAD pode ser resumido no Algoritmo 17, onde j_{max} é o número máximo de iterações do GPAD e $\mathbf{g}^{(p,l)}$ é o vetor de coeficientes da resposta ao degrau do sistema em malha aberta. O símbolo := denota a associação de um valor a uma variável, $\lambda_{max}(\mathbf{M})$ é o máximo autovalor da matriz $\mathbf{M}$ e a notação $\mathbf{V}_{n:m}$ representa um subvetor composto pelos elementos n até m do vetor $\mathbf{V}$.

8.2 ESTUDO DE CASO

Um estudo de caso descrito em Drgoňa *et al.* (2017) foi utilizado como uma referência para comparação. O sistema é uma coluna de destilação de laboratório que faz a separação da mistura de água e metanol. A variável controlada é a temperatura no topo da coluna e a variável manipulada é a taxa de refluxo na válvula, as quais variam entre 0% e 100%. O modelo utilizado em Drgoňa *et al.* (2017) é um modelo de tempo discreto em espaço de estados linearizado com 10 estados e obtido com uma frequência de amostragem de 1 Hz. O modelo de resposta ao degrau de malha aberta $\mathbf{g}$ é obtido a partir do modelo em espaço de estados e o seu pefil pode ser visto na Figura 23.

A simulação foi feita no ambiente do MATLAB/Simulink. O Algoritmo 17 foi implementado em Matlab e traduzido em linguagem C via biblioteca *codegen*. Os valores originais para o horizonte de controle, $N_u = 5$ e horizonte de predição $N_1 = 1$ e

Algoritmo 17 MIMO DMCGPAD

Entrada: $\mathbf{g}^{(1,1)}$, ..., $\mathbf{g}^{(n_o,n_i)}$, $\bar{\mathbf{R}}$, $\bar{\mathbf{r}}$, $u^{(1)}(k-1)$, ..., $u^{(n_i)}(k-1)$
Saída: $u^{(1)}(k)$,...,$u^{(n_o)}(k)$
Dados: $\mathbf{Q}_\lambda$, $\mathbf{Q}_\delta$, $\mathbf{N_u}$, $\mathbf{N_1}$, $\mathbf{N_2}$, $\mathbf{N_{ss}}$, $\boldsymbol{\psi}_0 = \boldsymbol{\psi}_{-1} = \mathbf{0}$, $\boldsymbol{\epsilon_f}$, $\boldsymbol{\epsilon_o}$, j_{max}
início
  para $p = 1 : n_o$ **faça**
    $N^{(p)} \leftarrow N_2^{(p)} - N_1^{(p)} + 1$;
    para $l = 1 : n_i$ **faça**
      para $i = 1 : N_u^{(l)}$ **faça**
        $\mathbf{G}^{(p,l)}_{i:N^{(p)},i} \leftarrow \mathbf{g}^{(p,l)}_{N_1^{(p)}:N_2^{(p)}+1-i}$;
    $\mathbf{G}_{mpl} \leftarrow \mathbf{G}^{(pl)}$;
    $\hat{\mathbf{f}}^{(p)} \leftarrow \mathbf{1}_{N_{ss}^{(p)}} y^{(p)}(0)$;
  $\mathbf{H} \leftarrow (\mathbf{G}_m^T \mathbf{Q}_\delta \mathbf{G}_m + \mathbf{Q}_\lambda)$;
  $\breve{\mathbf{H}} \leftarrow \mathbf{H}^{-1}$;
  $L \leftarrow \sqrt{\lambda_{max}((\bar{\mathbf{R}}\breve{\mathbf{H}}\bar{\mathbf{R}}^T)^2)}$;
  $k \leftarrow 0$;
  enquanto *modo automático* **faça**
    se *período de amostragem* **então**
      Obtém as saídas $y^{(1)}(k)$ ... $y^{(n_o)}(k)$;
      Define as referências $\mathbf{w}_m \leftarrow \left[\mathbf{w}^{(1)}(k) \ ... \ \mathbf{w}^{(n_o)}(k)\right]^T$;
      para $p = 1 : n_o$ **faça**
        $\hat{\mathbf{f}}^{(p)} \leftarrow \hat{\mathbf{f}}^{(p)} + \mathbf{g}^{(p)} \Delta u^{(p)}(k-1)$;
        $e^{(p)} \leftarrow (y^{(p)}(k) - \hat{f}_0^{(p)})$;
        $\hat{\mathbf{f}}^{(p)} \leftarrow [\hat{f}_1^{(p)} \ ... \ \hat{f}^{(p)}_{N_{ss}^{(p)}-1} \ \hat{f}^{(p)}_{N_{ss}^{(p)}-1}]^T$;
        $\mathbf{f}^{(p)} \leftarrow \hat{\mathbf{f}}^{(\mathbf{p})}_{N_1^{(p)}:N_2^{(p)}} + \mathbf{1}_{N^{(p)}\times 1} e^{(p)}$;
      $\mathbf{f}_m \leftarrow [\mathbf{f}^{(1)} \ ... \ \mathbf{f}^{(n_o)}]^T$;
      $\mathbf{b}^T \leftarrow (\mathbf{f}_m - \mathbf{w}_m)^T \mathbf{Q}_\delta \mathbf{G}_m$;
      enquanto $j < j_{max}$ **faça**
        se $j = 0$ **então**
          $\beta \leftarrow 0$;
        senão
          $\beta \leftarrow \frac{j-1}{j+2}$;
        $\mathbf{a} \leftarrow \boldsymbol{\psi}_j + \beta(\boldsymbol{\psi}_j - \boldsymbol{\psi}_{j-1})$;
        $\Delta \mathbf{u}_m \leftarrow -\breve{\mathbf{H}}(\mathbf{R}^T \mathbf{a} + \mathbf{b})$;
        $\mathbf{s} \leftarrow (1/L)(\mathbf{R}\Delta\mathbf{u}_m - \bar{\mathbf{r}})$;
        se $\mathbf{s}_i \leq \epsilon_f/L, \ \forall i = 1,...,n_{rin}$ **então**
          se $-\mathbf{a}^T\mathbf{s} < \epsilon_o/L$ **então**
            return;
        $\boldsymbol{\psi}_{j+1} \leftarrow \max(\mathbf{a} + \mathbf{s}, \mathbf{0})$;
        $j \leftarrow j + 1$;
      para $l = 1 : n_i$ **faça**
        $u^{(l)}(k) \leftarrow u^{(l)}(k-1) + \Delta u^{(l)}(k)$;
      $k \leftarrow k + 1$;
fim

$N_2 = 40$, foram utilizados. O mesmo é válido para as restrições $|\Delta\mathbf{u}(k+j)| \leq 10$ e $0 \leq \mathbf{u}(k+j) \leq 100$ com $j = 0,...,N_U - 1$, as quais são consideradas na formulação

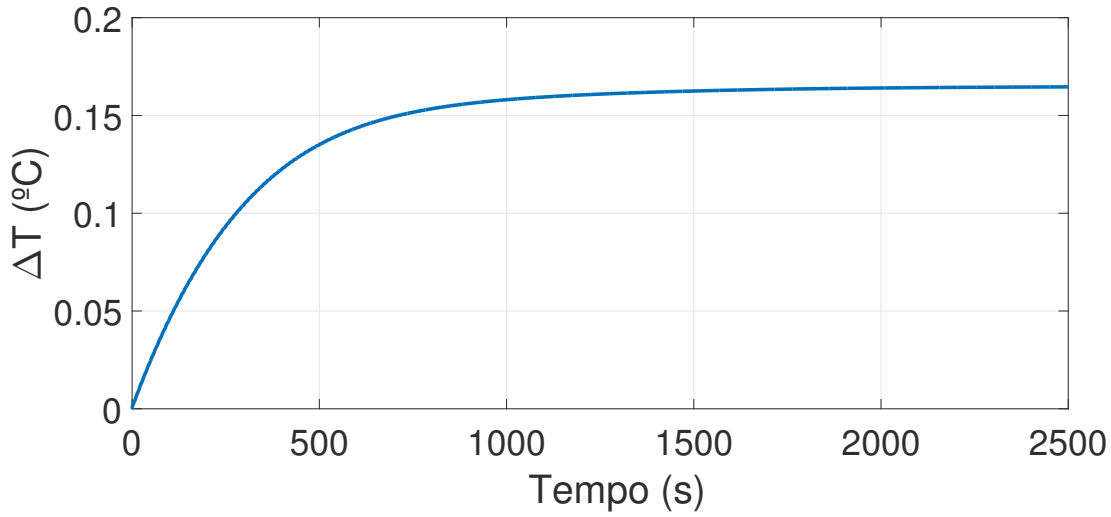


Figura 23 – Perfil da resposta ao degrau de referência para o estudo de caso da coluna de destilação.

original em Drgoňa *et al.* (2017). O horizonte do modelo foi definido como $N_{ss} = 1600$ e ambas as tolerâncias residuais foram escolhidas como $\epsilon_o = \epsilon_f = 10^{-3}$. O problema de otimização resultou em uma matriz $\mathbf{H} \in \mathbb{R}^{5\times5}$ e 20 restrições. As simulações foram feitas em um computador com um processador Intel Core i3 2.40 GHz de dois núcleos e com 12 GB de memória RAM. O tempo de cômputo foi medido utilizando a função *tic-toc* disponível no MATLAB.

Os resultados da simulação para o DMCGPAD, com os mesmos cenários de Drgoňa *et al.* (2017), podem ser vistos na Figura 24. Na Figura 24a, o perfil de temperatura e a respectiva referência são apresentados. Já na Figura 24b, é apresentada a ação de controle gerada pelo DMCGPAD. Analisando a resposta do controle, é possível verificar o rastreamento do degrau de referência, o qual é uma característica inerente ao DMC. Além disso, uma limitação na ação de controle pode ser vista nos momentos nos quais as restrições estão ativas. A comparação com o trabalho de Drgoňa *et al.* (2017) foi feita a partir da utilização de índices de desempenho. Os critérios de desempenho do rastreamento de referência utilizados em Drgoňa *et al.* (2017) foram a integral do erro quadrático (ISE, do inglês *Integral Squared Error*) por instante de amostragem e a integral do erro absoluto (IAE, do inglês *Integral Absolute Error*) por instante de amostragem. Além disso, para se avaliar a quantidade de produto utilizado como refluxo, a ação de controle média (ACA, do inglês *Average Control Action*) por instante de amostragem foi considerada, a qual é obtida calculando-se a integral da ação de controle e dividindo pelo número de instantes de amostragem. Uma comparação entre o DMCGPAD proposto e o ssMPC explícito considerado em Drgoňa *et al.* (2017) foi feita baseando-se nos critérios de desempenho detalhados acima e são apresentados na Tabela 10. A partir dos resultados do ISE, é possível verificar que o valor para o ssMPC explícito é menor do que o valor associado ao DMCGPAD. Entretanto, o oposto

é observado para o critério IAE. Para o critério de produção ACA, o DMCGPAD apresenta um valor maior. Essas diferenças são também esperadas devido à comparação entre um modelo simulado e uma planta real. Apesar dessas diferenças, o desempenho do controle pode ser considerado equivalente para ambas as abordagens. Esses resultados também enfatizam que a subotimalidade dos métodos de primeira ordem utilizados no problema de otimização não afetam significativamente o desempenho do controle. Uma razão para isso é que apenas a primeira ação de controle do horizonte é aplicada (WANG; BOYD, 2010). De fato, em aplicações práticas de controle realimentado, existem também algumas limitações físicas, tais como a resolução dos atuadores e sensores, o que implica que uma solução exata do sinal de controle não é necessária.

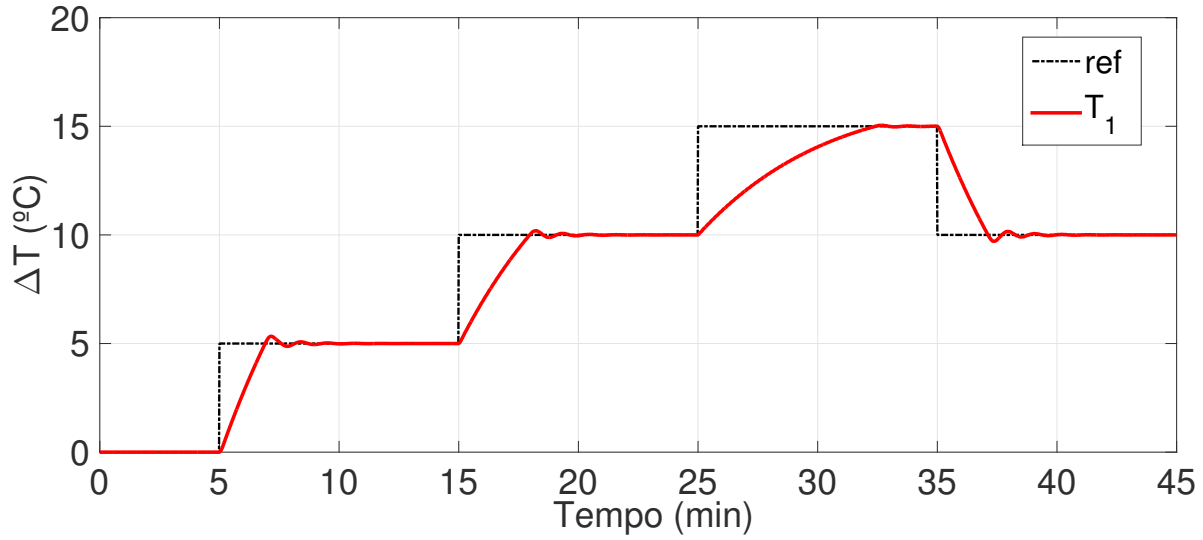
Tabela 10 – Índices de desempenho para o estudo de caso da coluna de destilação.

Método	ISE (°C^2)	IAE (°C)	ACA (%)
DMCGPAD	2,20	0,71	58,00
ssMPC explícito[1]	1,79	0,77	56,35

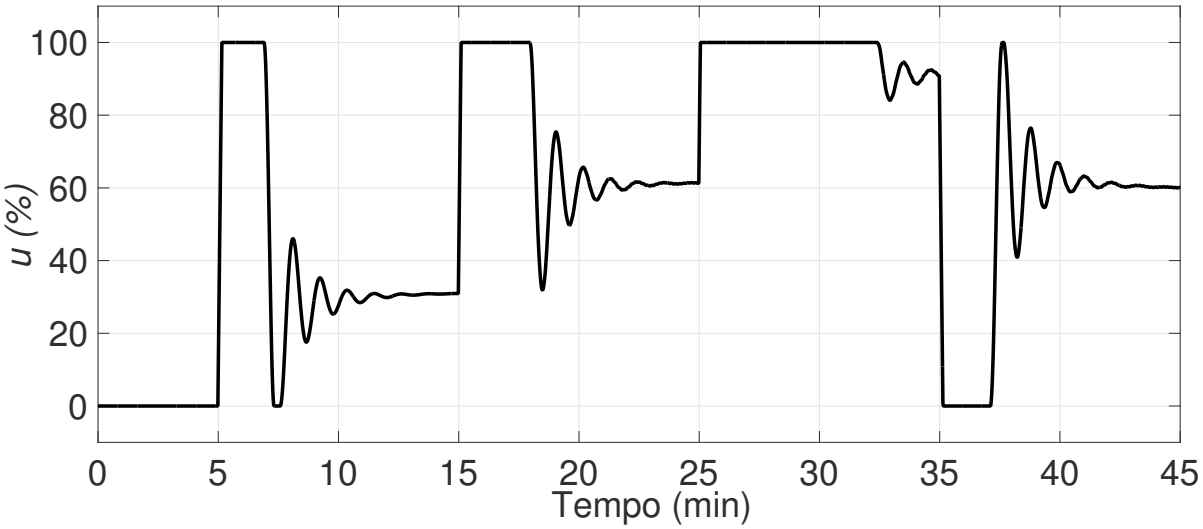
[1] Drgoňa *et al.* (2017) com $Q_y = 300$.

O perfil do tempo de computação para o DMCGPAD pode ser visto na Figura 25. Pode-se verificar que o tempo de computação tem um pico nos instantes em que há a troca em degrau de referência e apresenta valores muito menores quando as restrições não estão ativas. Essa é uma característica esperada para os métodos de primeira ordem (FERREAU *et al.*, 2017).

Na Tabela 11 é apresentada uma comparação do tempo de cômputo com os resultados de Drgoňa *et al.* (2017). O DMCGPAD apresentou um tempo de computação menor, sendo 1,86 vez mais rápido que o ssMPC explícito sem regiões e 2,11 vezes mais rápido que o ssMPC explícito baseado em regiões. Esse resultado mostra o quão rápido é o algoritmo DMCGPAD, mesmo com o processo de otimização com restrições calculado de forma online. É importante mencionar que os resultados de Drgoňa *et al.* (2017) foram obtidos em outra máquina o que complica a comparação direta de tempo, que depende de diversos fatores. Foi utilizado um hardware, Intel Core i7 de 2.30 GHz de quatro núcleos. Apesar das diferenças nos sistemas utilizados para a computação, é possível afirmar que o DMCGPAD apresenta uma solução rápida com a mesma ordem de grandeza de um ssMPC explícito para o estudo de caso considerado. Além disso, a abordagem online pode ser utilizada em sistemas MIMO de dimensões médias e grandes, casos nos quais a aplicação do ssMPC explícito é limitada por conta da explosão combinatória.



(a) Perfil de temperatura



(b) Perfil da ação de controle

Figura 24 – Resposta do controle para o DMCGPAD no estudo de caso da coluna de destilação.

Tabela 11 – Comparação dos tempos de execução do DMCGPAD, ssMPC explícito sem regiões e ssMPC explícito baseado em regiões para o estudo de caso da coluna de destilação.

Método	WCET
DMCGPAD	1,06 ms
ssMPC explícito sem regiões[1]	1,95 ms
ssMPC explícito baseado em regiões[1]	2,24 ms

[1] Drgoňa *et al.* (2017).

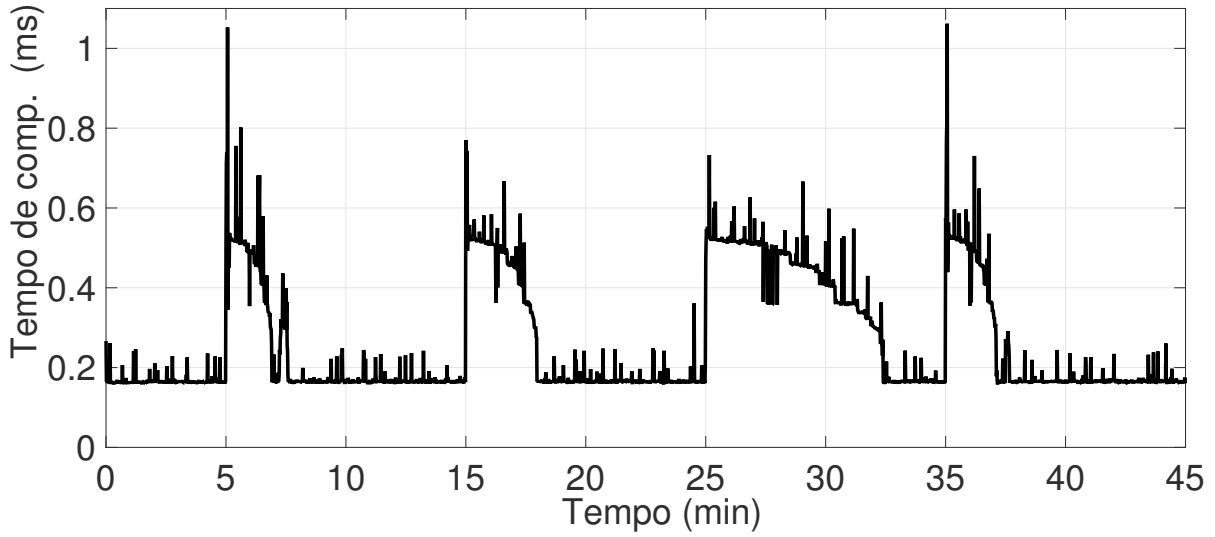


Figura 25 – Perfil do tempo de execução para o DMCGPAD no estudo de caso da coluna de destilação.

8.3 IMPLEMENTAÇÃO EM FPGA

O algoritmo DMCGPAD foi embarcado em um FPGA e o tempo de computação foi avaliado. Essa implementação foi possível uma vez que o algoritmo proposto apenas utiliza recursos computacionais e operações modestas. Um FPGA da Altera, família MAX 10, modelo 10M50DAF484C7G, foi utilizado. A família MAX 10 contém conversor analógico/digital e 50 000 elementos lógicos programáveis. O Algoritmo 17 foi implementado com uma linguagem de descrição de hardware chamada Verilog. Uma aritmética de ponto fixo de 16 bits foi utilizada para representar os números, com 10 bits para os decimais. Uma máquina de estados foi utilizada para sincronizar o algoritmo e todas as operações matriciais foram implementadas em computação paralela.

Os parâmetros de configuração do controlador são passados por comunicação externa (USB-Blaster) com o processador NIOS II implementado em um FPGA. A comunicação entre os blocos lógicos do DMC e o código embarcado do NIOS II foi feito via barramento Avalon. O bloco DMC instancia os blocos do GPAD e da resposta livre. As bibliotecas de multiplicação matricial e operações básicas em aritmética de ponto fixo foram também implementadas diretamente em lógica customizada. O algoritmo de multiplicação matricial paralela apresenta uma propriedade de complexidade dada por $O(\lceil \log_{10}(n) \rceil + 1)$ (PECCIN *et al.*, 2020b).

O pior caso observado em simulação para o estudo de caso em questão, com 718 iterações do GPAD, foi replicado para a implementação em FPGA. Os resultados obtidos, com os WCETs nas funções principais, são apresentados na Tabela 12. O tempo de execução é rápido, com um valor máximo de 258.76 μs, permitindo o uso do DMC em processos com dinâmicas rápidas utilizando períodos de amostragem

menores que 1 ms.

Tabela 12 – Tempo de execução do DMCGPAD em um FPGA.

Função	WCET observado
DMC	0,14 μs
GPAD	258,52 μs
Cálculo da resposta livre	0,10 μs
DMCGPAD (total)	258,76 μs

8.4 CONCLUSÃO

Este capítulo apresentou um algoritmo DMC baseado no método do gradiente projetado acelerado dual, chamado DMCGPAD. Em comparação com uma abordagem ssMPC explícito apresentada em Drgoňa *et al.* (2017), o algoritmo apresentou resultados similares em termos do sinal de controle resultante para o estudo de caso da coluna de destilação considerada nesse artigo. Além disso, o algoritmo DMCGPAD apresentou um tempo de execução pequeno considerando a otimização online, com a mesma ordem de grandeza observada no ssMPC explícito, o qual utiliza uma abordagem offline. A vantagem de utilizar o DMCGPAD ao invés do ssMPC explícito é reforçada para casos de problemas com dimensões que esbarram em questões de espaço de memória para o armazenamento de dados. A implementação em um FPGA Max 10 provou ser rápida, com um tempo de cômputo de 258.76 μs para o exemplo apresentado. Estes resultados contribuem para a utilização do DMC nos níveis mais baixos da automação e assim poder aproveitar as suas vantagens inerentes em sistemas que são tipicamente operados com controladores clássicos, devido à existência de dinâmicas rápidas.

9 CONSIDERAÇÕES FINAIS

Neste capítulo final da tese são discutidas as principais conclusões da pesquisa e as propostas para trabalhos futuros. Como no final de cada capítulo já são apresentadas as conclusões específicas de cada artigo, aqui são abordadas as conclusões e observações mais gerais sobre as contribuições.

9.1 CONCLUSÕES

O problema de pesquisa estava focado em como aplicar controladores preditivos, com restrições e formulações mais difundidas no meio industrial, em plantas com dinâmicas rápidas. O problema foi abordado tentando simplificar e reduzir o tempo de cômputo de três etapas: na obtenção das matrizes que definem o problema de controle a partir do modelo da planta; no método de otimização com restrições; na plataforma de implementação dos algoritmos. Pode-se dizer que os resultados foram alcançados com a disponibilização de algoritmos rápidos, completos, com otimizadores de características distintas e implementáveis até em arquiteturas paralelas e determinísticas de descrição em hardware. Os algoritmos propostos atenderam os requisitos para serem embarcáveis, sendo possível aplicá-los em plantas com dinâmicas da ordem de mili ou microssegundos.

Foram abordadas as formulações do GPC e do DMC por serem as mais utilizadas na indústria. Os métodos de otimização de primeira ordem foram os que tiveram os melhores resultados, destacando-se o GPAD, o FAMA e o ADMM. Para os dois primeiros, foi possível obter um limite superior do número de iterações necessários para o algoritmo entregar uma solução com uma dada tolerância. Essa característica é importante para garantir requisitos rígidos de tempo real. Dessa forma, para uma aplicação com dinâmica rápida, sugere-se que os primeiros algoritmos a serem avaliados sejam os baseados no FAMA e GPAD. O ADMM, além de não possuir uma certificação no número máximo de iterações, também apresenta o inconveniente de necessitar do ajuste de um parâmetro ρ, que nem sempre tem uma escolha trivial. O algoritmo baseado no método de barreira é uma solução que pode ser útil nos casos onde não são desejadas grandes variações no número de iterações do otimizador, apesar de ter um tempo de cômputo médio mais elevado. De modo a orientar a escolha do algoritmo de controle, na Tabela 13 é apresentada uma comparação qualitativa que destaca os algoritmos mais adequados com as métricas consideradas relevantes para a implementação.

Foram propostas arquiteturas de implementação em FPGA com características de paralelismo e determinismo e foram fornecidas equações para definir o tempo de cômputo das operações. Os esforços em implementar até operações básicas em hardware, como a multiplicação matricial paralela, foram fundamentais para os resultados

Tabela 13 – Tabela qualitativa para orientar a escolha do algoritmo de otimização.

Requisito	GPAD	FAMA	ADMM	IP	PFQP
Maior rapidez	✓	✓	✓	-	-
Menos recursos computacionais	✓	✓	✓	-	✓
Mais simples de codificar	✓	✓	✓	-	✓
Previsibilidade WCET	✓	✓	-	✓	-
Menor variabilidade temporal	-	-	-	✓	-
Maior precisão	-	-	-	✓	-
Paralelismo	-	✓	✓	-	✓

obtidos. Outra contribuição inédita que vale ser destacada foi o precondicionamento proposto que permite uma diminuição de até 11% no tempo de cômputo na arquitetura de implementação proposta. A aplicação em um estudo de caso simulado de um inversor trifásico com filtro LCL e conectado à rede apresentou bons resultados, abrindo uma possibilidade interessante de utilização do GPC e os seu benefícios inerentes em um equipamento central na área de energias renováveis. Os testes do controlador implementado em FPGA com uma planta real foram iniciados com um equipamento do tipo hardware no laço (HIL, do inglês *Hardware-In-the-Loop*) do inversor em frequência. Na ocasião, foram obtidos bons resultados com o controlador atingindo os resultados de tempo de computação previstos pelas equações das arquiteturas propostas, na ordem de microssegundos. O comportamento do controle também ficou dentro do esperado, entretanto, não foi utilizada a transformação em coordenadas *dq* e o GPC, na sua formulação convencional, não é capaz de fazer o seguimento de referência senoidal. O experimento prático foi descontinuado devido à pandemia e não foi possível completar os estudos para apresentar os resultados na presente tese. Trabalhos futuros devem investigar o comportamento das propostas de implementação dos algoritmos em aritmética de ponto fixo em diferentes plantas.

A produção de artigos diretamente ligados à pesquisa superou as expectativas, sendo produzidos seis artigos, dos quais três foram apresentados oralmente em congressos nacionais e internacionais, dois foram publicados em periódicos internacionais e o último foi submetido a um periódico internacional e encontra-se em fase de avaliação. O número de artigos publicados em veículos de organizações conceituadas indica a boa produção e o caráter inédito da pesquisa.

9.2 PROPOSTAS PARA TRABALHOS FUTUROS

Algumas propostas de trabalhos futuros para a continuação da pesquisa podem ser citados:

- a inclusão nos algoritmos de alguma forma de detectar e tratar casos de proble-

mas de programação quadrática infactíveis que podem aparecer;

- o estudo de melhoria de desempenho pela paralelização dos métodos de partição quando utilizados em problemas de grande dimensão;

- a avaliação da sensibilidade dos algoritmos quanto à escolha do tamanho das palavras em implementações com aritmética de ponto fixo;

- a inclusão, nos algoritmos, de métodos de garantia de robustez às variações paramétricas no modelo de predição;

REFERÊNCIAS

AHMADI-MOSHKENANI, P.; JOHANSEN, T. A.; OLARU, S. Combinatorial Approach Toward Multiparametric Quadratic Programming Based on Characterizing Adjacent Critical Regions. **IEEE Transactions on Automatic Control**, v. 63, n. 10, p. 3221–3231, 2018.

ALTERA, Corporation. **Altera Stratix IV Device Handbook**. [*S.l.*: *s.n.*], 2016. Disponível em: `https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/stratix-iv/stratix4_handbook.pdf`.

BEMPORAD, A. A Numerically Stable Solver for Positive Semidefinite Quadratic Programs Based on Nonnegative Least Squares. **IEEE Transactions on Automatic Control**, v. 63, n. 2, p. 525–531, fev. 2018.

BEMPORAD, A. A Quadratic Programming Algorithm Based on Nonnegative Least Squares With Applications to Embedded Model Predictive Control. **IEEE Transactions on Automatic Control**, v. 61, n. 4, p. 1111–1116, abr. 2016.

BEMPORAD, A.; MORARI, M.; DUA, V.; PISTIKOPOULOS, E. N. The explicit linear quadratic regulator for constrained systems. **Automatica**, v. 38, n. 1, p. 3–20, 2002.

BEMPORAD, A.; PATRINOS, P. Simple and Certifiable Quadratic Programming Algorithms for Embedded Linear Model Predictive Control. *In*: PROC. 4th IFAC Nonlinear Model Predictive Control Conference. Noordwijkerhout, The Netherlands: IFAC, ago. 2012. P. 14–20.

BORRELLI, F.; BAOTIĆ, M.; PEKAR, J.; STEWART, G. On the computation of linear model predictive control laws. **Automatica**, v. 46, n. 6, p. 1035–1041, 2010.

BOYD, S.; PARIKH, N.; CHU, E.; PELEATO, B.; ECKSTEIN, J. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. **Foundations and Trends on Machine Learning**, Now Publishers Inc., Hanover, MA, USA, v. 3, n. 1, p. 1–122, jan. 2011.

BOYD, S.; VANDENBERGHE, L. **Convex Optimization**. Cambridge, MA: Cambridge Univ. Press, mar. 2004.

BRAND, M.; CHEN, D. Parallel quadratic programming for image processing. *In*: 2011 18th IEEE International Conference on Image Processing. Brussels, Belgium: IEEE, set. 2011. P. 2261–2264.

CAI, X.; TIPPETT, M. J.; XIE, L.; BAO, J. Fast distributed MPC based on active set method. **Computers & Chemical Engineering.**, v. 71, p. 158–170, 2014.

CAIRANO, S. D.; BRAND, M.; BORTOFF, S. A. Projection-free parallel quadratic programming for linear model predictive control. **International Journal of Control**, v. 86, n. 8, p. 1367–1385, 2013.

CAMACHO, E. F.; BORDONS, C. **Model Predictive Control**. London: Springer, 2004. (Advanced Textbooks in Control and Signal Processing).

CAVANINI, L.; CIMINI, G.; IPPOLITI, G.; BEMPORAD, A. Model predictive control for pre-compensated voltage mode controlled DC–DC converters. **IET Control Theory & Applications**, v. 11, n. 15, p. 2514–2520, 2017.

CIMINI, G.; BEMPORAD, A. Exact Complexity Certification of Active-Set Methods for Quadratic Programming. **IEEE Transactions on Automatic Control**, v. 62, n. 12, p. 6094–6109, dez. 2017.

CLARKE, D W; MOHTADI, C; TUFFS, P S. Generalized Predictive Control - Part 1: The Basic Algorithm. **Automatica**, v. 23, n. 2, p. 137–148, mar. 1987.

COMBETTES, P. L.; PESQUET, J. C. Proximal Splitting Methods in Signal Processing. *In*: BAUSCHKE, H. H.; BURACHIK, R. S.; COMBETTES, P. L.; E., V.; LUKE, D. R.; WOLKOWICZ, H. (Ed.). **Fixed-Point Algorithms for Inverse Problems in Science and Engineering**. New York: Springer, 2011. v. 49. (Springer Optimization and Its Applications). P. 185–212.

CUTLER, C. R.; RAMAKER, B. L. Dynamic matrix control: A computer control algorithm. *In*: PROCEEDINGS of the Joint Automatic Control Conference. San Francisco, USA: AIChE, 1980. P. 72.

DANTZIG, G. B. **Linear programming and extensions**. Princeton, NJ: Princeton Univ. Press, 1963. (Rand Corporation Research Study).

DEVOLDER, O.; GLINEUR, F.; NESTEROV, Y. First-order methods of smooth convex optimization with inexact oracle. **Mathematical Programming.**, v. 146, n. 1, p. 37–75, ago. 2014.

DRGOŇA, J.; KLAUČO, M.; JANEČEK, F.; KVASNICA, M. Optimal control of a laboratory binary distillation column via regionless explicit MPC. **Computers & Chemical Engineering**, v. 96, p. 139–148, 2017.

FERNANDES, D.; HAQUE, M. E.; PALANKI, S.; RIOS, S. G.; CHEN, D. DMC controller design for an integrated Allam cycle and air separation plant. **Computers & Chemical Engineering**, v. 141, p. 107019, 2020.

FERREAU, H. J.; ALMÉR, S.; VERSCHUEREN, R.; DIEHL, M.; FRICK, D.; DOMAHIDI, A.; JEREZ, J. L.; STATHOPOULOS, G.; JONES, C. Embedded Optimization Methods for Industrial Automatic Control. *In*: PROC. 20th IFAC World Congress. Toulouse, France: IFAC, dez. 2017.

FIACCO, A. V.; MCCORMICK, G. P. **Nonlinear Programming**: Sequential Unconstrained Minimization Techniques. New York, NY, USA: John Wiley & Sons, 1968. Reprinted by SIAM Publications in 1990.

FIACCO, A. V.; MCCORMICK, G. P. **Nonlinear Programming**. Philadelphia: Society for Industrial e Applied Mathematics, 1990.

GOLDSTEIN, A. A. Convex programming in Hilbert space. **Bulletin of the American Mathematical Society**, v. 70, n. 5, p. 709–710, set. 1964.

GOLDSTEIN, T.; O'DONOGHUE, B.; SETZER, S.; BARANIUK, R. Fast Alternating Direction Optimization Methods. **SIAM Journal on Imaging Sciences**, v. 7, n. 3, p. 1588–1623, 2014.

GOLUB, G. H.; VAN LOAN, C. F. **Matrix Computations**. 3. ed. Baltimore: The Johns Hopkins University Press, 1996.

GULBUDAK, O.; SANTI, E. FPGA-Based Model Predictive Controller for Direct Matrix Converter. **IEEE Transactions on Industrial Electronics**, v. 63, n. 7, p. 4560–4570, jul. 2016.

GUROBI OPTIMIZATION, LLC. **Gurobi Optimizer Reference Manual**. [*S.l.*: *s.n.*], 2021. Disponível em: `http://www.gurobi.com`.

GUZMAN, R.; DE VICUÑA, L. G.; CAMACHO, A.; MIRET, J.; REY, J. M. Receding-Horizon Model-Predictive Control for a Three-Phase VSI With an LCL Filter. **IEEE Transactions on Industrial Electronics**, v. 66, n. 9, p. 6671–6680, 2019.

HARTLEY, E. N.; MACIEJOWSKI, J. M. Field programmable gate array based predictive control system for spacecraft rendezvous in elliptical orbits. **Optimal Control Applications and Methods**, v. 36, n. 5, p. 585–607, 2015.

HE, X.; LIMA, F. V. Development and implementation of advanced control strategies for power plant cycling with carbon capture. **Computers & Chemical Engineering**, v. 121, p. 497–509, 2019.

HERCEG, M.; JONES, C. N.; MORARI, M. Dominant speed factors of active set methods for fast MPC. **Optimal Control Applications and Methods**, v. 36, n. 5, p. 608–627, 2015.

IZMAILOV, A.; SOLODOV, M. **Otimização-Volume 1-Condições de Otimalidade, Elementos de Análise Convexa e de Dualidade**. 3. ed. Rio de Janeiro: IMPA, 2014.

JEREZ, J. L.; GOULART, P. J.; RICHTER, S.; CONSTANTINIDES, G. A.; KERRIGAN, E. C.; MORARI, M. Embedded Online Optimization for Model Predictive Control at Megahertz Rates. **IEEE Transactions on Automatic Control**, v. 59, n. 12, p. 3238–3251, dez. 2014.

JIN, T.; SHEN, X.; SU, T.; FLESCH, R. C. C. Model Predictive Voltage Control Based on Finite Control Set With Computation Time Delay Compensation for PV Systems. **IEEE Transactions on Energy Conversion**, v. 34, n. 1, p. 330–338, 2019.

JOHANSEN, T. A.; JACKSON, W.; SCHREIBER, R.; TONDEL, P. Hardware architecture design for explicit model predictive control. *In*: 2006 American Control Conference. Minneapolis,USA: IFAC, jun. 2006.

JUDEWICZ, M. G.; GONZÁLEZ, S. A.; ECHEVERRÍA, N. I.; FISCHER, J. R.; CARRICA, D. O. Generalized Predictive Current Control (GPCC) for Grid-Tie Three-Phase Inverters. **IEEE Transactions on Industrial Electronics**, v. 63, n. 7, p. 4475–4484, 2016.

JUDEWICZ, M. G.; GONZÁLEZ, S. A.; FISCHER, J. R.; MARTÍNEZ, J. F.; CARRICA, D. O. Inverter-Side Current Control of Grid-Connected Voltage Source Inverters With LCL Filter Based on Generalized Predictive Control. **IEEE Journal of Emerging and Selected Topics in Power Electronics**, v. 6, n. 4, p. 1732–1743, 2018.

KUKRER, O.; BAYHAN, S.; KOMURCUGIL, H. Model-Based Current Control Strategy With Virtual Time Constant for Improved Dynamic Response of Three-Phase Grid-Connected VSI. **IEEE Transactions on Industrial Electronics**, v. 66, n. 6, p. 4156–4165, 2019.

KUMM, M.; HARDIECK, M.; ZIPF, P. Optimization of Constant Matrix Multiplication with Low Power and High Throughput. **IEEE Transactions on Computers**, v. 66, n. 12, p. 2072–2080, dez. 2017.

KVASNICA, M.; TAKÁCS, B.; HOLAZA, J.; DI CAIRANO, S. On region-free explicit model predictive control. *In*: PROC. 54th IEEE Conference on Decision and Control (CDC). Osaka, Japan: IEEE, 2015. P. 3669–3674.

LEE, J. H.; MORARI, M.; GARCIA, C. E. State-space interpretation of model predictive control. **Automatica**, v. 30, n. 4, p. 707–717, 1994.

LEVITIN, E. S.; POLYAK, B. T. Constrained minimization problems. **USSR Computational Mathematics and Mathematical Physics**, p. 1–50, 1966.

LIM, C. S.; LEE, S. S.; CASSANDRA WONG, Y. C.; NUTKANI, I. U.; GOH, H. H. Comparison of Current Control Strategies Based on FCS-MPC and D-PI-PWM Control for Actively Damped VSCs With LCL-Filters. **IEEE Access**, v. 7, p. 112410–112423, 2019.

LIMA, D. M.; NORMEY-RICO, J. E.; PLUCENIO, A.; SANTOS, T. L.; GOMES, M. V. C. Improving robustness and disturbance rejection performance with industrial MPC. *In*: PROC. 20th Brazilian Conference on Automation (CBA). Belo Horizonte, Brazil: SBA, 2014. P. 3229–3236.

LING, K. V.; YUE, S. P.; MACIEJOWSKI, J. M. A FPGA implementation of model predictive control. *In*: 2006 American Control Conference. Minneapolis,USA: IFAC, jun. 2006.

MA, Z.; SAEIDI, S.; KENNEL, R. FPGA Implementation of Model Predictive Control With Constant Switching Frequency for PMSM Drives. **IEEE Transactions on Industrial Informatics**, v. 10, n. 4, p. 2055–2063, nov. 2014.

MACCARI JR, L. A.; LIMA, D. M.; KOCH, G. G.; MONTAGNER, V. F. Robust Model Predictive Controller Applied to Three-Phase Grid-Connected LCL Filters. **Journal of Control, Automation and Electrical Systems**, Springer, v. 31, n. 2, p. 447–460, 2020.

MACIEJOWSKI, J. M. **Predictive Control with Constraints**. London: Prentice Hall, 2002.

MAYNE, D.Q.; RAWLINGS, J. B.; RAO, C. V.; SCOKAERT, P. O. M. Constrained Model Predictive Control: Stability and Optimality. **Automatica**, v. 36, p. 789–814, jun. 2000.

MOUTON, T.; GEYER, T. Trajectory-based LQR Control of a Grid-connected Converter with an LCL Filter. *In*: PROC. 6th IFAC Conf. Nonlinear Model Predictive Control. Madison, USA: IFAC, 2018. P. 273–278.

NESTEROV, Y. A method of solving a convex programming problem with convergence rate o($1/k^2$). **Soviet Mathematics Doklady**, v. 27, n. 2, p. 372–376, 1983.

NESTEROV, Y. **Introductory Lectures on Convex Optimization**: A Basic Course. Boston: Springer Publishing Company, Incorporated, 2014.

NORMEY-RICO, J. E.; CAMACHO, E. F. **Control of Dead-time Processes**. London: Springer, 2007.

NORMEY-RICO, J. E.; CAMACHO, E. F. Multivariable generalised predictive controller based on the Smith predictor. **IEE Proceedings - Control Theory and Applications**, v. 147, n. 5, p. 538–546, 2000.

O'DONOGHUE, B.; STATHOPOULOS, G.; BOYD, S. A Splitting Method for Optimal Control. **IEEE Transactions on Control Systems Technology**, v. 21, n. 6, p. 2432–2442, nov. 2013.

PARIKH, N.; BOYD, S. Proximal Algorithms. **Foundations and Trends on Optimization**, v. 1, n. 3, p. 127–239, 2014.

PATRINOS, P.; BEMPORAD, A. An Accelerated Dual Gradient-Projection Algorithm for Embedded Linear Model Predictive Control. **IEEE Transactions on Automatic Control**, v. 59, n. 1, p. 18–33, jan. 2014.

PECCIN, V. B.; LIMA, D. M.; FLESCH, R. C. C.; NORMEY-RICO, J. E. Algoritmos GPC de Cômputo Rápido com Métodos de Ponto Interior e Programação Quadrática sem Projeção. *In*: ANAIS XXIII Congresso Brasileiro de Automática (CBA). Porto Alegre, Brasil: SBA, 2020a.

PECCIN, V. B.; LIMA, D. M.; FLESCH, R. C. C.; NORMEY-RICO, J. E. Fast algorithms for constrained generalised predictive control with on-line optimisation. **IET Control Theory & Applications**, v. 15, n. 4, p. 545–558, 2021a.

PECCIN, V. B.; LIMA, D. M.; FLESCH, R. C. C.; NORMEY-RICO, J. E. Fast Constrained Dynamic Matrix Control Algorithm with Online Optimization. Manuscrito submetido para avaliação. [*S.l.*], 2021b.

PECCIN, V. B.; LIMA, D. M.; FLESCH, R. C. C.; NORMEY-RICO, J. E. Fast Constrained Generalized Predictive Control with ADMM Embedded in an FPGA. **IEEE Latin America Transactions**, v. 18, n. 2, p. 422–429, 2020b.

PECCIN, V. B.; LIMA, D. M.; FLESCH, R. C. C.; NORMEY-RICO, J. E. Fast Generalized Predictive Control Based on Accelerated Dual Gradient Projection Method. *In*: PROCEEDINGS of 12th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems (DYCOPS). Florianópolis,Brazil: IFAC, 2019. P. 474–479.

PECCIN, V. B.; LIMA, D. M.; FLESCH, R. C. C.; NORMEY-RICO, J. E. Implementação de GPC e DMC para Sistemas Rápidos Usando ADMM. *In*: ANAIS XXII Congresso Brasileiro de Automática (CBA). João Pessoa, Brasil: SBA, 2018.

PEYRL, H.; ZANARINI, A.; BESSELMANN, T.; LIU, J.; BOÉCHAT, M. Parallel implementations of the fast gradient method for high-speed MPC. **Control Engineering Practice**, v. 33, p. 22–34, 2014.

PISTIKOPOULOS, E. N.; DIANGELAKIS, N. A.; OBERDIECK, R.; PAPATHANASIOU, M. M.; NASCU, I.; SUN, M. PAROC—An integrated framework and software platform for the optimisation and advanced model-based control of process systems. **Chemical Engineering Science**, v. 136, p. 115–138, 2015.

PU, Y.; ZEILINGER, M. N.; JONES, C. N. Complexity Certification of the Fast Alternating Minimization Algorithm for Linear MPC. **IEEE Transactions on Automatic Control**, v. 62, n. 2, p. 888–893, fev. 2017.

QASIM, S. M.; ABBASI, S. A.; ALMASHARY, B. A. Hardware realization of matrix multiplication using field programmable gate array. **MASAUM Journal of Computing**, v. 1, n. 1, p. 21–25, 2009.

RAGHUNATHAN, A. U.; CAIRANO, S. D. Infeasibility detection in alternating direction method of multipliers for convex quadratic programs. *In*: PROC. 53rd IEEE Conference on Decision and Control. Los Angeles, USA: IEEE, dez. 2014. P. 5819–5824.

RAO, C. V.; WRIGHT, S. J.; RAWLINGS, J. B. Application of Interior-Point Methods to Model Predictive Control. **Journal of Optimization Theory and Applications**, v. 99, n. 3, p. 723–757, dez. 1998.

RICHTER, S.; JONES, C. N.; MORARI, M. Computational Complexity Certification for Real-Time MPC With Input Constraints Based on the Fast Gradient Method. **IEEE Transactions on Automatic Control**, v. 57, n. 6, p. 1391–1403, jun. 2012.

RICHTER, S.; JONES, C. N.; MORARI, M. Real-time input-constrained MPC using fast gradient methods. *In*: PROC. 48th IEEE Conference on Decision and Control (CDC). Shanghai, China: IEEE, dez. 2009. P. 7387–7393.

ROLDAO-LOPES, A.; SHAHZAD, A.; CONSTANTINIDES, G. A.; KERRIGAN, E. C. More Flops or More Precision? Accuracy Parameterizable Linear Equation Solvers for Model Predictive Control. *In*: PROC. 17th IEEE Symposium on Field Programmable Custom Computing Machines. Napa,USA: IEEE, abr. 2009. P. 209–216.

ROSSITER, J. A. **Model-Based Predictive Control: A Practical Approach**. Boca Raton, FL: CRC Press, 2003. (Control Series).

SAMAD, T. A Survey on Industry Impact and Challenges Thereof [Technical Activities]. **IEEE Control Systems Magazine**, v. 37, n. 1, p. 17–18, fev. 2017.

STATHOPOULOS, G.; SHUKLA, H.; SZUCS, A.; PU, Y; JONES, C. N. Operator Splitting Methods in Control. **Foundations and Trends in Systems and Control**, v. 3, n. 3, p. 249–362, 2016.

TRUONG, T. T.; NGUYEN, H. T. Backtracking Gradient Descent Method and Some Applications in Large Scale Optimisation: Part 2: Algorithms and Experiments. **Applied Mathematics & Optimization**, Springer, p. 1–30, 2020.

TSENG, P. **On accelerated proximal gradient methods for convex-concave optimization**. Seattle, WA, 2008. (Technical Report).

WANG, J.; XU, Z.; SONG, C.; YAO, Y.; ZHAO, J. A distributed model predictive control algorithm with the gap metric output feedback decoupling. **Computers & Chemical Engineering**, p. 107167, 2020.

WANG, L. **Model Predictive Control System Design and Implementation Using MATLAB**. London: Springer Publishing Company, Incorporated, 2009.

WANG, Y.; BOYD, S. Fast Model Predictive Control Using Online Optimization. **IEEE Transactions on Control Systems Technology**, v. 18, n. 2, p. 267–278, mar. 2010.

WILLS, A.; MILLS, A.; NINNESS, B. FPGA Implementation of an Interior-Point Solution for Linear Model Predictive Control. *In*: PROC. 18th IFAC World Congr. Milano, Italy: IFAC, 2011.

WOJTULEWICZ, A.; ŁAWRYŃCZUK, M. Implementation of Multiple-Input Multiple-Output Dynamic Matrix Control Algorithm for Fast Processes Using Field Programmable Gate Array. *In*: PROC. 15th IFAC Conference on Programmable Devices and Embedded Systems (PDeS). Ostrava, Czech Republic: IFAC, 2018. P. 324–329.

XIE, J.; MEHER, P. K.; SUN, M.; LI, Y.; ZENG, B.; MAO, Z. Efficient FPGA Implementation of Low-Complexity Systolic Karatsuba Multiplier Over $GF(2^m)$ Based on NIST Polynomials. **IEEE Transactions on Circuits and Systems—Part I: Regular Papers**, v. 64, n. 7, p. 1815–1825, jul. 2017.

YANG, N.; LI, D.; ZHANG, J.; XI, Y. Model predictive controller design and implementation on FPGA with application to motor servo system. **Control Eng. Practice**, v. 20, n. 11, p. 1229–1235, 2012.

Apêndices

APÊNDICE A – DEFINIÇÕES DE OTIMIZAÇÃO CONVEXA

Neste apêndice são apresentadas as definições formais dos principais termos da área de otimização que são utilizados neste trabalho. Para um maior aprofundamento nessa temática, recomenda-se a leitura de Boyd e Vandenberghe (2004), material empregado como base para redação deste apêndice.

Definição A.1 (Conjunto convexo)**.** *Um conjunto $\Omega \subset \mathbb{R}^n$ é dito convexo, se* $\forall \theta \in [0,1]$*:*

$$\theta x + (1-\theta)y \in \Omega, \quad \forall x,y \in \Omega.$$

Definição A.2 (Função convexa)**.** *Uma função $f : \Omega \to \mathbb{R}$ definida sobre um conjunto convexo $\Omega \subset \mathbb{R}^n$ é dita convexa em Ω, se* $\forall \theta \in [0,1]$*:*

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad \forall x,y \in \Omega.$$

Definição A.3 (Função estritamente convexa)**.** *Uma função $f : \Omega \to \mathbb{R}$ definida sobre um conjunto convexo $\Omega \subset \mathbb{R}^n$ é dita estritamente convexa em Ω, se* $\forall \theta \in [0,1]$*:*

$$f(\theta x + (1-\theta)y) < \theta f(x) + (1-\theta)f(y), \quad \forall x,y \in \Omega.$$

Definição A.4 (Função Lipschitz contínua)**.** *Uma função é dita Lipschitz contínua no conjunto Ω se existir uma constante Lipschitz $L \geq 0$ que satisfaça a seguinte condição:*

$$\|f(\mathbf{x}) - f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \quad \forall \mathbf{x},\mathbf{y} \in \Omega$$

Definição A.5 (Programa quadrático)**.** *O problema de otimização*

$$\begin{aligned} QP : \min_{\mathbf{x}} \quad & \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{b}^T\mathbf{x} \\ s.\,a. \quad & \bar{\mathbf{R}}\mathbf{x} \leq \bar{\mathbf{r}} \\ & \bar{\mathbf{C}}\mathbf{x} = \bar{\mathbf{c}}, \end{aligned} \tag{175}$$

com $\mathbf{x} \in \mathbb{R}^n$ sendo o vetor variáveis de decisão, $\mathbf{H} \in \mathbb{R}^{n \times n}$ sendo uma matriz Hessiana positiva semidefinida simétrica, o vetor gradiente $\mathbf{b} \in \mathbb{R}^n$, a matriz de restrições de desigualdade $\bar{\mathbf{R}} \in \mathbb{R}^{n_{rin} \times n}$, o vetor de restrições de desigualdade $\bar{\mathbf{r}} \in \mathbb{R}^{n_{rin}}$, a matriz de restrições de igualdade $\bar{\mathbf{C}} \in \mathbb{R}^{n_{req} \times n}$ e o vetor de restrições de igualdade $\bar{\mathbf{c}} \in \mathbb{R}^{n_{req}}$, é chamado de programa quadrático e sua solução ótima é dada por $\mathbf{x}^*$.

Definição A.6 (Programa quadrático convexo)**.** *Um programa quadrático da forma de* (175) *é dito convexo, se e somente se, sua matriz Hessiana* $\mathbf{H}$ *for simétrica positiva definida, ou seja:*

$$\mathbf{H} = \mathbf{H}^T; \quad \nu^T \mathbf{H} \nu \succeq \mathbf{0}, \ \forall \nu \in \mathbb{R}^n.$$

Definição A.7 (Programa quadrático factível)**.** *Um programa quadrático da forma de* (175) *é dito factível, se e somente se, seu conjunto factível*

$$\mathcal{F} \triangleq \{\check{\mathbf{x}} \in \mathbb{R}^n | \bar{\mathbf{R}}\check{\mathbf{x}} \leq \bar{\mathbf{r}}, \ \bar{\mathbf{C}}\check{\mathbf{x}} = \bar{\mathbf{c}}\},$$

for não vazio.

Definição A.8 (Programa quadrático limitado)**.** *Um programa quadrático da forma de* (175) *é dito limitado (por baixo), se e somente se, existir um número* θ *tal que*

$$\theta \leq \frac{1}{2}\check{\mathbf{x}}^T \mathbf{H}\check{\mathbf{x}} + \mathbf{b}^T \check{\mathbf{x}}, \quad \forall \check{\mathbf{x}} \in \mathcal{F}.$$

Definição A.9 (Lagrangiano)**.** *O Lagrangiano* $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^{n_{rin}} \times \mathbb{R}^{n_{req}} \to \mathbb{R}$ *associado ao QP* (175) *é definido como:*

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\psi}, \boldsymbol{\nu}) \triangleq \frac{1}{2}\mathbf{x}^T \mathbf{H}\mathbf{x} + \mathbf{b}^T \mathbf{x} + \boldsymbol{\psi}^T(\bar{\mathbf{R}}\mathbf{x} - \bar{\mathbf{r}}) + \boldsymbol{\nu}^T(\bar{\mathbf{C}}\mathbf{x} - \bar{\mathbf{c}})$$

Definição A.10 (Função dual)**.** *Define-se a função dual* $f_d : \mathbb{R}^{n_{rin}} \times \mathbb{R}^{n_{req}} \to \mathbb{R}$ *como o mínimo valor do Lagrangiano sobre x, ou seja:*

$$f_d(\psi, \nu) \triangleq inf_x \ (\mathcal{L}(\mathbf{x}, \boldsymbol{\psi}, \boldsymbol{\nu})).$$

Definição A.11 (Programa quadrático dual)**.** *Define-se o programa quadrático dual referente ao QP* (175) *como*

$$\begin{aligned} QP_{dual} : \max_{\boldsymbol{\psi}} \quad & f_d(\psi) \\ s.\,a. \quad & \psi \geq 0 \end{aligned} \tag{176}$$

onde todas as quantidades são definidas como em (175) *e sua solução ótima é dada por* $\boldsymbol{\psi}^*$*.*

Definição A.12 (Dualidade forte)**.** *A dualidade forte se aplica para um QP do tipo* (175)*, se a brecha dual (duality gap) for nula, ou seja, o valor ótimo da função objetivo do problema dual d^* é igual ao valor ótimo da função objetivo primal p^*:*

$$p^* = d^*.$$

Definição A.13 (Condição de Slater)**.** *É dito que um QP do tipo* (175) *satisfaz a condição de Slater se ele é estritamente factível, ou seja:*

$$\exists\, \check{\mathbf{x}} \in \Omega : \bar{\mathbf{R}}\check{\mathbf{x}} < \bar{\mathbf{r}},\ \bar{\mathbf{C}}\check{\mathbf{x}} = \bar{\mathbf{c}}.$$

Teorema A.1 (Dualidade forte via condição de Slater)**.** *Se o problema primal* (175) *é convexo, e satisfaz a condição de Slater, então a dualidade forte se aplica.*

Demonstração. Uma prova da dualidade forte via condição de Slater pode ser vista em Boyd e Vandenberghe (2004, p. 234) □

Teorema A.2 (Condições Karush-Kuhn-Tucker (KKT) de otimalidade para QP)**.** *Assumindo um QP do tipo* (175) *estritamente convexo e factível, então existe um único $\mathbf{x}^* \in \mathbb{R}^n$ e um vetor $\psi^* \in \mathbb{R}^{n_{rin}}$ que satisfazem as seguintes condições:*

$$\begin{aligned} \bar{\mathbf{R}}\mathbf{x}^* - \bar{\mathbf{r}} &\leq \mathbf{0}, \\ \bar{\mathbf{C}}\mathbf{x}^* - \bar{\mathbf{c}} &= \mathbf{0}, \\ \psi^* &\geq \mathbf{0}, \\ {\psi^*}^T(\bar{\mathbf{R}}\mathbf{x}^* - \bar{\mathbf{r}}) &= \mathbf{0}, \\ \mathbf{H}\mathbf{x}^* + \mathbf{b} + \bar{\mathbf{R}}^T\psi^* &= \mathbf{0}, \end{aligned} \tag{177}$$

e além disso, $\mathbf{x}^*$ é o único mínimo global do QP primal (175)*, ($\mathbf{x}^*$,ψ^*) é a solução ótima do QP dual* (176) *e os valores ótimos das funções objetivo do QP primal e dual são iguais.*

Demonstração. Diversos livros de otimização apresentam demonstrações de que as condições KKT são necessárias e suficientes para caracterizar soluções de um QP convexo, recomenda-se Boyd e Vandenberghe (2004, p. 44). □

Definição A.14 (Projeção em um conjunto)**.** *Dados um conjunto $\Omega \subset \mathbb{R}^n$ e um ponto factível $\mathbf{x}_0 \in \Omega$ define-se a projeção de $\mathbf{x} \in \mathbb{R}^n$ sobre Ω, do seguinte modo:*

$$P(\mathbf{x}) \triangleq \mathit{argmin}\{\|\mathbf{x}_0 - \mathbf{x}\| : \mathbf{x}_0 \in \Omega\}$$

Teorema A.3 (Teorema da projeção)**.** *Seja $\Omega \subset \mathbb{R}^n$ um conjunto fechado e não-vazio. Então para todo ponto $\mathbf{x} \in \mathbb{R}^n$ existe uma projeção de $\mathbf{x}$ sobre Ω. Se, além de ser fechado, o conjunto Ω for convexo, então para todo ponto $\mathbf{x} \in \mathbb{R}^n$ existe uma única projeção de $\mathbf{x}$ sobre Ω.*

Demonstração. Uma demonstração pode ser encontrada nas páginas 10 e 94 de Izmailov e Solodov (2014). □