IMPLEMENTAÇÃO DE GPC E DMC PARA SISTEMAS RÁPIDOS USANDO ADMM

Vinícius Berndsen Peccin*, Daniel Martins Lima†, Rodolfo César Costa Flesch*, Julio Elias Normey-Rico*

*Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina
Florianópolis, Santa Catarina, Brasil

†Departamento de Engenharias
Universidade Federal de Santa Catarina
Blumenau, Santa Catarina, Brasil

Emails: vinicius.peccin@ifsc.edu.br, daniel.lima@ufsc.br, rodolfo.flesch@ufsc.br, julio.normey@ufsc.br

Abstract— In general, model predictive control (MPC) requires the computation of a quadratic programming problem (QP) at each sampling instant. This computation can be considered costly from the computational point of view and become a limitation for the use of MPC in plants with fast sampling rates. In order to circumvent this limitation and allow it to act on a larger variety of systems, special solvers which efficiently compute the control signal can be used and implemented using high-speed hardwares. Several works were proposed for this type of solution, but most of them focus on state space formulations for MPC, which are very popular in academia. This paper proposes a solution based on the Alternate Direction Method of Multipliers (ADMM) similar to the Operator Split for Control (OSC), applied to Generalized Predictive Control (GPC) and Dynamic Matrix Control (DMC), which are the most popular formulations in industry. The method is firstly validated using MATLAB and its results are compared with the ones presented by quadprog solver. A small size system is also evaluated in an FPGA with the QP computed in 15,6 μs.

Keywords— ADMM, embedded optimal control, FPGA, model predictive control.

Resumo— Em geral, o controle preditivo baseado em modelo (MPC) necessita do cômputo de um problema de programação quadrática (QP) a cada ciclo de controle. Esse cômputo pode ser considerado custoso do ponto de vista computacional e se tornar um limitante para a utilização do MPC em plantas com períodos de amostragem rápidos. Para contornar essa limitação e permitir a atuação em uma variedade maior de sistemas, pode-se utilizar um *solver* que faça o cômputo da ação de controle de forma eficiente e possa ser implementado em um *hardware* de rápida execução. Diversos trabalhos foram propostos para esse tipo de solução, mas a grande maioria focando nas características do MPC representado em espaço de estados, que é muito popular na academia. Neste trabalho é proposto um *solver* baseado no algoritmo Método dos Multiplicadores com Direções Alternadas (ADMM), semelhante ao Operador de Partição para Controle (OSC), aplicado ao Controle Preditivo Generalizado (GPC) e ao Controle por Matriz Dinâmica (DMC), que são as formulações mais populares na indústria. O método é validado a partir da implementação em MATLAB e comparado com o *solver* quadprog. Um exemplo de pequeno porte é também avaliado em um FPGA, sendo o QP computado em 15,6 μs.

Palavras-chave— ADMM, controle preditivo, controle ótimo embarcado, FPGA.

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. Possui muitas vantagens em sua aplicação, tais como tratamento explícito de restrições, aplicação em sistemas de uma entrada e uma saída (SISO, do inglês *Single Input Single Output*, múltiplas entradas e múltiplas saída (MIMO, do inglês *Multiple Input Multiple Output* e compensação intrínseca de tempo morto (Normey-Rico e 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 e Boyd, 2010). Essa característica advém do fato de que cada nova ação de controle é obtida através do cálculo de um problema de otimização *online* com restrições em um horizonte predefinido. 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.

As técnicas de otimização utilizadas para a solução do QP, adequadas para MPC, podem ser divididas como de cômputo *online* e *offline*. A principal técnica *offline* é chamada de MPC Explícito (EM, do inglês *Explicit MPC*). Entre as técnicas de otimização *online* merecem destaque os métodos de segunda e de primeira ordem. Entre as principais técnicas de segunda ordem podem-se citar as técnicas de Conjuntos Ativos (AS, do inglês *Active Set*) e as de ponto interior (IP, do inglês *Interior Point*). Os principais métodos de primeira ordem são o Método de Projeção de Gradiente (GPM, do inglês *Gradient Projection Method*) e Método de Operador de Partição (OSM, do inglês *Operator Split Method*). Estes últimos têm se destacado dos outros métodos pois permitem a paralelização do problema para um cômputo mais

eficiente (Ferreau et al., 2017).

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). As vantagens do FPGA em relação aos processadores são a velocidade de execução, podendo executar uma instrução a cada ciclo de *clock*, e a programação através de uma descrição de *hardware* e não de um software. Por se tratar de uma implementação em *hardware*, a execução das tarefas no FPGA é paralela e há uma garantia de determinismo. Os processadores executam os programas, de forma sequencial, através de um escalonador do sistema operacional e dependem de sistemas operacionais de tempo real para garantir o determinismo. 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 *solver* de controle possa ser embarcado:

1. computar uma ação de controle em um intervalo de amostragem definido em um *hardware* relativamente simples (microcontrolador, FPGA etc.);
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 é 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.

Existem diversas formulações de MPC, que se diferenciam pelo modelo utilizado na predição, pelas perturbações consideradas e pela função objetivo. 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 e Bordons, 2004). O GPC utiliza a função de transferência discreta para o cálculo das predições e o DMC os coeficientes da resposta ao degrau. Como é apresentado na seção 2, a grande maioria dos trabalhos que exploram algoritmos de MPC de cômputo rápido empregam a formulação em espaço de estados do MPC. Dessa forma, há uma lacuna entre os algoritmos do estado da arte de otimização e as formulações realmente utilizadas na indústria. Apesar de as formulações poderem ser equivalentes em alguma medida, elas ainda guardam algumas particularidades de implementação. Dessa forma, o objetivo deste trabalho é aplicar um algoritmo do estado da arte, que atenda aos requisitos citados, às particularidades do GPC e do DMC.

As principais contribuições deste artigo são:

- revisar as técnicas de otimização para MPC embarcado de cômputo rápido;
- aplicar uma técnica do estado da arte de otimização embarcada, que geralmente são desenvolvidas para a formulação MPC em espaço de estados, para a formulação GPC/DMC;
- validar a técnica proposta com um otimizador de uso geral;
- testar a implementação em um *hardware* de alto desempenho.

O artigo está organizado conforme segue. Na seção 2 é apresentada uma revisão das técnicas de otimização para MPC embarcado. Na seção 3 é apresentada a formulação geral do algoritmo ADMM. Na seção 4 apresenta-se uma formulação do ADMM aplicável ao problema de otimização decorrente do GPC e do DMC. Na seção 5 é apresentada a validação do algoritmo proposto, a partir da simulação de um GPC em MATLAB, e comparação com um *solver* de uso geral. Na seção 6 é apresentada a aplicação do algoritmo em um FPGA e demonstrada a rapidez do algoritmo. Na seção 7 são apresentadas as conclusões.

2 Técnicas de Otimização para MPC Embarcado

Nesta seção é apresentada uma breve revisão das técnicas de otimização para MPC embarcado, feita com base em trabalhos da literatura.

2.1 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). Os autores apresentam 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 e Boyd, 2010).

2.2 Métodos de Conjuntos Ativos (AS)

O AS 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, se resolve o QP apenas com restrições de igualdade, que é mais simples de ser resolvido por multiplicadores de Lagrange. O AS tem uma convergência muito rápida e com boa precisão na solução para QPs de pequeno a médio tamanho. 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 o QP, 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).

2.3 Métodos de Ponto Interior (IP)

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. Após a reformulação, o QP é então resolvido pelo método de Newton e seus 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 *solver* genérico. O método de otimização proposto é baseado nas técnicas de ponto interior, mais especificamente, o método de barreira logarítmica com passo de Newton infactível. A partir dessa técnica são propostas 3 alterações importantes para acelerar a computação da ação de controle. A primeira, e mais importante, é na exploração da estrutura das matrizes do QP através de uma eliminação de blocos de variáveis e decomposição Cholesky. A segunda é na 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.

2.4 Métodos de projeção de Gradiente (GPM)

Os GPM rápido foi proposto por Nesterov (1983). 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. Porém, como essa 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. Essa variação foi denominada Projeção de Gradiente 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.

2.5 Métodos de Operador de Partição (OSM)

A classe de OSMs 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 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 para 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. 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 desses métodos, o ADMM se apresenta como uma boa solução para as soluções modernas de *hardware*, como os processadores de múltiplos núcleos e os FPGAs. Em O'Donoghue et al. (2013) é apesentada 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 possibilita 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.

2.6 MPC de Cômputo Rápido Aplicado

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 restrições e 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 implementa 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. De modo a acelerar o processo de otimização, os cálculos dos produtos internos entre vetores são feitos em paralelo aproveitando essa característica do FPGA para reduzir drasticamente o tempo. 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), a aplicação de MPC rápido em FPGA para o acoplamento de espaçonaves em órbitas elípticas.

3 Formulação ADMM

A partir da revisão apresentada, foi escolhido o algoritmo ADMM como um dos mais promissores para o atendimento dos requisitos e possibilidade de paralelização. Ele se apresenta na forma de um algoritmo de decomposição/coordenação, no qual as soluções de pequenos subproblemas locais são coordenados para encontrar uma solução para um problema global de grande escala (O'Donoghue et al., 2013).

O algoritmo ADMM resolve problemas da seguinte forma:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & f(\mathbf{x}) + g(\mathbf{y}) \\ \text{s.a.} \quad & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{c} \end{aligned} \tag{1}$$

com $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^m, \mathbf{A} \in \mathbb{R}^{p\times n}, \mathbf{B} \in \mathbb{R}^{p\times m}$, $\mathbf{c} \in \mathbb{R}^p$ e f e g sendo funções convexas.

Para incluir a restrição na função objetivo, a partir de (1), pode ser formado o Lagrangeano aumentado:

$$L_\rho(\mathbf{x},\mathbf{y},\psi) = f(\mathbf{x}) + g(\mathbf{y}) + \psi^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{c}) + \\ + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{c}\|_2^2.$$

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 $\mathbf{x}^*$ pode ser obtido através do ponto ótimo dual ψ^*:

$$\mathbf{x}^* = \text{argmin } L\rho(\mathbf{x},\psi^*)$$

portanto o ADMM pode ser computado de forma recursiva, alternando entre $\mathbf{x}$ e $\mathbf{y}$ da seguinte forma:

$$\mathbf{x}^{k+1} := \text{argmin } L_\rho(\mathbf{x}^k,\mathbf{y}^k,\psi^k) \tag{2}$$

$$\mathbf{y}^{k+1} := \text{argmin } L_\rho(\mathbf{x}^{k+1},\mathbf{y}^k,\psi^k) \tag{3}$$

$$\psi^{k+1} := \psi^k + \rho(\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{y}^{k+1} - \mathbf{c}). \tag{4}$$

Uma prova da convergência do Algoritmo ADMM pode ser encontrado em Boyd et al. (2011). A partir da prova de convergência do algoritmo, pode-se concluir que as iterações do ADMM satisfazem o seguinte:

- convergência residual: $r^k \to 0$ se $k \to \infty$;
- convergência do objetivo: $f(\mathbf{x}^k)+g(\mathbf{y}^k) \to \mathbf{p}^*$ se $k \to \infty$, onde $\mathbf{p}^*$ é o ponto ótimo primal;
- convergência da variável dual: $\psi^k \to \psi^*$ se $k \to \infty$, onde ψ^* é o ponto ótimo dual;

assumindo f e g funções próprias, convexas e fechadas que o Lagrangeano associado tem um ponto de sela e que o resíduo é definido como:

$$\mathbf{r}^k = \mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{y}^{k+1} - \mathbf{c}.$$

4 Proposta de Modelagem ADMM para GPC/DMC

O problema de programação quadrática, típica de um GPC/DMC, pode ser escrita como:

$$\begin{aligned} \min_{\mathbf{u}} \quad & \frac{1}{2}\mathbf{u}^T\mathbf{H}\mathbf{u} + \mathbf{b}^T\mathbf{u} \\ \text{s.a.} \quad & \mathbf{R}\mathbf{u} \leq \bar{\mathbf{r}} \end{aligned} \tag{5}$$

com $\mathbf{u} \in \mathbb{R}^{n_u}$ sendo o vetor de incrementos de controle futuros, $\mathbf{H} \in \mathbb{R}^{n_u \times n_u}$ sendo uma matriz Hessiana positiva semi-definida, o vetor gradiente $\mathbf{b} \in \mathbb{R}^{n_u}$, a matriz de restrições $\mathbf{R} \in \mathbb{R}^{n_r \times n_u}$ e o vetor $\bar{\mathbf{r}} \in \mathbb{R}^{n_r}$, n_u é o horizonte de predição do controle e n_r o número de restrições.

O primeiro passo proposto é representar as restrições através de uma função indicadora $I_-(x)$ definida como:

$$I_-(x) = \begin{cases} 0, & \text{se } x \leq 0 \\ \infty, & \text{se } x > 0 \end{cases}.$$

Pode-se definir $f(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T\mathbf{H}\mathbf{u} + \mathbf{b}^T\mathbf{u}$ e reescrever o problema rearranjando as restrições através da função indicadora $I_-(x)$ na função objetivo:

$$\min_{\mathbf{u}} \quad f(\mathbf{u}) + I_-(\mathbf{R}\mathbf{u} - \bar{\mathbf{r}}). \tag{6}$$

Para obter-se a forma padrão do ADMM, pode-se definir uma nova variável $\mathbf{z} = \mathbf{R}\mathbf{u} - \bar{\mathbf{r}}$ para a segunda equação e inserir uma restrição para manter o acoplamento entre as variáveis:

$$\begin{aligned} \min_{\mathbf{u},\mathbf{z}} \quad & f(\mathbf{u}) + I_-(\mathbf{z}) \\ \text{s.a.} \quad & \mathbf{R}\mathbf{u} - \mathbf{z} = \bar{\mathbf{r}}. \end{aligned} \tag{7}$$

A partir da forma padrão obtida em (7) pode-se definir o Lagrangeano aumentado:

$$L(\mathbf{u},\mathbf{z},\psi) = f(\mathbf{u}) + I_-(\mathbf{z}) + \psi^T(\mathbf{R}\mathbf{u} - \mathbf{z} - \bar{\mathbf{r}}) + \\ + \frac{\rho}{2}\|\mathbf{R}\mathbf{u} - \mathbf{z} - \bar{\mathbf{r}}\|_2^2.$$

O ADMM pode ser computado recursivamente na forma escalonada, com $\phi = (\frac{1}{\rho})\psi^k$ da seguinte forma:

$$\mathbf{u}^{k+1} := \operatorname{argmin}\left(f(\mathbf{u}^{\mathrm{k}}) + \frac{\rho}{2}\|\mathbf{R}\mathbf{u} - \mathbf{z}^{\mathrm{k}} - \bar{\mathbf{r}} + \phi^{\mathrm{k}}\|_2^2\right) \tag{8}$$

$$\mathbf{z}^{k+1} := \operatorname{argmin}\left(I_-(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{R}\mathbf{u}^{\mathrm{k+1}} - \mathbf{z}^{\mathrm{k}} - \bar{\mathbf{r}} + \phi^{\mathrm{k}}\|_2^2\right) \tag{9}$$

$$\phi^{k+1} := \phi^k + (\mathbf{R}\mathbf{u}^{k+1} - \mathbf{z}^{k+1} - \bar{\mathbf{r}}). \tag{10}$$

Para computar o passo da equação (8) pode-se abrir o termo da norma e agrupar da seguinte forma:

$$\mathbf{u}^{k+1} := \operatorname{argmin}\left(\frac{1}{2}\mathbf{u}^T(\mathbf{H} + \mathbf{R}^T\rho\mathbf{R})\mathbf{u} + \right. \\ \left. + (\mathbf{b}^T + \rho\mathbf{R}^T(\phi - \mathbf{z} - \bar{\mathbf{r}}))\mathbf{u}\right).$$

Dessa forma, se for definido $\tilde{\mathbf{H}} = \mathbf{H} + \mathbf{R}^T\rho\mathbf{R}$ e $\tilde{\mathbf{b}} = \mathbf{b}^T + \rho\mathbf{R}^T(\phi - \mathbf{z} - \bar{\mathbf{r}}))$ recai-se num problema padrão de programação quadrática sem restrições:

$$\min_{\mathbf{u}} \quad \frac{1}{2}\mathbf{u}^T\tilde{\mathbf{H}}\mathbf{u} + \tilde{\mathbf{b}}^T\mathbf{u} \tag{11}$$

que possui solução analítica:

$$\mathbf{u}^{*k+1} := -\tilde{\mathbf{H}}^{-1}\tilde{\mathbf{b}}.$$

Como em um problema típico de GPC/DMC 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 (9), pode-se verificar que ela pode ser reescrita através do operador proximal (Stathopoulos et al., 2016) da seguinte forma:

$$\mathbf{z}^{k+1} := \operatorname{prox}_{I_-,\rho}(\mathbf{R}\mathbf{u}^{\mathrm{k+1}} - \bar{\mathbf{r}} + \phi^{\mathrm{k}})$$

Como nesse caso $I_-(x)$ é uma função do tipo indicadora, a operação proximal se torna uma projeção no domínio de $I_-(x)$ como uma operação de saturação da seguinte forma:

$$\mathbf{z}^{k+1} = \begin{cases} \mathbf{R}\mathbf{u}^{k+1} - \bar{\mathbf{r}} + \phi^k, \text{ se } \mathbf{R}\mathbf{u}^{\mathrm{k+1}} - \bar{\mathbf{r}} + \phi^{\mathrm{k}} \leq 0 \\ 0, \text{ se } \mathbf{R}\mathbf{u}^{\mathrm{k+1}} - \bar{\mathbf{r}} + \phi^{\mathrm{k}} > 0 \end{cases}$$

É importante ressaltar que, por ser separável, esta saturação pode ser executada de forma paralela, para cada elemento do vetor $\mathbf{z}$, e torna a solução desta equação independente do número de restrições aplicadas.

O passo da equação (10) pode ser resolvido de forma direta, entretanto é importante interpretá-la como uma forma de rastreamento para a convergência do algoritmo. Pode-se perceber que o termo entre parênteses de (10) é o resíduo primal do problema de otimização e dessa forma esse

passo se torna como o cálculo de um integrador em ϕ de modo a levar o resíduo para zero.

A partir do desenvolvimento proposto, pode-se resumir a implementação no algoritmo 1, onde ϵ_pri e ϵ_dual representam as tolerâncias esperadas como critério de parada para os resíduos primal e dual, respectivamente.

Algoritmo 1 ADMM proposto para GPC/DMC

Entrada: $\mathbf{H}$,$\mathbf{b}^T$,$\mathbf{R}$,$\bar{\mathbf{r}}$,$\mathbf{u}^0$ factível
Saída: $\mathbf{u}$
Dados: $\rho > 0$, $\phi^0 = \mathbf{0}$, $\epsilon_{pri} > 0$, $\epsilon_{dual} > 0$, k_{max}
início
  $\tilde{\mathbf{H}} = \mathbf{H} + \mathbf{R}^T\rho\mathbf{R}$;
  armazena $\tilde{\mathbf{H}}^{-1}$;
  $\mathbf{z}^0 = \mathbf{R}\mathbf{u}^0 - \bar{\mathbf{r}}$;
  $k = 0$;
  enquanto $k < k_{max}$ **faça**
    $\tilde{\mathbf{b}} = \mathbf{b}^T + \rho\mathbf{R}^T(\phi^k - \mathbf{z}^k - \bar{\mathbf{r}})$;
    $\mathbf{u}^{k+1} := -\tilde{\mathbf{H}}^{-1}\tilde{\mathbf{b}}$;
    $\mathbf{z}^{k+1} := \max(-\mathbf{R}\mathbf{u}^{k+1} + \bar{\mathbf{r}} - \phi^k, \mathbf{0})$;
    $\phi^{k+1} := \phi^k + (\mathbf{R}\mathbf{u}^{k+1} - \mathbf{z}^{k+1} - \bar{\mathbf{r}})$;
    se $\|\mathbf{R}\mathbf{u}^{k+1} - \mathbf{z}^{k+1} - \bar{\mathbf{r}}\|_2 \leq \epsilon_{pri}$ *e* $\|\rho\mathbf{R}^T(\mathbf{z}^{k+1} - \mathbf{z}^k)\|_2 \leq \epsilon_{dual}$ **então**
      retorna;
    fim
    $k = k + 1$;
  fim
fim

5 Validação

Para fazer a validação do algoritmo proposto foi implementado em MATLAB um controlador GPC e aplicado em uma planta de exemplo cujo modelo é:

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

e a função custo do GPC utilizada:

$$J(n_1,n_2,n_u) = \sum_{j=n_1}^{n_2} \delta(j)\left[\hat{y}(t+j|t) - w(t+j)\right]^2 + \sum_{j=1}^{n_u} \lambda(j)\left[\Delta u(t+j-1)\right]^2$$

com o 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 $w(t+j)$ conhecidas.

Foi inserida uma restrição no incremento de controle $|\mathbf{u}(k+j)| \leq 0{,}05$ com $j = 0,...,n_u - 1$. O problema de otimização recaiu então em uma matriz $\mathbf{H} \in \mathbb{R}^{5\times5}$ com 10 restrições. O GPC foi testado com o algoritmo proposto e comparado com o algoritmo de IP da função quadprog do MATLAB. Ambos foram ajustados para uma tolerância do resíduo de 0,0001. A simulação foi realizada por 100 amostras discretas e com duas trocas de referência do tipo degrau. Os resultados podem ser vistos na Figura 1. São apresentados a saída da planta, o incremento de controle aplicado, o tempo de cômputo da otimização e o número de iterações. Pode-se perceber que o comportamento do sistema de controle é equivalente para ambos os algoritmos e o esforços de controle semelhantes. Nas trocas de referência pode-se perceber a saturação atuando no incremento de controle em 0,05. A diferença maior pode ser vista no tempo de execução dos algoritmos. Percebe-se claramente que o tempo de execução é bem menor do ADMMGPC, pois mesmo tendo um aumento do número de iterações durante as trocas de referência, cada iteração é muito menos custosa do ponto de vista computacional do que o IP do quadprog. Essa é a principal característica desejada para o algoritmo. Apesar de não se ter muito interesse no valor absoluto do tempo de execução por estar sendo executado no MATLAB, serve como uma base de comparação entre os algoritmos e pode-se ver na Tabela 1 a comparação dos tempos de execução médio e máximo.

Tabela 1: Comparação dos tempos de cômputo para o exemplo de controle GPC simulado em MATLAB.

	ADMMGPC	QUADPROG
tempo médio	0,13 ms	14,00 ms
tempo máximo	0,60 ms	26,20 ms

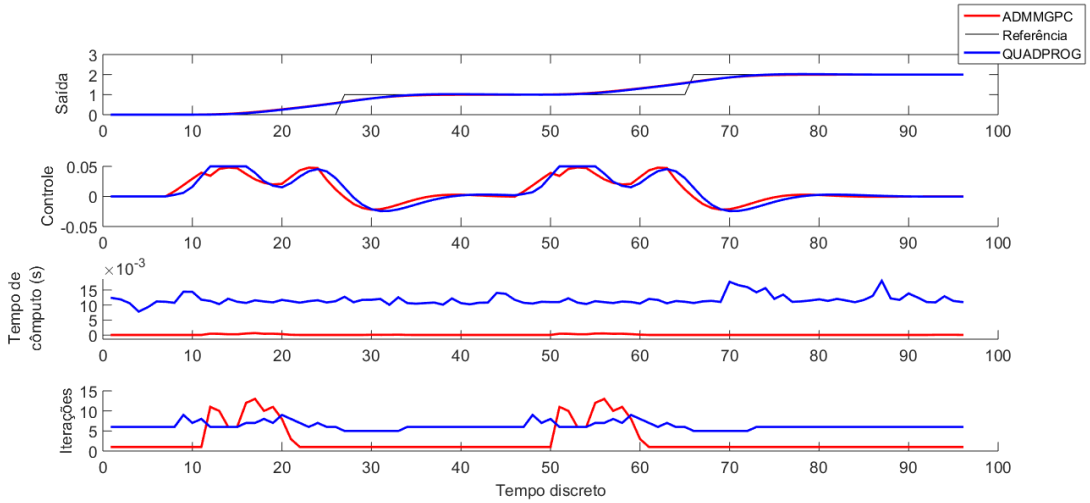

Figura 1: Validação do solver proposto com a aplicação de um exemplo de controle GPC simulado em MATLAB e comparação com o solver quadprog.

6 Teste em um FPGA

Para se avaliar o tempo de cômputo do algoritmo em um *hardware* de alto desempenho, foi implementado o exemplo de controle anterior em um FPGA. O FPGA utilizado foi da marca Altera, da família MAX 10 e do modelo 10M50DAF484C7G. A família MAX 10 apresenta conversores analógicos/digitais (A/D) embarcados no FPGA e 50 000 elementos lógicos programáveis. O solver foi implementado em linguagem de descrição de *hardware* Verilog. Para representar os números foi utilizada uma aritmética de ponto fixo com 32 bits, sendo 16 bits para os decimais. A estrutura escolhida foi de uma máquina de 5 estados para a implementação e com os cálculos de operação das matrizes em paralelo.

Os resultados obtidos para o tempo de cômputo do solver implementado no FPGA podem ser vistos na tabela 2. O *clock* utilizado no FPGA foi de 50 MHz e, devido ao paralelismo implementado, pode-se perceber que o tempo de execução é bastante rápido, com valor máximo de 15,6 μs. Percebe-se que o tempo é coerente com outras soluções apresentadas em hardwares semelhantes como em Wills et al. (2011) usando IP em um FPGA que foi de 30,0 μs e em Yang et al. (2012) que usa AS e levou 20,0 μs.

O tempo total de uma aplicação real ainda seria dependente de outros sistemas, como as conversões A/D e o número de iterações necessárias. Entretanto, o número de iterações poderia ser limitado para um valor máximo e o algoritmo entregar um valor subótimo. Essa abordagem de antecipação de parada foi testada em Wang e Boyd (2010) e demonstrou-se que, devido à realimentação, o sistema de controle é pouco sensível a uma solução subótima.

Tabela 2: Tempo de execução do ADMMGPC no FPGA

tempo por iteração	tempo máximo (13 iter.)
1,2 μs	15,6 μs

7 Conclusões

Este trabalho desenvolveu um algoritmo de otimização de cômputo rápido para problemas de controle preditivo presentes nas formulações GPC/DMC. O algoritmo baseado em ADMM que utiliza apenas operações básicas em cada iteração apresentou consistência na comparação com o solver quadprog no MATLAB. Chegou-se em uma solução que atende os requisitos apresentados e a implementação em FPGA se mostrou bastante rápida, com um tempo de cômputo de 15,6 μs para o exemplo apresentado. A implementação em FPGA ainda pode ser aperfeiçoada com a utilização de recursos como *pipeline*, que serão investigados no futuro.

Agradecimentos

Este trabalho foi apoiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), projetos 305785/2015-0 e 311024/2015-7, e pelos Programas de Formação de Recursos Humanos PRH34-ANP e PFRH34-Petrobras.

Referências

Bemporad, A. (2016). A quadratic programming algorithm based on nonnegative least squares with applications to embedded model predictive control, *IEEE Transactions on Automatic Control* **61**(4): 1111–1116.

Bemporad, A. (2018). A numerically stable solver for positive semidefinite quadratic programs based on nonnegative least squares, *IEEE Transactions on Automatic Control* **63**(2): 525–531.

Bemporad, A., Morari, M., Dua, V. e Pistikopoulos, E. N. (2002). The explicit linear quadratic regulator for constrained systems, *Automatica* **38**(1): 3–20.

Boyd, S., Parikh, N., Chu, E., Peleato, B. e Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers, *Found. Trends Mach. Learn.* **3**(1): 1–122.

Cai, X., Tippett, M. J., Xie, L. e Bao, J. (2014). Fast distributed MPC based on active set method, *Computers and Chemical Engineering* **71**: 158–170.

Camacho, E. e Bordons, C. (2004). *Model Predictive Control*, Advanced Textbooks in Control and Signal Processing, Springer London.

Cimini, G. e Bemporad, A. (2017). Exact complexity certification of active-set methods for quadratic programming, *IEEE Transactions on Automatic Control* **62**(12): 6094–6109.

Dantzig, G. (1963). *Linear programming and extensions*, Rand Corporation Research Study, Princeton Univ. Press, Princeton, NJ.

Ferreau, H., Almér, S., Verschueren, R., Diehl, M., Frick, D., Domahidi, A., Jerez, J., Stathopoulos, G. e Jones, C. (2017). Embedded optimization methods for industrial automatic control, *IFAC-PapersOnLine* **50**(1): 13194–13209. 20th IFAC World Congress.

Gulbudak, O. e Santi, E. (2016). FPGA-based model predictive controller for direct matrix converter, *IEEE Transactions on Industrial Electronics* **63**(7): 4560–4570.

Hartley, E. N. e Maciejowski, J. M. (2015). Field programmable gate array based predictive control system for spacecraft rendezvous in elliptical orbits, *Optimal Control Applications and Methods* **36**(5): 585–607.

Johansen, T. A., Jackson, W., Schreiber, R. e Tondel, P. (2006). Hardware architecture design for explicit model predictive control, *2006 American Control Conference*.

Ling, K. V., Yue, S. P. e Maciejowski, J. M. (2006). A FPGA implementation of model predictive control, *2006 American Control Conference*.

Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate $o(1/k^2)$, *Soviet Mathematics Doklady* **27**(2): 372–376.

Normey-Rico, J. E. e Camacho, E. F. (2007). *Control of Dead-time Processes*, Springer London.

O'Donoghue, B., Stathopoulos, G. e Boyd, S. (2013). A splitting method for optimal control, *IEEE Transactions on Control Systems Technology* **21**(6): 2432–2442.

Patrinos, P. e Bemporad, A. (2014). An accelerated dual gradient-projection algorithm for embedded linear model predictive control, *IEEE Transactions on Automatic Control* **59**(1): 18–33.

Peyrl, H., Zanarini, A., Besselmann, T., Liu, J. e Boéchat, M.-A. (2014). Parallel implementations of the fast gradient method for high-speed MPC, *Control Engineering Practice* **33**: 22–34.

Raghunathan, A. U. e Cairano, S. D. (2014). Infeasibility detection in alternating direction method of multipliers for convex quadratic programs, *53rd IEEE Conference on Decision and Control*, pp. 5819–5824.

Rao, C. V., Wright, S. J. e Rawlings, J. B. (1998). Application of interior-point methods to model predictive control, *Journal of Optimization Theory and Applications* **99**(3): 723–757.

Richter, S., Jones, C. N. e Morari, M. (2012). Computational complexity certification for real-time mpc with input constraints based on the fast gradient method, *IEEE Transactions on Automatic Control* **57**(6): 1391–1403.

Stathopoulos, G., Shukla, H., Szucs, A., Pu, Y. e Jones, C. N. (2016). Operator splitting methods in control, *Foundations and Trends in Systems and Control* **3**(3): 249–362.

Wang, Y. e Boyd, S. (2010). Fast model predictive control using online optimization, *Control Systems Technology, IEEE Transactions on* **18**(2): 267–278.

Wills, A., Mills, A. e Ninness, B. (2011). FPGA implementation of an interior-point solution for linear model predictive control, *IFAC Proceedings Volumes* **44**(1): 14527–14532. 18th IFAC World Congress.

Yang, N., Li, D., Zhang, J. e Xi, Y. (2012). Model predictive controller design and implementation on FPGA with application to motor servo system, *Control Engineering Practice* **20**(11): 1229 – 1235.