UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL

FACULDADE DE ENGENHARIAS, ARQUITETURA E URBANISMO E GEOGRAFIA

IMPLEMENTAÇÃO E SIMULAÇÃO DA ESTRATÉGIA DE CONTROLE PREDITIVO EM LINGUAGEM VHDL - TESTES EM COSIMULAÇÃO E FPGA-IN-THE-LOOP

Márcio Afonso Soleira Grassi

Campo Grande – MS

Outubro de 2019

UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL

FACULDADE DE ENGENHARIAS, ARQUITETURA E URBANISMO E GEOGRAFIA

IMPLEMENTAÇÃO E SIMULAÇÃO DA ESTRATÉGIA DE CONTROLE PREDITIVO EM LINGUAGEM VHDL - TESTES EM COSIMULAÇÃO E FPGA-IN-THE-LOOP

Márcio Afonso Soleira Grassi

Orientador: Prof. Dr. Edson Antonio Batista

Dissertação apresentada à Universidade Federal de Mato Grosso do Sul na Faculdade de Engenharias, Arquitetura e Urbanismo e Geografia, como requisito para a obtenção do grau de Mestre em Engenharia Elétrica.

Campo Grande – MS

Outubro de 2019

RESUMO

Este trabalho apresenta uma série de resultados para o projeto do controle preditivo, baseado em modelo (MPC), feito para sistemas embarcados. O tema central deste trabalho está no desenvolvimento de uma estrutura de hardware que facilite o projeto de controle preditivo sintetizado em FPGA. Um dos desafios encontrados para implementar a lei de controle do MPC em FPGA foi no sincronismo entre o tempo de amostragem e a atualização das variáveis. Este trabalho mostra que com a adição de um contador no algoritmo de controle é possível contornar este problema e projetar controladores preditivos para diversas plantas, tais como: sistema turbo-gerador, rastreamento de velocidade de máquina síncrona, reator CSTR e reator nuclear. Os resultados dos experimentos indicam que os controladores projetados têm reais possibilidades de ser utilizados em aplicações industriais, pois os testes foram realizados utilizando FPGA-in-the-loop.

Palavras-Chave: MPC, sistemas embarcados, reator nuclear, cosimulação, FPGA-in-the-loop.

ABSTRACT

This paper presents a series of results for the model based predictive control (MPC) project, designed for embedded systems. The central theme of this work is the development of a hardware structure that facilitates the predictive control design synthesized in FPGA. One of the challenges to implement the MPC control law in FPGA was the synchronization between sampling time and update of variables. This work shows that with the addition of a counter in the control algorithm it is possible to circumvent this problem and to design predictive controllers for several plants, such as: turbo-generator system, synchronous machine speed tracking, CSTR reactor and nuclear reactor. The results of the experiments indicate that the controllers designed have real possibilities of being used in industrial applications, since the tests were performed using FPGA-in-the-loop.

Keywords: MPC, embedded systems, nuclear reactor, cosimulation, FPGA-in-the-loop.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES

MPC	Controle preditivo baseado em modelo (*Model-based predictive control*)
VHDL	Linguagem de descrição de hardware (*VHSIC Hardware Description Language*)
VHSIC	Circuitos Integrados de alta velocidade (*Very High Speed Integrated Circuits*)
FIL	FPGA-in-the-loop
PID	Proporcional, integral e derivativo
MIMO	Sistemas com múltiplas entradas e múltiplas saídas (*Multiple-input, Multiple-output*)
HDL Coder	Codificador de linguagem de descrição de hardware (*Hardware Description Language Coder*)
GPC	Controle Preditivo Generalizado (*Generalized Predicitve Control*)
FCS-MPC	Modelo Preditivo de Controle com Horizonte de Controle Finito (*Finite-Control Set Model Predictive Control*)
CSTR	Reator de Tanque não Adiabático (*Continuous Stirred-Tank Reactor*)
PAR	Protótipo de Reator Avançado (*Prototypical Advanced Reactor*)

1 INTRODUÇÃO

O desenvolvimento de estratégia de controle preditivo baseado em modelo (MPC) para implementação em hardware tem atraído a atenção dos pesquisadores nos últimos anos, principalmente com a finalidade de obter um algoritmo com custo computacional adequado. O interesse é desenvolver metodologias para que esta abordagem de controle possa ser implementada em processos na indústria e na área de engenharia.

Nos últimos anos, grande parte dos trabalhos desenvolvidos para sintetizar o controlador preditivo utilizam a linguagem de programação C, devido à sua simplicidade e facilidade para descrever as equações do MPC. O uso de outras linguagens como a linguagem de descrição de hardware VHDL (*VHSIC Hardware Description Language*) tem sido pouco utilizada, pois além de conhecimentos de programação, o projetista precisa de experiência na área de circuitos digitais para desenvolver algoritmos eficientes. Recentemente, o uso de ferramentas de geração automática de código, sendo que neste caso a linguagem C também é a mais utilizada, trouxe maior facilidade para os pesquisadores obterem controladores preditivo de forma rápida. A utilização destes códigos tem sido feita para aplicações específicas e embarcadas em microcontroladores, kits de desenvolvimento com processador embarcado e, mais recentemente nas placas FPGA (*Field-programmable gate array*).

Neste sentido, o objetivo deste trabalho é desenvolver um algoritmo de controle preditivo em VHDL e implementar na plataforma FPGA para realizar testes de cosimulação e *FIL (FPGA-in-the-loop)*. Almeja-se que este código seja dinâmico, ou seja, com poucas modificações nos parâmetros do MPC seja possível obter um novo controlador, com a atual configuração, de forma prática.

Este trabalho está dividido da seguinte forma: no capítulo 1 é feita uma revisão sobre a teoria do controle preditivo com um exemplo, além de uma revisão bibliográfica sobre o foi publicado nesta área nos últimos anos; no capítulo 2 é discutida à metodologia das simulações e a apresentação das plantas e processos que foram controlados; o capítulo 3 mostra os resultados das simulações e testes, sendo que estes são analisados e discutidos; e por fim, no capítulo 4, descrevem-se as conclusões deste trabalho.

1.1 MODELO PREDITIVO DE CONTROLE

A estratégia de controle preditivo, neste texto resumido para o acrônimo MPC, é uma técnica de controle que tem a atenção da indústria, não só pela habilidade de entregar um controlador confiável, mas também por permitir a utilização de restrições em relação aos parâmetros a serem controlados. Entretanto conforme [1] , os engenheiros industriais ainda preferem utilizar controladores PID por estes serem simples, transparentes e, em alguns casos, robustos. Porém, este tipo de controle pode levar a limitações de desempenho e, segundo [2] , o MPC possui características que o tornam uma abordagem melhor, dentre as quais:

1) conceitos intuitivos, fáceis de entender e implementar para uma variedade de sistemas.
2) capacidade de lidar com sistemas MIMO sem nenhuma modificação.
3) facilidade de incorporar restrições no projeto do controle.

As principais aplicações do MPC são em processos lentos como na indústria química e petroquímica, mas nos últimos anos também tem sido abordado para outras áreas, tais como: aeroespacial, robótica, automotiva, metalúrgica, entre outras.

De acordo com a tradução de [2] , o modelo preditivo filosoficamente reflete o comportamento humano, pelo qual seleciona-se uma atitude ou toma-se uma decisão baseado no que se imagina que irá levar ao resultado predito desejado. Para isso, é feito o uso de um modelo interno do processo em que se atualizam constantemente as decisões de acordo com as novas observações acessíveis. De um modo geral é possível afirmar que uma lei de controle preditivo possui 4 componentes:

1- A lei de controle depende do comportamento predito;

2- As previsões da saída são calculadas utilizando um modelo do processo;

3- A entrada atual é determinada com a otimização de uma função custo;

4- O princípio de horizonte retrocedente: a entrada de controle é atualizada a cada instante de amostragem

Várias modelagens podem ser feitas para desenvolver o MPC, mas a utilizada neste trabalho é a de [3], no qual o modelo a ser utilizado é o espaço de estados discreto, como mostra as Equações 1 e 2:

$$x_m(k+1) = A_m x(k) + B_m u(k) \quad (1)$$

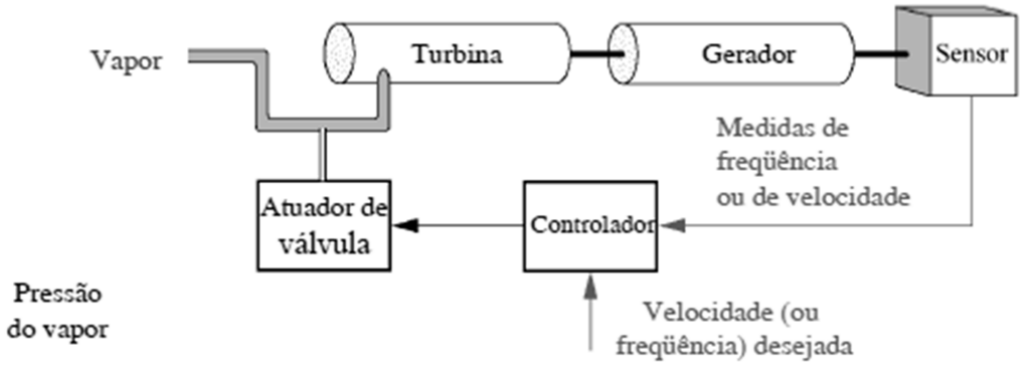
$$y(k) = C_m x(k) \quad (2)$$

O modelo em espaço de estado fornece um método unificado para modelagem, análise e projeto a uma ampla faixa de sistemas usando álgebra matricial. Apesar de a operação ser em tempo contínuo, o modelo discreto é utilizado pois a tomada de decisão é feita de maneira discreta, mesmo no controle convencional. Além disso, de acordo com [4] por demandar tempo de processamento e lidar com entradas, saídas e otimizações, não é uma boa ideia que as ações sejam feitas de maneira instantânea. Neste sentido, a taxa de amostragem é um parâmetro de seleção importante, mas que pode ser alterado pelo projetista. Porém, é recomendável utilizar o padrão de obter 10 pontos de transitório com relação ao tempo de estabilização do sistema, porque uma taxa maior pode prejudicar o desempenho do controle e uma taxa menor aumenta o número de variáveis de decisão e dificulta a otimização.

Em geral, há uma matriz D_m que multiplica a entrada u(k) na Equação 2. Entretanto, devido ao princípio do horizonte retrocedente em que para calcular a entrada e formar as predições é preciso as informações atuais da planta, considera-se a matriz D_m igual a zero, e desta forma a entrada não afeta a saída no mesmo instante.

Nesta seção, é desenvolvido um exemplo com base na teoria do MPC. Na Figura 1 mostra-se um sistema turbina gerador visto em [5], no qual o intuito é realizar o controle da frequência de saída de energia elétrica. Segundo [5], ao regular a velocidade, o sistema de controle assegura que a frequência gerada permaneça dentro da tolerância. Os desvios a partir da velocidade desejada são medidos e uma válvula de vapor é alterada para compensar o erro de velocidade.

Figura 1 - Sistema turbina gerador



Fonte: [5]

O espaço de estado discreto desta planta conforme as Equações 1 e 2 apresentadas anteriormente, e com taxa de amostragem (T_s) de 0,442 segundos pois o tempo de estabilização em malha aberta é de 4,42 segundos, é visto nas Equações 3 e 4:

$$x_m(k+1) = \overbrace{\begin{bmatrix} -0{,}1159 & -1{,}3371 & -0{,}5785 \\ 0{,}0193 & 0{,}1541 & -0{,}5080 \\ 0{,}0169 & 0{,}2563 & 0{,}8822 \end{bmatrix}}^{A_m} x(k) + \overbrace{\begin{bmatrix} 0{,}0193 \\ 0{,}0169 \\ 0{,}0039 \end{bmatrix}}^{B_m} u(k) \tag{3}$$

$$y(k) = \overbrace{\begin{bmatrix} 0 & 0 & 500 \end{bmatrix}}^{C_m} x(k) \tag{4}$$

Destarte, ainda consideramos um outro parâmetro nas equações de estado para representar os distúrbios. Porém este parâmetro foi omitido pois conforme [2] , a melhor forma de rejeição do distúrbio é ter um modelo interno do mesmo e o modelo mais utilizado para representá-lo é adicionar um integrador na planta. O modelo de espaço de estados com integrador é visto nas Equações 5 e 6 e segue o modelo apresentado em [1] , no qual as variáveis de estado passam a ser a variação dos estados originais da planta e a sua própria saída:

$$\overbrace{\begin{bmatrix} \Delta x_m(k+1) \\ y_m(k+1) \end{bmatrix}}^{x(k+1)} = \overbrace{\begin{bmatrix} A_m & 0_m^T \\ C_m A_m & 1 \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}}^{x(k)} + \overbrace{\begin{bmatrix} B_m \\ C_m B_m \end{bmatrix}}^{B} \Delta u(k) \tag{5}$$

$$y(k) = \overbrace{\begin{bmatrix} 0_m & 1 \end{bmatrix}}^{C} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} \tag{6}$$

As matrizes A, B e C são chamadas de matrizes aumentadas e 0_m é um vetor de zeros com a dimensão de A_m. Para o exemplo do sistema turbina gerador, as matrizes aumentadas são vistas nas Equações 7 e 8.

$$\overbrace{\begin{bmatrix}\Delta x_m(k+1)\\ y_m(k+1)\end{bmatrix}}^{x(k+1)} = \overbrace{\begin{bmatrix}-0{,}1159 & -1{,}3371 & -0{,}5785 & 0\\ 0{,}0193 & 0{,}1541 & -0{,}5080 & 0\\ 0{,}0169 & 0{,}2563 & 0{,}8862 & 0\\ 8{,}4662 & 128{,}1679 & 441{,}0765 & 1\end{bmatrix}}^{A} \overbrace{\begin{bmatrix}\Delta x_m(k)\\ y(k)\end{bmatrix}}^{x(k)} + \overbrace{\begin{bmatrix}0{,}0193\\ 0{,}0169\\ 0{,}0039\\ 1{,}9641\end{bmatrix}}^{B} \Delta u(k) \quad (7)$$

$$y(k) = \overbrace{\begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}}^{C} \begin{bmatrix}\Delta x_m(k)\\ y(k)\end{bmatrix} \quad (8)$$

O modelo discreto por padrão já faz uma predição de uma amostragem futura. Por exemplo, os estados no tempo *k+1* são dados pela Equação 3, e para saber a saída no tempo *k+1* basta substituir a Equação 3 na Equação 4 e tem-se y(k+1). Para uma predição de n-amostras futuras, basta repetir o processo anterior até onde o projetista quer prever, sendo que esta previsão se nomeia de horizonte de predição, representado por N_p. Também define-se o número de parâmetros usados para capturar a futura trajetória de controle e a este parâmetro é dado o nome de N_c. Nas Equações 9 e 10 mostram-se os vetores da saída e do controle dependentes de N_p e N_c, no tempo k_i.

$$Y = \begin{bmatrix}y(k_i+1|k_i) & y(k_i+2|k_i) & y(k_i+3|k_i) & \cdots & y\big(k_i+N_p\big|k_i\big)\end{bmatrix}^T \quad (9)$$

$$\Delta U = [\Delta U(k_i) \quad \Delta U(k_i+1) \quad \Delta U(k_i+2) \quad \cdots \quad \Delta U(k_i+N_c-1)]^T \quad (10)$$

A dimensão de Y é N_p e a dimensão de ΔU é N_c. Na Equação 11 apresenta-se como calcular a saída para predições de n-amostras futuras.

$$Y = Fx(k_i) + \varphi\Delta U \quad (11)$$

As matrizes F e φ são obtidas pelas Equações 12 e 13:

$$F = \begin{bmatrix}CA\\ CA^2\\ CA^3\\ \vdots\\ CA^{N_p}\end{bmatrix} \quad (12)$$

$$\varphi = \begin{bmatrix}CB & 0 & 0 & \cdots & 0\\ CAB & CB & 0 & \cdots & 0\\ CA^2B & CAB & CB & \cdots & 0\\ \vdots & \vdots & \vdots & \cdots & \vdots\\ CA^{N_p-1}B & CA^{N_p-2}B & CA^{N_p-3}B & \cdots & CA^{N_p-N_c}B\end{bmatrix} \quad (13)$$

No exemplo do sistema de turbina gerador, as matrizes F e φ para um horizonte de predição $N_p = 4$ e $N_c = 2$, são vistas nas Equações 14 e 15.

$$F = \begin{bmatrix} 8{,}4662 & 128{,}1679 & 441{,}0765 & 1 \\ 17{,}4249 & 249{,}6569 & 760{,}1703 & 1 \\ 24{,}1322 & 338{,}1899 & 974{,}7647 & 1 \\ 28{,}6956 & 397{,}8690 & 1115{,}2174 & 1 \end{bmatrix} \quad (14)$$

$$\varphi = \begin{bmatrix} 1{,}9641 & 0 \\ 6{,}0302 & 1{,}9641 \\ 9{,}5135 & 6{,}0302 \\ 11{,}9849 & 9{,}5135 \end{bmatrix} \quad (15)$$

O modelo preditivo é uma técnica baseada na redução de uma função de otimização que chama-se de função custo ou função objetivo e é simbolizada pela letra J. Uma típica equação para a função custo é vista na Equação 16.

$$J = (R_s - Y)^T (R_s - Y) + \Delta U^T \bar{R} \Delta U \quad (16)$$

Os parâmetros desta equação consistem em penalizar a lei de controle, caso não atinja o que se deseja. A primeira variável é o erro, ou seja, a diferença entre a saída da planta e o sinal de referência, a fim de obter o rastreamento. A referência $r(k_i)$ na Equação 16 encontra-se na variável R_s que é vista na Equação 17.

$${R_s}^T = \overbrace{[1 \quad 1 \quad \cdots \quad 1]}^{N_p} r(k_i) = \overline{R_s} r(k_i) \quad (17)$$

O segundo parâmetro, da Equação 16, é a variação da lei de controle para evitar que uma entrada muito divergente cause danos ao sistema a ser controlado. O elemento $\overline{\mathrm{R}}$ é visto na Equação 18, no qual o parâmetro r_w é chamado de peso e sua função é ajustar o desempenho em malha fechada, e I_{NcxNc} é a matriz identidade quadrada de dimensão N_c.

$$\bar{R} = r_w I_{N_c \times N_c}, (r_w \geq 0) \quad (18)$$

Para encontrar a melhor trajetória de controle futura basta achar o mínimo da função custo e isto é feito ao derivar a Equação 16 em relação ao sinal de controle e igualá-la a zero. Na Equação 19 apresenta-se o resultado da derivação e o isolamento da variável ΔU, o incremento da lei de controle.

$$\Delta U = \left(\varphi^{\mathrm{T}} \varphi + \overline{\mathrm{R}}\right)^{-1} \varphi^T (\overline{R_s} r(k_i) - F x(k_i)) \quad (19)$$

Devido ao princípio do horizonte retrocedente somente utiliza-se o primeiro elemento do ΔU calculado pela Equação 19. Desta maneira reescrevendo a fórmula do incremento da lei de controle tem-se a Equação 20 em função dos ganhos de realimentação que são descritos pelas Equações 21, 22 e 23.

$$\Delta U = K_y r(k_i) - K_{mpc} x(k_i) \quad (20)$$

$$K_{mpc} = [1 \quad 0 \quad \cdots \quad 0]\left(\varphi^{\mathrm{T}}\varphi + \bar{\mathrm{R}}\right)^{-1}(\varphi^{T}F) \tag{21}$$

$$K_{mpc} = [K_x \quad K_y] \tag{22}$$

$$K_y = [1 \quad 0 \quad \cdots \quad 0]\left(\varphi^{\mathrm{T}}\varphi + \bar{\mathrm{R}}\right)^{-1}(\varphi^{T}\begin{bmatrix}1\\1\\\vdots\\1\end{bmatrix}) \tag{23}$$

Para o exemplo do sistema turbina gerador, considerando o peso $r_w = 0$, os ganhos K_{mpc} e K_y são vistos nas Equações 24 e 25.

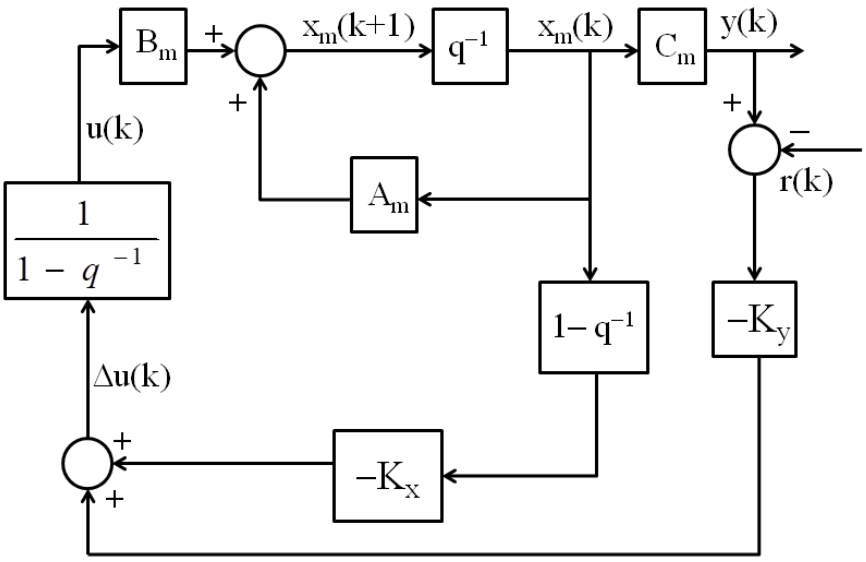
$$K_{mpc} = [3{,}4554 \quad 50{,}8388 \quad 164{,}5619 \quad 0{,}2823] \tag{24}$$

$$K_y = 0{,}2823 \tag{25}$$

Na Figura 2 mostra-se o bloco diagrama do MPC generalizado que segue a metodologia proposta, sendo a realimentação dos estados multiplicada pelo ganho K_x e a realimentação do erro ou da diferença entre a saída e a referência sendo multiplicada pelo ganho K_y.

No método MPC descrito recebe o nome de modelo preditivo generalizado (GPC), assume-se que todas as variáveis de estado são mensuráveis, na prática nem sempre isto é possível. Para contornar este problema, uma técnica, muita utilizada no campo da engenharia e da ciência, é estimar as variáveis de estado que não podem ser medidas por meio de um instrumento chamado observador.

Figura 2 - Bloco diagrama do MPC generalizado



Fonte: [1]

Na Figura 3 apresenta-se o bloco diagrama do MPC com observador. A diferença, para a metodologia anterior sem observador, é que a predição dos

estados estimados é calculada pela Equação 26, e depois esta predição é utilizada para obter a lei de controle por meio dos ganhos K_y e K_{mpc}, sendo então aplicada na planta real como visto na Equação 27.

$$\hat{x}(k_i + 1) = A\hat{x}(k_i) + B\Delta u(k_i) + K_{ob}(y(k_i) - C\hat{x}(k_i)) \quad (26)$$

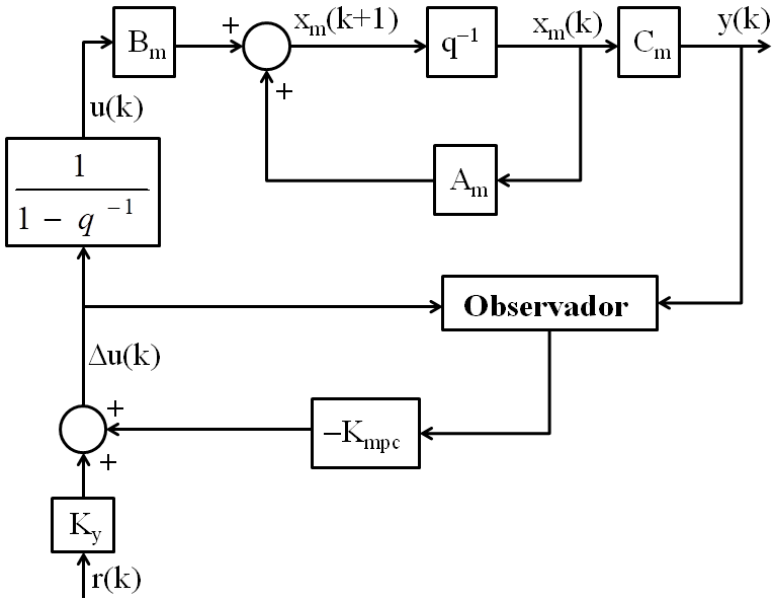
$$x(k + 1) = Ax(k) + BK_y r(k) - BK_{mpc}\hat{x}(k) \quad (27)$$

O ganho do observador representado pela variável K_{ob} é calculado pelo método de alocação de pólos. O valor do ganho K_{ob} é obtido pela fórmula de Ackermann ou pelo comando *place* do MATLAB.

No exemplo do sistema de turbina e gerador, os pólos discretos escolhidos por simplicidade foram próximos de zero e são: 0,001 0,00105 0,0011 0,00108. Os valores do ganho K_{ob} são vistos na Equação 28.

$$K_{ob} = [-0{,}0012 \quad -0{,}0011 \quad 0{,}0018 \quad 1{,}9161] \quad (28)$$

Figura 3 - Bloco diagrama do MPC com observador



Fonte: [1]

Neste trabalho foram realizadas simulações com as duas abordagens de modelo preditivo generalizado, sem e com observador, e os resultados são apresentados no capítulo 3.

1.2 MODELO PREDITIVO DE CONTROLE COM RESTRIÇÕES

Como mencionado na seção anterior uma das vantagens de utilizar a estratégia de controle preditivo é a facilidade de incorporar restrições. Logo, esta seção tem por objetivo descrever como implementar as restrições na teoria do MPC.

Esta atribuição é fundamental para garantir o funcionamento correto e eficiente de alguns processos industriais. Por exemplo, válvulas não podem estar mais do que 100% abertas ou mais fechadas do que 0%. Em algumas plantas pode ser necessário que a operação ocorra em determinados valores de temperatura, tensão, etc.

Em geral, há três principais tipos de restrições encontradas nas aplicações de controle. As duas primeiras com relação as variáveis de controle *u(k)* e a terceira com relação a saída *y(k)* ou os estados *x(k)*. Normalmente estas restrições são apresentadas como desigualdades lineares, como visto nas Equações 29, 30 e 31.

Na Equação 29 indica-se à restrição na amplitude de controle, que é a mais encontrada. Esta restrição é considerada rígida, pois geralmente a variável de controle (u(k)) não pode ultrapassar seu limite superior ou inferior.

Na Equação 30 indica-se a restrição na variação incremental da lei de controle. Esta restrição também é considerada rígida, pois há casos em que a taxa de mudança no controle é limitada em algum valor, como por exemplo a inclusão de maior concentração de reagentes em um processo químico não pode ser maior que um determinado valor. Segundo [1] , esta restrição ainda pode impor uma direção a lei de controle, por exemplo se u(k) apenas puder aumentar e não diminuir, neste caso o limite mínimo de Δu(k) deve ser igual zero.

$$u^{min} \leq u(k) \leq u^{max} \tag{29}$$

$$\Delta u^{min} \leq \Delta u(k) \leq \Delta u^{max} \tag{30}$$

$$y^{min} \leq y(k) \leq y^{max} \tag{31}$$

Na Equação 31 apresenta-se a restrição da saída da planta. Entretanto, este tipo de restrição é geralmente implementado como uma restrição “suave” e, a ela é adicionado uma variável de folga $s_v > 0$, conforme mostra a Equação 32.

$$y^{min} - s_v \leq y(k) \leq y^{max} + s_v \tag{32}$$

Conforme explica [1] , o motivo de adicionar a variável de folga e transformar a restrição da saída em “suave” é que esta restrição impacta em grandes mudanças em ambas variáveis tanto de controle quanto de seu incremento o que pode levar a estas variáveis a violar suas próprias restrições o que leva a um problema de conflito

de restrições. A variável de folga é então selecionada quando as restrições das variáveis de controle são mais essenciais para a operação da planta.

Uma vez que as restrições são formuladas como parte importante para o funcionamento do processo é necessário agora escrevê-las como inequações lineares e relacioná-las com a teoria do MPC. O ponto chave para isto é parametrizar as variáveis que possuem restrição com o mesmo parâmetro usado no projeto do MPC, o vetor ΔU. Na literatura de otimização este vetor ΔU é frequentemente chamado de variável de decisão. Para combinar as restrições com a função custo do MPC é preciso decompor as inequações em duas partes para representar o limite inferior e o superior. Neste sentido, as restrições impostas a taxa de variação da lei de controle $\Delta u(k)$ são expressadas pelas Equações 33 e 34.

$$-\Delta U \leq -\Delta U^{min} \tag{33}$$

$$\Delta U \leq \Delta U^{max} \tag{34}$$

No formato matricial compacto as Equações 33 e 34 se tornam a Equação 35, sendo que ΔU^{min} e ΔU^{max} são vetores coluna com N_c elementos de Δu^{min} e Δu^{max}, respectivamente.

$$\begin{bmatrix} -I \\ I \end{bmatrix} \Delta U \leq \begin{bmatrix} -\Delta U^{min} \\ \Delta U^{max} \end{bmatrix} \tag{35}$$

No caso da restrição na variável de controle, de acordo com [1] tem-se a Equação 36.

$$\begin{bmatrix} u(k_i) \\ u(k_i+1) \\ u(k_i+2) \\ \vdots \\ u(k_i+N_c-1) \end{bmatrix} = \overbrace{\begin{bmatrix} I \\ I \\ I \\ \vdots \\ I \end{bmatrix}}^{C_1} u(k_i-1) + \overbrace{\begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ I & I & 0 & \cdots & 0 \\ I & I & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ I & I & I & \cdots & I \end{bmatrix}}^{C_2} \begin{bmatrix} \Delta u(k_i) \\ \Delta u(k_i+1) \\ \Delta u(k_i+2) \\ \vdots \\ \Delta u(k_i+N_c-1) \end{bmatrix} \tag{36}$$

Reescrevendo a Equação 36 no formato matricial compacto com C1 e C2 correspondendo as respectivas matrizes vistas anteriormente, têm-se as Equações 37 e 38, sendo que U^{min} e U^{max} são vetores coluna com N_c elementos de u^{min} e u^{max}, respectivamente.

$$-(C_1 u(k_i-1) + C_2 \Delta U) \leq -U^{min} \tag{37}$$

$$(C_1 u(k_i-1) + C_2 \Delta U) \leq U^{max} \tag{38}$$

As restrições de saída expressas em função do vetor ΔU são vistas na Equação 39.

$$Y^{min} \leq Fx(k_i) + \varphi \Delta U \leq Y^{max} \tag{39}$$

O modelo preditivo na presença de restrições rígidas é baseado em encontrar o vetor ΔU que minimiza a função de custo da Equação 40, sujeito as desigualdades restritas da equação 41.

$$J = \left(R_S - Fx(k_i)\right)^T \left(R_S - Fx(k_i)\right) - 2\Delta U^T \varphi^T \left(R_S - Fx(k_i)\right) + \Delta U^T (\varphi^T \varphi + \bar{R}) \Delta U \tag{40}$$

$$\begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix} \Delta U \leq \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix} \tag{41}$$

Conforme [1] as matrizes M_1, M_2 e M_3 junto com as matrizes N_1, N_2 e N_3 são expressadas nas Equações 42, 43, 44, 45, 46 e 47.

$$M_1 = \begin{bmatrix} -C_2 \\ C_2 \end{bmatrix} \tag{42}$$

$$N_1 = \begin{bmatrix} -U^{min} + C_1 u(k_i - 1) \\ U^{max} - C_1 u(k_i - 1) \end{bmatrix} \tag{43}$$

$$M_2 = \begin{bmatrix} -I \\ I \end{bmatrix} \tag{44}$$

$$N_2 = \begin{bmatrix} -\Delta U^{min} \\ \Delta U^{max} \end{bmatrix} \tag{45}$$

$$M_3 = \begin{bmatrix} -\varphi \\ \varphi \end{bmatrix} \tag{46}$$

$$N_3 = \begin{bmatrix} -Y^{min} + Fx(k_i) \\ Y^{max} - Fx(k_i) \end{bmatrix} \tag{47}$$

Como a função de custo J é quadrática e as restrições são inequações lineares, o problema de encontrar a ótima lei de controle preditiva se torna a solução ótima de um problema padrão de programação quadrática. Na Equação 48 define a forma compacta da Equação 41.

$$M\Delta U \leq \gamma \tag{48}$$

A matriz M reflete as restrições com número de linhas iguais ao número de restrições e número de colunas igual a dimensão de ΔU.

A solução numérica de um processo com restrições é um problema frequente para o MPC. Como a função objetivo J é quadrática, o cálculo para encontrar a solução ótima é chamado na literatura de programação quadrática e vários métodos têm sido propostos para resolvê-la. Neste trabalho, foi utilizado o

algoritmo de *Hildreth*. Primeiro, é preciso ajustar a função de custo considerando as restrições. Na Equação 49 mostra-se a expressão de Lagrange, de acordo com a formalidade da literatura da área, em que x é a variável de decisão.

$$J = \frac{1}{2}x^T Ex + x^T F + \lambda^T(Mx - \gamma) \tag{49}$$

As matrizes E, F, M e γ são matrizes e vetores compatíveis e fazem parte do problema de programação quadrática, sendo que a matriz E é considerada ser simétrica e positiva definida.

Os elementos do vetor λ são chamados de multiplicadores de Lagrange e para a minimização desta função basta calcular a derivada parcial com relação aos vetores x e λ e depois igualá-las a zero, como apresentado nas Equações 50 e 51.

$$\frac{\partial J}{\partial x} = Ex + F + M^T\lambda = 0 \tag{50}$$

$$\frac{\partial J}{\partial \lambda} = Mx - \gamma = 0 \tag{51}$$

Os valores ótimos de x e λ são calculados de forma direta, isolando estas variáveis das Equações 50 e 51, tem-se como resposta as Equações 52 e 53.

$$\lambda = -(ME^{-1}M^T)^{-1}(\gamma + ME^{-1}F) \tag{52}$$

$$x = -E^{-1}(M^T\lambda + F) \tag{53}$$

Na Equação 53 pode-se ser escrita com dois termos, como mostra a equação 54.

$$x = -E^{-1}F - E^{-1}M^T\lambda = x^0 - E^{-1}M^T\lambda \tag{54}$$

O primeiro termo da Equação 54, $x^0 = -E^{-1}F$ é a solução ótima global do mínimo da função de custo sem as restrições, enquanto que o segundo termo é a correção devido as restrições.

Em geral, as restrições são apresentadas em forma de inequações lineares e, neste caso, pode acontecer de o número de restrições ser maior que o número de variáveis de decisão, o que aumenta a carga computacional na busca pela solução. Para resolver este dilema, uma forma simples é eliminar do processo de otimização as restrições que não são ativas.

Os multiplicadores de Lagrange, chamados na literatura de otimização de variáveis duais, podem definir se uma restrição é ativa (multiplicador com sinal positivo) ou inativa (multiplicador com sinal negativo). Então, um processo sistemático pode ser utilizado para identificar as restrições que não são ativas e encontrar a solução ótima do problema de minimização com restrições.

Este processo sistemático é a programação quadrática de *Hildreth*, um algoritmo com procedimentos de programação simples para resolver o problema. Neste algoritmo, os vetores de direção são selecionados para serem iguais aos vetores de base $e_i = [0\ 0\ ...\ 1\ ...\ 0\ 0]^T$ e desta forma o vetor λ pode ser variado um componente por vez. Por exemplo, em uma certa etapa do processo, sendo obtido um vetor $\lambda \geq 0$, o algoritmo começa com um único componente λ_i. A função custo passa a ser considerada como uma função quadrática neste componente único e então ajusta-se λ_i para minimizar a função objetivo. Caso $\lambda_i < 0$, então configura-se $\lambda_i = 0$. Em ambos os casos a função objetivo é diminuída, e o próximo componente λ_{i+1} passa a ser calculado. De acordo com [1] o método pode ser expresso pelas Equações 55 e 56.

$$\lambda_i^{m+1} = \max(0, w_i^{m+1}) \tag{55}$$

$$w_i^{m+1} = \frac{1}{h_{ii}}\left[k_i + \sum_{j=1}^{i-1} h_{ij}\,\lambda_j^{m+1} + \sum_{j=i+1}^{n} h_{ij}\,\lambda_j^{m}\right] \tag{56}$$

O escalar h_{ij} é o elemento da i-ésima linha e j-ésima coluna da matriz H vista na equação 57 e k_i é o elemento i-ésimo do vetor K visto na equação 58.

A principal vantagem da progamação quadrática de *Hildreth* é que por sua busca ser elemento por elemento, não há a necessidade de realizar uma inversão de matriz.

$$H = ME^{-1}M^T \tag{57}$$

$$K = \gamma + ME^{-1}F \tag{58}$$

As variáveis duais irão convergir se as restrições ativas forem linearmente independentes e seu número for menor ou igual ao número de variáveis de decisão. Não haverá convergência se um destes requerimentos ou os dois forem violados. Este método iterativo possui um contador que quando atinge o seu valor máximo encerra o procedimento. Além disso, o algoritmo de *Hildreth* possui a habilidade de se recuperar automaticamente de um problema com restrições mal condicionado, o fundamental para garantir a segurança da operação da planta.

Uma vez descritas as restrições na formalidade matemática e com o método de otimização a ser calculado, falta aplicá-los ao problema do MPC. A função objetivo de um MPC pode ser definida pela equação 59.

$$J = \Delta U^T(\varphi^T\varphi + \bar{R})\Delta U - 2\Delta U^T\varphi^T(R_s - Fx(k_i)) \tag{59}$$

As matrizes E e F ligam a Equação 59 com a Equação 49 são vistas nas Equações 60 e 61:

$$E = 2(\varphi^T \varphi + \bar{R}) \tag{60}$$

$$F = -2\varphi^T (R_s - Fx(k_i)) \tag{61}$$

Mais detalhes sobre os métodos de solução da programação quadrática podem ser encontrados em [1] e [6].

1.3 MPC COM FUNÇÕES DE LAGUERRE

Como visto em [1] , a ideia base do MPC discreto visto das seções anteriores é a otimização de uma futura trajetória de controle futura. Assumindo um horizonte de controle finito N_c, a diferença do sinal de controle $\Delta u(k)$ para $k = 0,1,2,\dots,N_c - 1$ é capturada pelo vetor de controle ΔU enquanto o resto de $\Delta u(k)$ para $k = N_c, N_c + 1, \dots, N_P$ é presumido ser zero. Entretanto, há casos em que nem sempre a trajetória de controle negligenciada é zero, apesar de ter uma magnitude pequena. O desenvolvimento da teoria de MPC com a introdução de um conjunto de funções de base ortonormais discreta para capturar a dinâmica do processo pode permitir um grande horizonte de controle sem utilizar uma grande quantidade de parâmetros.

Na Equação 10 vista na seção anterior indica o horizonte de controle que no instante k_i, qualquer elemento pode ser representado pela função discreta δ em conjunto do ΔU, como observado na Equação 62.

$$\Delta u(k_i + i) = [\delta(i) \quad \delta(i - 1) \quad \cdots \quad \delta(i - N_c + 1)]\Delta U \tag{62}$$

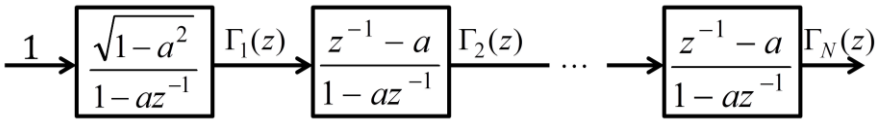
A função discreta δ é chamada de operador pulso sendo que $\delta(i) = 1$ se $i = 0$ e $\delta(i) = 0$ se $i \neq 0$. A função $\delta(i\text{-}d)$ desloca o pulso conforme a variação do parâmetro *d*. Pela Equação 62 é claro perceber que a função pulso é utilizada para capturar a sequência Δu, e por isso o uso de outra base ortonormal como as funções de Laguerre para aproximar a trajetória de controle futura é fundamental para controle de processo com dinâmica complicada em malha aberta.

A rede de filtros discretos de Laguerre é gerada da discretização da rede contínua de Laguerre. As transformadas z dos filtros discretos são vistas na Equação 63. O parâmetro *a*, chamado na literatura de fator escalar, é o pólo da rede de Laguerre com o objetivo de garantir a estabilidade da rede. Caso *a* seja igual a

zero, as funções de Laguerre se tornam um conjunto de pulsos. O parâmetro N é a ordem da cadeia de filtros. Na Figura 4 mostra-se a rede de Laguerre.

$$\begin{aligned}\Gamma_1(z) &= \frac{\sqrt{1-a^2}}{1-az^{-1}} \\ \Gamma_2(z) &= \frac{\sqrt{1-a^2}}{1-az^{-1}}\frac{z^{-1}-a}{1-az^{-1}} \\ &\vdots \\ \Gamma_N(z) &= \frac{\sqrt{1-a^2}}{1-az^{-1}}\left(\frac{z^{-1}-a}{1-az^{-1}}\right)^{N-1}\end{aligned} \tag{63}$$

Figura 4 - Rede de Laguerre Discreta



Fonte: [1]

As funções de Laguerre discretas são obtidas com a transformada z inversa da rede de Laguerre e expressas por um vetor L(k) como apresentada na Equação 64.

$$L(k) = [l_1(\mathrm{k}) \quad l_2(\mathrm{k}) \quad \cdots \quad l_N(\mathrm{k})]^T \tag{64}$$

As funções discretas de Laguerre satisfazem a Equação diferencial 65, no qual a matriz A_l é uma matriz de Toeplitz e tem dimensão NxN e seus elementos são dependentes dos parâmteros *a* e $\beta = (1\text{-}a^2)$.

$$L(k+1) = A_l L(k) \tag{65}$$

A condição inicial é dada pela Equação 66.

$$L(0)^T = \sqrt{\beta}[1 \quad -a \quad a^2 \quad -a^3 \quad \cdots \quad (-1)^{N-1}a^{N-1}] \tag{66}$$

Na Equação 67 mostra-se a matriz A_l.

$$A_l = \begin{bmatrix} a & 0 & 0 & 0 & \cdots & 0 \\ \beta & a & 0 & 0 & \cdots & 0 \\ -a\beta & \beta & a & 0 & \cdots & 0 \\ a^2\beta & -a\beta & \beta & a & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ (-1)^{N-2}a^{N-2}\beta & (-1)^{N-3}a^{N-3}\beta & \cdots & \cdots & \beta & a \end{bmatrix} \tag{67}$$

No desenvolvimento do MPC as funções de Laguerre são utilizadas no domínio do tempo e por isso sua característica de ortonormalidade se mantém neste domínio e é expressa pelas Equações 68 e 69.

$$\sum_{k=0}^{\infty} l_i(k) l_j(k) = 0 \quad para \quad i \neq j \tag{68}$$

$$\sum_{k=0}^{\infty} l_i(k) l_j(k) = 1 \quad para \quad i = j \tag{69}$$

As aplicações das redes de Laguerre estão mais concentradas nas áreas de identificação de sistemas, no qual as funções de Laguerre modelam a dinâmica de um sistema por meio da resposta ao impulso discreto. Dado a resposta ao impulso H(k) de um sistema estável e um número N de termos, a resposta ao impulso é representada pela Equação 70.

$$H(k) = c_1 l_1(k) + c_2 l_2(k) + \cdots + c_N l_N(k) \tag{70}$$

Os coeficientes c_1, c_2, ..., c_N da equação 70 são determinados dos dados do sistema a ser controlado. Devido a propriedade ortonormal das funções de Laguerre os coeficientes da rede de Laguerre são definidos pela Equação 71.

$$c_i = \sum_{k=0}^{\infty} H(k) l_i(k) \tag{71}$$

É a descrição da resposta ao impulso no tempo discreto que direciona o projeto do controle preditivo utilizando funções de Laguerre, sendo atribuído, pelo usuário, os parâmetros do pólo de Laguerre (*a*) e o número termos N.

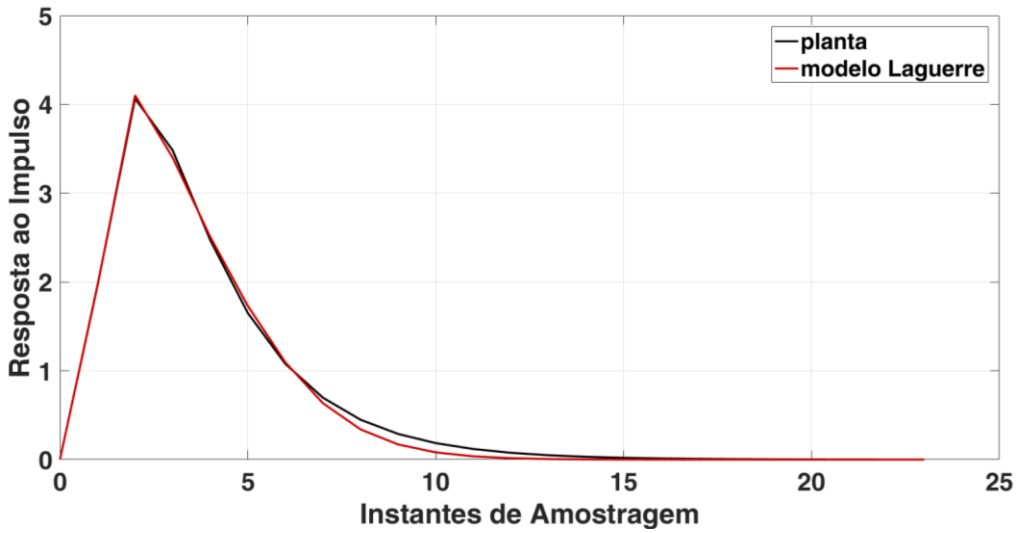
No sistema turbina gerador a resposta ao impulso é mapeada com N = 5 termos e o pólo de Laguerre a = 0,3, conforme visto na Figura 5.

Mais uma vez é ressaltado que o uso das funções de Laguerre no desenvolvimento do MPC se faz necessário em condições que ocorram solução de péssimo condicionamento numérico tais como rápida amostragem, dinâmicas de processo complicadas e/ou alta demanda em desempenho na malha fechada. Estes sistemas exigem alta quantidade de parâmetros para uma aproximação satisfatória do sinal de controle o que leva a alta carga computacional quando implementado on-line.

No instante inicial k_i, a futura trajetória de controle é considerada como a resposta ao impulso de um sistema dinâmico estável. Logo, um conjunto de funções de Laguerre é utilizado para capturar a dinâmica da resposta com um conjunto de

coeficientes de Laguerre que será determinado do processo. Na Equação 72 indica a futura trajetória de controle em um instante k.

Figura 5 – Resposta ao impulso mapeada com funções de Laguerre



Fonte: Do autor

$$\Delta u(k_i + k) = \sum_{j=1}^{N} c_j(k_i) l_j(k) \tag{72}$$

Na estrutura da Equação 72, o horizonte de controle N_c da abordagem do MPC generalizado desaparece, sendo que a complexidade da trajetória agora é descrita pelo número de termos N e pelo polo *a* de Laguerre. Um grande valor do parâmetro *a* pode ser selecionado para capturar um longo horizonte de controle com menor valor de N para o procedimento de otimização. Quando *a* = 0 então N = N_c e a metodologia do MPC se torna igual a das seções anteriores. Na Equação 72 pode ser escrita em formato vetorial como mostra a Equação 73.

$$\Delta u(k_i + k) = L(k)^T \eta \tag{73}$$

O parâmetro $L(k)^T$ é o vetor transposto das funções de Laguerre definido pela Equação 65. O parâmetro η é um vetor com os N coeficientes de Laguerre como mostra na Equação 74.

$$\eta = [c_1 \quad c_2 \quad \cdots \quad c_N]^T \tag{74}$$

Substituindo o vetor ΔU pela Equação 73 no espaço de estado aumentado, a predição das futuras variáveis de estado e de saída no instante *m* é visto nas Equações 75 e 76.

$$x(k_i + m|k_i) = A^m x(k_i) + \sum_{i=0}^{m-1} A^{m-i-1} BL(i)^T \eta \tag{75}$$

$$y(k_i + m|k_i) = CA^m x(k_i) + \sum_{i=0}^{m-1} CA^{m-i-1} BL(i)^T \eta \tag{76}$$

Ambas as Equações 75 e 76 estão em função do vetor de coeficientes η da rede de Laguerre e é este vetor que será otimizado e calculado no projeto do MPC.

A função objetivo que antes era descrita em formato matricial, passa a ser escrita na forma de vetores como visto na Equação 77, em que R_L é uma matriz diagonal (N x N) com rw ≥ 0 na sua diagonal e $r(k_i)$ sendo a referência para a saída no instante k_i.

$$J = \sum_{m=1}^{N_p} \big(r(k_i) - y(k_i + m|k_i)\big)^T \big(r(k_i) - y(k_i + m|k_i)\big) + \eta^T R_L \eta \tag{77}$$

Por questões de simplicidade é comum a função custo ser baseada na minimização do erro entre o sinal de referência e a saída medida. Porém, no MPC com funções de Laguerre esta função custo é reformulada de modo estabelecer uma conexão com os reguladores quadráticos lineares discretos no tempo (DLQR). Na Equação 78 apresenta-se a função custo reformulada comas matrizes de peso Q ≥ 0 e R_L> 0.

$$J = \sum_{m=1}^{N_p} x(k_i + m|k_i)^T Q x(k_i + m|k_i) + \eta^T R_L \eta \tag{78}$$

A matriz Q tem dimensão igual ao número das variáveis de estado e a matriz R_L tem dimensão igual ao vetor de coeficientes η. Em geral é atribuída a matriz Q = C^TC e o sinal de referência é incluído nas variáveis de estado. Na abordagem tradicional as variáveis de estado aumentada eram x(k) = $[\Delta xm(k)^T\ y(k)]^T$. Aqui as variáveis de estado utilizadas são as da Equação 79.

$$x(k_i + m|k_i) = \begin{bmatrix} \Delta x(k_i + m|k_i)^T & y(k_i + m|k_i) - r(k_i) \end{bmatrix}^T \tag{79}$$

É importante observar que a inclusão da referência não altera o modelo utilizado para as predições.

Antes de minimizar a função custo é preciso reescrever a Equação 75, onde introduziremos a matriz ϕ conforme a Equação 80. É importante notar que o número de linhas de ϕ é igual ao número de linhas de η. Na Equação 81 indicam-se as variáveis de estado reescritas.

$$\phi(m)^T = \sum_{i=0}^{m-1} A^{m-i-1} B L(i)^T \tag{80}$$

$$x(k_i + m|k_i) = A^m x(k_i) + \phi(m)^T \eta \tag{81}$$

Substituindo a nova equação das variáveis de estado na função custo, a Equação 82 é obtida.

$$\begin{aligned} J = \eta^T \left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right) \eta + 2\eta^T \left(\sum_{m=1}^{N_p} \phi(m) Q A^m \right) x(k_i) \\ + \sum_{m=1}^{N_P} x(k_i)^T (A^T)^m Q A^m x(k_i) \end{aligned} \tag{82}$$

Sem considerar restrições, o mínimo da equação 82 é obtido com a derivada parcial da função custo visto na Equação 83.

$$\frac{\partial J}{\partial \eta} = 2 \left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right) \eta + 2 \left(\sum_{m=1}^{N_p} \phi(m) Q A^m \right) x(k_i) \tag{83}$$

Assumindo que $\left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right)^{-1}$ exista quando $\frac{\partial J}{\partial \eta} = 0$, a solução ótima do vetor é visto na Equação 84.

$$\eta = -\left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right)^{-1} \left(\sum_{m=1}^{N_p} \phi(m) Q A^m \right) x(k_i) \tag{84}$$

Por simplicidade, definem-se as variáveis Ω (ômega) e Ψ (psi), vistas nas Equações 85 e 86.

$$\Omega = \left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right) \tag{85}$$

$$\Psi = \left(\sum_{m=1}^{N_p} \phi(m) Q A^m \right) \tag{86}$$

A Equação 84 reescrita na forma simples é vista na Equação 87.

$$\eta = -\Omega^{-1} \Psi x(k_i) \tag{87}$$

No sistema turbo-gerador para a configuração apresentada anteriormente e considerando $N_p = 5$, temos as seguintes matrizes ômega e psi, vistas nas Equações 88 e 89.

$$\Omega = \begin{bmatrix} 684{,}2115 & 344{,}2536 & 59{,}8453 & -39{,}4374 & -5{,}3247 \\ 344{,}2536 & 202{,}2225 & 46{,}4560 & -24{,}4517 & -5{,}2914 \\ 59{,}8453 & 46{,}4560 & 18{,}2278 & -5{,}6271 & -3{,}3546 \\ -39{,}4374 & -24{,}4517 & -5{,}6271 & 3{,}4848 & 0{,}3448 \\ -5{,}3247 & -5{,}2914 & -3{,}3546 & 0{,}3448 & 1{,}0657 \end{bmatrix} \tag{88}$$

$$\Psi = \begin{bmatrix} 1365{,}3977 & 18998{,}3536 & 53756{,}0925 & 51{,}3059 \\ 648{,}8768 & 8970{,}5049 & 24947{,}6633 & 21{,}3873 \\ 102{,}6419 & 1406{,}8390 & 3821{,}8630 & 2{,}9020 \\ -72{,}8245 & -1004{,}8424 & -2780{,}1668 & -2{,}3305 \\ -8{,}4513 & -115{,}1478 & -307{,}6347 & -0{,}2110 \end{bmatrix} \tag{89}$$

Com a definição das matrizes Ω e Ψ, o cálculo das predições é feito por recursividade por meio da soma de convolução. Por exemplo, considere a Equação 90.

$$S_c(m) = \sum_{i=0}^{m-1} A^{m-i-1} BL(i)^T \tag{90}$$

Na Equação 91 apresenta-se um exemplo para três amostras, onde é utilizado a equação diferencial $L(k+1) = A_l L(k)$ para gerar o conjunto das funções de Laguerre.

$$\begin{gathered} S_c(1) = BL(0)^T \\ S_c(2) = ABL(0)^T + BL(1)^T = ABL(0)^T + BL(0)^T {A_l}^T = AS_c(1) + S_c(1){A_l}^T \\ S_c(3) = A^2 BL(0)^T + ABL(1)^T + BL(2)^T = AS_c(2) + S_c(1)\left({A_l}^2\right)^T \end{gathered} \tag{91}$$

A recursão pode ser escrita de forma reduzida como na Equação 92.

$$S_c(m) = AS_c(m-1) + S_c(1)\left({A_l}^{m-1}\right)^T \tag{92}$$

Após o cálculo das predições por soma de convolução e da solução ótima do vetor de coeficientes η, a lei de controle $\Delta u(k)$ pode ser escrita na forma de realimentação de estados com o ganho K_{mpc} como mostram as Equações 93, 94 e 95, o sinal de referência está contido nas variáveis de estado.

$$\Delta u(k_i) = L(0)^T \eta \tag{93}$$

$$\Delta u(k_i) = -K_{mpc} x(k) \tag{94}$$

$$K_{mpc} = L(0)^T \left(\left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right)^{-1} \sum_{m=1}^{N_p} \phi(m) Q A^m \right) \tag{95}$$

O ganho Kmpc pode ser escrito no formato mais simplificado com as matrizes Ω e Ψ (Equação 96).

$$K_{mpc} = L(0)^T \Omega^{-1} \Psi \tag{96}$$

1.4 REVISÃO BIBLIOGRÁFICA

Nesta seção é feita uma análise sobre alguns artigos mais recentes publicados na área de controle de preditivo. A implementação do MPC em FPGA teve um grande volume de trabalho entre os anos de 2012 a 2015. Em primeiro lugar, descreve-se o artigo que cita os problemas de realizar os ensaios do MPC em meio físico. Em seguida, descrevem-se as ferramentas que estão sendo utilizadas para criar os algoritmos que implementam o MPC. Por fim, alguns artigos mais recentes publicados na área.

No próximo capítulo é detalhado o problema ocorrido com o atraso de tempo das variáveis amostradas para implementação do MPC em meio digital. Este problema foi abordado com soluções diferentes da proposta por este trabalho. Por exemplo, em 2015 na ROPEC (*IEEE International Autumn Meeting on Power, Eletronics and Computing*) os pesquisadores do departamento de Eletrônica do INAOE (*National Institute of Astrophysics, Optics and Eletronics*) de Puebla no México desenvolveram um MPC, baseado na tecnologia FPGA, para controle de máquina síncrona de imã permanente.

Neste artigo [7] é discutido de forma ampla o problema da implementação digital do MPC e a principal contribuição dele é uma técnica para compensação do atraso dos dados, afinal este atraso afeta o desempenho do sistema. Do artigo em [7], a proposta para compensação do atraso é resumida da seguinte forma:

1 – Aplicação da lei de controle no tempo de amostragem anterior;

2 – Medição das variáveis de estado;

3 – Estimação da saída da planta;

4 – Estimação das variáveis de estado para cálculo da lei de controle no próximo tempo de amostragem;

5 – Estimação da saída para aplicação da lei de controle no próximo tempo de amostragem;

6 – Predição das variáveis de estado para cada possibilidade de lei de controle no tempo de amostragem k+2;

7 – Predição das variáveis de saída da planta para cada possível estado no tempo de amostragem k+2;

8 – Avaliação da função de custo para cada predição;

9 – Minimização da função de custo;

10 – Aplicação da nova lei de controle.

Este artigo foi explorado neste trabalho, pois o controlador também foi implementado em linguagem VHDL e todas as variáveis foram codificadas utilizando o complemento a 2 e aritmética de ponto fixo, com 15 bits para a parte real, 16 bits para a parte fracionária e 1 bit para o sinal. A cosimulação, porém, foi executada entre o Matlab/Simulink e a ferramenta Active-HDL e não houve teste de *FPGA-in-the_loop.*

Os resultados mostraram que após a aplicação da técnica de compensação do atraso houve uma significativa redução da ondulação (*ripple*) nas variáveis de controle. Segundo os pesquisadores, esta técnica é simples e muito eficiente, além de não requerer uma alta quantidade de recursos para implementação digital.

Para testes em *FPGA-in-the-loop*, dois artigos muito citados de pesquisadores que desenvolveram ferramentas para embarcar o MPC são o PROTOIP [8] e o mμAO-MPC [9].

De [8], o PROTOIP é um *framework* de código aberto para projetar, validar e prototipar algoritmos, de forma rápida, em plataformas FPGA e foi desenvolvido por pesquisadores do Imperial College London. Neste artigo dois processos foram controlados, uma ponte rolante e um pêndulo invertido, sendo que para cada processo uma teoria diferente de MPC foi projetada, *fast gradient MPC* e o *Explicit MPC*. A grande contribuição foi realizar testes de *FPGA-in-the-loop* e comparar o circuito gerado pelo PROTOIP com o gerado pelo HDL (Hardware description language) Coder.

Utilizar o HDL Coder [10] do *MATLAB* ou o *LABVIEW* [11] da National Instruments, permite um maior nível de abstração o que facilita o desenvolvimento de projetos, além de ser uma ferramenta flexível a ponto de permitir a implementação em diversas plataformas de hardware. A desvantagem é a qualidade do circuito digital gerado com alto consumo de potência e baixa eficiência em aplicações onde é exigido uso intensivo de computação.

A PROTOIP basicamente executa um papel intermediário entre as ferramentas de síntese de alto nível (HLS) da Xilinx (Vivado) e algoritmos descritos em linguagem C. As HLS estão se tornando cada vez mais inteligente a ponto de conseguir identificar a configuração ideal do ciruito bem como permitir que os usuários alternem a representação dos dados sem se preocuparem com a sincronização dos dados.

Um dos responsáveis pela criação do PROTOIP, o pesquisador Eric Kerrigan participou de outros artigos que ajudaram na identificação de problemas para a implementação do MPC no meio digital. Em [12] e [13], são discutidos alguns problemas da representação de dados com a utilização do ponto fixo, tais como overflow, erros de cálculo nas operações de multiplicação e na conversão de outro tipo de dado para o ponto fixo. O processo controlado em [12] foi um microscópio de força atômica (AFM) e sua a maior contribuição foi a implementação do método de gradiente rápido para resolver a otimização do MPC. Em [13] foram feitas duas arquiteturas computacionais para resolver dois métodos diferentes de MPC, o gradiente rápido de Nesterov e ADMM (*alternating direction method of multipliers*). O processo controlado foi um conjunto de massas oscilantes presas na parede. A grande contribuição é uma análise para determinar o número mínimo de bits para utilizar o mínimo de recurso e garantir estabilidade numérica e precisão da solução.

Em [14], foi projetado um MPC para controle de uma aplicação em aeronave. Outro algoritmo para realizar a otimização foi implementado em FPGA, o PIDP (*primal dual interior point*). O interessante deste artigo é que os pesquisadores fizeram um levantamento dos algoritmos de solução da otimização que foram implementados em FPGA. Por exemplo, em [15] pesquisadores da *University of Sydney* implemetaram o procedimento de programação de *Hildreth* junto com a teoria do MPC com funções de Laguerre. Neste artigo, em que o processo a ser controlado é uma estabilização de satélite, foi desenvolvido um processador com 5 estágios de *pipeline* para realizar os cálculos do MPC. Porém, não foram feitos os testes de *FPGA-in-the-loop*, sendo feito apenas a cosimulação.

Em [9] [15] é apresentado o mμAO-MPC desenvolvida pelos pesquisadores da *Universität Berlin* em conjunto com a *University of Zaragoza* e *University Magdeburg*. Diferente da PROTOIP que usa o ambiente MATLAB, esta ferramenta foi feita em linguagem Phyton e possui uma utilização mais amigável para o usuário,

de maneira que não especialistas na área podem projetar e otimizar automaticamente uma implementação de MPC em FPGA.

Basicamente, o mμAO-MPC é um gerador de código C automático e o MPC desenvolvido por esta ferramenta já foi testado em várias plataformas que dão suporte a essa linguagem, tais como: processadores x86/AMD64, microcontroladores ARM Cortex-M, Lego Mindstorms NXT, emicrocontroladores Arduino.

No artigo [16], o mμAO-MPC é utilizado junto com a ferramenta de síntese de alto nível Vivado para controlar dois processos, um motor DC e uma cadeia de massas. A otimização foi resolvida com o método do gradiente rápido (FGM) em combinação com o Langrangeano aumentado (ALM). Este artigo tem uma ótima contribuição com a apresentação de resultados de área e consumo de potência para vários testes da implementação do MPC em FPGA.

Outras ferramentas também têm sido utilizadas para gerar código C que implementam o MPC, mas não são descritas neste trabalho, tais como: CVXGEN [17], ECOS/QMCL [18], FalcOpt [19].

Além dos problemas supracitados, a operação de calcular a matriz inversa também é um problema em hardware, como é visto no capítulo 3. Por isso, em [20] é proposta uma nova forma de calcular a inversa quando o MPC é implementado em FPGA. Basicamente a função inversa é substituída por um processo de iteração que envolve operações de multiplicação e adição, com menos operações de divisão, podendo ser calculadas de forma mais rápida se aproveitar o paralelismo e pipeline do hardware. Neste artigo o MPC proposto controlou uma máquina de moldagem por injeção.

Nos últimos anos a teoria do MPC tem sido utilizada em processos específicos. Por exemplo, em [21] o MPC teve como objetivo criar planos personalizados de terapia antiangiogênica em pacientes com câncer. Nesta aplicação o MPC foi comparado com a técnica RFPT (Robust Fixed Point Transformation). Como a planta simulada era instável e não linear, os controles não lineares mostraram melhor desempenho, sendo o MPC eficiente com diferentes níveis de saturação e referência constante e o RFPT com resultados precisos para a referência variante no tempo.

Em [22] o MPC foi projetado para melhorar o desempenho e segurança de robôs móveis controlados pelo cérebro. O MPC tinha como objetivo rastrear a intenção do usuário e garantir a segurança do robô. Foi utilizada uma interface cérebro-máquina desenvolvida pelos próprios pesquisadores e os resultados mostraram que o MPC garantiu melhor confiabilidade e eficiência para os robôs do que a aplicação sem MPC.

Em [23] o MPC foi utilizado para controlar uma planta de pressão de gás. Esta planta não linear foi controlada para comparação de dois controladores, o MPC e o controle PI. Os resultados revelaram que o controlador PI apresentou alta oscilação o que podia provocar danos ao equipamento e às válvulas de controle, além de alto tempo de estabilização e lenta convergência em comparação ao MPC.

O controle preditivo foi utilizado para controlar plantas reais. Em 2017 pesquisadores da Chalmers University of Technology em Göteborg na Suécia publicaram um artigo [24] no journal IET Generation, Transmission & Distribution sobre testes de campo para o controle de tensão de um sistema de distribuição. A principal contribuição deste artigo é a realização de testes em um sistema real, pois a maior parte dos artigos publicados de MPC em controle de sistemas elétricos é feito em ambiente de simulação. Estes testes são necessários e muito importantes para avaliar o uso prático de um algoritmo baseado em otimização como o MPC em um sistema de tempo real.

Neste artigo, para o projeto do MPC foi utilizada a mesma metodologia do capítulo 1 deste trabalho com espaço de estado discreto no qual a tensão da barra remota é um estado e a entrada do controle é a variação da potência reativa da saída de um conversor de tensão (VSC), os valores de N_p é 4 e N_c é 2 com taxa de amostragem de 10 segundos. Como este é um processo real, foi preciso usar restrições para manter a tensão da barra, e por este motivo foi necessário a ferramenta de algoritmos de programação quadrática para lidar com as restrições na tensão da barra remota e na saída da potência reativa do VSC e suas variações. A restrição da tensão foi mais flexibilizada neste trabalho pois em alguns casos o limite de potência reativa tornava a otimização inviável.

Um Arduino Uno transformava o sinal de tensão em dados digitais e um computador executava o algoritmo MPC no ambiente do MATLAB. A saída do MPC fornecia um sinal de referência para um controlador PI local de potência reativa do

VSC. Uma *lookup table* fornecida pelos operadores do VSC traduzia a tensão de 0 a 5V da saída do MPC para os valores reais de potência reativa.

Os resultados mostraram que o MPC conseguiu regular a tensão mesmo com significativos erros no modelo em espaço de estado. Outras observações foram que uma realimentação da medição de tensão suavizada assim como um atraso de tempo inicial, são críticos para o funcionamento satisfatório do controlador.

Por fim, foram analisados os trabalhos publicados por pesquisadores de universidades brasileiras. Recentemente, o autor deste trabalho participou do XXII Congresso Brasileiro de Automática (CBA) e poucos trabalhos implementaram o MPC em meio digital. Em [25], pesquisadores da Universidade Federal de Santa Catarina e da Universidade Federal da Bahia realizaram um estudo comparativo entre algumas das ferramentas listadas nesta seção para desenvolver o MPC em meio digital e prototipá-lo em sistemas embarcados. Em [26], pesquisadores da Universidade Federal do Pará projetaram um MPC com funções de Laguerre para controle de um sistema MIMO baseado na coluna de destilação binária de Wood e Berry, mas a implementação em sistemas embarcados não foi realizada. Em [27], pesquisadores da Universidade de São Paulo e da Embrapa, em São Carlos, fizeram em MPC com restrições embarcado em um microcontrolador ARM. O código do MPC foi escrito em linguagem C e o sistema a ser controlado foi um pulverizador agrícola com injeção direta.

2 METODOLOGIA E APLICAÇÕES DO MPC

Nesta seção é detalhada a metodologia de desenvolvimento do projeto do MPC e as aplicações de processo em que o controle proposto foi testado, além do sistema turbo-gerador apresentado na seção anterior.

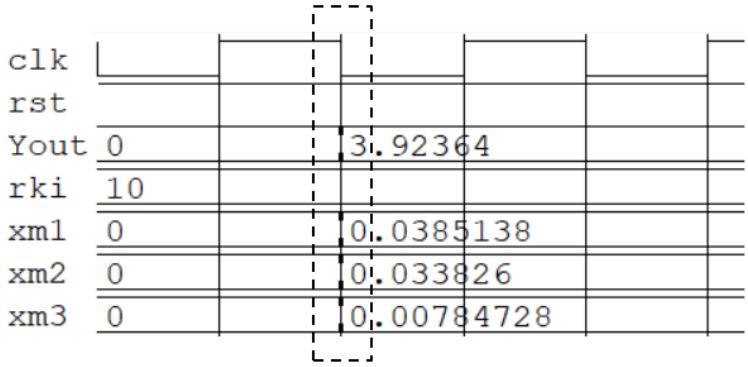
O algoritmo do MPC foi escrito em linguagem de descrição de *hardware* VHDL e a finalidade é realizar testes de cosimulação, em que o controlador é simulado pelo ModelSim PE da Mentor Graphics e a dinâmica das plantas são simuladas pelo Simulink do MATLAB, programa da MathWorks. Em seguida, deseja-se realizar também testes de *FPGA-in-the-loop*, em que o controlador é embarcado em uma plataforma FPGA DE2-115 da Altera e o processo a ser controlado segue sendo simulado pelo Simulink.

Uma configuração importante para o desenvolvimento do código VHDL que implementa o MPC em um sistema embarcado é qual o tipo de representação dos dados feito para as variáveis e sinais do VHDL. Entre duas escolhas possíveis (ponto fixo ou ponto flutuante), optou-se pela representação em ponto fixo com sinal, pois é de conhecimento que em comparação com o ponto flutuante, o ponto fixo tem execução mais rápida e ocupa menos hardware (aproximadamente três vezes menos). A biblioteca utilizada neste trabalho tem por padrão representar os dados com lógica de complemento a dois e 32 bits para os dados, sendo: 1 bit mais significativo para representar o sinal, 15 bits para representar a parte inteira e 16 bits para a parte fracionária.

Conforme visto na revisão bibliográfica no capítulo 1.4, o grande desafio de implementar o MPC em meio digital é que há um atraso de tempo no momento em que os dados são amostrados e o momento em que a lei de controle é aplicada. Na Figura 6 demonstra-se um exemplo de uma simulação de controle, sem observador, de um processo com 3 estados em que ocorre o atraso nos dados amostrados, sendo que *clk* refere-se ao *clock* do cirucito, *rst* é o sinal de reset, *Yout* é a saída da planta, *rki* o sinal de referência (*set-point*) e *xm1*, *xm2* e *xm3* os sinais dos estados. Em geral, os projetos de circuitos digitais são feitos para atualização dos dispositivos, com memória, ocorrerem em ciclos de transição do *clock* para lógica alta, mas a amostragem dos valores de saída acontece com meio ciclo de atraso na transição para lógica baixa. Se o projeto for alterado para atualização dos

dispositivos ser na transição para lógica baixa, o atraso também ocorre e neste caso a amostragem dos dados ocorre na transição para lógica alta.

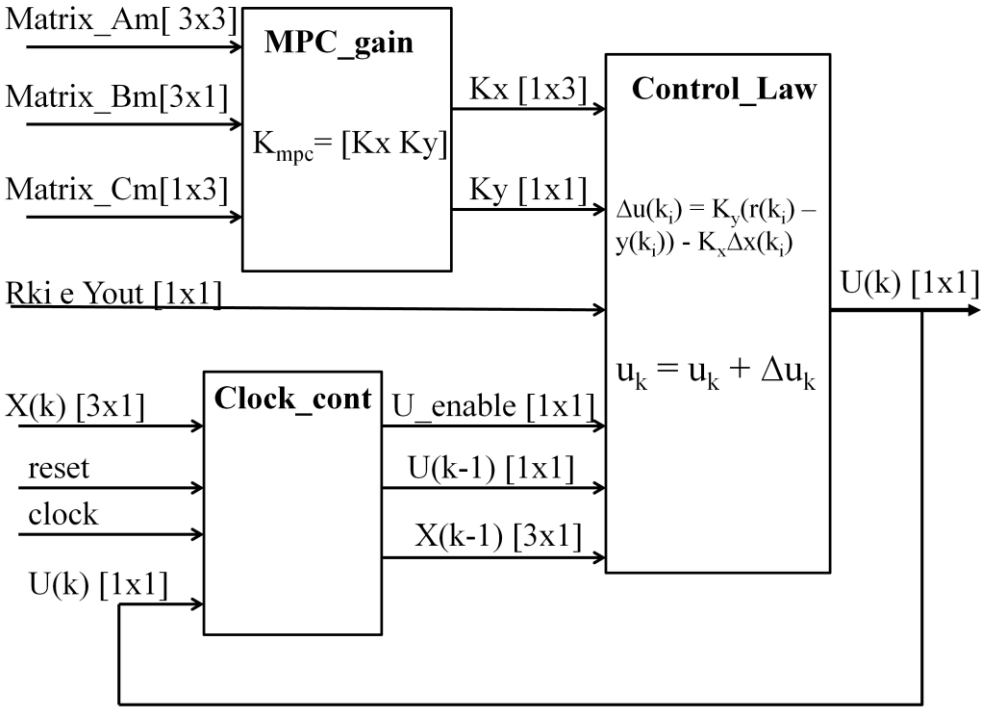
Figura 6 - Atraso de dados



Fonte: Do autor

Para contornar este problema, um contador foi incorporado ao circuito digital do MPC para que a lei de controle seja aplicada no mesmo instante de tempo em que as amostras são obtidas. Na Figura 7 mostra-se o esquemático do circuito digital do MPC sem um observador.

Figura 7 - Esquemático do MPC digital projetado

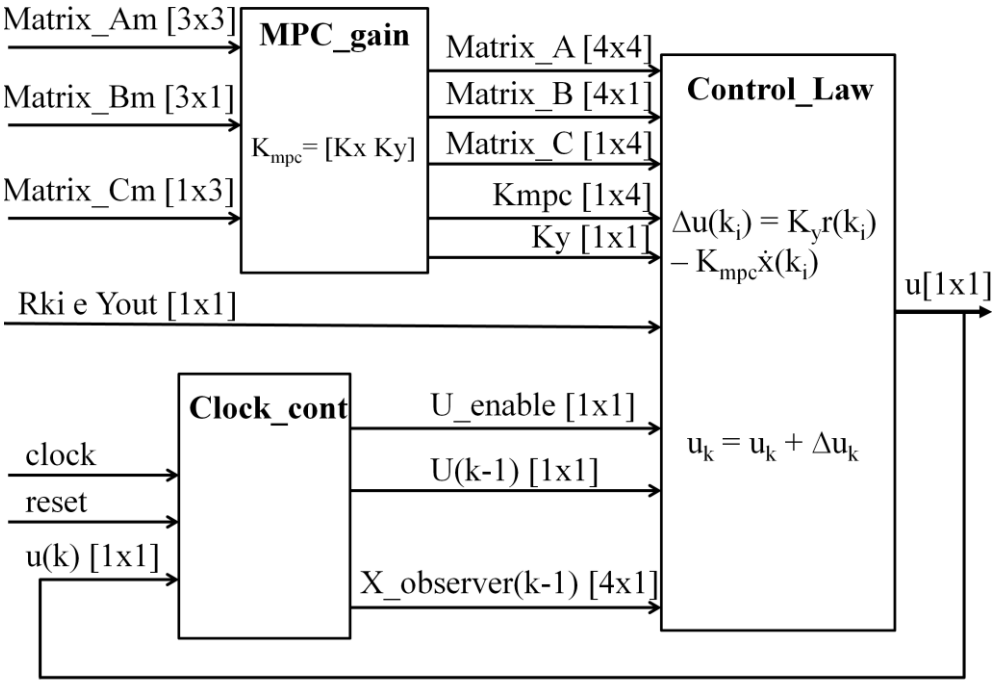


Fonte: Do autor

Basicamente, são três componentes sendo que o primeiro componente *MPC_gain* tem a função de calcular os ganhos K_{mpc} conforme as equações vistas no capítulo 1.1. O segundo componente *Clock_cont* implementa o contador e tem a função de gerar um sinal *U_enable* para sincronizar a aplicação da lei de controle no tempo correto. O terceiro componente *Control_law* tem a função de calcular a lei de controle e fornecê-la na saída do circuito.

O mesmo esquemático é utilizado para implementar o MPC com observador de estados como visto na Figura 8, sendo que as poucas mudanças são na passagem das matrizes do espaço de estado para cálculo da lei de controle conforme visto na Equação 26.

Figura 8 - Esquemático do MPC digital com observador



Fonte: Do autor

A separação dos circuitos em componentes facilita a modificação do algoritmo para testes em outras plantas, mas a principal vantagem que espera-se obter é que apenas os componentes *Clock_cont* e *Control_Law* tenham um maior consumo de potência, pois são os que possuem circuitos sequenciais. O componente *MPC_gain* foi projetado para conter apenas circuitos combinacionais.

Esta arquitetura proposta permite uma flexibilidade quando o projetista precisa modificar algum parâmetro e obter um MPC com uma nova configuração. Além disso, a implementação de novas teorias de MPC, como restrições, implicaria modificações apenas no terceiro componente. O MPC com funções de Laguerre implica mudanças também no componente *MPC_gain*, mas todos os códigos mantiveram a lógica de implementar o contador descrito.

O primeiro processo controlado por esta metodologia foi o sistema turbina gerador descrito no capítulo anterior junto com a apresentação da teoria do MPC. A seguir apresentam-se outras plantas em que também foi projetado o MPC para realizar o controle.

2.1 CONTROLE DE MÁQUINA SÍNCRONA

Uma das vantagens da técnica de controle preditivo é a facilidade em lidar com sistemas de múltiplas entradas e múltiplas saídas (MIMO). Por isso, após os resultados do sistema turbo-gerador, que gerou um artigo para o XXII Congresso Brasileiro de Automática (CBA), a fim de tornar o código mais dinâmico e habilitá-lo para dar suporte a sistemas MIMO, foi projetado um MPC para realizar o controle de velocidade de uma máquina síncrona de imã permanente (PMSM). Esta planta também serviu para cobrir uma crítica ao artigo do CBA, na qual o controlador MPC foi testado para controlar a planta não linear para o qual foi projetado.

No controle de máquina síncrona a abordagem com a estratégia de controle preditivo pode ser dividida em duas: FCS-MPC (finite control set MPC) e a dead-beat MPC que estima as referências de tensão e corrente a ser geradas por um modulador.

Como algumas técnicas de MPC utilizam a derivada da corrente do estator para cálculo da lei de controle, neste projeto foi feito uma simples manipulação matemática para que a derivada fosse evitada sem afetar o algoritmo do MPC. Entretanto, os valores das referências da corrente de eixo *d* e da velocidade são necessários para o cálculo dos ganhos do MPC, sendo que para este trabalho a referência de velocidade é igual a 1 e a referência da corrente de eixo *d* é igual a zero.

O espaço de estados que modelou a planta da máquina síncrona é visto nas Equações 97 e 98.

$$\dot{x}_m(k+1) = \overbrace{\begin{bmatrix} \frac{-b}{j_r} & \frac{1{,}5n\phi}{j_r} & 0 \\ \frac{-n\phi}{L} & \frac{-r}{L} & -n\omega_m \\ 0 & n\omega_m & \frac{-r}{L} \end{bmatrix}}^{A_m} x(k) + \overbrace{\begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{L} \\ \frac{1}{L} & 0 \end{bmatrix}}^{B_m} u(k) \tag{97}$$

$$y(k) = \overbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}^{C_m} x(k) \tag{98}$$

Os parâmetros da máquina síncrona utilizados são descritos na Tabela 1.

Tabela 1 – Parâmetros da máquina síncrona

Parameter	Value
r	0,85 Ω
L	6 mH
ϕ	0,135 Wb
j_r	5x10^{-3} kg-m^2
b	5x10^{-4} N-m-s
n	3
ω_0	100 rad/s

Fonte: [28]

Após a substituição destes parâmetros e com o tempo de amostragem para a discretização da planta de 10^{-4} segundos, o espaço de estados utilizado, para este trabalho, passa a ser o que é o visto nas Equações 99 e 100.

$$\dot{x}_m(k+1) = \overbrace{\begin{bmatrix} 0{,}9999 & 0{,}0121 & -1{,}8052 \\ -0{,}0067 & 0{,}9854 & -0{,}0296 \\ 1{,}0029e^{-4} & 0{,}0296 & 0{,}9855 \end{bmatrix}}^{A_m} x(k) + \overbrace{\begin{bmatrix} -1{,}0053e^{-6} & 1{,}0077e^{-4} \\ -2{,}4763e^{-4} & 0{,}0165 \\ 0{,}0165 & 2{,}4763e^{-4} \end{bmatrix}}^{B_m} u(k) \tag{99}$$

$$y(k) = \overbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}^{C_m} x(k) \tag{100}$$

As matrizes aumentadas para o projeto do MPC são vistas nas Equações 101 e 102.

$$\dot{x}_m(k+1) = \overbrace{\begin{bmatrix} 0{,}9999 & 0{,}0121 & -1{,}8052e^{-4} & 0 & 0 \\ -0{,}0067 & 0{,}9854 & -0{,}0296 & 0 & 0 \\ -1{,}0029e^{-4} & 0{,}0296 & 0{,}9855 & 0 & 0 \\ 0{,}9999 & 0{,}0121 & -1{,}8052e^{-4} & 1 & 0 \\ -1{,}0029e^{-4} & 0{,}0296 & 0{,}9855 & 0 & 1 \end{bmatrix}}^{A} x(k)$$
$$+ \overbrace{\begin{bmatrix} -1{,}0053e^{-6} & 1{,}0077e^{-4} \\ -2{,}4763e^{-4} & 0{,}0165 \\ 0{,}0165 & 2{,}4763e^{-4} \\ 1{,}0053e^{-6} & 1{,}0077e^{-4} \\ 0{,}0165 & 2{,}4763e^{-4} \end{bmatrix}}^{B} u(k) \quad (101)$$

$$y(k) = \overbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}}^{C} x(k) \quad (102)$$

A configuração utilizada para a simulação desta planta tem $N_p = 5$ e $N_c = 3$ com peso $r_w = 10^{-8}$. As matrizes F e φ para esta configuração são vistas nas Equações 103 e 104.

$$F = \begin{bmatrix} 0{,}9999 & 0{,}0121 & -1{,}8052e^{-4} & 1 & 0 \\ -1{,}0029e^{-4} & 0{,}0296 & 0{,}9855 & 0 & 1 \\ 1{,}9998 & 0{,}0360 & -8{,}9566e^{-4} & 1 & 0 \\ -4{,}9759e^{-4} & 0{,}0879 & 1{,}9558 & 0 & 1 \\ 2{,}9994 & 0{,}0716 & -0{,}0025 & 1 & 0 \\ -0{,}0014 & 0{,}1740 & 2{,}9103 & 0 & 1 \\ 3{,}9987 & 0{,}1188 & -0{,}0053 & 1 & 0 \\ -0{,}0029 & 0{,}2871 & 3{,}8484 & 0 & 1 \\ 4{,}9976 & 0{,}1772 & -0{,}0096 & 1 & 0 \\ -0{,}0054 & 0{,}4262 & 4{,}7696 & 0 & 1 \end{bmatrix} \quad (103)$$

$$\varphi = \begin{bmatrix} -1{,}0053e^{-6} & 1{,}0077e^{-4} & 0 & 0 & 0 & 0 \\ 0{,}0165 & 2{,}4763e^{-4} & 0 & 0 & 0 & 0 \\ -7{,}9846e^{-6} & 4{,}0107e^{-4} & -1{,}0053e^{-6} & 1{,}0077e^{-4} & 0 & 0 \\ 0{,}0328 & 9{,}8099e^{-4} & 0{,}0165 & 2{,}4763e^{-4} & 0 & 0 \\ -2{,}6752e^{-5} & 8{,}9781e^{-4} & -7{,}9846e^{-6} & 4{,}0107e^{-4} & -1{,}0053e^{-6} & 1{,}0077e^{-4} \\ 0{,}0489 & 0{,}0022 & 0{,}0328 & 9{,}8099e^{-4} & 0{,}0165 & 2{,}4763e^{-4} \\ -6{,}2945e^{-5} & 0{,}0016 & -2{,}6752e^{-5} & 8{,}9781e^{-4} & -7{,}9846e^{-6} & 4{,}0107e^{-4} \\ 0{,}0647 & 0{,}0038 & 0{,}0489 & 0{,}0022 & 0{,}0328 & 9{,}8099e^{-4} \\ -1{,}2202e^{-4} & 0{,}0025 & -6{,}2945e^{-5} & 0{,}0016 & -2{,}6752e^{-5} & 8{,}9781e^{-4} \\ 0{,}0802 & 0{,}0060 & 0{,}0647 & 0{,}0038 & 0{,}0489 & 0{,}0022 \end{bmatrix} \quad (104)$$

Os ganhos K_x e K_y calculados são vistos nas Equações 105 e 106.

$$K_x = \begin{bmatrix} -76{,}37 & 0{,}5005 & 59{,}5696 \\ 5102{,}6 & 85{,}9715 & -0{,}9150 \end{bmatrix} \quad (105)$$

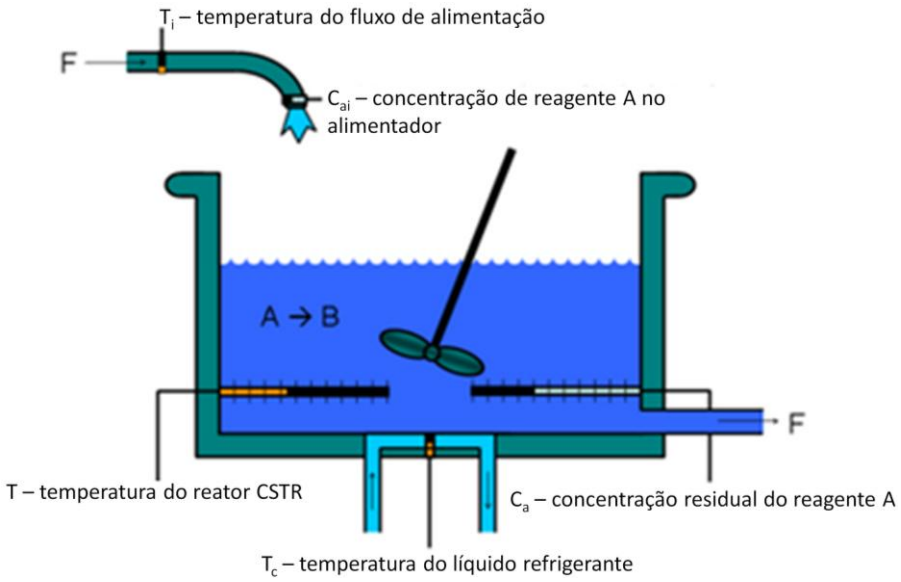
$$K_y = \begin{bmatrix} -41{,}1576 & 0 \\ 2750{,}02 & 0 \end{bmatrix} \quad (106)$$

Para as simulações desta planta não foram implementadas restrições ou a teoria de Laguerre.

2.2 REATOR CSTR

O CSTR (continuous stirred-tank reactor) é um reator de tanque não adiabático, abrigado comumente utilizado em processos industriais, como visto na Figura 9.

Figura 9 - Reator CSTR



Fonte: Adaptado de [29]

O modelo CSTR possui três entradas:

C_{ai} - concentração de reagente A no alimentador em kgmol/m^3;

T_i – temperatura do fluxo de alimentação;

T_c - temperatura do líquido refrigerante.

As duas saídas do modelo são:

T - temperatura do reator CSTR medida em K.;

C_a - concentração residual medida em kgmol/m^3, que é a concentração de reagente A no produto.

As equações diferenciais que representam o CSTR são vistas nas Equações 107 e 108 e representam uma reação exotérmica geradora de calor [30].

$$\frac{dC_a'}{dt} = a_{11}C_a' + a_{12}T' + b_{11}T_c' + b_{12}C_{ai}' \tag{107}$$

$$\frac{dT'}{dt} = a_{21}C_a' + a_{22}T' + b_{21}T_c' + b_{22}C_{ai}' \tag{108}$$

Uma das linearizações desta planta indica o espaço de estado visto nas Equações 109 e 110. Deve-se observar que as matrizes estão na sua forma contínua e para a aplicação do MPC é preciso discretizá-las.

$$\dot{x}_m = \overbrace{\begin{bmatrix} -0{,}0285 & -0{,}014 \\ -0{,}0371 & -0{,}1476 \end{bmatrix}}^{A_m} x + \overbrace{\begin{bmatrix} -0{,}0850 & 0{,}0238 \\ 0{,}0802 & 0{,}4462 \end{bmatrix}}^{B_m} u \tag{109}$$

$$y = \overbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}^{C_m} x \tag{110}$$

Neste trabalho o tempo de amostragem para discretizar esta planta foi de 1 segundo. As matrizes digitais desta operação são vistas nas Equações 111 e 112.

$$x_m(k+1) = \overbrace{\begin{bmatrix} 0{,}9719 & -0{,}0013 \\ -0{,}0340 & 0{,}8628 \end{bmatrix}}^{A_m} x(k) + \overbrace{\begin{bmatrix} -0{,}0839 & 0{,}0232 \\ 0{,}0761 & 0{,}4144 \end{bmatrix}}^{B_m} u(k) \tag{111}$$

$$y(k) = \overbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}^{C_m} x(k) \tag{112}$$

Para o projeto do MPC, as matrizes aumentadas são calculadas conforme descrito na seção 1 e podem ser vistas nas Equações 113 e 114.

$$x(k+1) = \overbrace{\begin{bmatrix} 0{,}9719 & -0{,}0013 & 0 & 0 \\ -0{,}0340 & 0{,}8628 & 0 & 0 \\ -0{,}0340 & 0{,}8628 & 1 & 0 \\ 0{,}9719 & -0{,}0013 & 0 & 1 \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}}^{x(k)} + \overbrace{\begin{bmatrix} -0{,}0839 & 0{,}0232 \\ 0{,}0761 & 0{,}4144 \\ 0{,}0761 & 0{,}4144 \\ -0{,}0839 & 0{,}0232 \end{bmatrix}}^{B} \Delta u(k) \tag{113}$$

$$y(k) = \overbrace{\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}^{C} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} \tag{114}$$

Em geral, a medição das concentrações de reagente é muito difícil e em alguns casos praticamente impossível. Porém, neste exemplo foi considerado que todas as saídas são medíveis e as grandezas C_{ai} e T_c como as entradas a serem controladas.

A toolbox do MPC no MATLAB utiliza por padrão a configuração de $N_p = 9$ e $N_c = 1$ com peso $r_w = 0.1$. Como o intuito é comparar a resposta da cosimulação com a da toolbox, as mesmas configurações foram implementadas no código VHDL com

exceção do peso r_w que foi configurado para ser igual a 0 (zero). As matrizes F e φ do CSTR para a confguração apresentada são vistas nas Equações 115 e 116.

$$F = \begin{bmatrix} -0{,}0340 & 0{,}8628 & 1 & 0 \\ 0{,}9719 & -0{,}0013 & 0 & 1 \\ -0{,}0964 & 1{,}6073 & 1 & 0 \\ 1{,}9166 & -0{,}0036 & 0 & 1 \\ -0{,}1823 & 2{,}2497 & 1 & 0 \\ 2{,}8349 & -0{,}0069 & 0 & 1 \\ -0{,}2876 & 2{,}8040 & 1 & 0 \\ 3{,}7274 & -0{,}0109 & 0 & 1 \\ -0{,}4089 & 3{,}2825 & 1 & 0 \\ 4{,}5951 & -0{,}0154 & 0 & 1 \\ -0{,}5430 & 3{,}6955 & 1 & 0 \\ 5{,}4385 & -0{,}0205 & 0 & 1 \\ -0{,}6873 & 4{,}0519 & 1 & 0 \\ 6{,}2585 & -0{,}0259 & 0 & 1 \\ -0{,}8398 & 4{,}3597 & 1 & 0 \\ 7{,}0556 & -0{,}0317 & 0 & 1 \\ -0{,}9984 & 4{,}6254 & 1 & 0 \\ 7{,}8305 & -0{,}0377 & 0 & 1 \end{bmatrix} \quad (115)$$

$$\varphi = \begin{bmatrix} 0{,}0761 & 0{,}4144 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0{,}0839 & 0{,}0232 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0{,}1445 & 0{,}7712 & 0{,}0761 & 0{,}4144 & 0 & 0 & 0 & 0 \\ -0{,}1655 & 0{,}0452 & -0{,}0839 & 0{,}0232 & 0 & 0 & 0 & 0 \\ 0{,}2064 & 1{,}0783 & 0{,}1445 & 0{,}7712 & 0{,}0761 & 0{,}4144 & 0 & 0 \\ -0{,}2448 & 0{,}0661 & -0{,}1655 & 0{,}0452 & -0{,}0839 & 0{,}0232 & 0 & 0 \\ 0{,}2624 & 1{,}3425 & 0{,}2064 & 1{,}0783 & 0{,}1445 & 0{,}7712 & 0{,}0761 & 0{,}414 \\ -0{,}3221 & 0{,}0860 & -0{,}2448 & 0{,}0661 & -0{,}1655 & -0{,}1655 & -0{,}0839 & 0{,}023 \\ 0{,}3134 & 1{,}5698 & 0{,}2624 & 1{,}3425 & 0{,}2064 & 1{,}0783 & 0{,}1445 & 0{,}771 \\ -0{,}3972 & 0{,}1050 & -0{,}3221 & 0{,}0860 & -0{,}2448 & 0{,}0661 & -0{,}1655 & -0{,}16 \\ 0{,}3600 & 1{,}7653 & 0{,}3134 & 1{,}5698 & 0{,}2624 & 1{,}3425 & 0{,}2064 & 1{,}078 \\ -0{,}4703 & 0{,}1232 & -0{,}3972 & 0{,}1050 & -0{,}3221 & 0{,}0860 & -0{,}2448 & 0{,}066 \\ 0{,}4026 & 1{,}9333 & 0{,}3600 & 1{,}7653 & 0{,}3134 & 1{,}5698 & 0{,}2624 & 1{,}342 \\ -0{,}5415 & 0{,}1407 & -0{,}4703 & 0{,}1232 & -0{,}3972 & 0{,}1050 & -0{,}3221 & 0{,}086 \\ 0{,}4418 & 2{,}0777 & 0{,}4026 & 1{,}9333 & 0{,}3600 & 1{,}7653 & 0{,}3134 & 1{,}569 \\ -0{,}6106 & 0{,}1574 & -0{,}5415 & 0{,}1407 & -0{,}4703 & 0{,}1232 & -0{,}3972 & 0{,}105 \\ 0{,}4780 & 2{,}2017 & 0{,}4418 & 2{,}0777 & 0{,}4026 & 1{,}9333 & 0{,}3600 & 1{,}765 \\ -0{,}6779 & 0{,}1735 & -0{,}6106 & 0{,}1574 & -0{,}5415 & 0{,}1407 & -0{,}4703 & 0{,}123 \end{bmatrix} \quad (116)$$

Os ganhos K_{mpc} e K_y calculados são vistos nas Equações 117 e 118.

$$K_{mpc} = \begin{bmatrix} -11{,}0529 & 0{,}5621 & 0{,}6346 & -11{,}3500 \\ 1{,}9463 & 1{,}9788 & 2{,}2966 & 2{,}0828 \end{bmatrix} \quad (117)$$

$$K_y = \begin{bmatrix} 0{,}6346 & -11{,}3500 \\ 2{,}2966 & 2{,}0828 \end{bmatrix} \quad (118)$$

Para as simulações desta planta não foram implementadas restrições ou a teroria de Laguerre. Apesar de as dificuldades desta planta serem bem semelhantes as do controle da máquina síncrona, os problemas encontrados foram importantes para garantir melhor eficiência no código VHDL do MPC.

2.3 REATOR PAR

O maior desafio da aplicação do MPC neste trabalho é o reator nuclear que envolve um Protótipo de Reator Avançado (Prototypical Advanced Reactor – PAR), cuja modelagem e implementação no MATLAB/Simulink foi feito pelo Departamento de Engenharia Nuclear da Universidade do Tennessee (DNE-UT).

Segundo [31], O DNE-UT é uma referência mundial na área Nuclear, sendo detentora de diversas publicações, patentes e formação de recursos humanos. Além disso, o DNE-UT é uma das gestoras do importante projeto Norte americano TerraPower, financiado pelo bilionário Bill Gates, cujo objetivo visa a construção de novos modelos de Reatores Nucleares para operação segura e de baixo custo buscando maior aplicabilidade destes modelos em diversos segmentos.

A modelagem do reator PAR no ambiente Simulink teve como foco, produzir um modelo flexível para produção, manutenção, segurança operacional e com capacidade de produção de energia, sendo essencial para simular, testar e desenvolver novas ideias de projeto. Na Figura 10 pode observar o esquema do Protótipo de Reator Avançado utilizado neste trabalho.

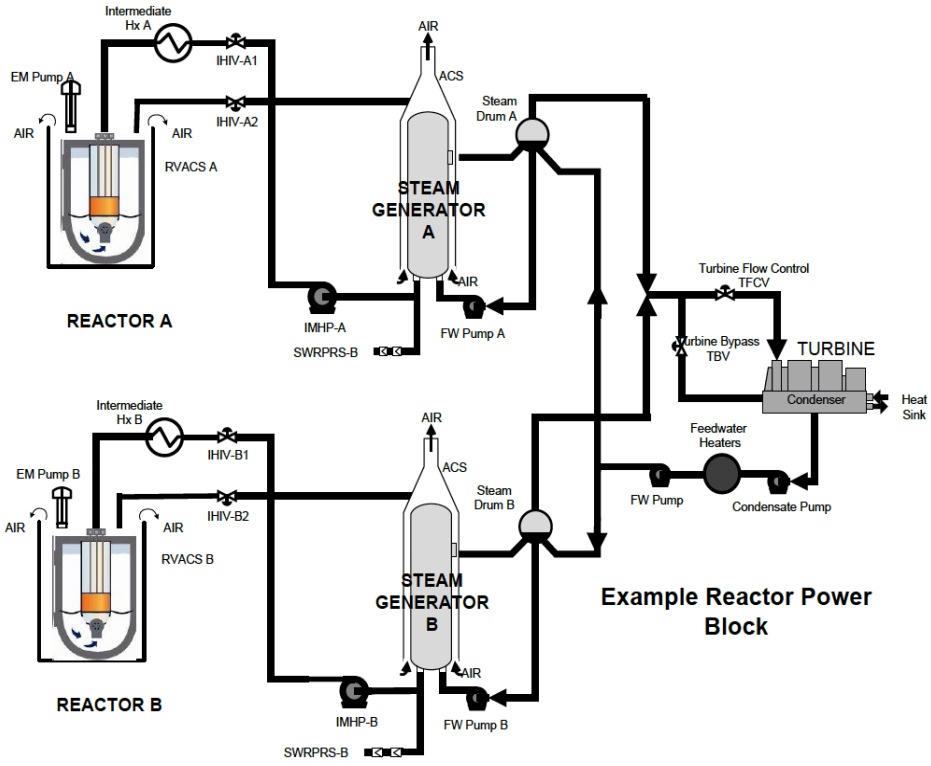
Basicamente, trata-se de uma planta baseada no EBR-II (*Experimental Breeder Reactor-II*), mas com operação multimodular. Segundo [32], o PAR possui dois núcleos de reatores, cada um conectado a um trocador de calor intermediário (IHX - *Intermediate Heat Exchanger*) dedicado e um gerador de vapor. A saída desses dois geradores de vapor é então conectada a um comum sistema de balanceamento (BOP - *Balance of Plant*). O BOP inclui tambores de vapor, turbina, condensador, bombas de água de alimentação e aquecedores de água de alimentação. A operação multimodular para produção de energia elétrica torna o sistema mais flexível para manutenção e com menos vulnerabilidade a falta de produção de energia.

Portanto, as simulações e os estudos de controle foram feitos com um modelo linear que inclui o reator, o trocador de calor intermediário (IHX) e o gerador

de vapor. Entretanto, neste trabalho apenas o modelo linearizado do reator com 100% de potência nominal é utilizado para o projeto do MPC, sendo utilizado para os outros componentes, os demais modelos descritos em [33].

Em consonância com os objetivos descritos em [33], a idéia principal não é propor uma nova estratégia de controle para o EBR-II, mas estudar as capacidades dos controladores modernos quando substituem os controladores existentes no EBR-II para realizar tarefas simples.

Figura 10 - Protótipo de reator avançado



Fonte: [32]

Conforme relatado em [33], o EBR-II é um reator de resfriamento rápido de metal líquido projetado como uma instalação para teste de engenharia, e fica localizado na Estação Nacional de Teste de Reator em Idaho. Os testes e as demonstrações do EBR-II têm um papel muito importante. O futuro do desenvolvimento de tecnologia para reatores avançados de metal líquido demanda várias atividades de pesquisa relacionadas com as dinâmicas do sistema e o controle.

Segundo [32], o EBR-II possui 62,5 MWth com saída de 20 MWe. Então, a operação multimodular com dois reatores fornece 40 MWe de energia elétrica na saída. Os principais componentes identificados na Figura 10 que exigem modelos físicos incluem: núcleo do reator, IHX, gerador de vapor e BOP. A planta do EBR-II também inclui um sistema de processamento de combustível que possui sistemas para desmontagem, descontaminação, fabricação e montagem de elementos combustíveis e subconjuntos.

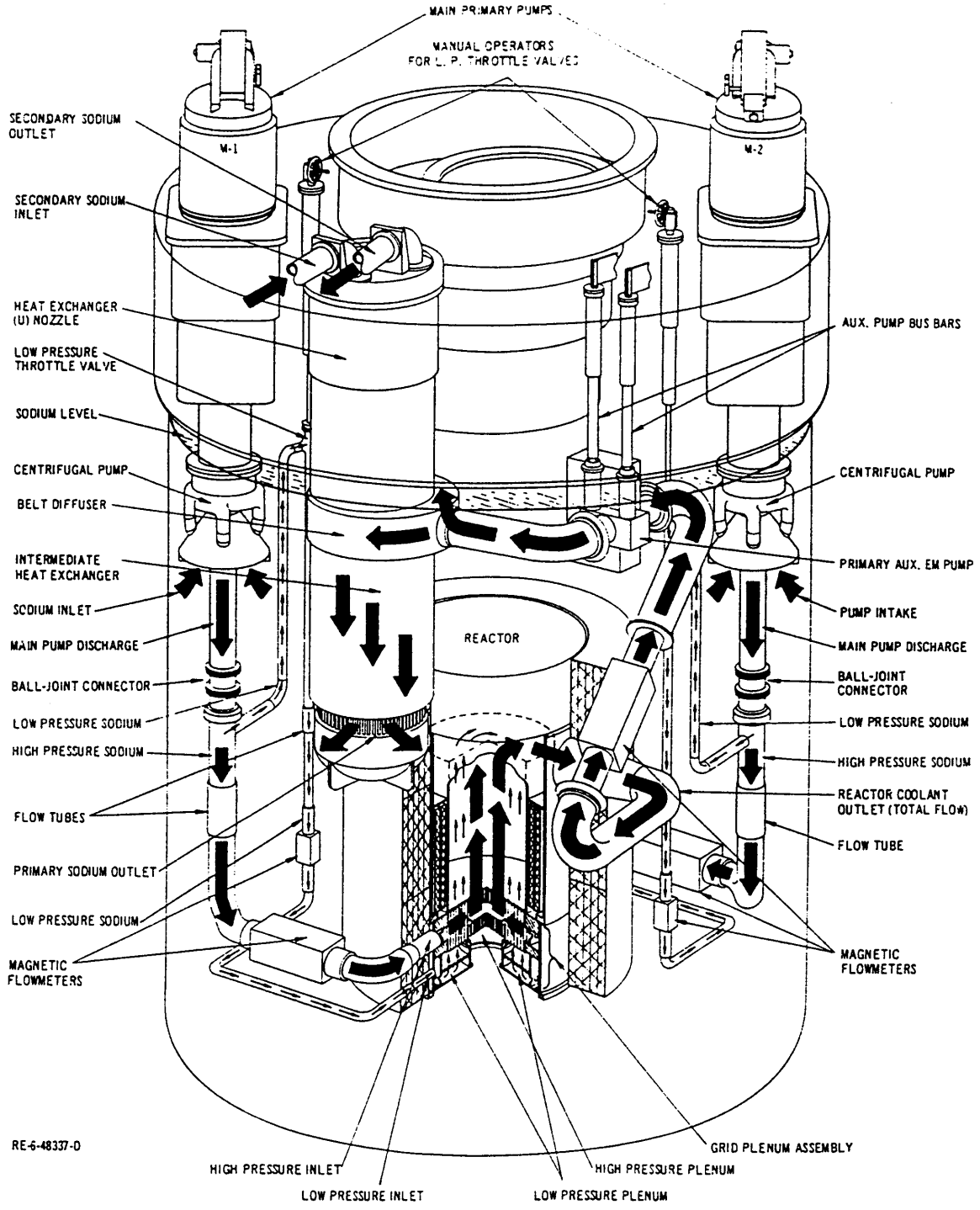
Tanto o reator quanto as instalações de reciclagem de combustível são submersas em um grande tanque primário de sódio e este conceito é referido como projeto tipo piscina (como os reatores de metal líquido Phoenix e Super Phoenix na França). Atualmente, o propósito do EBR-II foi redirecionado para fornecer serviços de irradiação para o desenvolvimento de combustíveis e materiais estruturais.

Uma breve explicação do funcionamento do EBR-II é: o calor gerado pela fissão nuclear no núcleo é absorvido pelo sódio líquido que circula através do reator. O calor absorvido é então transferido por meio de uma tubulação tipo concha para a troca de calor intermediária que repassa o calor para o sódio secundário não radioativo. O sódio secundário transfere o calor para o sistema de vapor que está no gerador de vapor. O resultado é um vapor superaquecido que movimenta uma turbina-gerador para produção de eletricidade e é condensada em água no condensador. O calor transferido para o condensador é dissipado para a atmosfera por torres de resfriamento. Na Figura 11 é visto o esquemático do sistema de resfriamento primário.

O sistema secundário de sódio é um circuito fechado intermediário entre o sistema primário e o sistema de vapor, no qual absorve o calor produzido no primário e transfere para o sistema de vapor. O fluxo deste sódio secundário é regulado manualmente para absorver todo o calor em excesso e manter a temperatura do volume de sódio primário constante na temperatura de 700 F.

A transferência de calor entre o sódio primário e o secundário ocorre no trocador de calor intermediário que é localizado submerso no volume de sódio do tanque primário. O calor do sódio secundário flui através da tubulação para a caldeira de sódio do gerador de vapor. A transferência de calor do sódio secundário para o sistema de vapor é efetuada nesta interface e produz o vapor superaquecido que é utilizado para movimentar a turbina gerador.

Figura 11 - Sistema de resfriamento primário do EBR-II



Fonte: [33]

Na Tabela 2 mostram alguns dados operacionais do EBR-II usados para a linearização do reator.

Tabela 2 – Configuração do reator com 100% da potência nominal

100% da potência nominal	**Dados**
Saída de calor, Mw	62,5
Temperatura de sódio primário, F (p/ reator)	700
Temperatura de sódio primário, F (do reator)	883
Fluxo de sódio, através do reator, gpm	9000
Fluxo de sódio secundário, gpm	5890

Fonte: [33]

O sistema primário da planta do reator contém o núcleo do reator, sistema de resfriamento primário e de desligamento, escudo de neutrôns, instrumentação, sistemas de acionamento de controle e segurança, sistema de manuseio de combustível, entre outros sistemas.

O reator contém um núcleo central de material fissionável que está completamente cercado por *blankets* radiais e axiais. O combustível e o material dos blankets estão contidos no recipiente do reator em formato cilíndrico. Um escudo de nêutrons também cerca o reator nas direções axiais e radiais. O núcleo do reator possui 53 subconjuntos de combustível assim como 12 subconjuntos de hastes de controle e 2 subconjuntos de hastes de segurança, sendo estes subconjuntos dispostos em formato hexagonal. As 2 hastes de segurança têm função de interromper a inserção de reatividade no reator e são movidos pela parte baixo do núcleo.

Atualmente, um sistema de acionamento da haste controlado por computador controla a energia do reator durante o estado estacionário e também altera a potência do reator. Por este motivo, nas simulações desta planta a variável de controle é a inserção de reatividade e a saída a ser medida da planta é a potência nominal.

A modelagem da planta do EBR-II é composta de três subsistemas: o sistema primário, o sistema gerador de vapor e o sistema turbina-condensador. O sistema primário inclui o núcleo do reator, o tanque de sódio e o trocador de calor intermediário (IHX).

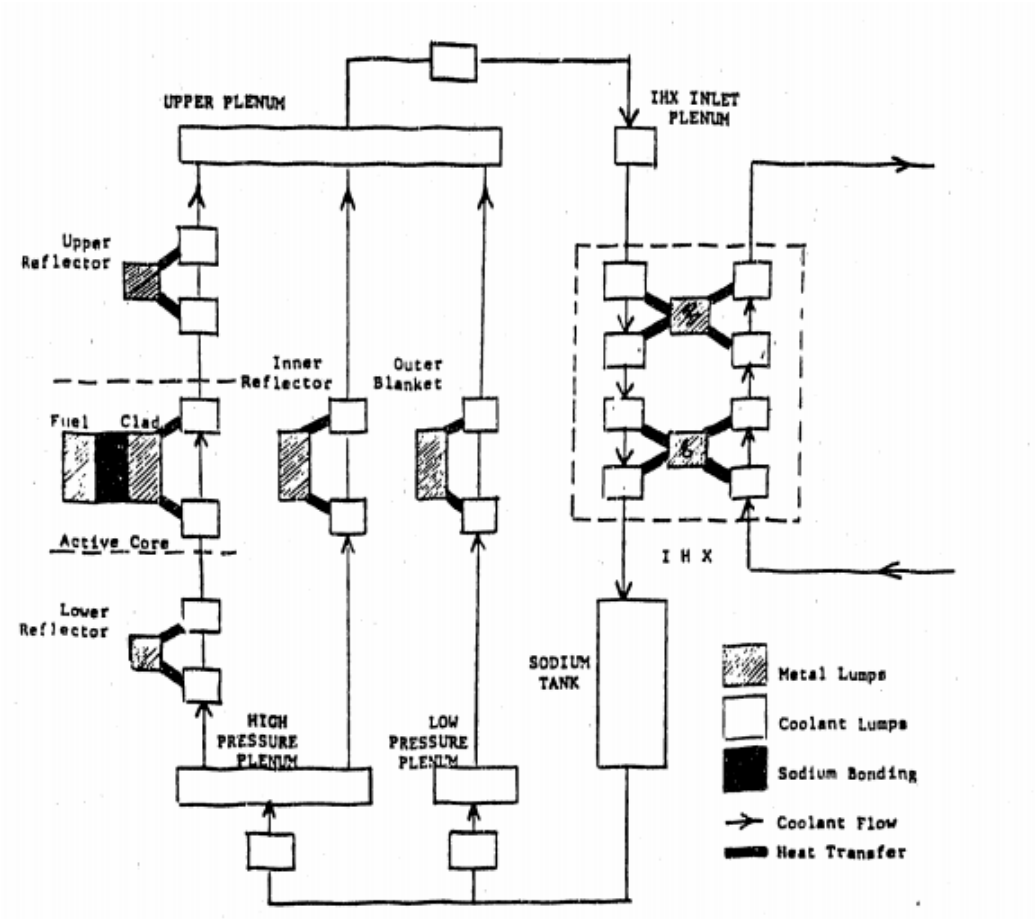
A representação nodal do sistema primário, incluindo o núcleo, refletores, plenum e IHX, é vista na Figura 12. De acordo com [32], o núcleo é um modelo de 25 nódulos, que inclui o núcleo ativo, *blankets* internos e externos, refletores inferior

e superior e tubulações. As suposições para a região da tubulação incluem: densidade constante do líquido refrigerante, nenhuma condução de calor axial e nenhum ganho ou perda de calor na tubulação.

Para a transferência de calor do núcleo, na qual o modelo de Mann foi utilizado, presume-se que a temperatura da saída do fluido refrigerante inferior represente a temperatura média global do sistema porque este é um parâmetro de acoplamento para a força motriz de transferência de calor entre o metal e a região do refrigerante.

Outras suposições consideradas são a negligência, no modelo do EBR-II, dos efeitos da reatividade na expansão das hastes de controle e curvatura do núcleo.

Figura 12 - Representação nodal do sistema primário do EBR-II



Fonte: [33]

O trocador de calor intermediário foi modelado usando doze variáveis de estado: dez nós no IHX, indicados na caixa tracejada da Figura 12; o plenum de

entrada IHX no lado primário; e o tanque de sódio na saída lateral primária. O núcleo e o IHX são acoplados em um único modelo. As Equações 119, 120, 121, 122, 123 e 124 descrevem cada subsistema, sendo que os valores para os parâmetros constantes são vistos em [33].

$$\dot{P}_c = \frac{-\beta_T}{\Lambda} P_c + \frac{\rho P_c}{\Lambda} + \bar{\lambda} C \tag{119}$$

$$\rho = \rho_{external} + \rho_{feedback} \tag{120}$$

$$\rho_{feedback} = \sum_i \alpha_i (T_i - T_{io}) \tag{121}$$

$$\dot{C}_c = \frac{\beta_T}{\Lambda} P_c - \bar{\lambda} C \tag{122}$$

$$C = C_c + C_o \tag{123}$$

$$C_o = \frac{\beta_T P_{co}}{\Lambda \bar{\lambda}} \tag{124}$$

Sendo:

$\dot{P}_c$= Fração de Potência no Núcleo

β_T = Fração total do atraso dos Neutron

Λ = Tempo de geração do Neutron

ρ = Reatividade

$\bar{\lambda}$ = Constante Média de Atraso

C = Concentração

α_i = Temperatura da Reatividade T_i

T_i = Temperatura Atual

T_{io} = Temperatura Estável para 100% da Potência.

A transferência de calor no núcleo ativo é modelada utilizando 5 equações diferenciais, que correspondem aos 5 nódulos vistos na Figura 13. Estes nódulos representam as regiões do combustível, sódio, do revestimento, da entrada e da saída do líquido refrigerante. A dinâmica de transferência de calor entre as regiões de revestimento e refrigerante é representada usando o modelo de Mann. As

Equações 125, 126 e 127 são a do modelo de Mann e as temperaturas representadas nas Equações 128 e 129 são da perna fria e perna quente, respectivamente.

$$\dot{T}_F = \frac{P_f P_o}{(MC_p)_F} P_c - \frac{1}{R_1 (MC_p)_F} \quad (T_F - T_B) \tag{125}$$

$$\dot{T}_B = \frac{1}{R_1 (MC_p)_B} (T_F - T_B) - \frac{1}{R_2 (MC_p)_B} \quad (T_B - T_C) \tag{126}$$

$$\dot{T}_C = \frac{1}{R_2 (MC_p)_B} (T_B - T_C) - \frac{1}{R_3 (MC_p)_\theta} \quad (T_B - \theta_1) \tag{127}$$

$$\dot{\theta}_1 = \frac{1}{R_2 (MC_p)_\theta} (T_C - \theta_1) + \frac{2}{\tau} (\gamma_2 - \theta_1) \tag{128}$$

$$\dot{\theta}_2 = \frac{1}{R_3 (MC_p)_\theta} (T_C - \theta_1) + \frac{2}{\tau} (\theta_1 - \theta_2) \tag{129}$$

Sendo:

T_F = Temperatura do Combustível.

T_B = Temperatura do Sódio.

T_C = Temperatura de Resfriamento do Combustível.

θ_i = Temperatura relacionada com a refrigeração.

R_1, R_2, R_3 = resistência de transferência de Calor,

γ_2 = temperatura do refletor de saída do refrigerante

τ = Tempo de refrigeração do núcleo

P_F = Fração da Potência depositada no combustível

$(C_p)_F$ = Capacidade específica do combustível.

$(C_p)_B$ = Capacidade específica do material de revestimento

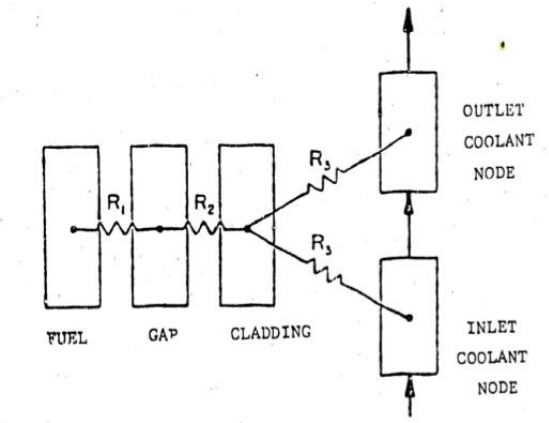
$(C_p)_\theta$ = Capacidade térmica específica do refrigerante.

M_F = Massa do combustível

M_B = Massa do material do revestimento

M_θ = massa do refrigerante

Figura 13 - Modelo de Mann para transferência de calor do núcleo



Fonte: [32]

Os refletores e os *blankets* cercam o núcleo ativo no sistema primário do EBR-II. O modelo completo do núcleo inclui 12 nódulos adicionais representando as zonas dos refletores axiais e radias e a região radial do *blanket*. O mesmo princípio de transferência de calor é realizado no desenvolvimento das equações de estado, como no modelo de transferência de calor do núcleo. As equações gerais para regiões refletoras e dos *blankets* são descritas por um conjunto de três equações para cada uma, como mostrado nas equações 130, 131 e 132.

$$\dot{T}_M = \frac{P_i}{(MC_p)_M} P_c - \frac{U * A}{(MC_p)_M} (T_M - T_1) \tag{130}$$

$$\dot{T}_1 = \frac{U * A}{(MC_p)_T} (\delta T_M - \delta T_1) + \frac{2}{\tau} (\theta_{in} - T_1) \tag{131}$$

$$\dot{T}_2 = \frac{U * A}{(MC_p)_T} (\delta T_M - \delta T_1) + \frac{2}{\tau} (T_1 - T_2) \tag{132}$$

Sendo:

T_M = Temperatura do Metal

T_1 = Temperatura da Primeira Região do Refrigerante.

T_2 = Temperatura da Segunda Região do Refrigerante.

A = Área Total de Transferência de Calor

τ = Tempo de resistividade do refrigerante na região do reflector ou do revestimento.

U = Coeficiente metálico para transferência de calor

θ_{in} = Temperatura do líquido refrigerante de entrada.

$(C_p)_M$ = Capacidade térmica específica do metal.

$(C_p)_T$ = Capacidade Térmica específica do refrigerante.

O modelo inclui seis partes que representam o *plenum* de baixa pressão, o *plenum* de alta pressão, o *plenum* superior e a região da tubulação de entrada e saída do núcleo. Um atraso de transferência de primeira ordem foi considerado para todas as tubulações. As outras premissas são: densidade constante de refrigeração, ausência de condução de calor axial e ausência de ganho ou perda de calor na tubulação. Nas Equações 133, 134, 135, 136, 137 e 138 descrevem-se o modelo do *plenum* e da tubulação.

$$\dot{T_U} = \frac{M_1 C_{p_2}}{(MC_p)_u}\gamma_4 + \frac{M_2 C_{p_2}}{(MC_p)_u}\gamma_6 + \frac{M_3 C_{p_3}}{(MC_p)_u}\gamma_8 - \left[\frac{M_1 C_{p_2}}{(MC_p)_u} + \frac{M_2 C_{p_2}}{(MC_p)_u} + \frac{M_3 C_{p_3}}{(MC_p)_u}\right] T_U \quad (133)$$

$$\dot{T_{out}} = \frac{1}{\tau_1} T_U - \frac{1}{\tau_1} T_{out} \quad (134)$$

$$\dot{T_{LI}} = \frac{1}{\tau_3}\theta_p - \frac{1}{\tau_3} T_{LI} \quad (135)$$

$$\dot{T_{HI}} = \frac{1}{\tau_4}\theta_p - \frac{1}{\tau_4} T_{HI} \quad (136)$$

$$\dot{T_H} = \frac{1}{\tau_5} T_{HI} - \frac{1}{\tau_5} T_H \quad (137)$$

$$\dot{T_L} = \frac{1}{\tau_6} T_{LI} - \frac{1}{\tau_6} T_L \quad (138)$$

Sendo:

T_U = Temperatura Superior do Plenum

T_{out} = Temperatura de Saída do Reator

θ_p = Temperatura do Tank de Sódio no primário

T_{LI} = Temperatura de Entrada no Plenum em baixa pressão.

T_{HI} = Temperatura de Entrada no Plenum em alta pressão.

T_H = Temperatura no Plenum em alta pressão.

γ_4 = Temperatura de saída da parte superior do reflector.

γ_6 = Temperatura interna da saída do reflector.

γ_8 = Temperatura de saída da região de revestimento.

T_L = Temperatura no Plenum em baixa pressão.

τ_1 = tempo de resistência do Sódio na saída da tubulação do reator.

τ_2 = tempo de resistência do Sódio no recipiente

τ_3 = tempo de resistência do Sódio do recipiente para a tubulação do reator em baixa pressão.

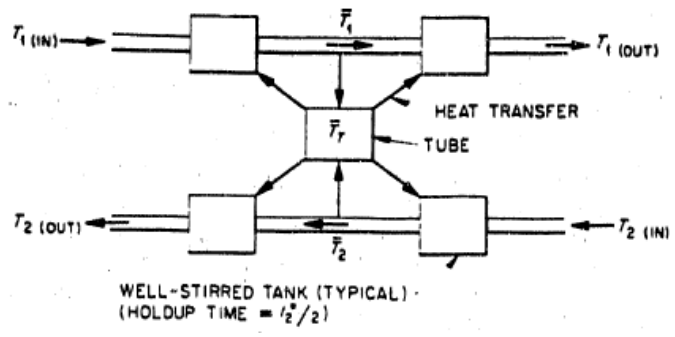
τ_4 = tempo de resistência do Sódio do recipiente para a tubulação do reator em alta pressão.

τ_5 = tempo de resistência do Sódio no Plenum em alta pressão.

τ_6 = tempo de resistência do Sódio no Plenum em baixa pressão.

O trocador de calor intermediário (IHX) é modelado por uma aproximação de parâmetros agrupados de 10 nódulos de um trocador de calor de contrafluxo; uma metade do IHX é mostrada na Figura 14. O *plenum* de entrada primário e o tanque de sódio são representados por aproximações de atraso de transporte de primeira ordem. A transferência de calor do sódio do primário para o secundário é modelada usando a técnica de Mann. Os nódulos primários e secundários são numerados na direção do fluxo. Nas equações 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148 e 149 representam-se o modelo do IHX.

Figura 14 - Diagrama de transferência de calor do IHX



Fonte: [32]

$$\dot{P}_1 = \frac{2}{\tau_{HXP}} T_{HP} - \left(\frac{(UA)_P}{(MC_p)_P} + \frac{2}{\tau_{HXP}} \right) P_1 + \frac{(UA)_P}{(MC_p)_P} M_1 \quad (138)$$

$$\dot{P}_2 = \left(\frac{2}{\tau_{HXP}} - \frac{(UA)_P}{(MC_p)_P} \right) P_1 - \frac{2}{\tau_{HXP}} P_2 + \frac{(UA)_P}{(MC_p)_P} M_1 \quad (139)$$

$$\dot{M}_1 = \frac{(UA)_P}{(MC_p)_M} P_1 - \left(\frac{(UA)_P + (UA)_S}{(MC_p)_M} \right) M_1 + \frac{(UA)_S}{(MC_p)_M} S_3 \quad (140)$$

$$\dot{S}_4 = \frac{(UA)_S}{(MC_p)_S} M_1 - \left(\frac{(UA)_S}{(MC_p)_S} - \frac{2}{\tau_{HXS}} \right) S_3 - \frac{2}{\tau_{HXS}} S_4 \quad (141)$$

$$\dot{S}_3 = \frac{(UA)_S}{(MC_p)_S} M_1 - \left(\frac{(UA)_S}{(MC_p)_S} + \frac{2}{\tau_{HXS}} \right) S_3 + \frac{2}{\tau_{HXS}} S_2 \quad (142)$$

$$\dot{P}_3 = \frac{2}{\tau_{HXP}} P_2 - \left(\frac{(UA)_P}{(MC_p)_P} + \frac{2}{\tau_{XHP}} \right) P_3 + \frac{(UA)_P}{(MC_p)_P} M_2 \quad (143)$$

$$\dot{P}_4 = \left(\frac{2}{\tau_{HXP}} - \frac{(UA)_P}{(MC_p)_P} \right) P_3 - \frac{2}{\tau_{XHP}} P_4 + \frac{(UA)_P}{(MC_p)_P} M_2 \quad (144)$$

$$\dot{M}_2 = \frac{(UA)_P}{(MC_p)_M} P_3 - \left(\frac{(UA)_P + (UA)_S}{(MC_p)_M} \right) M_2 + \frac{(UA)_S}{(MC_p)_M} S_1 \quad (145)$$

$$\dot{S}_2 = \frac{(UA)_S}{(MC_p)_S} M_2 - \left(\frac{(UA)_S}{(MC_p)_S} - \frac{2}{\tau_{XHS}} \right) S_1 - \frac{2}{\tau_{XHS}} S_2 \quad (146)$$

$$\dot{S}_1 = \frac{(UA)_S}{(MC_p)_S} M_2 - \left(\frac{(UA)_S}{(MC_p)_S} + \frac{2}{\tau_{HXS}} \right) S_1 + \frac{2}{\tau_{HXS}} S_{in} \quad (147)$$

$$\dot{P_{in}} = \frac{1}{\tau_7} T_{out} - \frac{1}{\tau_7} P_{in} \quad (148)$$

$$\dot{\theta}_P = \frac{1}{\tau_2} P_4 - \frac{1}{\tau_2} \theta_P \quad (149)$$

Sendo:

P_1 = Primeira Temperatura do nó primário

P_2 = Segunda Temperatura do nó secundário.

M_1 = Primeira temperatura na parede da tubulação (superior)

S_4 = Quarta temperatura do nó secundário.

S_3 = Terceira Temperatura do nó secundário.

P_3 = Terceira temperatura do nó primário.

P_4 = Quarta temperatura do nó primário.

M_2 = segunda temperatura na parede da tubulação (inferior).

S_2 = Segunda temperatura do secundário.

S_1 = Primeira temperatura do nó secundário.

P_{in} = Temperatura de entrada no primário do Plenum.

T_{out} = Temperatura de saída do Reator.

S_{in} = Temperatura de entrada do sódio no secundário.

θ_p = Temperatura do tanque de sódio.

τ_{HXP} = Tempo de resistência do nó primário

τ_{HXS} = tempo de resistência do nó secundário.

τ_7 = tempo de resistência na saída do primário do Plenum.

Com a modelagem do reator e do IHX, o próximo passo é avaliar as técnicas de controle. A viabilidade do emprego de modernas técnicas de controle em reatores avançados tem sido estudada nos últimos anos, mesmo que ainda sem muita aplicação, pois os controles convencionais são eficientes e confiáveis em controlar sistemas nucleares. Outra razão pelo uso de técnicas convencionais está na limitação de ferramentas de hardware e software, pois o mais simples reator nuclear possui um grande número de variáveis de estado.

No entanto, conforme [33] em plantas nucleares a recuperação de transitórios não esperados requer manobras operacionais seguras. Portanto, o desenvolvimento de novas estratégias de controle pode resolver estes problemas, já que além de lidar com as consequências de falhas e ações de controle que não sejam ótimas, estratégias melhores podem evitar esforço redundante de controle e também melhorar o desempenho do sistema na operação diária.

Conforme mencionado, toda a modelagem e desenvolvimento do PAR no ambiente Simulink foi desenvolvido pelo DNE-UT e neste trabalho apenas são realizados testes de controle utilizando a abordagem do modelo preditivo.

As especificações técnicas do reator indicam duas restrições sendo que a primeira está relacionada com a inserção de reatividade que não pode ultrapassar o

limite de 0,01 $/s (relacionado com segurança). A segunda restrição está realcionada a diferença de temperatura da saída (*outlet*) e da entrada (*inlet*) do reator que não pode ser maior do que 2 F. No projeto do MPC apenas é inserida a primeira restrição, como mostra as Equações 150 e 151.

$$-\Delta U \leq -0{,}01 \tag{150}$$

$$\Delta U \leq 0{,}01 \tag{151}$$

Além disso também foi incluído uma restrição na qual a saída do reator não poderia ser maior do que 1 pu, como mostra a Equação 152.

$$Fx(k_i) + \varphi \Delta U \leq 1 \tag{152}$$

Logo, para a configuração de N_p = 20 e N_c = 5, as matrizes M e γ que implementam as restrições são mostradas na Equação 153, sendo 1_{5x1} um vetor de números 1 com cinco linhas e uma coluna, I_{5x5} a matriz identidade com dimensão igual a cinco.

$$\overbrace{\begin{bmatrix} I_{5x5} \\ -I_{5x5} \\ \varphi \end{bmatrix}}^{M} \Delta U \leq \overbrace{\begin{bmatrix} 0{,}01 * 1_{5x1} \\ 0{,}01 * 1_{5x1} \\ 1 - F * x(k_i) \end{bmatrix}}^{\gamma} \tag{153}$$

Os ganhos K_{mpc} e K_y, calculados para um tempo de amostragem de 1 segundo, são vistos nas Equações 154 e 155. Não foram implementadas a teoria de Laguerre para esta planta.

$$K_{mpc} = \begin{bmatrix} 0{,}000112487447480132 \\ -0{,}000131611599304771 \\ -0{,}0167685829893894 \\ -0{,}00175443624878168 \\ -0{,}000806961942113647 \\ -2{,}76832120594111e^{-6} \\ 1{,}90849998565976e^{-7} \\ 3{,}15176627599685e^{-7} \\ -0{,}000335150997041954 \\ -3{,}16636351672682e^{-6} \\ -2{,}86158804218572e^{-7} \\ 3{,}32477956174549e^{-7} \\ -0{,}000738618616909888 \\ 7{,}42598089532817e^{-7} \\ 9{,}07687668722300e^{-7} \\ 7{,}65932123290953e^{-5} \\ 4{,}82883218474109e^{-5} \\ 7{,}63942851848427e^{-5} \\ 9{,}81160768383770e^{-5} \\ 0{,}000186446647252015 \\ 9{,}75435010175911e^{-5} \\ 0 \\ 1{,}33508724411314e^{-5} \\ -0{,}00254599508577674 \\ 0{,}000225461280980730 \\ 0{,}000357709736152062 \\ 0{,}000248267577114179 \\ 1{,}20387782042375e^{-5} \\ 3{,}62753000153510e^{-5} \\ 0{,}105427517008775 \\ 4{,}72886414197368e^{-5} \\ -0{,}000496284488941755 \\ -1{,}97375599973165e^{-5} \\ 2{,}32856088941656e^{-7} \\ 3{,}48077891315654e^{-6} \\ -0{,}0374253032859981 \\ -0{,}0888358155249860 \\ 122{,}230363609565 \end{bmatrix} \tag{154}$$

$$K_y = 122{,}230363609565 \tag{155}$$

3 RESULTADOS

Nesta seção são detalhadas as simulações dos controles preditivos projetados para controlar as plantas e processos mencionados nos capítulos 1 e 2.

As simulações destas plantas são feitas primeiro no MATLAB, por um *script* baseado no tutorial do livro em [1] ou no ambiente Simulink com o bloco MPC da *toolbox*. Em seguida, tenta-se replicar estes resultados na cosimulação onde o código VHDL é simulado no programa ModelSim PE da Mentor Graphics, sendo que nestas simulações as variáveis do código para implementar o MPC são todas do tipo real.

Os ensaios de *FPGA-in-the-loop* do MPC com e sem observador são feitos com as variáveis declaradas como ponto fixo, utilizando o padrão da biblioteca da Altera de 1 bit para o sinal, 15 bits para a parte inteira e 16 bits para a parte fracionária, sendo a representação em complemento a 2, conforme visto no capítulo 2. Nestes experimentos o MPC é embarcado fisicamente na plataforma FPGA DE2-115 e o processo a ser controlado é simulado no ambiente Simulink.

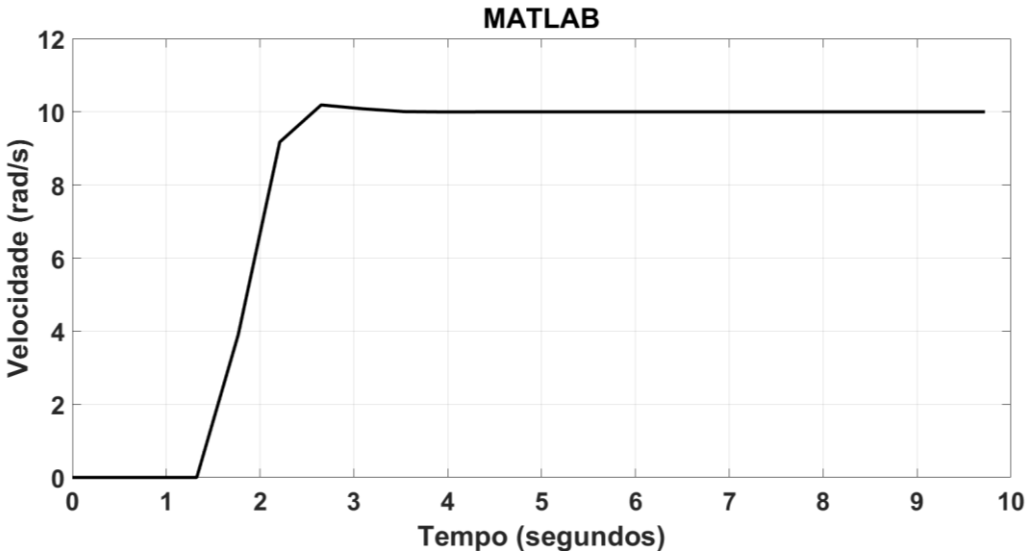
A ordem de apresentação dos resultados segue a ordem vista neste texto. Começa-se pelo sistema turbo-gerador, pois é a planta mais fácil de ser controlada e, por isso, com ela foi desenvolvida todas as teorias do MPC apresentadas neste trabalho. Com os erros e dificuldades aprendidos, foi projetado o MPC para o controle da máquina síncrona onde o código foi habilitado para dar suporte a sistemas de múltiplas entradas e múlitplas saídas (MIMO). A seguir apresentam-se os resultados e análises para o reator CSTR. Por fim, os resultados do projeto MPC para a planta que representa o maior desafio deste trabalho que é o protótipo de reator avançado (PAR).

3.1 SISTEMA TURBINA GERADOR

O sistema turbina gerador, descrito junto com a teoria no capítulo 1, é o primeiro processo a ser simulado. A primeira simulação foi realizada por um *script* feito com base no tutorial de [1] . O código desta simulação se encontra disponível no anexo ao final do trabalho.

Para a simulação inicial foi considerado valores mínimos para os parâmetros do MPC, sendo N_p = 2 e N_c = 1, com o peso r_w = 0. Na Figura 15 observa-se o resultado da saída do processo.

Figura 15 - Simulação do *script* para Np = 2 e Nc = 1



Fonte: Do autor

Para esta configuração inicial o *overshoot* foi de 1,88% e o tempo de assentamento foi de pouco mais de 1 segundo. Conforme visto no capítulo 1, o MPC projetado proporcionou uma melhora no tempo de estabilização, sendo que para a malha aberta este tempo era de 4,42 segundos.

A simulação neste e em todos os casos começa em 1 segundo porque o controle digital teve a porta *reset* desabilitada neste intervalo de tempo.

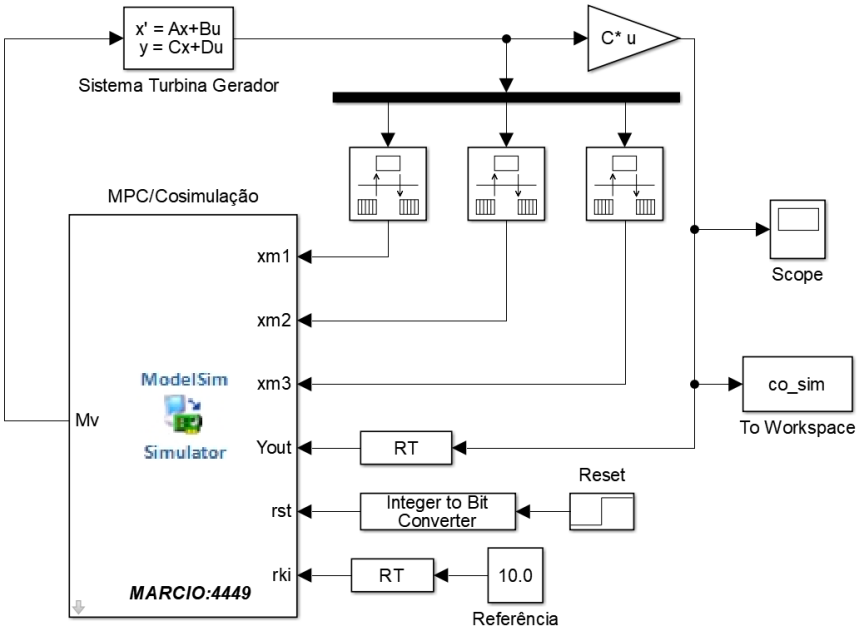
Com o resultado do *script*, o próximo passo é realizar a cosimulação para comparar as duas respostas. Na Figura 16 apresenta-se o ambiente de simulação do Simulink com o controle sendo simulado pelo Modelsim.

As primeiras simulações do MPC com o sistema turbo-gerador têm como objetivo verificar se o código VHDL está correto, ou seja, se os cálculos dos ganhos e a aplicação da lei de controle ocorre nos tempos corretos sem atraso nos dados que prejudique o projeto do controlador. Por isso, foi configurado o MPC para valores mínimos de N_p, N_c e peso r_w, sendo estes incrementados para identificação de possíveis problemas.

A resposta da cosimulação com os mesmos parâmetros de N_p, N_c e r_w foi similar ao visto com o *script* baseado no tutorial, como visto na Figura 17 a saída da

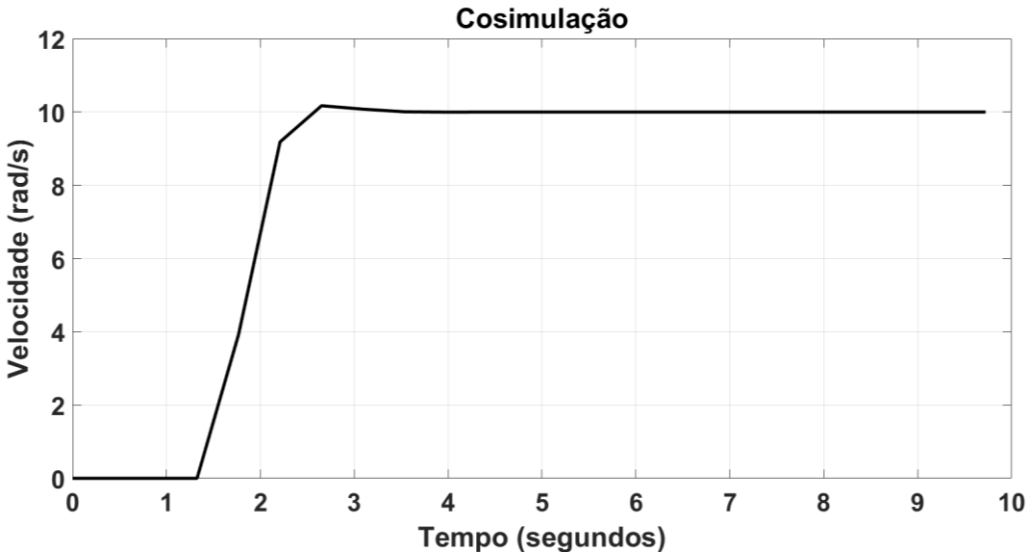
planta. O gráfico da lei de controle também apresentou resultado similar, conforme é apresentado na Figura 18 o comparativo entre a lei de controle calculada pelo script baseado no tutorial e a lei de controle aplicada no ensaio da cosimulação.

Figura 16 - Cosimulação do MPC no Simulink



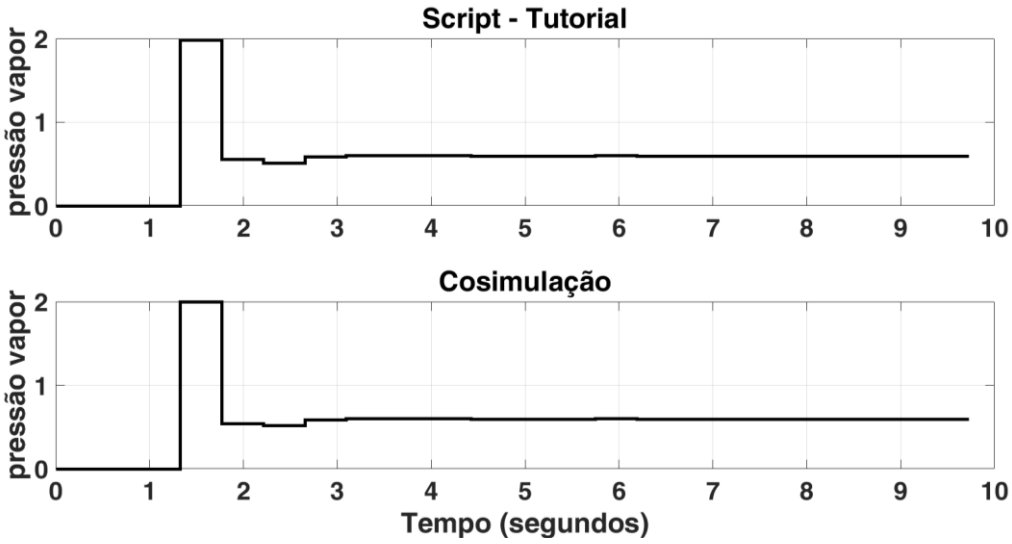
Fonte: Do autor

Figura 17 - Cosimulação para Np = 2 e Nc = 1



Fonte: Do autor

Figura 18 – Comparação da lei de controle na cosimulação e no *script*



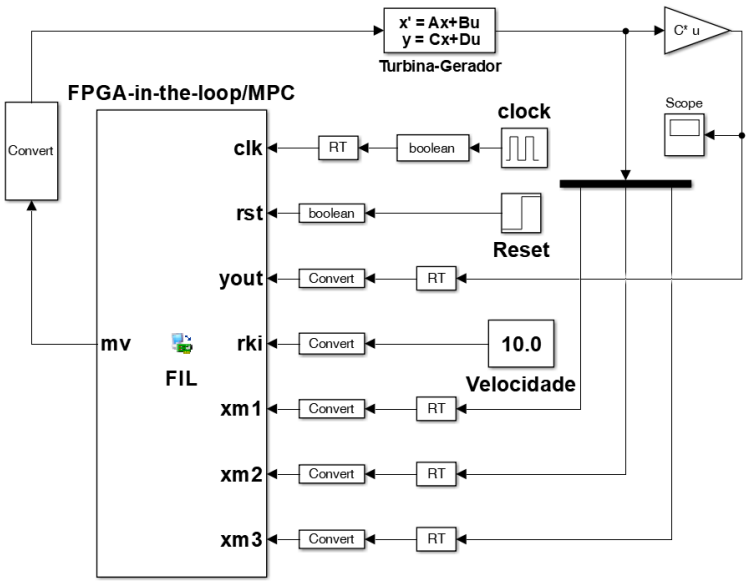
Fonte: Do autor

Este ensaio permite observar que a resposta da cosimulação é quase idêntica a resposta gerada pelo *script* desenvolvido. Isto indica uma forte tendência de que o algoritmo VHDL desenvolvido para implementar o MPC possa funcionar em meio físico. Em consequência disso, o próximo passo é realizar a simulação de FPGA-in-the-loop com os mesmos parâmetros do MPC e comparar com os outros dois resultados anteriores. Na Figura 19 apresenta-se o ambiente do Simulink com o controle sendo simulado na FPGA. Nota-se a inclusão de blocos de conversão pois neste ensaio as variáveis do código VHDL estão representadas com o tipo ponto fixo com sinal (sfixed) e o MATLAB precisa converter este tipo dado para dar suporte a simulação no ambiente Simulink.

Na Figura 20 apresenta-se o resultado da saída do processo para o experimento FPGA-in-the-loop e mais uma vez o resultado é quase idêntico ao visto pela simulação do *script*. Apesar de os gráficos serem muito parecidos, uma análise mais profunda mostra que os ensaios envolvendo a FPGA e Modelsim implica pequenas diferenças númericas nos resultados da saída e da lei de controle. Por isso, não é afirmado que os três ensaios deram resultados iguais, mas semelhantes.

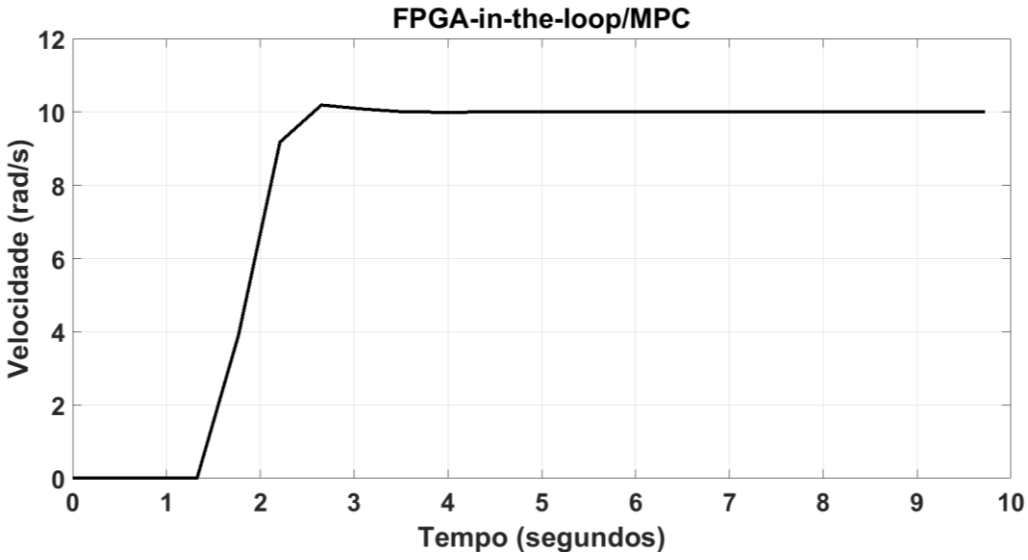
Todas as simulações anteriores foram realizadas para o MPC generalizado sem observador de estados, mas a abordagem com observador também foi simulada com os mesmos parâmetros e os mesmos resultados foram obtidos. Na Figura 21 mostra-se o ensaio de *FPGA-in-the-loop* para o MPC com observador.

Figura 19 - FPGA-in-the-loop do MPC para Turbina-gerador



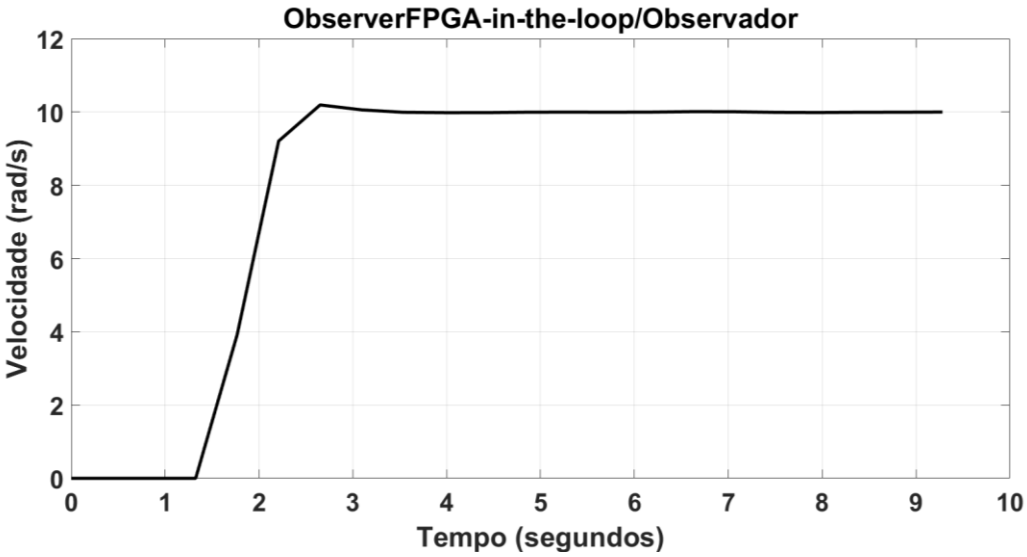
Fonte: Do autor

Figura 20 - FPGA-in-the-loop do MPC para Np = 2 e Nc = 1



Fonte: Do autor

Figura 21 - FPGA-in-the-loop com observador para Np = 2 e Nc = 1



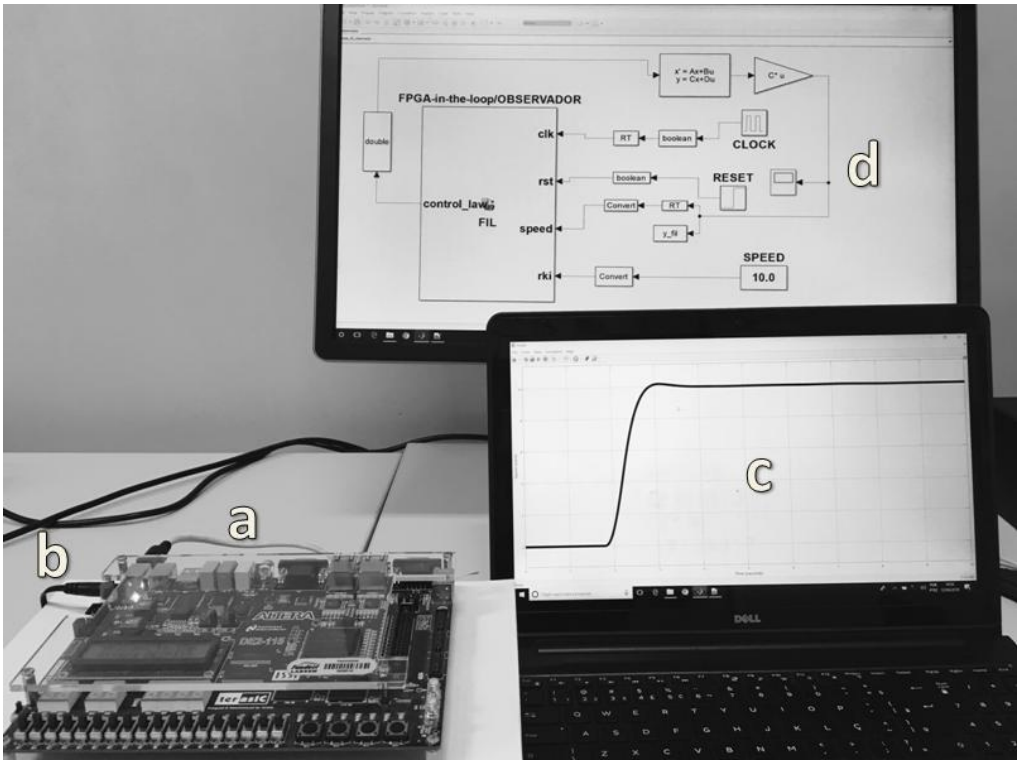
Fonte: Do autor

Pelos resultados dos ensaios anteriores podemos observar que o MPC obteve o mesmo desempenho para as configurações mínimas de N_p e N_c. Na Figura 22 mostra-se uma foto no qual está sendo executado o ensaio FPGA-in-the-loop do MPC com observador. Na Figura 22, ‘a’ indica o cabo USB-blaster responsável pela comunicação da FPGA com o computador, ‘b’ é o cabo de alimentação da FPGA, ‘c’ é o *scope* do Simulink com a resposta do ensaio de FPGA-in-the-loop, ‘d’ é o ambiente Simulink que está sendo feita a simulação.

Por esta planta ser simples tendo os pólos em malha aberta todos no eixo real e com dinâmica estável, além de o tempo de estabilização ser de pouco mais de 4 segundos, a configuração adotada é uma boa escolha para controlar este tipo de planta conforme é demonstrado na abordagem prática do MPC em [2] .

Um dos objetivos é tornar o código dinâmico, ou seja, com simples alterações, como nos valores de N_p e N_c, o algoritmo deve permitir gerar um MPC com uma configuração diferente. Por isso, os ensaios vistos anteriormente foram repetidos para outras configurações a fim de validar o código.

Figura 22 – Foto com o ensaio do FPGA-in-the-loop



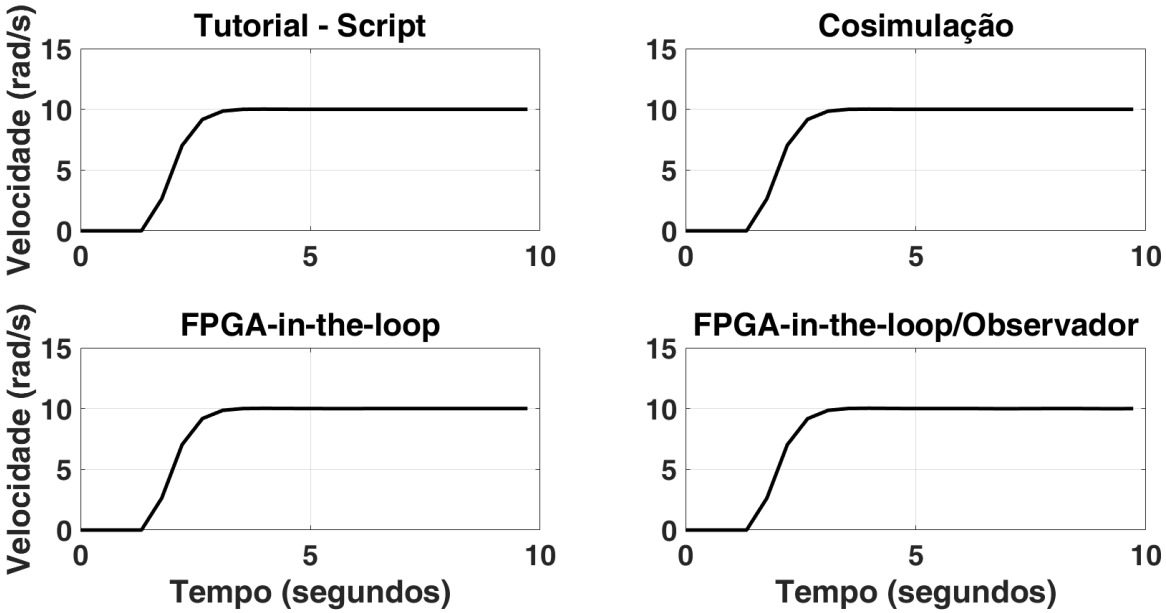
Fonte: Do autor

Primeiro foi variado o parâmetro do horizonte de predição, o N_p, com os outros parâmetros mantidos constantes. Os resultados para o N_p sendo igual a 3 podem ser vistos na Figura 23. Neste caso, a saída não apresenta nenhum percentual de ultrapassagem e o tempo de estabilização continua pouco maior que 1 segundo, afinal é de conhecimento que o N_p quando se aproxima do tempo de estabilização do sistema proporciona uma resposta melhor.

O horizonte de predição continou sendo incrementado e mais duas simulações foram feitas com o N_p sendo igual a 4 e 5. Na Figura 24 observa-se os resultados para o N_p = 4 e na Figura 25 apresenta-se os resultados para o Np = 5.

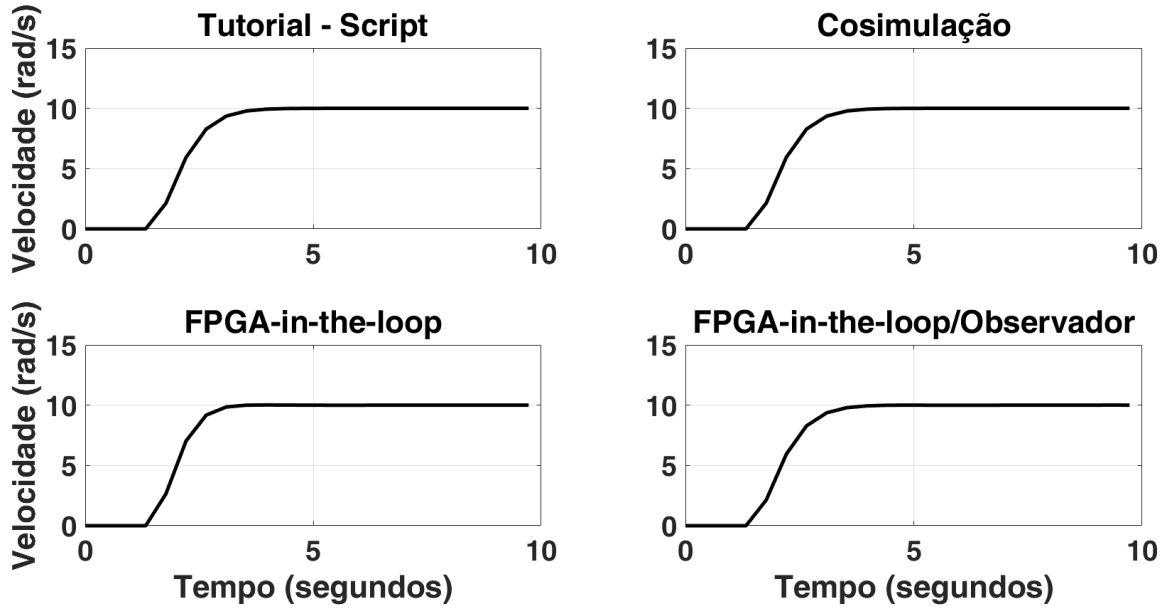
É possível observar que os ensaios com o Np = 4 apresentam um resultado bem semelhante ao do Np = 3.

Figura 23 - FPGA-in-the-loop com observador para Np = 3 e Nc = 1



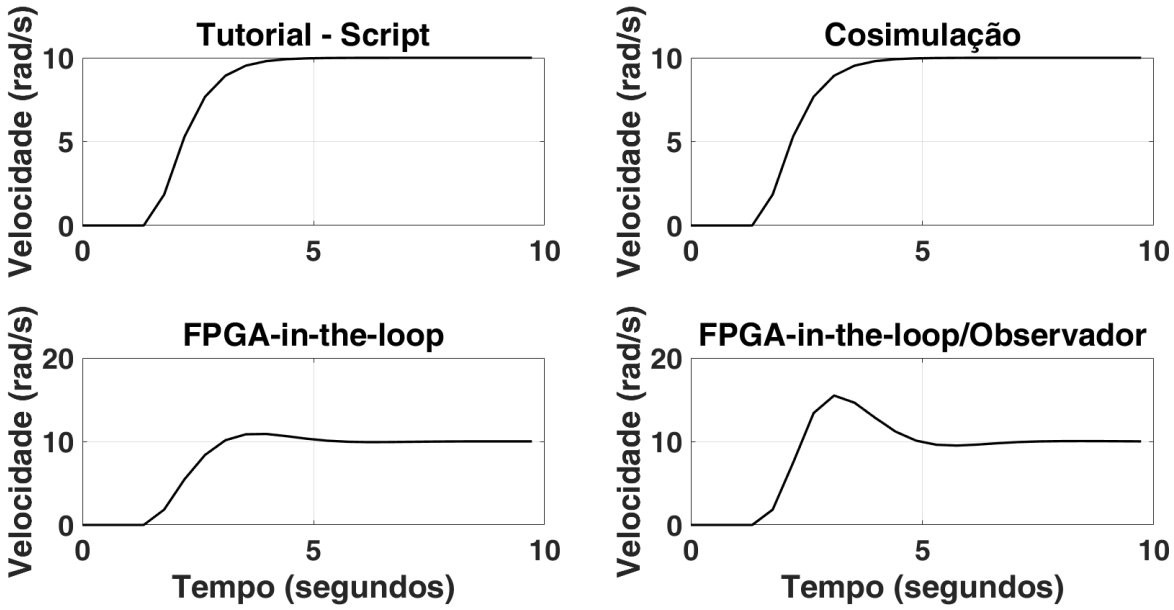
Fonte: Do autor

Figura 24 - FPGA-in-the-loop com observador para Np = 4 e Nc = 1



Fonte: Do autor

Figura 25 - FPGA-in-the-loop com observador para Np = 5 e Nc = 1



Fonte: Do autor

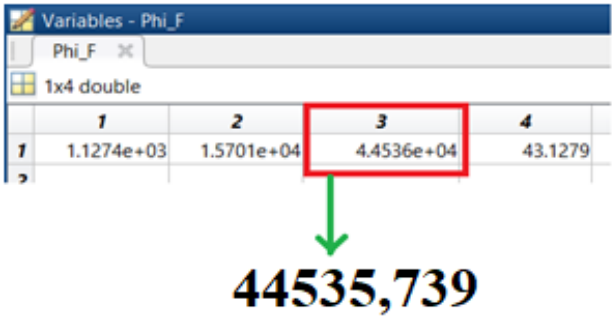
Na Figura 25 fica claro que os ensaios em FPGA-in-the-loop para o Np = 5 não deram o mesmo tipo de reposta que o script do MATLAB e a cosimulação. Isto ocorreu porque o padrão da biblioteca da Altera é para que os valores numéricos sejam representados, em complemento a dois, por 15 bits da parte inteira e 16 bits da parte fracionária, ou seja, há uma certa limitação de representação, por isso conforme o valor de N_p ou N_c aumenta é provável que o ensaio de FPGA-in-the-loop não represente a mesma resposta que os ensaios feitos pelo *script* do MATLAB e pela cosimulação.

De acordo com [34], os limites para a parte inteira em complemento a dois e com 15 bits é:

$$-2^{15}\ at\acute{e} + (2^{15} - 1)$$
$$-32768\ at\acute{e} + 32767$$

Para entender melhor o que aconteceu, na Figura 26 é vista a matriz φ*F para o cálculo dos ganhos do MPC com a configuração N_p = 5. Um dos valores ultrapassa os limites de representação do padrão da biblioteca de matrizes da Altera. Logo, para esta representação é necessário um aumento no número de bits que representam os dados. Na Tabela 3 pode-se observar o valor do ganho K_{mpc} correto comparado com o que acabou sendo calculado com o limite extrapolado.

Figura 26 - FPGA-in-the-loop com observador para Np = 4 e Nc = 2



Fonte: Do autor

Tabela 3 – Comparação dos ganhos Kmpc

K_{mpc}	$K_x(1,1)$	$K_x(1,2)$	$K_x(1,3)$	K_y
Script/Cosimulação	2,4493	34,1118	96,7572	0,0937
FPGA-in-the-loop	2,4414	33,9945	71	0,0934

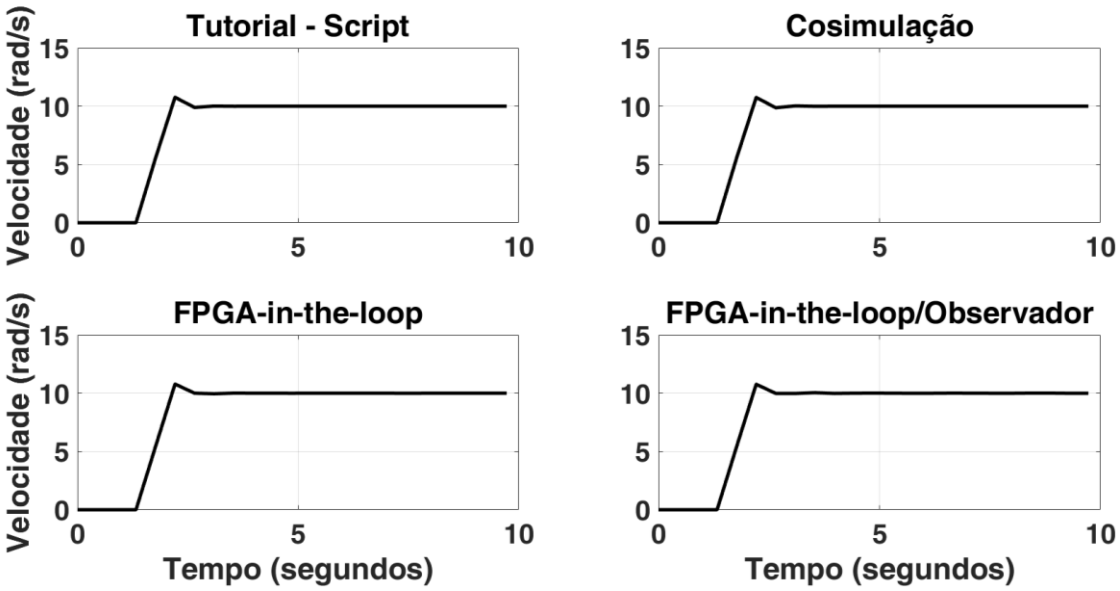
Fonte: Do autor

Tendo identificado o erro referente ao aumento do horizonte de predição, o próximo passo foi tentar identificar se ocorre algo estranho quando o horizonte de controle aumenta. Por isso, para os próximos ensaios, o horizonte de predição foi ajustado e mantido sempre $N_p = 4$ e incrementa-se o $N_c = 2$. Na Figura 27 mostra-se os resultados do experimento.

É de conhecimento que um aumento do N_c acarreta em tornar a resposta, para este tipo de planta, mais oscilante. Conforme mencionado, o intuito é identificar os possíveis problemas com o código quando os parâmetros possuem grande valor numérico e não comparar as respostas para as diversas configurações. Neste caso a resposta para os quatro ensaios foi igual, portanto, não houve nenhum problema por enquanto.

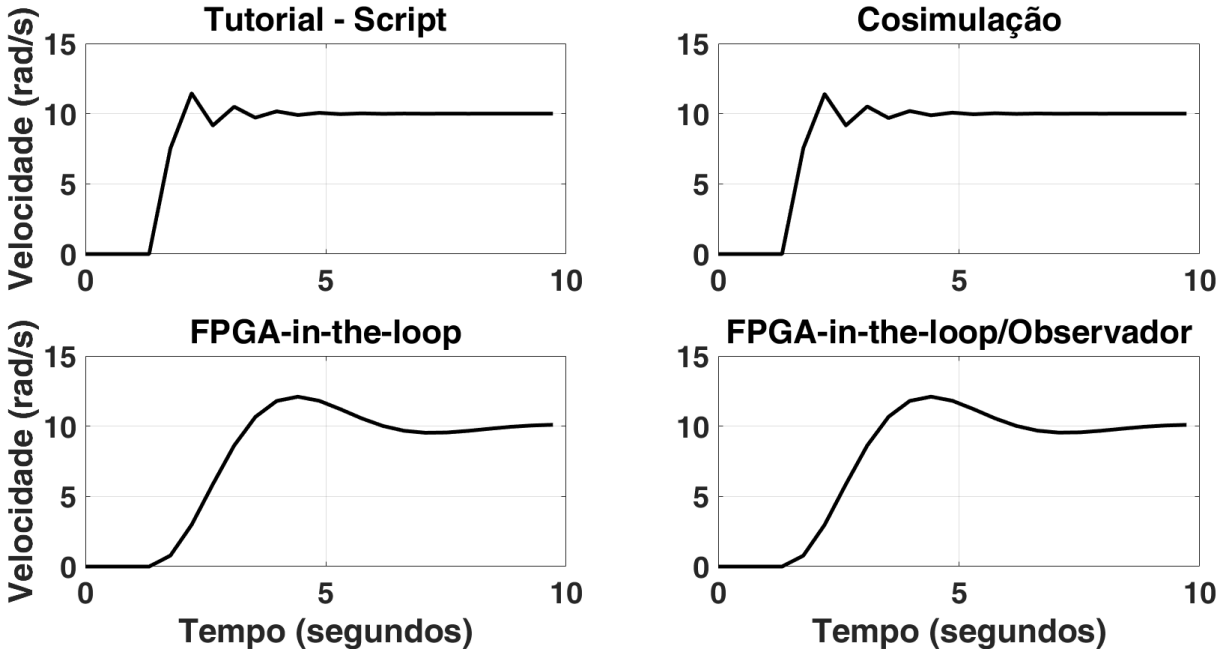
Em continuidade, o horizonte de controle é mais uma vez incrementado para $N_c = 3$, e na Figura 28, é possível observar que os ensaios em FPGA-in-the-loop não tiveram o mesmo tipo de resposta que o da cosimulação e do *script*.

Figura 27 - FPGA-in-the-loop com observador para Np = 4 e Nc = 2



Fonte: Do autor

Figura 28 - FPGA-in-the-loop com observador para Np = 4 e Nc = 3

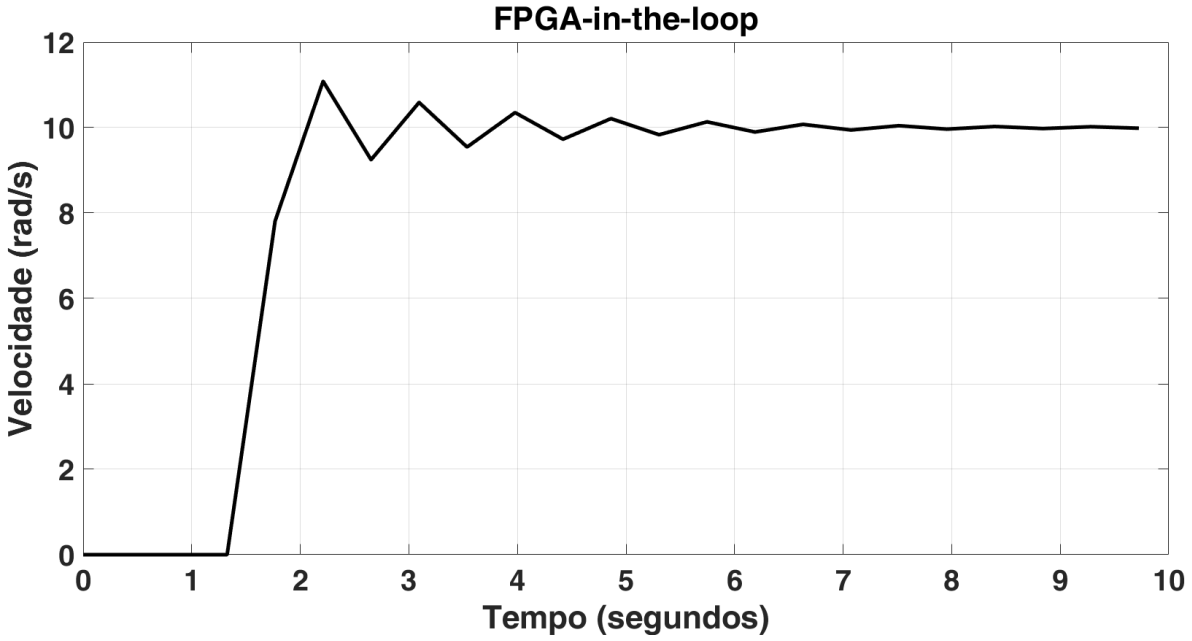


Fonte: Do autor

O aumento do horizonte de controle N_c altera a dimensão da matriz φ conforme a (21). Quando o peso r_w é igual a zero, para o cálculo do ganho K_{mpc}, é necessário calcular a inversa da multiplicação desta matriz com a sua transposta

(matriz $\varphi^T\varphi$). O problema é que a função inversa de uma matriz é difícil de implementar em hardware e para a biblioteca utilizada, apesar de aparentemente os cálculos e a lógica de programação estarem corretos, a resposta da função inversa nem sempre condiz com os valores verdadeiros (mais sobre este erro é visto na 3.3 na simulação do reator CSTR). Neste caso, o resultado da inversa teve valores dez vezes menores que os verdadeiros. Ainda não se sabe o que ocasiona essa mudança na função inversa, mas ao se corrigir manualmente os valores o resultado do ensaio FPGA-in-the-loop se torna semelhante ao visto nos outros ensaios, como visto na Figura 29.

Figura 29 - FPGA-in-the-loop com observador para Np = 4 e Nc = 3 corrigido

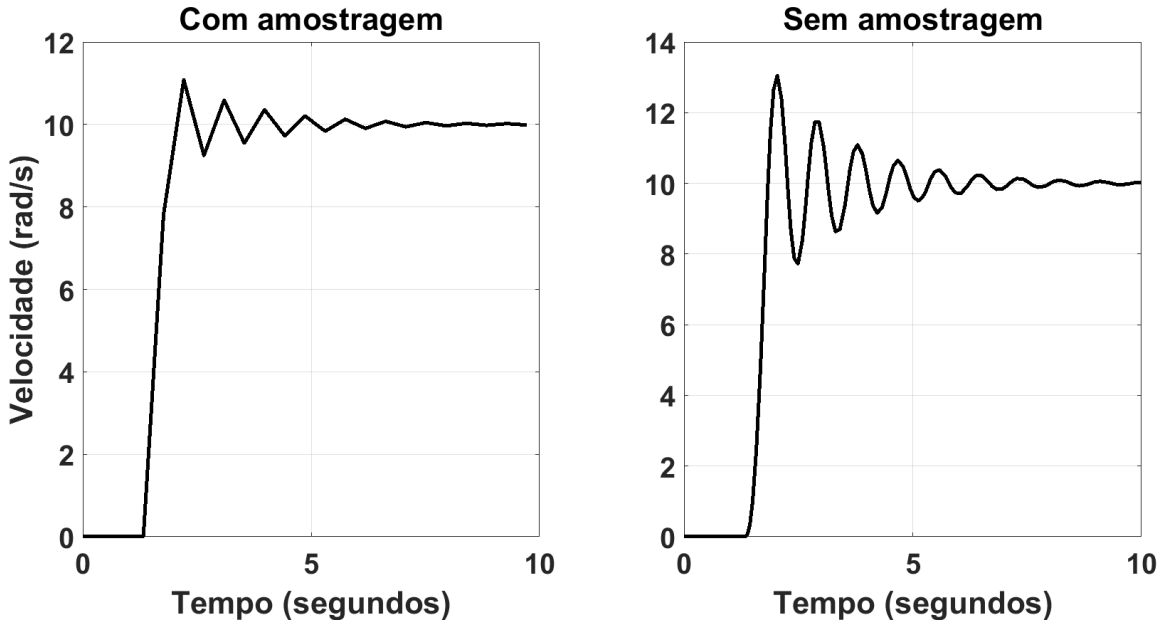


Fonte: Do autor

O leitor deve tomar cuidado com o ajuste das configurações do MPC. Neste caso por exemplo, para esta planta com o aumento do horizonte de controle a resposta da saída da planta tende a ficar mais oscilatória. Porém, se analisar o gráfico da Figura 29, o leitor pode pensar que o overshoot desta resposta é 13,98%. O problema é que esta resposta e também as vistas anteriormente são para os dados amostrados de acordo com o tempo de discretização da planta. O Simulink possui um passo de simulação muitas vezes menor que o tempo de amostragem, por isso ao conferir o gráfico com o *scope* do Simulink, este *overshoot* se torna maior. Na Figura 30 demonstra-se a comparação dos dados amostrados para a

resposta sem amostragem gerada pelo Simulink e pode-se identificar que o overshoot na verdade é de 28,54%.

Figura 30 – Diferença das respostas com e sem amostragem para Np = 4 e Nc = 3



Fonte: Do autor

Na Tabela 4 estão os valores dos ganhos calculados pelo *script* e cosimulação comparados com os vistos no ensaio *FPGA-in-the-loop*. No processo de calcular a matriz inversa o algoritmo perde um pouco de precisão o que é normal quando os dados são representados em ponto fixo, mas fica claro que os valores estão em uma proporção dez vezes menores.

Tabela 4 – Comparação dos ganhos Kmpc

Kmpc	Kx(1,1)	Kx(1,2)	Kx(1,3)	Ky
Script/Cosimulação	3,8382	57,2897	191,3920	0,3827
FPGA-in-the-loop	0,4001	5,9683	19,9290	0,0398

Fonte: Do autor

A variação no parâmetro do peso r_w não implica mudança nas dimensões das matrizes do MPC, por isso não houve erros no código e não são mostrados resultados para simulações com esta alteração neste trabalho. O peso r_w é discutido

com mais detalhes na 3.4, pois ele terá um papel importante na simulação do reator PAR.

Para esta planta também foi realizado um ensaio para comparar a área ocupada pelo circuito digital que implementa o MPC e a variação do tamanho do circuito em relação não só aos incrementos nos horizontes de predição e controle, mas também a abordagem com observador.

Na Tabela 5 mostra-se os dados de recursos da placa utilizados para sintetizar o MPC sem observador quando ocorre a variação no horizonte de predição. Pode-se notar que esta variação não implica grandes mudanças na área ocupada pelo circuito do MPC. Basicamente, houve pouco aumento de blocos DSP (Processamento Digital de Sinais - *Digital Signal Processing*) e elementos lógicos (EL) utilizados.

Tabela 5 – Área ocupada pelo MPC sem observador com N_c fixo

Recursos	$N_c = 1$			Total DE2-115
	$N_p = 2$	$N_p = 3$	$N_p = 4$	
EL	3103 (3%)	3328 (3%)	3550 (3%)	114480
Registradores lógicos dedicados	257 (<1%)	257 (<1%)	257 (<1%)	114480
DSP	76 (14%)	84 (16%)	92 (17%)	532

Fonte: Do autor

Na Tabela 6 pode-se observar os dados do MPC sem observador para quando o horizonte de controle é variado. Pode-se notar que esta variação tem maior impacto na área ocupada pelo MPC porque este parâmetro muda a dimensão da matriz φ, essencial para o cálculo dos ganhos. Há um aumento considerável de blocos DSP e elementos lógicos (EL) conforme o N_c aumenta.

Tabela 6 - Área ocupada pelo MPC sem observador com N_p fixo

Recursos	$N_p = 4$			Total DE2-115
	$N_c = 1$	$N_c = 2$	$N_c = 3$	
EL	3550 (3%)	6124 (5%)	7618 (7%)	114480
Registradores lógicos dedicados	257 (<1%)	257 (<1%)	257 (<1%)	114480
DSP	92 (17%)	152 (29%)	196 (37%)	532

Fonte: Do autor

Nas Tabelas 7 e 8 analisam-se as mudanças com o MPC com observador. É possível identificar que a inclusão de um observador de estados também acarreta um aumento considerável na área ocupada pelo MPC. Isto acontece devido a necessidade de armazenar mais informações para estimar os estados da planta.

Tabela 7 - Área ocupada pelo MPC com observador com N_c fixo

Recursos	$N_c = 1$			Total DE2-115
	$N_p = 2$	$N_p = 3$	$N_p = 4$	
EL	7517 (7%)	7746 (7%)	7966 (7%)	114480
Registradores lógicos dedicados	193 (<1%)	193 (<1%)	193 (<1%)	114480
DSP	180 (34%)	188 (35%)	196 (37%)	532

Fonte: Do autor

Tabela 8 - Área ocupada pelo MPC com observador com N_p fixo

Recursos	$N_p = 4$			Total DE2-115
	$N_c = 1$	$N_c = 2$	$N_c = 3$	
EL	7966 (7%)	10898 (10%)	12641 (11%)	114480
Registradores lógicos dedicados	193 (<1%)	193 (<1%)	193 (<1%)	114480
DSP	196 (37%)	264 (50%)	316 (59%)	532

Fonte: Do autor

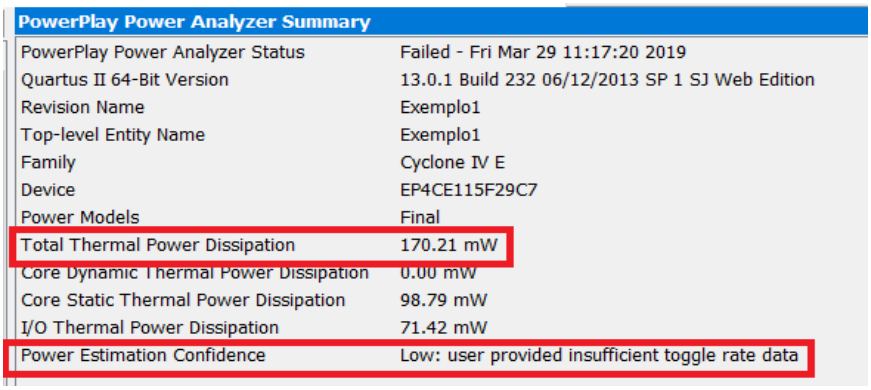
Além dos resultados de área ocupada pelo circuito digital projetado, é conveniente também que o projeto não consuma muita energia para realizar os cálculos. O Quartus possui uma ferramenta de nome *PowerPlay Power Analyzer* que estima o consumo de potência do circuito projetado.

Para realizar este teste basta executar uma simulação no Modelsim tendo seguido os passos de configuração vistos em [35]. Neste trabalho foram executados os passos descritos, mas o Modelsim PE não conseguiu gravar no arquivo VCD as informações de transição das variáveis do código VHDL. Deste modo, a estimativa gerada pela ferramenta do Quartus, vista na Figura 31, fica com confiabilidade baixa.

Os testes de consumo de potência ficarão para um trabalho futuro, mas se este dado for confirmado ou a ferramenta gerar uma estimativa mais confiável que seja próxima da Figura 31, o MPC em VHDL desenvolvido aqui pode ser considerado bem eficiente, pois nos trabalhos em [7] e [15], apesar de controlar

plantas um pouco mais complexas, os circuitos projetados consumiram mais energia.

Figura 31 – Resultados do consumo de potência



PowerPlay Power Analyzer Summary	
PowerPlay Power Analyzer Status	Failed - Fri Mar 29 11:17:20 2019
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	Exemplo1
Top-level Entity Name	Exemplo1
Family	Cyclone IV E
Device	EP4CE115F29C7
Power Models	Final
Total Thermal Power Dissipation	170.21 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	98.79 mW
I/O Thermal Power Dissipation	71.42 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

Fonte: Do autor

3.1.1 TURBINA-GERADOR COM RESTRIÇÕES

Nesta seção são apresentados os resultados do MPC com restrições para o sistema turbo-gerador. De acordo com a teoria vista no capítulo 1, o algoritmo de otimização utilizado é o procedimento de programação quadrática de Hildreth.

Na Figura 32 são vistos os resultados para a saída da planta no experimento com o *script* e a cosimulação do código VHDL implementado. As restrições feitas nestes ensaios foram apenas na variação da lei de controle. As Equações 156 e 157 mostram as restrições incorporadas ao projeto do MPC. Para este ensaio $N_p = 4$, $N_c = 3$ e o peso $r_w = 0$.

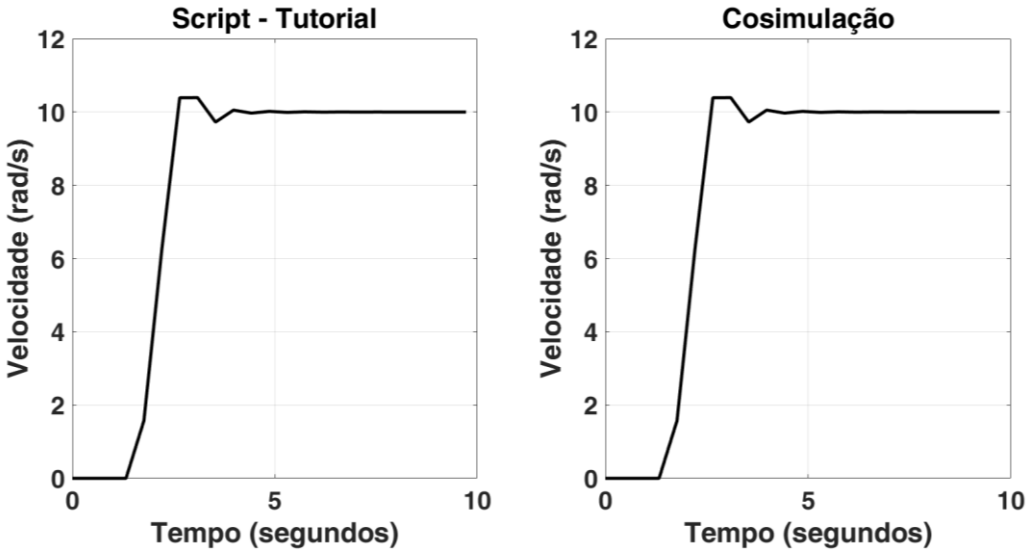
$$-\Delta U \leq -0{,}8 \tag{156}$$

$$\Delta U \leq 0{,}8 \tag{157}$$

Logo, a matriz M e γ para esta configuração são vistas na Equação 158.

$$\overbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}}^{M} \Delta U \leq \overbrace{\begin{bmatrix} 0{,}8 \\ 0{,}8 \\ 0{,}8 \\ 0{,}8 \\ 0{,}8 \\ 0{,}8 \end{bmatrix}}^{\gamma} \tag{158}$$

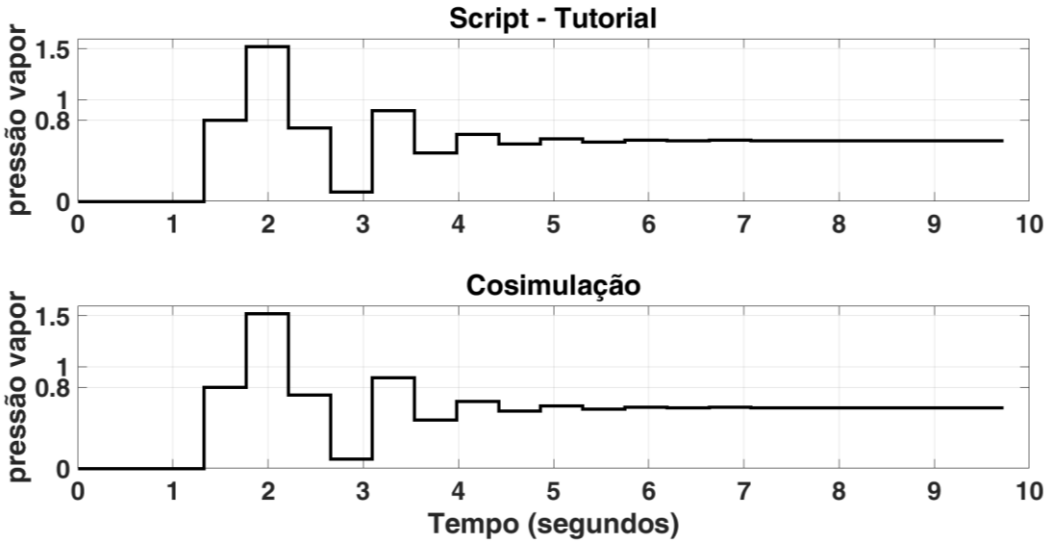
Figura 32 - Bloco diagrama do MPC com observador



Fonte: Do autor

Tanto a resposta da cosimulação quanto a do tutorial apresentaram overshoot de 3,92%. Na Figura 33 é visto a lei de controle dos dois experimentos e é possível identificar que a restrição imposta foi respeitada.

Figura 33–Resposta da lei de controle do MPC com restrição

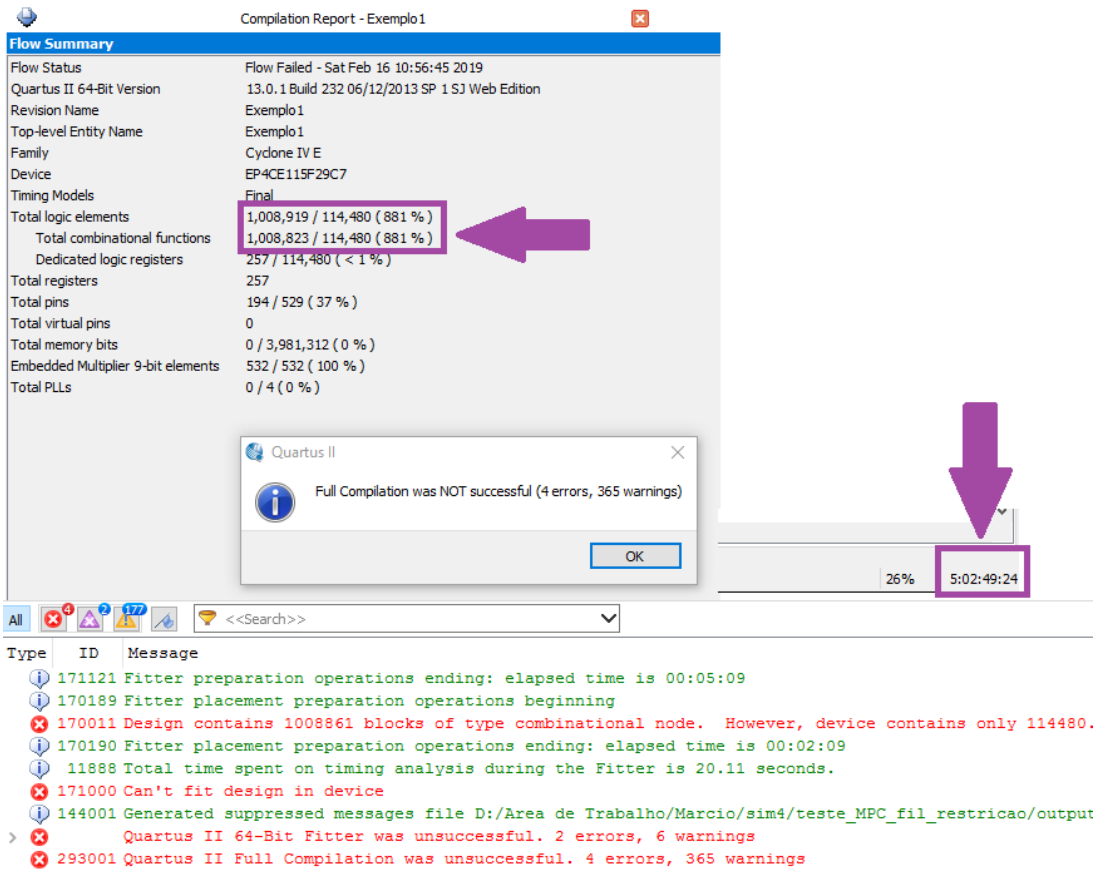


Fonte: Do autor

Em seguida, desejava-se realizar os testes de *FPGA-in-the-loop*, porém não foi possível realizar tais ensaios devido a complexidade de sintetizar o circuito gerado pelo código. Como mostra a Figura 34 o código VHDL ficou mais de 5 dias

compilando no programa Quartus e o circuito necessitava de pouco mais de um milhão de elementos lógicos para sua síntese, o que não há suporte a este volume no *kit* educacional da DE2-115. Recentemente, a Intel-Altera lançou uma nova família de FPGAs Stratix 10 com placas que possuem de 1,5 a 2 milhões de elementos lógicos, e são estas placas que poderiam sintetizar o MPC com restrições gerado por este código. Por isso, nas simulações das próximas plantas focaremos apenas nas cosimulações pois este experimento consegue fornecer uma forte evidência de que o MPC, se implementado em meio físico, pode funcionar também. O código desta simulação se encontra disponível no anexo ao final do trabalho.

Figura 34 – Erro da síntese do MPC no Quartus II



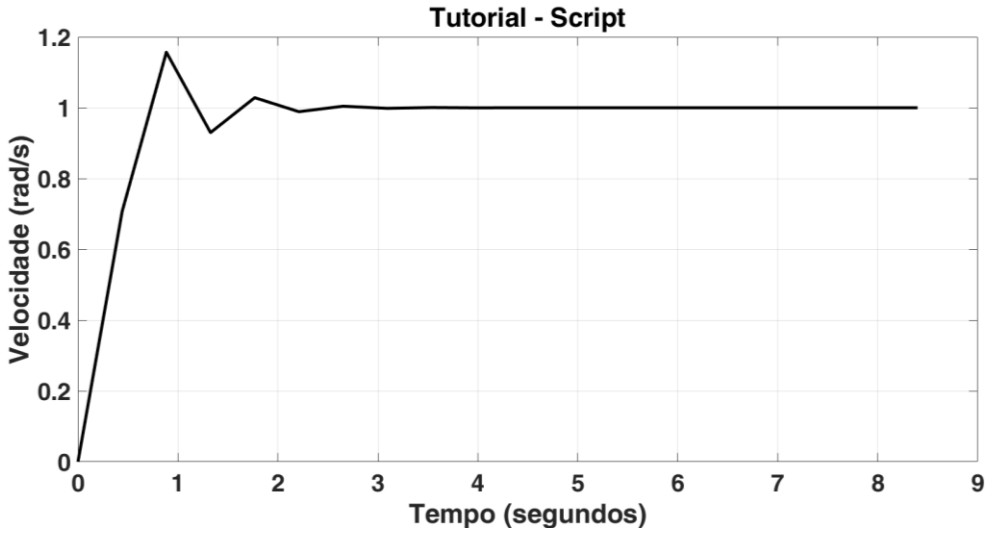
Fonte: Do autor

3.1.2 TURBINA-GERADOR COM LAGUERRE

Na Figura 35 apresenta-se a saída do processo turbo-gerador com o controlador implementado com funções de Laguerre, descrito no capítulo 1. Para

relembrar, neste caso o pólo de Laguerre considerado foi a = 0,3, o número de termos N = 5 e o horizonte de predição N_p = 5.

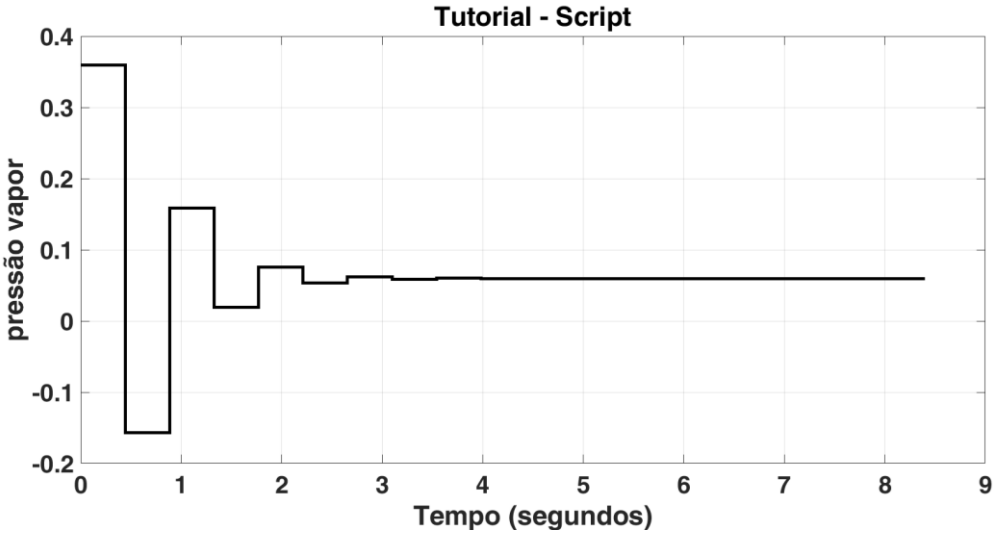
Figura 35 – Resposta da saída para MPC com funções de Laguerre



Fonte: Do autor

A resposta do MPC com as funções de Laguerre apresentou um overshoot de 15,69%. Para este tipo de planta, já é conhecido que o MPC generalizado com horizonte de controle unitário costuma ser a melhor abordagem de projeto, por isso o overshoot foi menor nestas abordagens. Na Figura 36 mostra-se a lei de controle aplicada.

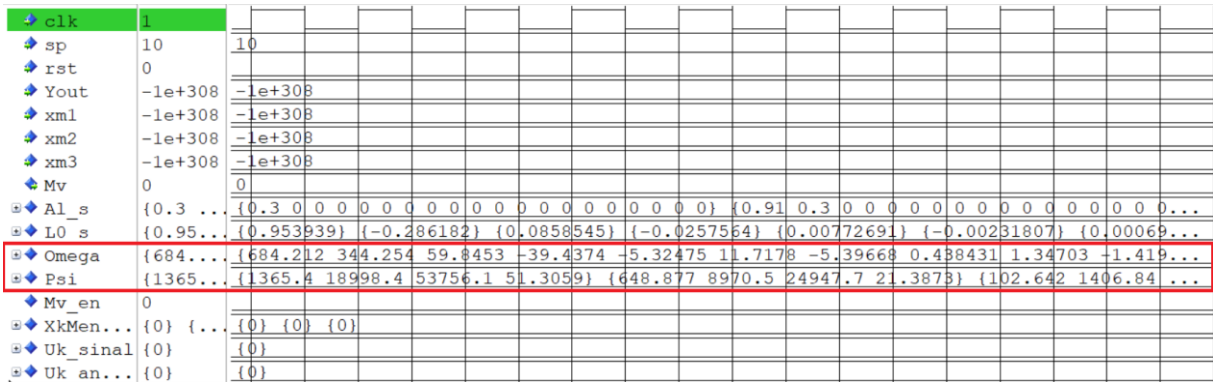
Figura 36 – Resposta do controle para MPC com funções de Laguerre



Fonte: Do autor

A cosimulação do código VHDL do MPC com as funções de Laguerre foi executada, porém o computador ficou vários dias para calcular a primeira lei de controle e a simulação não prosseguia. Por este motivo não serão apresentados resultados deste experimento. Uma simulação com valores forçados utilizando apenas o Modelsim PE indica que as matrizes Ω e Ψ, vistas no capítulo 141.3, são calculadas corretamente, como mostra na Figura 37.

Figura 37 – Matrizes Ω e Ψ simuladas no ModelSim



Fonte: Do autor

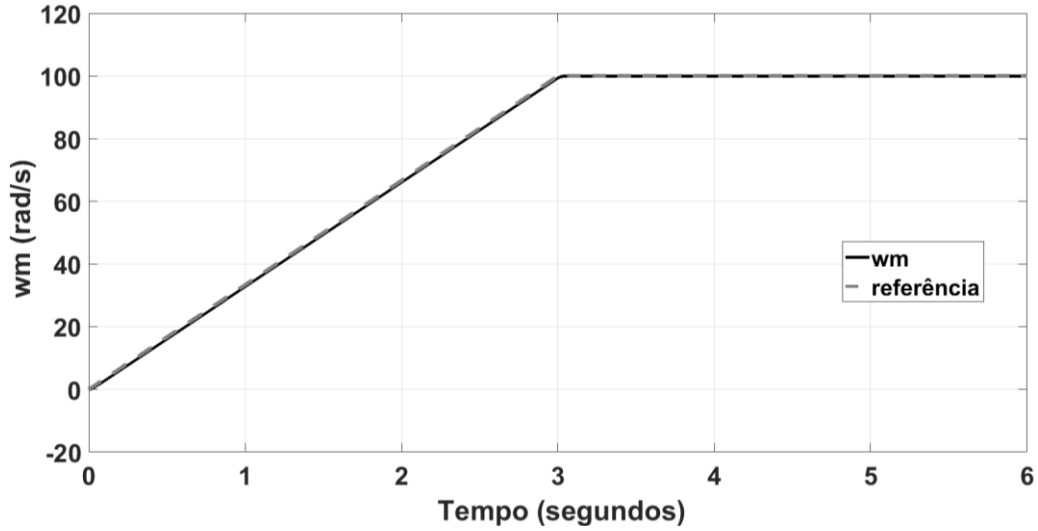
Aparentemente o código do MPC com as funções de Laguerre está funcionando corretamente, porém é preciso mais recursos computacionais para confirmar por meio da cosimulação a exatidão e precisão do código. Os ensaios de FPGA-in-the-loop não foram testados devido a este problema da cosimulação. O código desta simulação se encontra disponível no anexo ao final do trabalho.

3.2 MÁQUINA SÍNCRONA

O controle da máquina síncrona teve como parâmetros $N_p = 5$ e $N_c = 3$, com peso $r_w = 10^{-8}$. O tempo de amostragem, já visto no capítulo 2, foi de 10^{-4} segundos para discretização da planta.

O resultado do rastreamento da velocidade mecânica é visto na Figura 38.

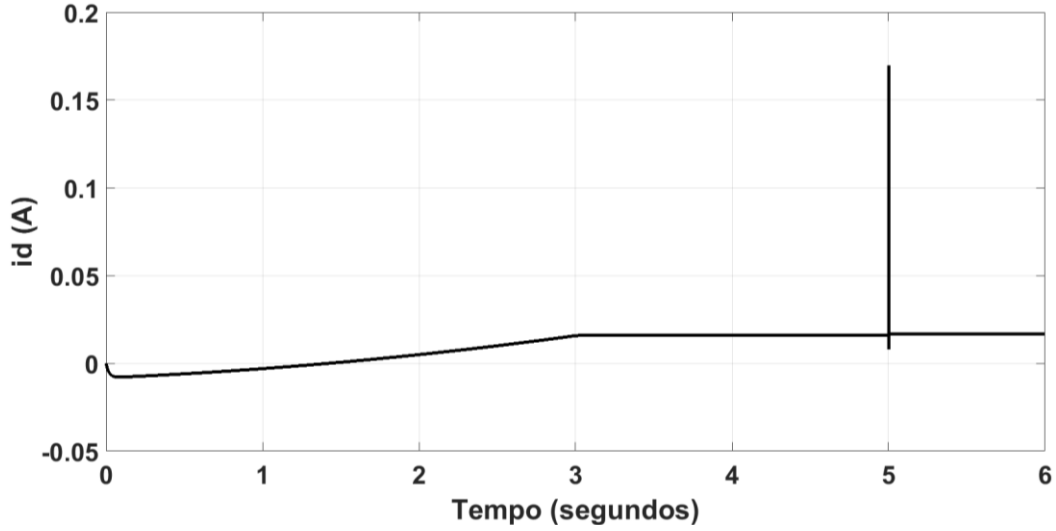
Figura 38 – Resposta da cosimulação para a velocidade mecânica



Fonte: Do autor

O resultado da corrente de eixo d é visto na Figura 39.

Figura 39 – Resposta da cosimulação para a corrente do eixo *d*.



Fonte: Do autor

A cosimulação desta planta serviu como comparação para a técnica dead-beat MPC com a modificação descrita no capítulo 2. O ensaio de FPGA-in-the-loop com a técnica proposta garantiu não só o rastreamento da velocidade mecânica, como também erro de regime permante nulo para a corrente de eixo d. No ensaio da cosimulação, a corrente *id* teve como valor final aproximadamente 20 mA. Os

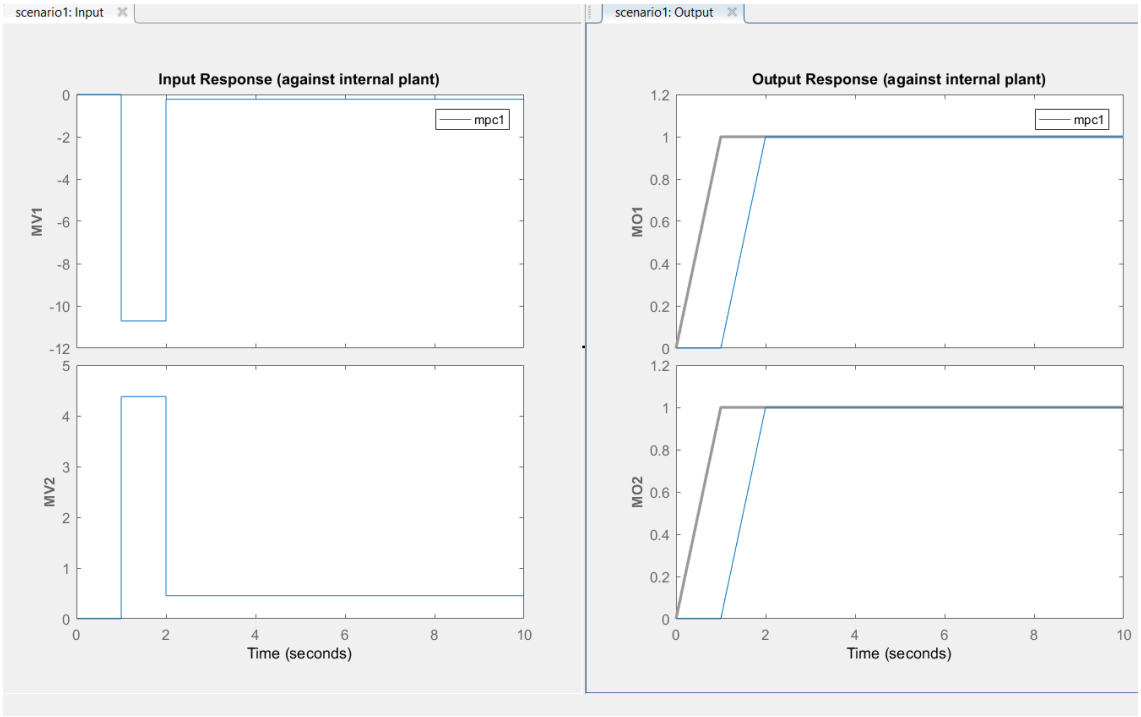
resultados para esta planta geraram um artigo que foi submetido para o Journal of Emerging and Selected Topics in Power Eletronics (JESTPE).

3.3 REATOR CSTR

A simulação para o reator CSTR foi feita com a configuração $N_p = 9$, $N_c = 4$, peso $r_w = 0$. Nesta simulação foi considerado que as duas entradas são controladas e as duas saídas são medidas.

A toolbox do MPC do MATLAB já indica a resposta do reator para esta configuração, como visto na Figura 40, tendo um degrau unitário como referência para as duas variáveis de saída.

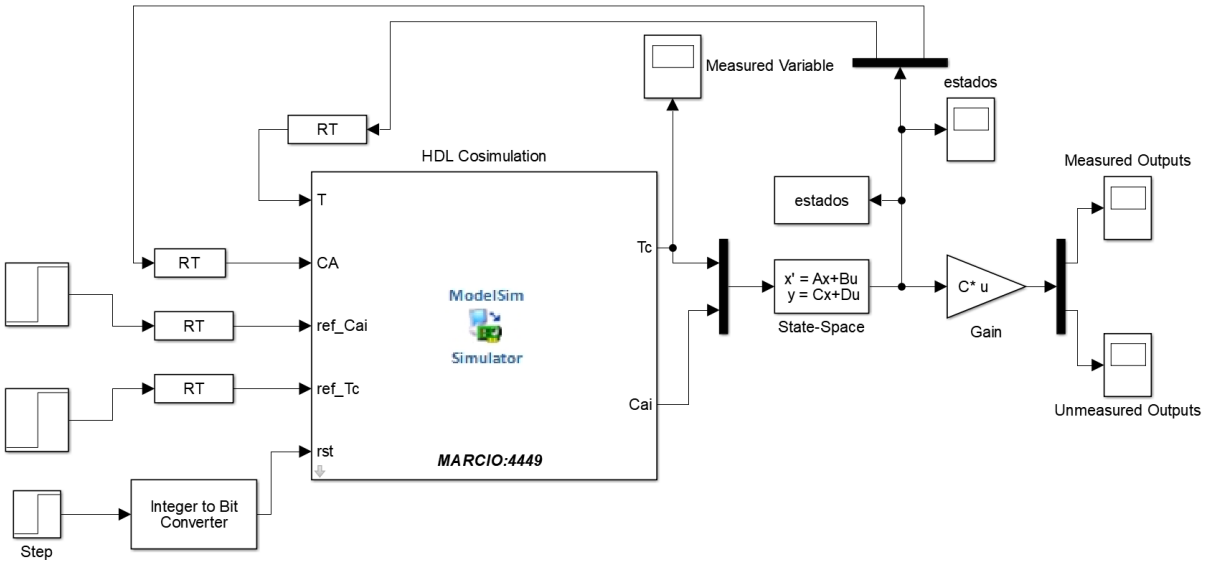
Figura 40 – Resposta do MPC do MATLAB para o CSTR



Fonte: Do autor

Nesta planta apenas a cosimulação foi realizada devido aos problemas relatados no experimento do processo turbo-gerador. O objetivo aqui é tentar replicar a resposta gerada pela toolbox. Na Figura 41 mostra-se o ambiente de simulação do Simulink para este processo.

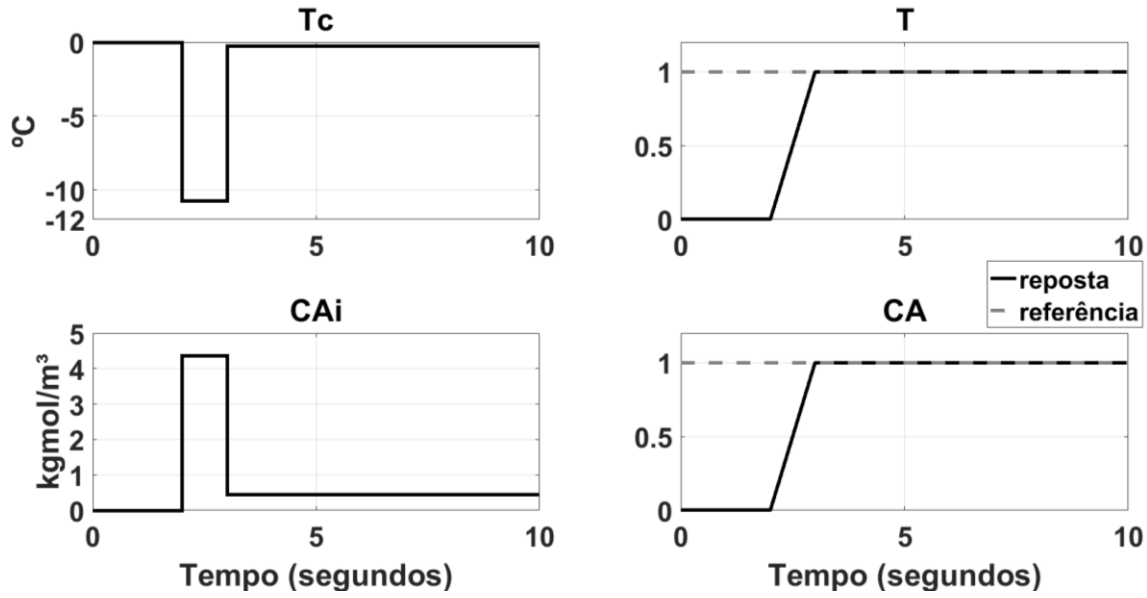
Figura 41 – Cosimulação do MPC para o reator CSTR



Fonte: Do autor

As repostas obtidas com a cosimulação são vistas na Figura 42.

Figura 42 – Resposta da Cosimulação para o reator CSTR



Fonte: Do autor

Pode-se observar que a resposta para a cosimulação é semelhante a resposta apresentada pela toolbox do MATLAB. O controle desta planta tinha como objetivo a produção de um artigo para o congresso interacional INDUSCON, porém devido ao *deadline* ter sido muito próximo da correção do artigo para o CBA e

também porque os resultados anteriores para a máquina síncrona já encobriram grande parte das críticas do artigo do CBA, optou-se por não continuar a trabalhar com esta planta, e sim partir para o desafio de controlar uma planta real.

A importância deste resultado apresentado é que foi a primeira cosimulação em que foi identificado o problema da matriz inversa visto na seção de resultados da turbina gerador. O detalhe é que aqui, como o MPC projetado não foi embarcado, ficou mais fácil de visualizar o problema, pois os dados sendo do tipo real, além de facilitar a compreensão, a compilação do código no ModelSim é mais rápida,afinal não há a síntese do circuito e isto possibilita que mais testes sejam feitos em menor tempo.

Para entender o que houve no CSTR, a Equação 159 mostra-se a inversa da matriz $\varphi^T\varphi$, baseada na matriz φ descrita no capítulo 2. O resultado da matriz inversa tem dimensão 8x8, mas para a comparação somente basta mostrar as 3 primeiras colunas.

$$inv(\varphi^T\varphi) = \begin{bmatrix} 129{,}2253 & -22{,}1830 & -255{,}0329 & \cdots \\ -22{,}1830 & 9{,}6124 & 43{,}9137 & \cdots \\ -255{,}0329 & 43{,}9137 & 632{,}5491 & \cdots \\ 43{,}0178 & -18{,}2107 & -107{,}3317 & \cdots \\ 125{,}8076 & -21{,}7307 & -503{,}3239 & \cdots \\ -20{,}8349 & 8{,}5983 & 84{,}2529 & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad (159)$$

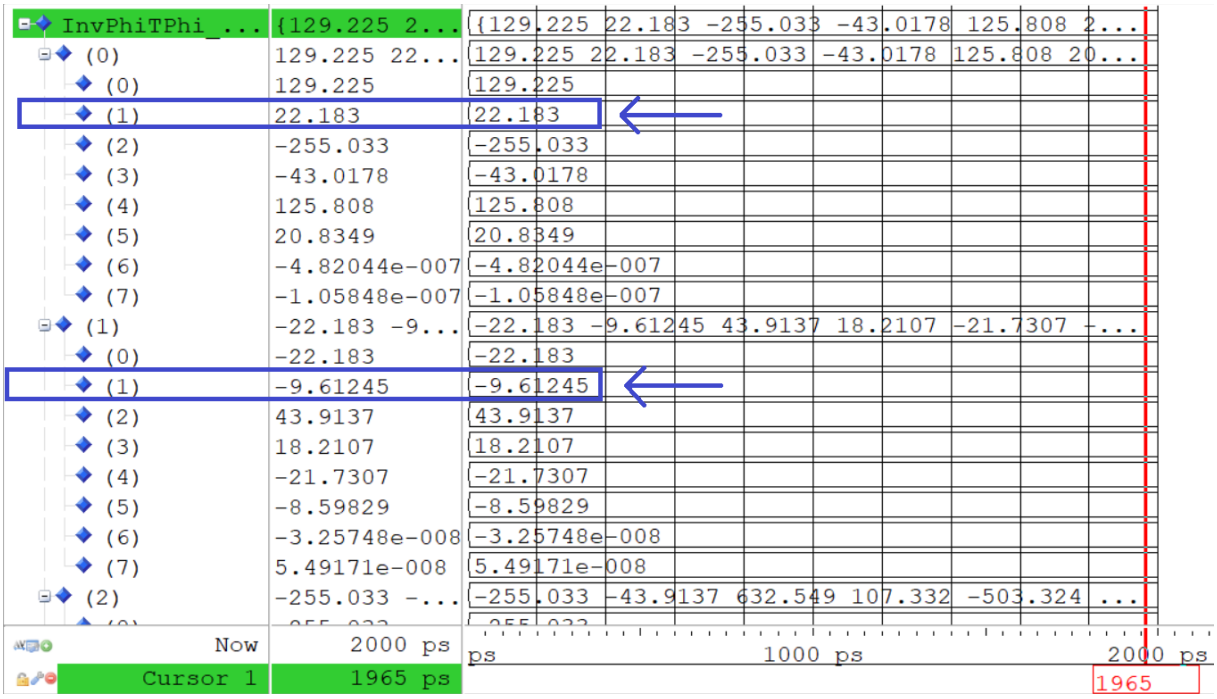
O resultado da cosimulação para esta matriz inversa é visto na Equação 160 e na Figura 42.

$$inv(\varphi^T\varphi) = \begin{bmatrix} 129{,}2253 & 22{,}1830 & -255{,}0329 & \cdots \\ -22{,}1830 & -9{,}6124 & 43{,}9137 & \cdots \\ -255{,}0329 & -43{,}9137 & 632{,}5491 & \cdots \\ 43{,}0178 & 18{,}2107 & -107{,}3317 & \cdots \\ 125{,}8076 & 21{,}7307 & -503{,}3239 & \cdots \\ -20{,}8349 & -8{,}5983 & 84{,}2529 & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad (160)$$

A comparação das Equações 159 e 160 e a Figura 43 mostram que mesmo com os dados sendo representados pelo tipo real, quando o N_c for maior do que 3 a função da biblioteca do VHDL que calcula a matriz inversa pode apresentar defeitos, sendo que neste caso, apesar de os cálculos terem sido feitos corretamente já que os valores são iguais, todas as colunas pares da matriz ficam com o sinal invertido. Quando o parâmetro N_c posuir um valor elevado, pode ser necessário a correção do resultado da matriz inversa manualmente.

O código da biblioteca foi examinado, mas na análise do autor não foi encontrado nenhum problema com VHDL, sendo próvavel que seja apenas um erro de *software*.

Figura 43 – Matriz inversa na cosimulação



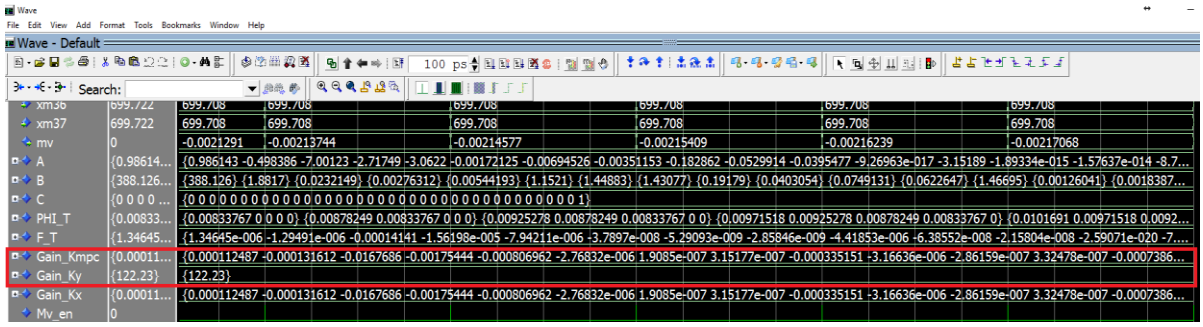
Fonte: Do autor

3.4 REATOR PAR

Nesta seção apresenta-se os resultados para a planta que é o maior desafio deste trabalho. O tempo de amostragem é de 1 segundo.

Para a primeira simulação, a configuração utilizada foi de $N_p = 20$, $N_c = 5$ e $r_w = 0$. Felizmente, para esta planta não foi necessário corrigir a matriz inversa como nos processos anteriores. Na Figura 44 apresenta-se um recorte da janela *Wave* do ModelSim PE para demonstrar que o cálculo dos ganhos K_{mpc} e K_y estão com o mesmo valor dos vistos no capítulo 2.3. Não foi implementado o observador de estado, sendo esta simulação com realimentação de todos os 37 estados.

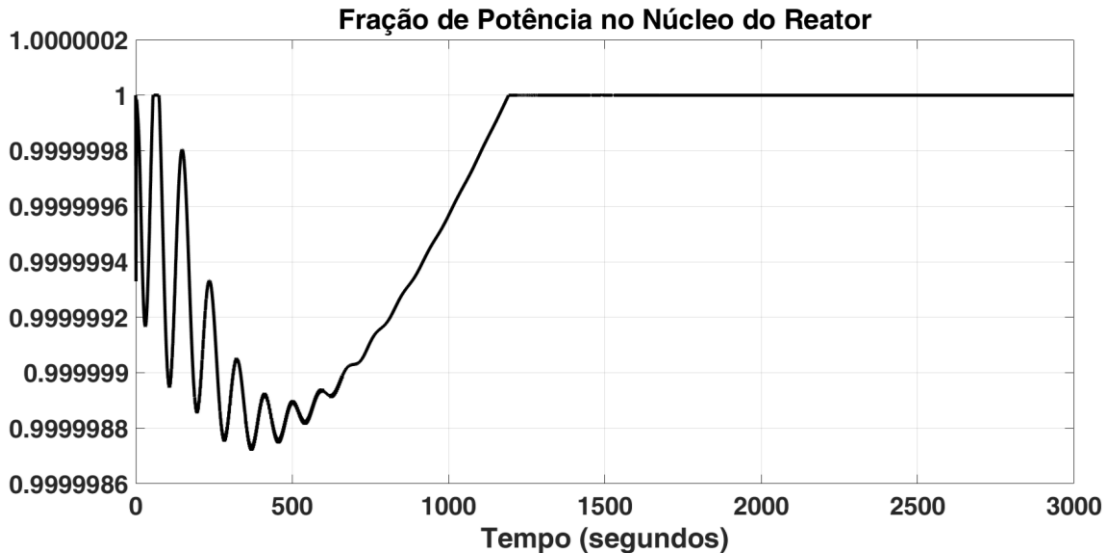
Figura 44 – Cálculo dos ganhos para a cosimulação



Fonte: Do autor

Na Figura 45 mostra-se o resultado da potência no experimento de cosimulação para o código em que as restrições informadas no capítulo 2 foram incorporadas.

Figura 45 – Resposta da potência para a cosimulação

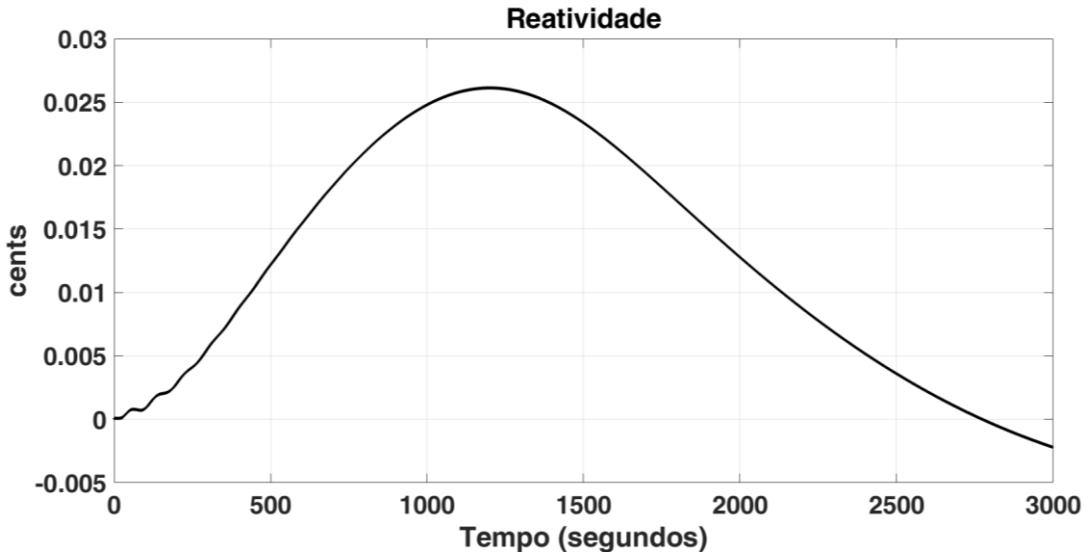


Fonte: Do autor

O controlador MPC projetado consegue manter a potência operando próximo da restrição e rastreando a referência, apesar de uma oscilação transitória. A restrição da saída foi respeitada sendo que em nenhum momento a potência ultrapassou o valor unitário.

A reatividade também tem a restrição respeitada conforme visto na Figura 46.

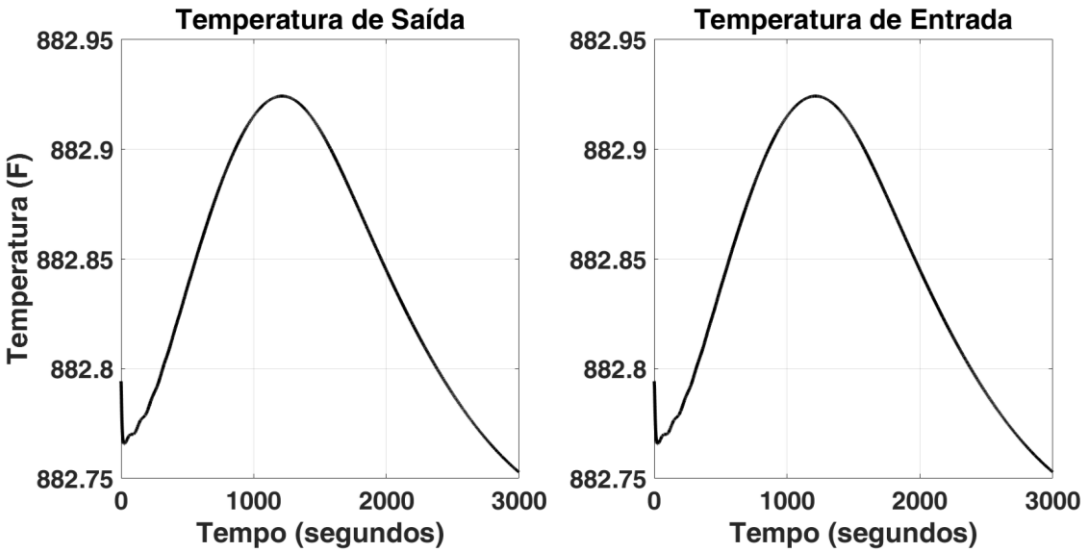
Figura 46 – Reatividade aplicada na cosimulação



Fonte: Do autor

A diferença de temperatura entre a entrada (*inlet*) e saída (*outlet*) do núcleo do reator não passa dos 2 F de acordo com o que mostra a Figura 47.

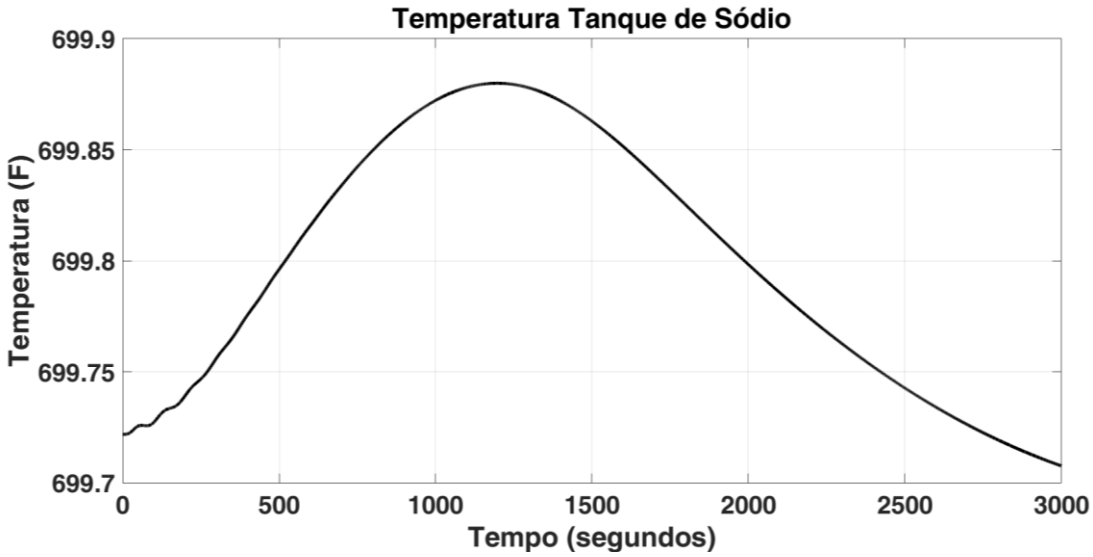
Figura 47 – Respostas das Temperaturas do reator na cosimulação



Fonte: Do autor

Os fluxos conseguem manter a temperatura do tanque de sódio abaixo do valor de 700 F como mostra a Figura 48.

Figura 48 – Resposta da Temperatura do tanque de sódio



Fonte: Do autor

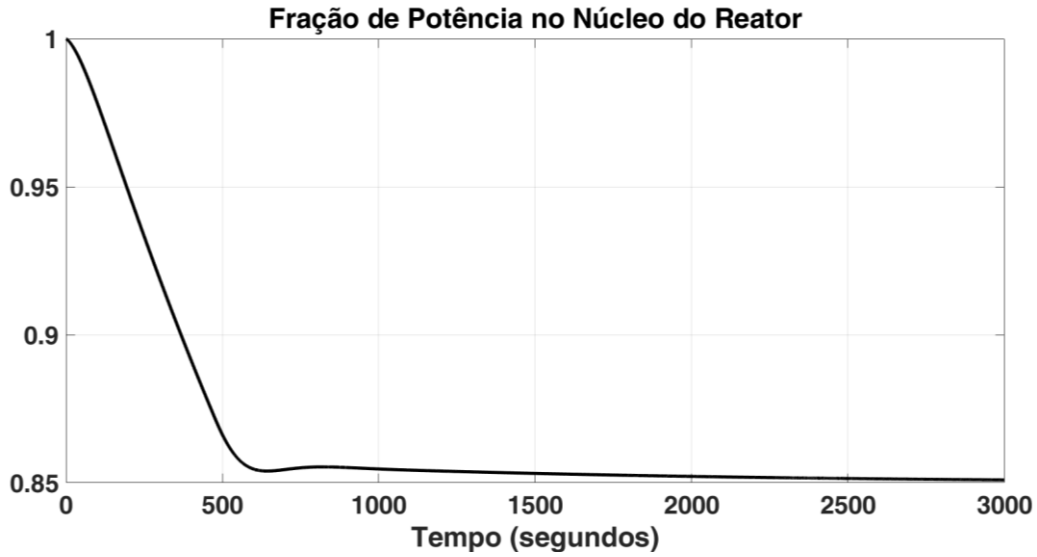
Foi proposto pelos pesquisadores do Tennessee dois cenários para experimentos de teste do MPC. No primeiro as configurações são mantidas iguais e a referência do sinal de potência é alterado. No segundo ensaio, a referência é mantida no seu valor original, assim como todas as configurações do reator, porém ocorre uma perturbação de 10% dos valores de regime estacionário nos fluxos primário e secundário.

3.4.1 CENÁRIO 1 – MUDANÇA DE REFERÊNCIA

O primeiro cenário para simulação do reator PAR envolve a mudança no valor da referência e a ação do MPC para efetuar o rastreamento. A referência (*set-point*) foi alterado para 0,85. O peso r_w teve que ser ajustado para 0,5 para que a restrição da reatividade fosse respeitada e os horizontes de controle e predição foram mantidos com os mesmos valores do ensaio anterior.

A resposta da potência para a cosimulação do código com restrições é vista na Figura 49.

Figura 49 – Cosimulação potência para mudança de referência

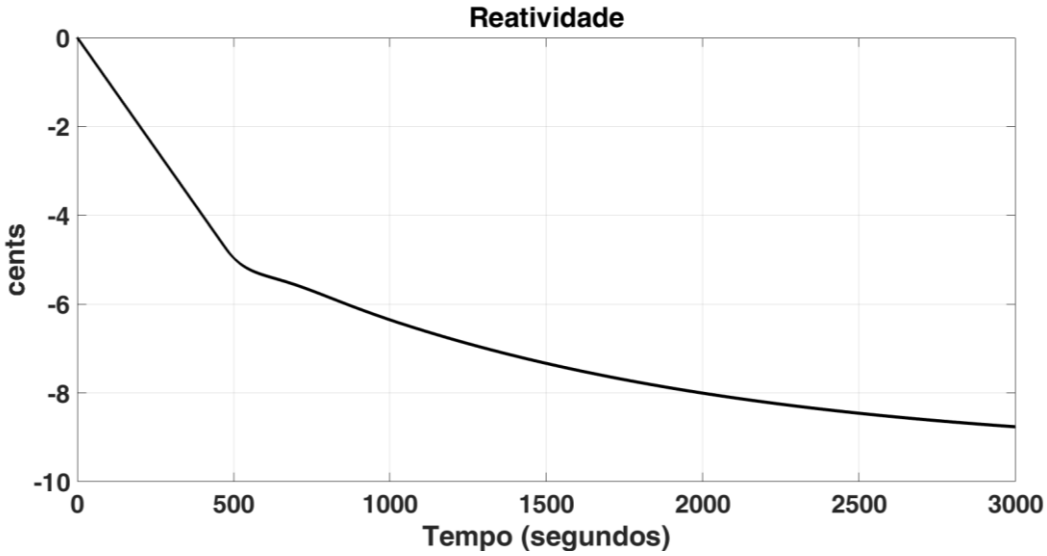


Fonte: Do autor

A potência atinge um valor próximo da referência em pouco mais de 500 segundos e demora para estabilizar sem erro de regime permanente.

Na Figura 50 apresenta-se o resultado da reatividade para este ensaio e a restrição imposta é respeitada.

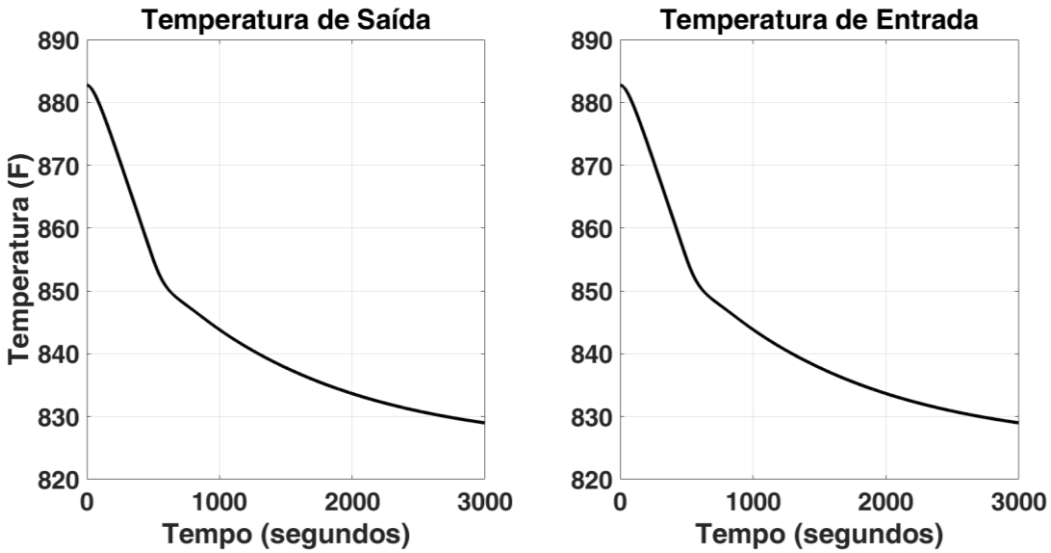
Figura 50 – Resposta da reatividade na cosimulação para mudança de referência



Fonte: Do autor

O mesmo ocorre para a restrição da diferença entre as temperaturas da entrada (*inlet*) e da saída (*outlet*) do reator. Na Figura 51 mostra-se o comparativo entre estas duas temperaturas.

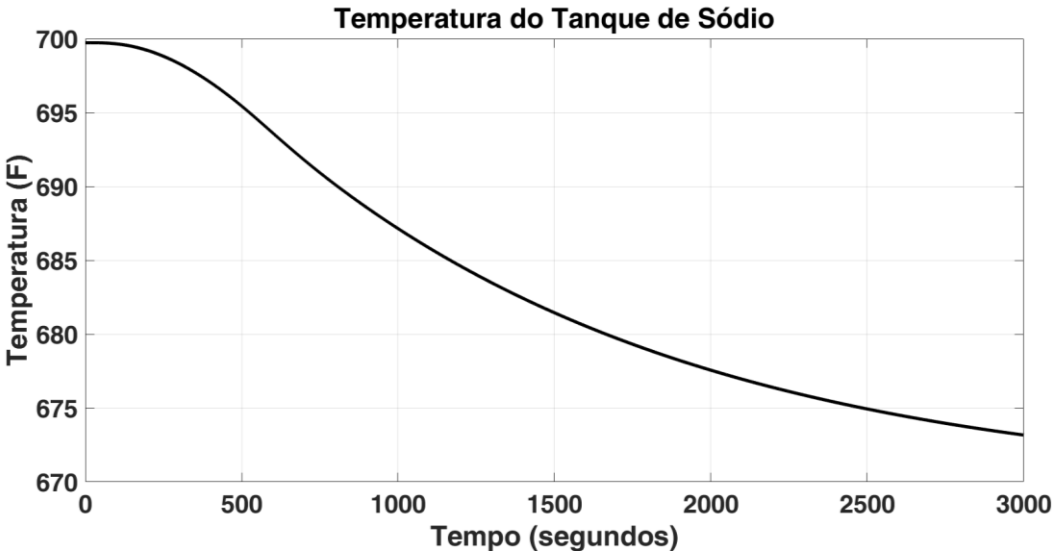
Figura 51 – Cosimulação temperaturas para mudança de referência



Fonte: Do autor

Para este cenário, a Temperatura do tanque de sódio não ultrapassou o limite de 700 F, como visto na Figura 52.

Figura 52 – Resposta da Temperatura do tanque de sódio



Fonte: Do autor

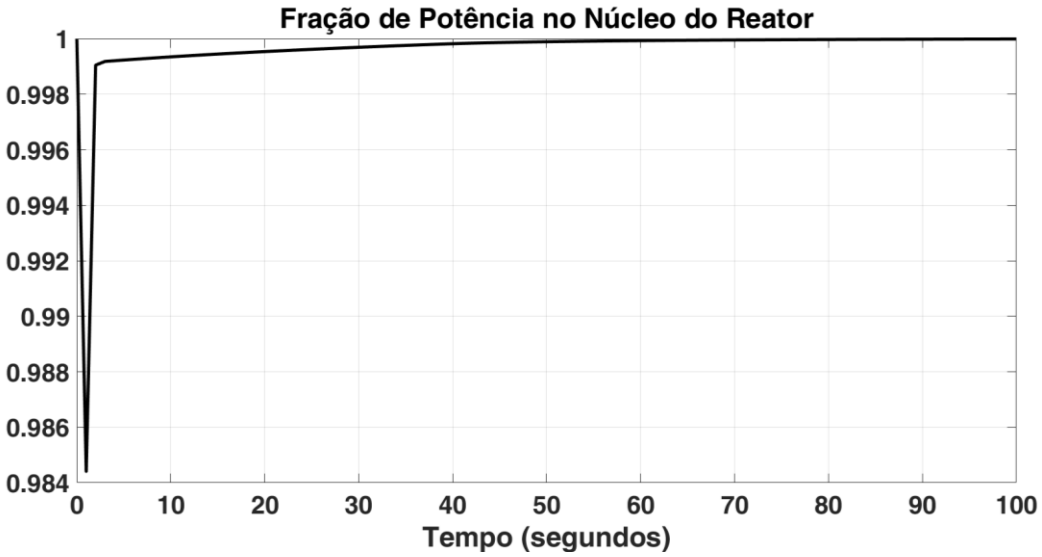
3.4.2 CENÁRIO 2 – PERTURBAÇÃO NOS FLUXOS

O segundo cenário para a simulação do reator PAR envolve perturbar os fluxos primário e secundário do líquido refrigerante. A sugestão descrita pelos pesquisadores do Tennessee indica que os fluxos devem começar com um valor 10% menor que o valor em que foi linearizada a planta.

Para esta simulação, os fluxos começam com 8100 gpm para o primário e 5301 gpm para o secundário, sendo que após 1 segundo os fluxos retornam para o valor original. A referência (*set-point*) nesta simulação é mantido em 1 e as configurações do reator são mantidas sem alteração.

Na Figura 53 mostra-se o resultado da potência para a cosimulação. O código utilizado para esta simulação é o que foi implementado as restrições. Neste caso, é possível observar que em apenas 100 segundos o MPC conseguiu atuar para que haja o rastreamento da potência.

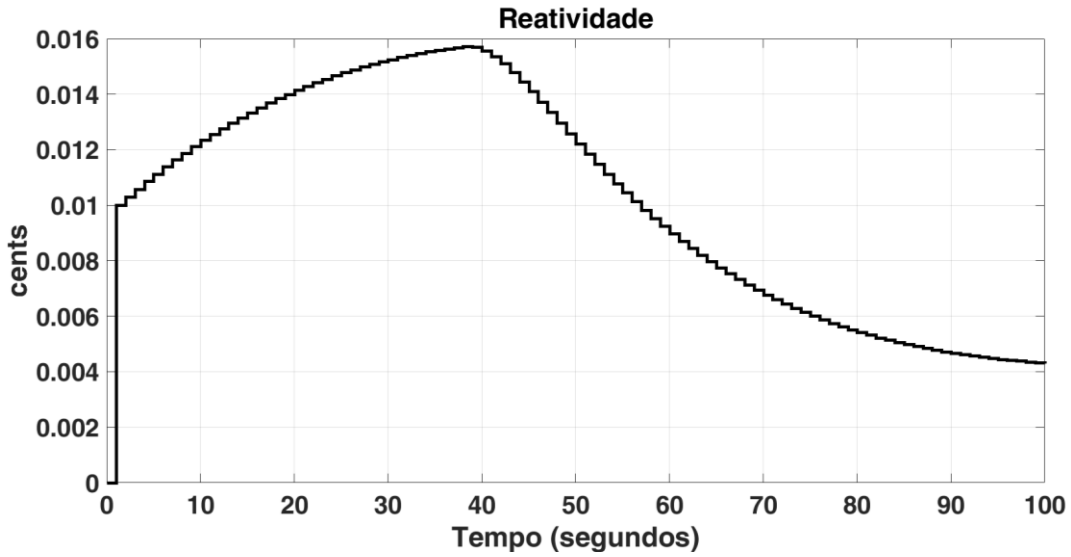
Figura 53 – Cosimulação da potência para pertubação nos fluxos



Fonte: Do autor

A Figura 54 mostra o resultado da reatividade para a cosimulação e é possível observar que a restrição da reatividade é respeitada.

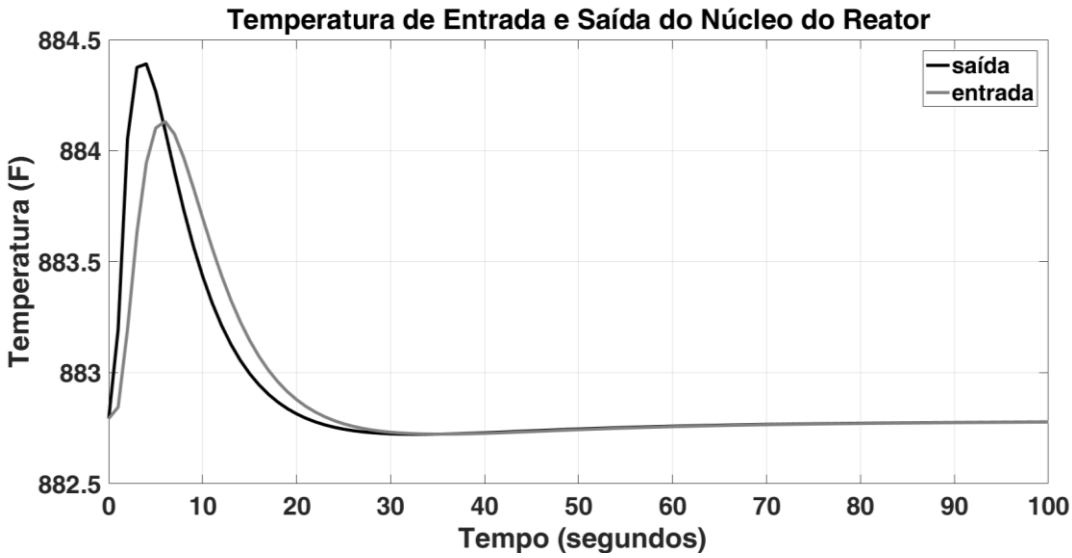
Figura 54 – Cosimulação da reatividade para pertubação nos fluxos



Fonte: Do autor

Na Figura 55 apresenta-se o resultado das temperaturas da entrada (*inlet*) e da saída (*outlet*) do núcleo do reator. É possível identificar que a diferença entre as duas temperaturas respeita a restrição e não ultrapassa o valor de 2 F entre as mesmas.

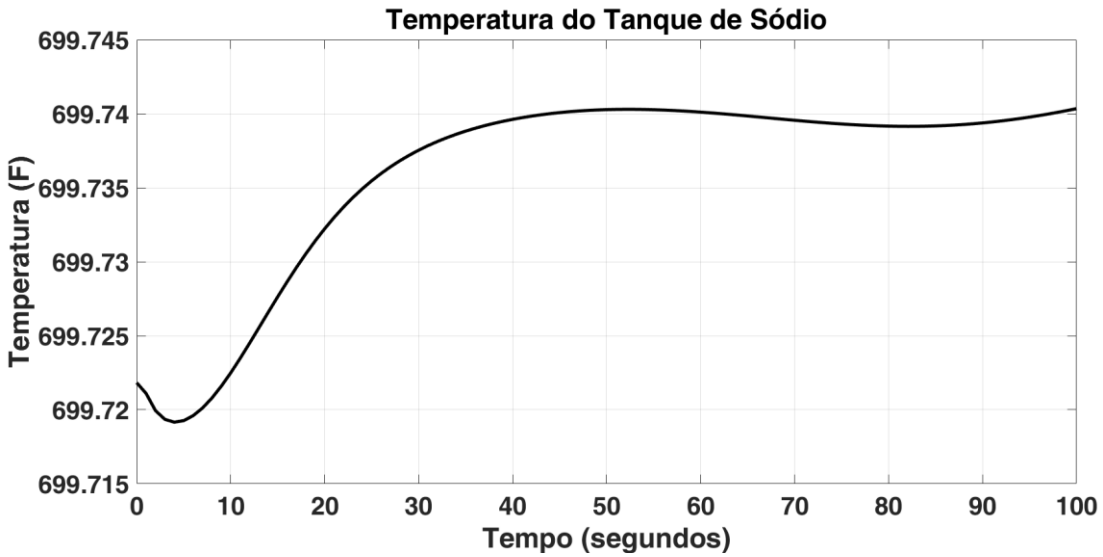
Figura 55 – Cosimulação da temperatura para pertubação nos fluxos



Fonte: Do autor

A perturbação nos fluxos não alterou de modo drástico a temperatura do tanque de sódio, como visto na Figura 56 a temperatura não ultrapassou 700 F.

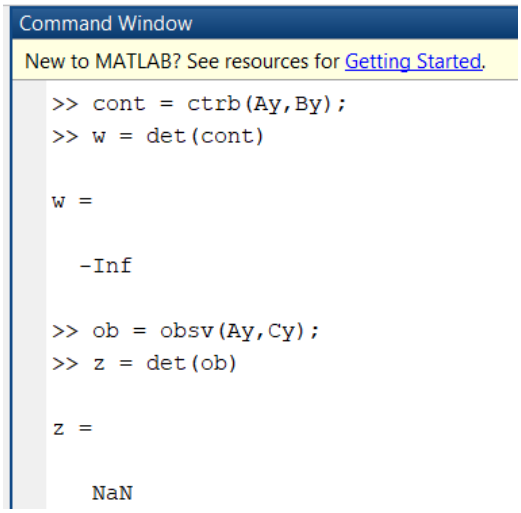
Figura 56 – Resposta da temperatura do tanque de sódio



Fonte: Do autor

O MPC generalizado com realimentação de estados conseguiu controlar o reator PAR em todos os experimentos. Na prática os estados do reator não são mensuráveis e por isso haveria a necessidade de projetar um estimador de estados para implementar o controle proposto na vida real. Porém, esta planta demonstrou um problema númerico em relação a observabilidade e deste modo não foi possível projetar um estimador de estados e repetir os experimentos. Na Figura 57 mostra-se o cálculo do determinante das matrizes de observabilidade e controlabilidade.

Figura 57 – Resposta de observabilidade e controlabilidade



Fonte: Do autor

Nota-se que por ter uma dimensão elevada o MATLAB considera a controlabilidade como infinito, ou seja, a planta é contrável, mas o mesmo não acontece com a observabilidade, em que a resposta do determinante é NaN (Not a number). Este resultado ocorreu pois como a dimensão das matrizes do modelo do reator PAR são de alto valor numérico, o MATLAB ao realizar as multiplicações para o cálculo do determinante substitui os altos valores por um valor simbólico infinito. Entretanto, uma observação mais detalhada da matriz de observabilidade do reator PAR mostra que a coluna 22 é composta apenas de elementos com valor numérico zero. É de conhecimento que uma matriz genérica, independente da sua dimensão, se possuir uma linha ou coluna inteiramente composta de zeros, o cálculo do seu determinante deve ser zero. O problema é que o MATLAB ao multiplicar o número zero por infinito retorna como resposta um *Not a Number*. Por este motivo, é possível afimar que a planta do reator PAR não é observável, apesar de o determinante da matriz de observabilidade ser *Not a Number*.

4 CONCLUSÕES

Com os resultados vistos neste trabalho, conclui-se que a estratégia de controle preditivo foi implementada, em linguagem VHDL, de forma correta e com resultados satisfatórios visto que as cosimulações e os testes em *FPGA-in-the-loop*, quando feitos, obtiveram uma resposta semelhante ao do *script* tutorial ou respeitaram as restrições de segurança da planta.

Nos processos controlados, foi utilizado a técnica do MPC em meio digital para plantas lineares com boa resposta dinâmica em malha aberta (plantas estáveis como o sistema turbina gerador) e para plantas reais (reator PAR e CSTR). As críticas recebidas pelo artigo submetido ao XXII Congresso Brasileiro de Automática questionavam se a metodologia proposta de implementação do MPC em meio digital conseguiria controlar plantas que precisam ser linearizadas ou ter uma ação de controle mais rápida, pois estas representariam um desafio maior do que o controle do sistema turbo-gerador. Os resultados do reator CSTR, PAR e o controle da máquina síncrona descritos neste trabalho mostraram que a metodologia proposta consegue controlar estes tipos de processos.

A grande contribuição deste trabalho é a possibilidade de uma nova implementação do MPC apenas com o uso de circuitos digitais simples, como um contador. Além disso, o código desenvolvido é dinâmico de forma que com poucas modificações, um MPC com uma nova configuração é implementado de forma rápida.

Infelizmente, nem todos os ensaios puderam ser realizados, como a cosimulação do MPC com funções de Laguerre devido a falta de recursos computacionais e o FPGA-in-the-loop do MPC com restrições, devido a falta de uma FPGA que suporte o circuito gerado. Porém, os resultados dos testes apresentados neste trabalho trazem um forte indício de que o controle poderia funcionar em meio físico.

Trabalhos futuros podem conter novos algoritmos de otimização como os vistos na seção da revisão bibliográfica; testes de consumo de potência com maior confiabilidade; desenvolvimento de novas metodologias de implementação do MPC e de execução dos testes, bem como para conferência de confiabilidade numérica.

5 REFERÊNCIAS BIBLIOGRÁFICAS

[1] Wang, L.; Model Predictive Control System Design and Implementation Using Matlab; 2009, Springer.

[2] Rossiter, J. A.; Model-based Predictive Control - A Practical Approach; CRC Press.

[3] Ricker, N. L.; Model-predictive control: state of the art. Proc. Fourth International Conference on Chemical Process Control, Padre Island, Texas, 271–296, 1991.

[4] Rossiter, J. A.; Lectures and Resources in Modelling and Control: Introduction to Predictive Control for Beginners. Disponível em: http://controleducation.group.shef.ac.uk/indexwebbook.html.

[5] Nise, N. S. (2002). Engenharia de Sistemas de Controle. 3ª edição. Rio de Janeiro: LTC.

[6] Sakamoto, Suzana Sayuri. *Projeto de Sistemas de Controle Preditivo: Estudo de Caso sobre um Controlador de Nível de Insulina.* 2018. Trabalho de Conclusão de Curso – UFMS, Campo Grande, 2018.

[7] Sandre-Hernandez, O., Rangel-Magdaleno, J. J., Tlelo-Cuatle, E., Morales-Caporal, R. (2015). FPGA-based Delay Compensation on Model Predictive Control for a PM Synchronous Machine. IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC).

[8] A. Suardi, E. C. Kerrigan, G. A. Constantinides, and R. Findeisen, "Fast FPGA prototyping toolbox for embedded optimization.," in Proc. of the European Control Conference, 2015, pp. 2589-2594.

[9] P. Zometa, M. Kögel, and R. Findeisen, "muAO-MPC: A free code generation tool for embedded real-time linear model predictive control," in Proc. American Control Conf., 2013, pp. 5320-5325.

[10] MathWorks. (2014, Oct.) HDL coder. [Online]. Disponível em: http://www.mathworks.co.uk/products/hdl-coder

[11] National Instruments. (2015, Mar.) National instruments website. [Online]. Disponível em: http://www.ni.com/

[12] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari," Embedded Predictive Control on an FPGA using the Fast Gradient Method," European Control Conference, pp. 3614-3620, 2013.

[13] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari, "Embedded online optimization for model predictive control at megahertz rates," IEEE Transactions on Automatic Control, pp. 3238-3251, 2014.

[14] E. Hartley, J. Jerez, A. Suardi, J. Maciejowski, E. Kerrigan, and G. Constantinides, “Predictive control using an FPGA with application to aircraft control,” IEEE Transactions on Control Systems Technology, vol. 22, no. 3, pp. 1006–1017, May 2014.

[15] X. Chen and X. Wu, “Design and implementation of model predictive control algorithms for small satellite three-axis stabilization,” in Proc.Int. Conf. Inf. Autom., Jun. 2011, pp. 666–671.

[16] S. Lucia, D. Navarro, O. Lucía, P. Zometa, R. Findeisen, “Optimized FPGA Implementation of Model Predictive Control for Embedded Systems Using High-Level Synthesis Tool”, in IEEE Transactions on Industrial Informatics, pp. 137-145, 2017.

[17] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," Optimization and Engineering, vol. 13, pp. 1-27,2012.

[18] E. Chu, N. Parikh, A. Domahidi, and S. Boyd, "Code generation for embedded second-order cone programming," in Proc. European Control Conf., 2013, pp. 1547-1552.

[19] FalcOpt (First-order Algorithm via Linearization of Constraints for OPTimization) - Solving efficient nonlinear Model Predictive Control problems. [Online]. Disponível em: https://github.com/torrisig/FalcOpt/wiki

[20] Y. Xu, D. Li, Y. Xi, J. Lan, and T. Jiang, “An Improved Predictive Controller on the FPGA by Hardware Matrix Inversion,” IEEE Transactions on Industrial Electronics, Vol. 65, no. 9, September, 2018.

[21] B. Czakó, J. Sápi, L. Kovács, “Model-based Optimal Control Method for Cancer Treatment Using Model Predictive Control and Robust FixedPoint Method,” International Conference on Intelligent Engineering Systems (INES 2017), pp. 271–276, Cyprus, 2017.

[22] F. He, L. Bi, Y. Lu, H. Li, and L. Wang, “Model Predictive Control for a Brain-controlled Mobile Robot,” IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 3184-3188, Canada, 2017.

[23] E. Hasan, R. Ibrahim, K. Bingi, S. M. Hassan, and S. F. Gilani, “Real-Time Model Predictive Control for Nonlinear Gas Pressure Process Plant,” IEEE International

Conference on Signal and Image Processing Applications (IEEE ICSIPA 2017), pp. 299-303, Malaysia, 2017.

[24] P. Balram, O. Carlson, L. A. Tuan, “Demonstration of Voltage Control in a Real Distribution System Using Model Predictive Control,” IET Generation, Transmission & Distribution, pp. 3922-3929, 2017.

[25] Brandão, A. S. M., Lima, D. M., Filho, M. V. A. da C., Normey-Rico, J. E. A Comparative Study on Embedded MPC for Industrial Processes. XXII Congresso Brasileiro de Automática, 2018.

[26] Pinheiro, T. C. F., Araújo, M. da S., Silveira, A. da S., Dutra, B. G. Controle Preditivo Multivariável Usando Funções de Laguerre. XXII Congresso Brasileiro de Automática, 2018.

[27] Mercaldi, H. V., Peñaloza, E. A. G., Oliveira, V. A., Cruvinel, P. E. Controlador Preditivo com Restrições Embarcado. XXII Congresso Brasileiro de Automática, 2018.

[28] García, R. C., Batista, E. A., Andrea, C. Q., Grassi, M. A. S. FPGA-in-the-loop and Cosimulation Tests of a Modified MIMO-MPC Speed Control of PMSM. 2018. A ser publicado.

[29] MathWorks. (2016, Oct.) System Identification Toolbox. [Online]. Disponível em: https://www.mathworks.com/products/sysid.html

[30] Seborg, D. E., T. F. Edgar, and D. A. Mellichamp, Process Dynamics and Control, 2nd Edition, Wiley, 2004, pp. 34–36 and 94–95.

[31] Batista, E. A. Projeto de um Controlador Preditivo Aplicadoaum Protótipo de Reator Nuclear Avançadoe Testes Realizadosem FPGA-IN-THE-LOOP. Relatório de Estágio Pós Doutoral.

[32] Coble, J. and Liu, X. Development of a Prototypical Advanced Reactor Model to Support the Enhanced Risk Monitor. Knoxville, TN, University of Tennessee.

[33] Berkan, R. C. and B. R. Upadhyaya (1988). Dynamic Modeling of EBR-II for Simulation and Control. Knoxville, TN, University of Tennessee. UTNE/BRU88-1.

[34] TOCCI, Ronald J.; WIDMER, Neal S.; MOSS, Gregory L. Sistemas digitais: princípios e aplicações. 11. ed. São Paulo, SP: Pearson, 2014. xx, 817 p. ISBN 9788576059226.

[35] PowerPlay Power Analysis [Online]. Disponível em: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/qts/qts_qii53013.pdf

ANEXO I

Neste anexo são apresentados os códigos dos algoritmos que implementaram o MPC em linguagem VHDL. Todos os códigos são do projeto MPC para a planta do sistema turbina-gerador, sendo o primeiro do MPC generalizado com os dados todos representados com o tipo real para realizar a cosimulação e o segundo do MPC generalizado com os dados representados pelo tipo ponto fixo para os ensaios de FPGA-in-the-loop.

MPC generalizado sem observador de estados - Cosimulação

```
library ieee;
library ieee_proposed;

use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

use ieee_proposed.real_matrix_pkg.all;

entity Exemplo1 is
generic (
NP : integer := 2; -- Horizonte de predição
NC : integer := 1
);
port( clk : in std_logic;
rst : in std_logic;
Yout : in real; -- Recebe a saída da planta Y
rki : in real; -- Porta para o sinal de referência (setpoint)
xm1 : in real; -- realimentação de estado X1
xm2 : in real; -- realimentação de estado X2
xm3 : in real; -- realimentação de estado X3
Mv : out real);-- Lei de controle - saída do MPC

end entity;

architecture horizon of Exemplo1 is

signal A : real_matrix (0 to 3, 0 to 3) := zeros(4,4); -- Matriz aumentada A
signal B : real_matrix (0 to 3, 0 to 0) := zeros(4,1); -- Matriz aumentada B
signal C : real_matrix (0 to 0, 0 to 3) := zeros(1,4); -- Matriz aumentada C

signal Gain_Kmpc : real_matrix (0 to 0, 0 to (A'length(2)-1)) := zeros(1,A'length(2)); --
Ganho Kmpc
signal Gain_Ky : real_matrix (0 to 0, 0 to 0) := zeros(1,1); -- Ganho Ky
signal Gain_Kx : real_matrix (0 to 0, 0 to (A'length(2)-2)) := zeros(1,A'length(2)-1); --
Ganho Ky
--=============================================================
signal Mv_en : std_logic;

signal XkMenosHum_sinal : real_matrix (0 to 2, 0 to 0):= zeros(3,1);
signal Uk_sinal : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
signal Uk_antigo : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
```

```
begin

MPC_gain : process(rki)

-- Matrizes do modelo
variable am : real_matrix (0 to 2, 0 to 2):=zeros(3,3);
variable cm : real_matrix (0 to 0, 0 to 2):=zeros(1,3);
variable bm : real_matrix (0 to 2, 0 to 0):=zeros(3,1);

-- Matrizes Aumentadas
variable Ae : real_matrix (0 to (Cm'length(1)+Cm'length(2)-1), 0 to (Cm'length(1)+Cm'
length(2)-1)):=eye((Cm'length(1)+Cm'length(2)),(Cm'length(1)+Cm'length(2)));

variable Ce : real_matrix (0 to (Cm'length(1)-1), 0 to (Cm'length(1)+Cm'length(2)-1)):=
zeros((Cm'length(1)),(Cm'length(1)+Cm'length(2)));

variable Be : real_matrix (0 to (Cm'length(1)+Cm'length(2)-1), 0 to (Bm'length(2)-1)):=
zeros((Cm'length(1)+Cm'length(2)),Bm'length(2));

variable um_mat: real_matrix (0 to (Cm'length(1)-1), 0 to (Cm'length(1)-1)):= eye(Cm'
length(1),Cm'length(1));
variable so_zero: real_matrix (0 to 0, 0 to 0):= zeros(1,1);

--Matrizes F e Phi
variable Fv : real_matrix (0 to NP-1, 0 to (A'length(2)-1)) := zeros(NP,A'length(2)); --
Matrix F
variable Phiv : real_matrix (0 to NP-1, 0 to NC-1) := zeros(NP,NC); -- Matrix Phi ?

------------------------------------------------------------
-- Phi transposto
variable PhiT5 : real_matrix(0 to NC-1, 0 to NP-1) := zeros(NC,NP);
------====================----------------====================
--PhiTranspose*Phi, Inversa
variable PhiT_Phi : real_matrix(0 to NC-1, 0 to NC-1) := zeros(NC,NC);
variable PhiTPhi_Rbarra : real_matrix(0 to NC-1, 0 to NC-1):= zeros(NC,NC) ;
variable InvPhiTPhi_Rbarra : real_matrix(0 to NC-1, 0 to NC-1):= zeros(NC,NC);
--=========================================================
--Phi tranposto * F
variable PhiT_F : real_matrix(0 to NC-1, 0 to A'length(2)-1):= zeros(NC,A'length(2));

--============================================================
-- Matriz Rbarra
variable RbNc : real_matrix (0 to NC-1, 0 to NC-1) := eye(NC,NC);
variable RbarraW : real_matrix (0 to NC-1, 0 to NC-1) := zeros(NC,NC);
variable peso : real;
--============================================================
-- Calculo dos ganhos
variable vectorHumZero : real_matrix (0 to 0, 0 to NC-1) := zeros(1,NC);
variable vectorHum : real_matrix (0 to NP-1, 0 to 0) := ones(NP,1);
variable Kmpc : real_matrix (0 to 0, 0 to (A'length(2)-1)):= zeros(1,A'length(2));
variable Kx : real_matrix (0 to 0, 0 to (A'length(2)-2)):= zeros(1,A'length(2)-1);
variable Ky : real_matrix (0 to 0, 0 to 0):= zeros(1,1);

begin

am := ((-0.1159, -1.3371, -0.5785), (0.0193, 0.1541, -0.5080), (0.0169, 0.2563, 0.8822));

bm(0,0) := 0.0193;
```

```
bm(1,0) := 0.0169;

bm(2,0) := 0.0039;

cm(0,0) := 0.0;
cm(0,1) := 0.0;
cm(0,2) := 500.0;

peso := 0.0; -- rw

vectorHumZero(0,0) := 1.0;

--Matriz A
buildmatrix(Am, Ae, 0,0);
buildmatrix(Cm*Am, Ae, (Am'length(1)),0);

--Matriz B
buildmatrix(Bm, Be, 0,0);
buildmatrix(Cm*Bm, Be, Bm'length(1),0);
--Matriz C
buildmatrix(um_mat, Ce, 0,Am'length(2));

Matriz_F : for n in 0 to NP-1 loop
buildmatrix(Ce*Ae**(n+1), Fv, n, 0);
end loop;

Matriz_Phi : for linha in 0 to NP-1 loop
for coluna in 0 to (NC - 1) loop
if linha = coluna then
buildmatrix(Ce*Be, Phiv, linha, coluna);
elsif linha < coluna then
buildmatrix(so_zero, Phiv, linha, coluna);
elsif linha > coluna then
if coluna = 0 then
buildmatrix(Ce*(Ae**linha)*Be, Phiv, linha, coluna);
else
buildmatrix(Ce*(Ae**(linha-coluna))*Be, Phiv, linha, coluna);
end if;
end if;
end loop;
end loop;

--=================================================
-- Phi transposto
PhiT5 := transpose(Phiv);
--=================================================
--PhiTPhi, ou seja, Phitransposto vezes Phi

PhiT_Phi := PhiT5*Phiv;
PhiTPhi_Rbarra := PhiT_Phi + RbarraW;
InvPhiTPhi_Rbarra := inv(PhiTPhi_Rbarra);
--=====================================================
-- Matriz Phi Transposto * F
PhiT_F := PhiT5*Fv;
 --=====================================================
-- Cálculo do Ganho Kmpc
Kmpc := vectorHumZero*InvPhiTPhi_Rbarra*PhiT_F;
--=====================================================
-- Ganho Ky
Ky := vectorHumZero*InvPhiTPhi_Rbarra*PhiT5*vectorHum;
```

```
--=====================================================
-- Ganho Kk
kx(0,0) := Kmpc(0,0);
kx(0,1) := Kmpc(0,1);
kx(0,2) := Kmpc(0,2);

--=========================================================
Gain_Kmpc <= Kmpc;
Gain_Kx <= Kx;
Gain_Ky <= Ky;
A <= Ae;
B <= Be;
C <= Ce;

end process;

Clock_cont: process(clk, rst)

variable cnt : integer := 0;
constant Tempo_amostragem : integer := 442; -- clock 1kHz

variable XkMenosHum : real_matrix (0 to 2, 0 to 0):= zeros(3,1); -- X(k-1)

begin
if (rst = '0') then
cnt := 0;
Mv_en <= '0';

XkMenosHum(0,0) := 0.0;
XkMenosHum(1,0) := 0.0;
XkMenosHum(2,0) := 0.0;

elsif (clk'event and clk = '1') then
cnt := cnt + 1;

if cnt = Tempo_amostragem then
cnt := 0;
Mv_en <= '1';
else
Mv_en <= '0';
end if;

if cnt = (Tempo_amostragem - 1) then
XkMenosHum(0,0) := xm1;
XkMenosHum(1,0) := xm2;
XkMenosHum(2,0) := xm3;

Uk_sinal <= Uk_antigo;
end if;
XkMenosHum_sinal <= XkMenosHum;

end if;
end process;
--=================================================
Control_law: process(rst, xm1,xm2,xm3, Yout)

variable Xkm : real_matrix (0 to 2, 0 to 0):= zeros(3,1); -- X(K)
variable DeltaXk : real_matrix (0 to 2, 0 to 0):= zeros(3,1); -- delta_x(k) = X(k) - X(K-1)
variable KxDeltaXk : real_matrix (0 to 0, 0 to 0):= zeros(1,1);--Kx*delta_x(k)
```

```
variable vectorDeltaU : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
variable Error : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
variable Uk : real_matrix (0 to 0, 0 to 0):= zeros(1,1);

begin

if (rst = '0') then

Xkm(0,0) := 0.0;
Xkm(1,0) := 0.0;
Xkm(2,0) := 0.0;

Mv <= 0.0;

elsif (xm1 = 0.0 and xm2 = 0.0 and xm3 = 0.0 and Yout = 0.0) then
Uk := Gain_ky*rki;
Mv <= Uk(0,0);
Uk_antigo <= Uk;
elsif (Mv_en = '1') then

-- Recebe a realimentação da planta pelo espaço de estado
Xkm(0,0) := xm1;
Xkm(1,0) := xm2;
Xkm(2,0) := xm3;

DeltaXk := Xkm - XkMenosHum_sinal;

KxDeltaXk := -(Gain_Kx)*DeltaXk; -- -Kx*[X(k) - X(k-1)]

Error := -Gain_Ky*(Yout - rki);

vectorDeltaU := Error + KxDeltaXk;

-- integrador
Uk := Uk_sinal + vectorDeltaU;

--============================================================
end if;
Uk_antigo <= Uk;
Mv <= Uk(0,0);
end process;
end architecture;
```

MPC generalizado sem observador de estados - FPGA-in-the-loop

```
library ieee;
library ieee_proposed;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
use ieee.fixed_float_types.all;
use ieee.fixed_pkg.all;
use ieee_proposed.fixed_matrix_pkg.all;
entity Exemplo1 is
generic (
NP : integer := 2;
NC : integer := 1
);
port( clk : in std_logic;
```

```
rst : in std_logic;
Yout : in std_logic_vector(31 downto 0); -- Recebe a saída da planta Y
rki : in std_logic_vector(31 downto 0); -- Porta para o sinal de referência (setpoint)
xm1 : in std_logic_vector(31 downto 0); -- realimentação de estado X1
xm2 : in std_logic_vector(31 downto 0); -- realimentação de estado X2
xm3 : in std_logic_vector(31 downto 0); -- realimentação de estado X3
Mv : out std_logic_vector(31 downto 0));-- Lei de controle - saída do MPC
end entity;
architecture horizon of Exemplo1 is
signal A : sfixed_matrix (0 to 3, 0 to 3) := zeros(4,4); -- Matriz aumentada A
signal B : sfixed_matrix (0 to 3, 0 to 0) := zeros(4,1); -- Matriz aumentada B
signal C : sfixed_matrix (0 to 0, 0 to 3) := zeros(1,4); -- Matriz aumentada C
signal Gain_Ky : sfixed_matrix (0 to 0, 0 to 0) := zeros(1,1); -- Ganho Ky
signal Gain_Kx : sfixed_matrix (0 to 0, 0 to (A'length(2)-2)) := zeros(1,A'length(2)-1); -- Ganho
Kx
--================================================================
signal XkMenosHum_sinal : sfixed_matrix (0 to 2, 0 to 0):= zeros(3,1);
signal Uk_sinal : sfixed_matrix (0 to 0, 0 to 0):= zeros(1,1);
signal Uk_antigo : sfixed_matrix (0 to 0, 0 to 0):= zeros(1,1);
--================================================================
signal Mv_en : std_logic;

begin

MPC_gain : process(rki)
-- Matrizes do modelo
variable am : sfixed_matrix (0 to 2, 0 to 2):=zeros(3,3);
variable cm : sfixed_matrix (0 to 0, 0 to 2):=zeros(1,3);
variable bm : sfixed_matrix (0 to 2, 0 to 0):=zeros(3,1);
-- Matrizes Aumentadas
variable Ae : sfixed_matrix (0 to (Cm'length(1)+Cm'length(2)-1), 0 to (Cm'length(1)+Cm'length(2)-
1)):=eye((Cm'length(1)+Cm'length(2)),(Cm'length(1)+Cm'length(2)));
variable Ce : sfixed_matrix (0 to (Cm'length(1)-1), 0 to (Cm'length(1)+Cm'length(2)-1)):=zeros((
Cm'length(1)),(Cm'length(1)+Cm'length(2)));
variable Be : sfixed_matrix (0 to (Cm'length(1)+Cm'length(2)-1), 0 to (Bm'length(2)-1)):=zeros((
Cm'length(1)+Cm'length(2)),Bm'length(2));
variable um_mat: sfixed_matrix (0 to (Cm'length(1)-1), 0 to (Cm'length(1)-1)):= eye(Cm'length(1),
Cm'length(1));
variable so_zero: sfixed_matrix (0 to 0, 0 to 0):= zeros(1,1);
--Matrizes F e Phi
variable Fv : sfixed_matrix (0 to NP-1, 0 to (A'length(2)-1)) := zeros(NP,A'length(2)); --
Matrix F
variable Phiv : sfixed_matrix (0 to NP-1, 0 to NC-1) := zeros(NP,NC); -- Matrix Phi ?
-------------------------------------------------------------
-- Phi transposto
variable PhiT5 : sfixed_matrix(0 to NC-1, 0 to NP-1) := zeros(NC,NP);
------====================----------==========================
--PhiTransposto*Phi, Inversa
variable PhiT_Phi : sfixed_matrix(0 to NC-1, 0 to NC-1) := zeros(NC,NC);
variable PhiTPhi_Rbarra : sfixed_matrix(0 to NC-1, 0 to NC-1):= zeros(NC,NC) ;
variable InvPhiTPhi_Rbarra : sfixed_matrix(0 to NC-1, 0 to NC-1):= zeros(NC,NC);
--=====================================================
--Phi tranposto * F
variable PhiT_F : sfixed_matrix(0 to NC-1, 0 to A'length(2)-1):= zeros(NC,A'length(2));
--=============================================================
-- Matriz Rbarra
variable RbNc : sfixed_matrix (0 to NC-1, 0 to NC-1) := eye(NC,NC);
variable RbarraW : sfixed_matrix (0 to NC-1, 0 to NC-1) := zeros(NC,NC);
variable peso : sfixed(15 downto -16);
--=============================================================
```

```
-- Cálculo Ky
variable vectorHumZero : sfixed_matrix (0 to 0, 0 to NC-1) := zeros(1,NC);
variable vectorHum : sfixed_matrix (0 to NP-1, 0 to 0) := ones(NP,1);
variable Kmpc : sfixed_matrix (0 to 0, 0 to 3):= zeros(1,4);
variable Kx : sfixed_matrix (0 to 0, 0 to 2):= zeros(1,3);
variable Ky : sfixed_matrix (0 to 0, 0 to 0):= zeros(1,1);
begin
am := ((to_sfixed(-0.115892273831871,15,-16), to_sfixed(-1.33711207804134,15,-16), to_sfixed(-
0.578470168753942,15,-16)), (to_sfixed(0.0192823389584647,15,-16),
to_sfixed(0.154060471586635,15
,-16), to_sfixed(-0.507971502827361,15,-16)), (to_sfixed(0.0169323834275787,15,-16), to_sfixed(
0.256335706944567,15,-16), to_sfixed(0.882152958972519,15,-16)));
bm(0,0) := to_sfixed(0.0192823389584647,15,-16);
bm(1,0) := to_sfixed(0.0169323834275787,15,-16);
bm(2,0) := to_sfixed(0.00392823470091603,15,-16);
cm(0,0) := to_sfixed(0.0,15,-16);
cm(0,1) := to_sfixed(0.0,15,-16);
cm(0,2) := to_sfixed(500.0,15,-16);
peso := to_sfixed(0.0,15,-16); -- rw
vectorHumZero(0,0) := to_sfixed(1.0,15,-16);
--Matriz A
buildmatrix(Am, Ae, 0,0);
buildmatrix(Cm*Am, Ae, (Am'length(1)),0);
--Matriz B
buildmatrix(Bm, Be, 0,0);
buildmatrix(Cm*Bm, Be, Bm'length(1),0);
--Matriz C
buildmatrix(um_mat, Ce, 0,Am'length(2));
Matriz_F : for n in 0 to NP-1 loop
buildmatrix(Ce*(Ae**(n+1)), Fv, n, 0);
end loop;
Matriz_Phi : for linha in 0 to NP-1 loop
for coluna in 0 to (NC - 1) loop
if linha = coluna then
buildmatrix(Ce*Be, Phiv, linha, coluna);
elsif linha < coluna then
buildmatrix(so_zero, Phiv, linha, coluna);
elsif linha > coluna then
if coluna = 0 then
buildmatrix(Ce*(Ae**linha)*Be, Phiv, linha, coluna);
else
buildmatrix(Ce*(Ae**(linha-coluna))*Be, Phiv, linha, coluna);
end if;
end if;
end loop;
end loop;
--================================================
-- Phi transposto
PhiT5 := transpose(Phiv);
--================================================
--Rbarra
RbarraW := peso*(RbNc);
--================================================
--PhiTPhi, ou seja, Phitransposto vezes Phi
PhiT_Phi := PhiT5*Phiv;
PhiTPhi_Rbarra := PhiT_Phi + RbarraW;
InvPhiTPhi_Rbarra := inv(PhiTPhi_Rbarra);
--================================================
-- Matrix Phi Transpose * F
PhiT_F := PhiT5*Fv;
```

```
--=====================================================
-- Calculate of the Gain Kmpc
Kmpc := vectorHumZero*InvPhiTPhi_Rbarra*PhiT_F;
--=====================================================
-- Gain Ky
Ky := vectorHumZero*InvPhiTPhi_Rbarra*PhiT5*vectorHum;
--=====================================================
-- Gain Kk
Kx(0,0) := Kmpc(0,0);
kx(0,1) := Kmpc(0,1);
kx(0,2) := Kmpc(0,2);
--=========================================================
Gain_Kx <= Kx;
Gain_Ky <= Ky;
end process;

Clock_cont: process(clk, rst)

variable cnt : integer := 0;
constant Tempo_amostragem : integer := 442; -- clock 1kHz
variable XkMenosHum : sfixed_matrix (0 to 2, 0 to 0):= zeros(3,1); -- X(k-1)
begin
if (rst = '0') then
cnt := 0;
Mv_en <= '0';
XkMenosHum(0,0) := to_sfixed(0.0,15,-16);
XkMenosHum(1,0) := to_sfixed(0.0,15,-16);
XkMenosHum(2,0) := to_sfixed(0.0,15,-16);
elsif (clk'event and clk = '1') then
cnt := cnt + 1;
if cnt = Tempo_amostragem then
cnt := 0;
Mv_en <= '1';
else
Mv_en <= '0';
end if;
if cnt = (Tempo_amostragem - 1) then
XkMenosHum(0,0) := to_sfixed(xm1,15,-16);
XkMenosHum(1,0) := to_sfixed(xm2,15,-16);
XkMenosHum(2,0) := to_sfixed(xm3,15,-16);
Uk_sinal <= Uk_antigo;
end if;
XkMenosHum_sinal <= XkMenosHum;
end if;
end process;
--=====================================================

Control_law: process(rst, xm1,xm2,xm3, Yout, rki)

variable Xkm : sfixed_matrix (0 to 2, 0 to 0):= zeros(3,1); -- X(K)
variable DeltaXk : sfixed_matrix (0 to 2, 0 to 0):= zeros(3,1); -- delta_x(k) = X(k) - X(K-1)
variable KxDeltaXk : sfixed_matrix (0 to 0, 0 to 0):= zeros(1,1);--Kx*delta_x(k)
variable vectorDeltaU : sfixed_matrix (0 to 0, 0 to 0):= zeros(1,1);
variable Error : sfixed_matrix (0 to 0, 0 to 0):= zeros(1,1);
variable Uk : sfixed_matrix (0 to 0, 0 to 0):= zeros(1,1);
begin
if (rst = '0') then
Xkm(0,0) := to_sfixed(xm1,15,-16);
Xkm(1,0) := to_sfixed(xm2,15,-16);
Xkm(2,0) := to_sfixed(xm3,15,-16);
```

```
DeltaXk(0,0) := to_sfixed(0.0,15,-16);
DeltaXk(1,0) := to_sfixed(0.0,15,-16);
DeltaXk(2,0) := to_sfixed(0.0,15,-16);
KxDeltaXk(0,0) := to_sfixed(0.0,15,-16);
vectorDeltaU(0,0) := to_sfixed(0.0,15,-16);
Error(0,0) := to_sfixed(0.0,15,-16);
Mv <= to_slv(to_sfixed(0.0,15,-16));
elsif (xm1 = "00000000000000000000000000000000" and xm2 =
"00000000000000000000000000000000" and
xm3 = "00000000000000000000000000000000" and Yout =
"00000000000000000000000000000000") then
Uk := Gain_ky*(to_sfixed(rki,15,-16));
Mv <= to_slv(Uk(0,0));
Uk_antigo <= Uk;
elsif (Mv_en = '1') then
Xkm(0,0) := to_sfixed(xm1,15,-16);
Xkm(1,0) := to_sfixed(xm2,15,-16);
Xkm(2,0) := to_sfixed(xm3,15,-16);
DeltaXk := Xkm - XkMenosHum_sinal;
KxDeltaXk := (to_sfixed(-1.0,15,-16))*(Gain_Kx)*DeltaXk; -- -Kx*[X(k) - X(k-1)]
Error := (to_sfixed(-1.0,15,-16))*(Gain_Ky)*((to_sfixed(Yout,15,-16)) - (to_sfixed(rki,15,-16)));
vectorDeltaU := Error + KxDeltaXk;
-- integrador
Uk := Uk_sinal + vectorDeltaU;
--============================================================
Mv <= to_slv(Uk(0,0));
Uk_antigo <= Uk;
end if;
end process;
end architecture;
```

MPC generalizado com observador de estados

```
library ieee;
library ieee_proposed;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
use ieee_proposed.real_matrix_pkg.all;
entity Exemplo1 is
generic (
NP : integer := 2; -- Horizonte de predição
NC : integer := 1
);
port( clk : in std_logic;
rst : in std_logic;
Yout : in real; -- Recebe a saída da planta Y
rki : in real; -- Porta para o sinal de referência (setpoint)
Mv : out real); -- Lei de controle - saída do MPC
end entity;
architecture horizon of Exemplo1 is
signal A : real_matrix (0 to 3, 0 to 3) := zeros(4,4); -- Matriz aumentada A
signal B : real_matrix (0 to 3, 0 to 0) := zeros(4,1); -- Matriz aumentada B
signal C : real_matrix (0 to 0, 0 to 3) := zeros(1,4); -- Matriz aumentada C
signal Gain_Ky : real_matrix (0 to 0, 0 to 0) := zeros(1,1); -- Ganho Ky
signal Gain_Kmpc : real_matrix (0 to 0, 0 to (A'length(2)-1)) := zeros(1,A'length(2)); -- Ganho
Kmpc
--================================================================
signal XkMenosHum_sinal : real_matrix (0 to 2, 0 to 0):= zeros(3,1);
signal Uk_sinal : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
```

```
signal Uk_antigo : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
--================================================================
signal Xk_observer_sinal : real_matrix (0 to 3, 0 to 0) := zeros(4,1);
signal Xk_observer_antigo : real_matrix (0 to 3, 0 to 0) := zeros(4,1);
--================================================================
signal Mv_en : std_logic;

begin

MPC_gain : process(rki)
-- Matrizes do modelo
variable am : real_matrix (0 to 2, 0 to 2):=zeros(3,3);
variable cm : real_matrix (0 to 0, 0 to 2):=zeros(1,3);
variable bm : real_matrix (0 to 2, 0 to 0):=zeros(3,1);
-- Matrizes Aumentadas
variable Ae : real_matrix (0 to (Cm'length(1)+Cm'length(2)-1), 0 to (Cm'length(1)+Cm'length(2)-1
)):=eye((Cm'length(1)+Cm'length(2)),(Cm'length(1)+Cm'length(2)));
variable Ce : real_matrix (0 to (Cm'length(1)-1), 0 to (Cm'length(1)+Cm'length(2)-1)):=zeros((Cm'
length(1)),(Cm'length(1)+Cm'length(2)));
variable Be : real_matrix (0 to (Cm'length(1)+Cm'length(2)-1), 0 to (Bm'length(2)-1)):=zeros((Cm'
length(1)+Cm'length(2)),Bm'length(2));
variable um_mat: real_matrix (0 to (Cm'length(1)-1), 0 to (Cm'length(1)-1)):= eye(Cm'length(1),Cm
'length(1));
variable so_zero: real_matrix (0 to 0, 0 to 0):= zeros(1,1);
--Matrizes F e Phi
variable Fv : real_matrix (0 to NP-1, 0 to (A'length(2)-1)) := zeros(NP,A'length(2)); -- Matriz F
variable Phiv : real_matrix (0 to NP-1, 0 to NC-1) := zeros(NP,NC); -- Matriz Phi
-------------------------------------------------------
-- Phi transposto
variable PhiT5 : real_matrix(0 to NC-1, 0 to NP-1) := zeros(NC,NP);
------====================----------========================
--PhiTransposto*Phi, Inversa
variable PhiT_Phi : real_matrix(0 to NC-1, 0 to NC-1) := zeros(NC,NC);
variable PhiTPhi_Rbarra : real_matrix(0 to NC-1, 0 to NC-1):= zeros(NC,NC) ;
variable InvPhiTPhi_Rbarra : real_matrix(0 to NC-1, 0 to NC-1):= zeros(NC,NC);
--=======================================================
--Phi tranposto * F
variable PhiT_F : real_matrix(0 to NC-1, 0 to A'length(2)-1):= zeros(NC,A'length(2));
--================================================================
-- Matriz Rbarra
variable RbNc : real_matrix (0 to NC-1, 0 to NC-1) := eye(NC,NC);
variable RbarraW : real_matrix (0 to NC-1, 0 to NC-1) := zeros(NC,NC);
variable peso : real;
--================================================================
-- Cálculo ganhos
variable vectorHumZero : real_matrix (0 to 0, 0 to NC-1) := zeros(1,NC);
variable vectorHum : real_matrix (0 to NP-1, 0 to 0) := ones(NP,1);
variable Kmpc : real_matrix (0 to 0, 0 to 3):= zeros(1,4);
variable Kx : real_matrix (0 to 0, 0 to 2):= zeros(1,3);
variable Ky : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
begin
am := ((-0.115892273831871, -1.33711207804134, -0.578470168753942), (0.0192823389584647,
0.154060471586635, -0.507971502827361), (0.0169323834275787, 0.256335706944567,
0.882152958972519
));
bm(0,0) := 0.0192823389584647;
bm(1,0) := 0.0169323834275787;
bm(2,0) := 0.00392823470091603;
cm(0,0) := 0.0;
cm(0,1) := 0.0;
```

```
cm(0,2) := 500.0;
peso := 0.0; -- rw
vectorHumZero(0,0) := 1.0;
--Matriz A
buildmatrix(Am, Ae, 0,0);
buildmatrix(Cm*Am, Ae, (Am'length(1)),0);
--Matriz B
buildmatrix(Bm, Be, 0,0);
buildmatrix(Cm*Bm, Be, Bm'length(1),0);
--Matriz C
buildmatrix(um_mat, Ce, 0,Am'length(2));
Matriz_F : for n in 0 to NP-1 loop
buildmatrix(Ce*Ae**(n+1), Fv, n, 0);
end loop;
Matriz_Phi : for linha in 0 to NP-1 loop
for coluna in 0 to (NC - 1) loop
if linha = coluna then
buildmatrix(Ce*Be, Phiv, linha, coluna);
elsif linha < coluna then
buildmatrix(so_zero, Phiv, linha, coluna);
elsif linha > coluna then
if coluna = 0 then
buildmatrix(Ce*(Ae**linha)*Be, Phiv, linha, coluna);
else
buildmatrix(Ce*(Ae**(linha-coluna))*Be, Phiv, linha, coluna);
end if;
end if;
end loop;
end loop;
--=====================================================
-- Phi transposto
PhiT5 := transpose(Phiv);
--=====================================================
--Rbarra
RbarraW := peso*(RbNc);
--=====================================================
--PhiTPhi, ou seja, Phitransposto vezes Phi
PhiT_Phi := PhiT5*Phiv;
PhiTPhi_Rbarra := PhiT_Phi + RbarraW;
InvPhiTPhi_Rbarra := inv(PhiTPhi_Rbarra);
--=====================================================
-- Matriz Phi Transposto * F
PhiT_F := PhiT5*Fv;
--=====================================================
-- Cálculo do Ganho Kmpc
Kmpc := vectorHumZero*InvPhiTPhi_Rbarra*PhiT_F;
--=====================================================
-- Ganho Ky
Ky := vectorHumZero*InvPhiTPhi_Rbarra*PhiT5*vectorHum;
--=====================================================
-- Ganho Kx
Kx(0,0) := Kmpc(0,0);
kx(0,1) := Kmpc(0,1);
kx(0,2) := Kmpc(0,2);
--=============================================================
A <= Ae;
B <= Be;
C <= Ce;
Gain_Kmpc <= Kmpc;
Gain_Ky <= Ky;
```

```
end process;

Clock_cont: process(clk, rst)

variable cnt : integer := 0;
constant Tempo_amostragem : integer := 442; -- clock 1kHz
begin
if (rst = '0') then
cnt := 0;
Mv_en <= '0';
elsif (clk'event and clk = '1') then
cnt := cnt + 1;
if cnt = Tempo_amostragem then
cnt := 0;
Mv_en <= '1';
else
Mv_en <= '0';
end if;
if cnt = (Tempo_amostragem - 1) then
Xk_observer_sinal <= Xk_observer_antigo;
Uk_sinal <= Uk_antigo;
end if;
end if;
end process;
--=================================================

Control_law: process(rst, Yout, rki, clk)

variable XkmaisHum_observer : real_matrix (0 to 3, 0 to 0):= zeros(4,1);
variable KxDeltaXk : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
variable K_observer : real_matrix (0 to 3, 0 to 0):= zeros(4,1);
variable vectorDeltaU : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
variable Uk : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
variable A_xk : real_matrix (0 to 3, 0 to 0):= zeros(4,1);
variable B_deltaU : real_matrix (0 to 3, 0 to 0):= zeros(4,1);
variable K_error : real_matrix (0 to 3, 0 to 0):= zeros(4,1);
variable C_xk : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
begin
if (rst = '0') then
Xk_observer_antigo(0,0) <= 0.0;
Xk_observer_antigo(1,0) <= 0.0;
Xk_observer_antigo(2,0) <= 0.0;
Xk_observer_antigo(3,0) <= 0.0;
XkmaisHum_observer(0,0) := 0.0; --condição inicial
XkmaisHum_observer(1,0) := 0.0; --condição inicial
XkmaisHum_observer(2,0) := 0.0; --condição inicial
XkmaisHum_observer(3,0) := 0.0; --condição inicial
KxDeltaXk(0,0) := 0.0;
vectorDeltaU(0,0) := 0.0;
K_observer(0,0) := -0.0012;
K_observer(1,0) := -0.0011;
K_observer(2,0) := 0.0018;
K_observer(3,0) := 1.9161;
Mv <= 0.0;
elsif (Mv_en = '1') then
KxDeltaXk := -(Gain_Kmpc)*Xk_observer_sinal;
vectorDeltaU := Gain_Ky*rki + KxDeltaXk;
A_xk := A*Xk_observer_sinal;
B_deltaU := B*vectorDeltaU;
C_xk := C*Xk_observer_sinal;
```

```
K_error := K_observer*(Yout - C_xk(0,0));
XkmaisHum_observer := A_xk + B_deltaU + K_error;
-- integrador
Uk := Uk_sinal + vectorDeltaU;
--=========================================================
Mv <= Uk(0,0);
Uk_antigo <= Uk;
Xk_observer_antigo <= XkmaisHum_observer;
end if;
end process;
        end architecture;
```

MPC com restrições

```
library ieee;
library ieee_proposed;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
use ieee_proposed.real_matrix_pkg.all;
entity Exemplo1 is
generic (
NP : integer := 4;
NC : integer := 3
);
port( clk : in std_logic;
rst : in std_logic;
Yout : in real; -- Recebe a saída da planta Y
rki : in real; -- Porta para o sinal de referência (setpoint)
xm1 : in real; -- realimentação de estado X1
xm2 : in real; -- realimentação de estado X2
xm3 : in real; -- realimentação de estado X3
Mv : out real);-- Lei de controle - saída do MPC
end entity;
architecture horizon of Exemplo1 is
signal A : real_matrix (0 to 3, 0 to 3) := zeros(4,4); -- Matriz aumentada A
signal B : real_matrix (0 to 3, 0 to 0) := zeros(4,1); -- Matriz aumentada B
signal C : real_matrix (0 to 0, 0 to 3) := zeros(1,4); -- Matriz aumentada C
--Matrizes F e Phi
signal F : real_matrix (0 to NP-1, 0 to (A'length(2)-1)) := zeros(NP,A'length(2)); -- Matriz F
signal Phi_transpose : real_matrix(0 to NC-1, 0 to NP-1) := zeros(NC,NP);
signal Rs_sinal : real_matrix(0 to NP-1, 0 to 0):= zeros(NP,1);
--=============================================================
signal Mv_en : std_logic;
signal XkMenosHum_sinal : real_matrix (0 to 2, 0 to 0):= zeros(3,1);
signal Uk_sinal : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
signal Uk_antigo : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
-- Constraints
signal M : real_matrix(0 to 5, 0 to NC-1) := zeros(6,NC);
signal gama : real_matrix(0 to 5, 0 to 0) := zeros(6,1);
signal E : real_matrix(0 to NC-1, 0 to NC-1) := zeros(NC,NC);
signal lambda_teste : real_matrix(0 to 1, 0 to 0) := zeros(2,1);
--Programação quadrática de Hildreth
procedure QPhild (E_rest : in real_matrix; H_rest : in real_matrix; M_rest : in real_matrix;
gama_rest : in real_matrix; eta_rest : out real_matrix) is
variable restricao : integer := 0;
variable eta : real_matrix(0 to NC-1, 0 to 0) := zeros(NC,1);
variable linha_M : real_matrix(0 to 0, 0 to M_rest'length(2)-1) := zeros(1,M_rest'length(2));
variable inv_E_rest : real_matrix(0 to E_rest'length(1)-1, 0 to E_rest'length(2)-1) := zeros(
```

```
E_rest'length(1),E_rest'length(2));
variable M_rest_trans : real_matrix(0 to M_rest'length(2)-1, 0 to M_rest'length(1)-1) := zeros(
M_rest'length(2), M_rest'length(1));
variable lambda : real_matrix(0 to M_rest'length(1)-1, 0 to 0) := zeros(M_rest'length(1),1);
variable M_teste : real_matrix(0 to M_rest'length(1)-1, 0 to M_rest'length(2)-1 ) := zeros(
M_rest'length(1), M_rest'length(2));
variable teste : real_matrix(0 to M_rest'length(1)-1, 0 to 0) := zeros(M_rest'length(1),1);
variable P : real_matrix(0 to M_rest'length(1)-1, 0 to M_rest'length(1)-1) := zeros(M_rest'
length(1),M_rest'length(1));
variable K : real_matrix(0 to M_rest'length(1)-1, 0 to gama_rest'length(2)-1) := zeros(M_rest'
length(1),gama_rest'length(2));
variable teste2 : real_matrix(0 to 0, 0 to P'length(2)-1) := zeros(1,P'length(2));
variable w : real_matrix (0 to 0, 0 to 0) := zeros(1,1);
variable la : real_matrix (0 to 0, 0 to 0) := zeros(1,1);
variable op1 : real_matrix(0 to 0, 0 to 0) := zeros(1,1);
variable op2 : real_matrix(0 to 0, 0 to 0) := zeros(1,1);
--auxiliares
variable op_aux1 : real_matrix(0 to 0, 0 to 0) := zeros(1,1);
variable op_aux2 : real_matrix(0 to 0, 0 to 0) := zeros(1,1);
variable op_aux3 : real_matrix(0 to 0, 0 to 0) := zeros(1,1);
variable op_aux4 : real_matrix(0 to 0, 0 to 0) := zeros(1,1);
variable teste500 : real_matrix(0 to 0, 0 to 0) := zeros(1,1);
begin
M_teste := M_rest;
eta := -(inv(E_rest))*H_rest;
restricao := 0;
linha_M := zeros(1,M_teste'length(2));
lambda := zeros(M_rest'length(1),1);
for linha_Q in 0 to M_teste'length(1)-1 loop
linha_M := submatrix(M_teste,linha_Q,0, 1,M_rest'length(2));
buildmatrix(linha_M*eta, teste, linha_Q,0);
if(teste(linha_Q,0) > (gama_rest(linha_Q,0))) then
restricao := restricao + 1;
else
restricao := restricao + 0;
end if;
end loop;
if restricao = 0 then
eta_rest := eta;
else
inv_E_rest := inv(E_rest);
M_rest_trans := transpose(M_rest);
P := M_rest*(inv_E_rest)*(M_rest_trans); --matriz H
K := ((M_rest*(inv_E_rest)*H_rest))+gama_rest;
for km in 1 to 38 loop
for linha_K in 0 to (K'length(1)-1) loop
teste2 := submatrix(P, linha_K, 0 , 1, P'length(2));
op1 := (teste2*lambda);
op_aux1 := submatrix(P,linha_K,linha_K,1,1);
op_aux2 := submatrix(lambda,linha_K,0,1,1);
op2 := (op_aux1)*(op_aux2);
w := op1 - op2;
op_aux4 := submatrix(K,linha_K,0,1,1);
w := w + op_aux4;
op_aux3 := submatrix(P,linha_K,linha_K,1,1);
la := -(w)/op_aux3;
if la(0,0) < 0.0 then
lambda(linha_K,0):= 0.0;
else
teste500 := submatrix(la,0,0,1,1);
```

```
lambda(linha_K,0):= teste500(0,0);
end if;
end loop;
end loop;
eta_rest := -(inv_E_rest*H_rest) - (inv_E_rest*M_rest_trans*lambda);
end if;
end procedure;
begin
MPC_gain : process(rki)
-- Matrizes do modelo
variable am : real_matrix (0 to 2, 0 to 2):=zeros(3,3);
variable cm : real_matrix (0 to 0, 0 to 2):=zeros(1,3);
variable bm : real_matrix (0 to 2, 0 to 0):=zeros(3,1);
-- Matrizes Aumentadas
variable Ae : real_matrix (0 to (Cm'length(1)+Cm'length(2)-1), 0 to (Cm'length(1)+Cm'length(2)-1
)):=eye((Cm'length(1)+Cm'length(2)),(Cm'length(1)+Cm'length(2)));
variable Ce : real_matrix (0 to (Cm'length(1)-1), 0 to (Cm'length(1)+Cm'length(2)-1)):=zeros((Cm'
length(1)),(Cm'length(1)+Cm'length(2)));
variable Be : real_matrix (0 to (Cm'length(1)+Cm'length(2)-1), 0 to (Bm'length(2)-1)):=zeros((Cm'
length(1)+Cm'length(2)),Bm'length(2));
variable um_mat: real_matrix (0 to (Cm'length(1)-1), 0 to (Cm'length(1)-1)):= eye(Cm'length(1),Cm
'length(1));
variable so_zero: real_matrix (0 to 0, 0 to 0):= zeros(1,1);
--Matrizes F e Phi
variable Fv : real_matrix (0 to NP-1, 0 to (A'length(2)-1)) := zeros(NP,A'length(2)); -- Matrix F
variable Phiv : real_matrix (0 to NP-1, 0 to NC-1) := zeros(NP,NC); -- Matrix Phi ?
-------------------------------------------------------
-- Phi transposto
variable PhiT5 : real_matrix(0 to NC-1, 0 to NP-1) := zeros(NC,NP);
------====================----------==================
--PhiTranspose*Phi, Inversa
variable PhiT_Phi : real_matrix(0 to NC-1, 0 to NC-1) := zeros(NC,NC);
variable PhiTPhi_Rbarra : real_matrix(0 to NC-1, 0 to NC-1):= zeros(NC,NC) ;
variable InvPhiTPhi_Rbarra : real_matrix(0 to NC-1, 0 to NC-1):= zeros(NC,NC);
--=========================================================
--Phi tranposto * F
variable PhiT_F : real_matrix(0 to NC-1, 0 to A'length(2)-1):= zeros(NC,A'length(2));
--=========================================================
-- Vetor Rs
variable Rs : real_matrix(0 to 0, 0 to NP-1):= zeros(1,NP);
variable Rs_rki : real_matrix(0 to 0, 0 to NP-1):= zeros(1,NP);
variable TransposeRs : real_matrix(0 to NP-1, 0 to 0):= zeros(NP,1);
--=========================================================
-- Matriz Rbarra
variable RbNc : real_matrix (0 to NC-1, 0 to NC-1) := eye(NC,NC);
variable RbarraW : real_matrix (0 to NC-1, 0 to NC-1) := zeros(NC,NC);
variable peso : real;
--=========================================================
-- Calcule Ky
variable vectorHumZero : real_matrix (0 to 0, 0 to NC-1) := zeros(1,NC);
variable vectorHum : real_matrix (0 to NP-1, 0 to 0) := ones(NP,1);
variable Kmpc : real_matrix (0 to 0, 0 to (A'length(2)-1)):= zeros(1,A'length(2));
variable Kx : real_matrix (0 to 0, 0 to (A'length(2)-2)):= zeros(1,A'length(2)-1);
variable Ky : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
begin
am := ((-0.115892273831871, -1.33711207804134, -0.578470168753942), (0.0192823389584647,
0.154060471586635, -0.507971502827361), (0.0169323834275787, 0.256335706944567,
0.882152958972519
));
bm(0,0) := 0.0192823389584647;
```

```
bm(1,0) := 0.0169323834275787;
bm(2,0) := 0.00392823470091603;
cm(0,0) := 0.0;
cm(0,1) := 0.0;
cm(0,2) := 500.0;
peso := 0.0; -- rw
vectorHumZero(0,0) := 1.0;
--Matriz A
buildmatrix(Am, Ae, 0,0);
buildmatrix(Cm*Am, Ae, (Am'length(1)),0);
--Matriz B
buildmatrix(Bm, Be, 0,0);
buildmatrix(Cm*Bm, Be, Bm'length(1),0);
--Matriz C
buildmatrix(um_mat, Ce, 0,Am'length(2));
Matriz_F : for n in 0 to NP-1 loop
buildmatrix(Ce*Ae**(n+1), Fv, n, 0);
end loop;
Matriz_Phi : for linha in 0 to NP-1 loop
for coluna in 0 to (NC - 1) loop
if linha = coluna then
buildmatrix(Ce*Be, Phiv, linha, coluna);
elsif linha < coluna then
buildmatrix(so_zero, Phiv, linha, coluna);
elsif linha > coluna then
if coluna = 0 then
buildmatrix(Ce*(Ae**linha)*Be, Phiv, linha, coluna);
else
buildmatrix(Ce*(Ae**(linha-coluna))*Be, Phiv, linha, coluna);
end if;
end if;
end loop;
end loop;
--=====================================================
-- Phi transposto
PhiT5 := transpose(Phiv);
--=====================================================
-- Vetor Rs
Rs := ones(1,NP);
Rs_rki := rki*Rs;
TransposeRs := transpose(Rs_rki);
--=====================================================
--Rbarra
RbarraW := peso*(RbNc);
--=====================================================
--PhiTPhi, ou seja, Phitransposto vezes Phi
PhiT_Phi := PhiT5*Phiv;
PhiTPhi_Rbarra := PhiT_Phi + RbarraW;
InvPhiTPhi_Rbarra := inv(PhiTPhi_Rbarra);
--=========================================================
-- Matriz Phi Transposto * F
PhiT_F := PhiT5*Fv;
--=========================================================
A <= Ae;
B <= Be;
C <= Ce;
M(0,0) <= 1.0;
M(1,1) <= 1.0;
M(2,2) <= 1.0;
M(3,0) <= -1.0;
```

```
M(4,1) <= -1.0;
M(5,2) <= -1.0;
gama(0,0) <= 0.8;
gama(1,0) <= 0.8;
gama(2,0) <= 0.8;
gama(3,0) <= 0.8;
gama(4,0) <= 0.8;
gama(5,0) <= 0.8;
Phi_transpose <= PhiT5;
F <= Fv;
Rs_sinal <= TransposeRs;
E <= 2*PhiTPhi_Rbarra;
end process;

Clock_cont: process(clk, rst)

variable cnt : integer := 0;
constant Tempo_amostragem : integer := 442;
variable XkMenosHum : real_matrix (0 to 2, 0 to 0):= zeros(3,1); -- X(k-1)
begin
if (rst = '0') then
cnt := 0;
Mv_en <= '0';
XkMenosHum(0,0) := 0.0;
XkMenosHum(1,0) := 0.0;
XkMenosHum(2,0) := 0.0;
elsif (clk'event and clk = '1') then
cnt := cnt + 1;
if cnt = Tempo_amostragem then
cnt := 0;
Mv_en <= '1';
else
Mv_en <= '0';
end if;
if cnt = (Tempo_amostragem - 1) then
XkMenosHum(0,0) := xm1;
XkMenosHum(1,0) := xm2;
XkMenosHum(2,0) := xm3;
Uk_sinal <= Uk_antigo;
end if;
XkMenosHum_sinal <= XkMenosHum;
end if;
end process;
--================================================
Control_law: process(rst, xm1,xm2,xm3, Yout)
variable Xkm : real_matrix (0 to 2, 0 to 0):= zeros(3,1); -- X(K)
variable Xf : real_matrix (0 to Xkm'length(1), 0 to 0) := zeros(Xkm'length(1)+1,1);
variable DeltaXk : real_matrix (0 to 2, 0 to 0):= zeros(3,1); -- delta_x(k) = X(k) - X(K-1)
variable vectorDeltaU : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
variable Uk : real_matrix (0 to 0, 0 to 0) := zeros(1,1);
--Restrições (Constraints)
variable H :real_matrix(0 to NC-1 , 0 to 0) := zeros(NC,1);
variable eta : real_matrix(0 to NC-1, 0 to 0) := zeros(NC,1);
begin
if (rst = '0') then
Xkm(0,0) := 0.0;
Xkm(1,0) := 0.0;
Xkm(2,0) := 0.0;
Mv <= 0.0;
elsif (xm1 = 0.0 and xm2 = 0.0 and xm3 = 0.0 and Yout = 0.0) then
```

```
DeltaXk := Xkm;
buildmatrix(DeltaXk, Xf, 0, 0);
Xf(Xkm'length(1),0) := Yout;
H := -2*Phi_transpose*(Rs_sinal-F*Xf);
QPhild(E,H,M,gama, eta);
vectorDeltaU := submatrix(eta,0,0,1,1);
-- integrador
Uk := Uk_sinal + vectorDeltaU;
Mv <= Uk(0,0);
Uk_antigo <= Uk;
elsif (Mv_en = '1') then
Xkm(0,0) := xm1;
Xkm(1,0) := xm2;
Xkm(2,0) := xm3;
DeltaXk := Xkm - XkMenosHum_sinal;
buildmatrix(DeltaXk, Xf, 0, 0);
Xf(Xkm'length(1),0) := Yout;
H := -2*Phi_transpose*(Rs_sinal-F*Xf);
QPhild(E,H,M,gama, eta);
vectorDeltaU := submatrix(eta,0,0,1,1);
-- integrador
Uk := Uk_sinal + vectorDeltaU;
--============================================================
end if;
Uk_antigo <= Uk;
Mv <= Uk(0,0);
end process;
end architecture;
```

MPC com funções de Laguerre

```
library ieee;
library ieee_proposed;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
use ieee_proposed.real_matrix_pkg.all;
entity Exemplo1 is
generic (
a_pole : real := 0.3;
N : integer := 20;
NP : integer := 5
);
port( clk : in std_logic;
sp : in real; -- referência (set-point)
rst : in std_logic;
Yout : in real; -- Recebe a saída da planta Y
xm1 : in real; -- realimentação de estado X1
xm2 : in real; -- realimentação de estado X2
xm3 : in real; -- realimentação de estado X3
Mv : out real);-- Lei de controle - saída do MPC
end entity;
architecture teste of Exemplo1 is
--Funções de Laguerre
procedure lagd( a : in real; N : in integer; Al: out real_matrix; L0 : out real_matrix) is
variable v : real_matrix(0 to N-1, 0 to 0) := zeros(N,1);
variable Al_v : real_matrix(0 to N-1, 0 to N-1) := zeros(N,N);
variable L0_v : real_matrix(0 to N-1, 0 to 0) := zeros(N,1);
variable beta : real_matrix(0 to 0, 0 to 0) := zeros(1,1);
```

```
variable linha1_v : real_matrix(0 to N-1, 0 to 0) := zeros(N,1);
variable beta_r : real := 0.0;
variable beta_raiz : real_matrix(0 to 0, 0 to 0) := zeros(1,1);
variable so_zero : real := 0.0;
variable elemento_v : real_matrix(0 to 0, 0 to 0) := zeros(1,1);
--começo
begin
--primeiros elementos
Al_v := zeros(N,N);
v(0,0) := a;
L0_v(0,0) := 1.0;
beta(0,0) := (1.0 - (a*a));
--construção de L0 e primeira linha de Al
for k in 1 to N-1 loop
buildmatrix(((-a)**(k-1))*beta, v, k,0);
L0_v(k,0) := (-a)**(k);
end loop;
--Fechando L0
beta_raiz := sqrt(beta);
beta_r := beta_raiz(0,0);
L0 := beta_r*L0_v;
linha1_v := submatrix(v,0,0,v'length(1),1);
buildmatrix(linha1_v, Al_v, 0,0);
--Construindo Al
for linha in 0 to N-1 loop
for coluna in 0 to N-1 loop
if linha = coluna then --diagonal
elemento_v := submatrix(v,0,0,1,1);
buildmatrix(elemento_v,Al_v,linha,coluna);
elsif linha < coluna then -- triangulo superior
elemento_v(0,0) := so_zero;
buildmatrix(elemento_v,Al_v,linha,coluna);
elsif linha > coluna then -- triangulo inferior
if coluna = 0 then
elemento_v := submatrix(v,linha,0,1,1);
buildmatrix(elemento_v,Al_v,linha,coluna);
else
elemento_v := submatrix(v,linha-coluna,0,1,1);
buildmatrix(elemento_v,Al_v,linha,coluna);
end if;
end if;
end loop;
end loop;
Al := Al_v;
end procedure;
procedure dmpc( A_e : in real_matrix; B_e : in real_matrix; a: in real; N : in integer; Np : in
integer; Q : in real_matrix; R : in real_matrix; E : out real_matrix; H : out real_matrix) is
variable E_v : real_matrix(0 to N-1, 0 to N-1) := zeros(N,N);
variable H_v : real_matrix(0 to N-1, 0 to (B_e'length(1))-1) := zeros(N,(B_e'length(1)));
variable R_peso : real;
variable R_para : real_matrix(0 to N-1, 0 to N-1) := zeros(N,N);
variable S_in : real_matrix(0 to B_e'length(1)-1, 0 to N-1) := zeros(B_e'length(1),N);
variable Al : real_matrix(0 to N-1, 0 to N-1) := zeros(N,N);
variable L0 : real_matrix(0 to N-1, 0 to 0) := zeros(N,1);
variable transpose_Al : real_matrix(0 to N-1, 0 to N-1) := zeros(N,N);
variable L0_transpose : real_matrix(0 to 0, 0 to N-1) := zeros(1,N);
variable ident_N : real_matrix(0 to N-1, 0 to N-1) := eye(N,N);
variable S_sum : real_matrix(0 to B_e'length(1)-1, 0 to N-1) := zeros(B_e'length(1),N);
variable phi : real_matrix(0 to B_e'length(1)-1, 0 to N-1) := zeros(B_e'length(1),N);
variable phi_transpose : real_matrix(0 to N-1, 0 to B_e'length(1)-1) := zeros(N,B_e'length(1));
```

```
variable Eae : real_matrix(0 to A_e'length(1)-1, 0 to A_e'length(2)-1) := zeros(A_e'length(1),
A_e'length(2));
begin
R_peso := R(0,0);
R_para := (R_peso*ident_N);
lagd(a,N,Al,L0);
L0_transpose := transpose(L0);
S_in := B_e*L0_transpose;
S_sum := S_in;
phi := S_in;
phi_transpose := transpose(phi);
E_v := phi_transpose*Q*phi;
H_v := phi_transpose*Q*A_e;
for w in 2 to NP loop
Eae := A_e**(w);
transpose_Al := transpose(Al**(w-1));
S_sum := (A_e*S_sum) + (S_in*transpose_Al);
phi := S_sum;
phi_transpose := transpose(phi);
E_v := E_v + (phi_transpose*Q*phi);
H_v := H_v + (phi_transpose*Q*Eae);
end loop;
E := E_v + R_para;
H := H_v;
end procedure;
signal Al_s : real_matrix(0 to N-1, 0 to N-1);
signal L0_s : real_matrix(0 to N-1, 0 to 0);
signal Omega : real_matrix (0 to N-1, 0 to N-1):=zeros(N,N);
signal Psi : real_matrix (0 to N-1, 0 to 3):=zeros(N,4);
signal Mv_en : std_logic;
signal XkMenosHum_sinal : real_matrix (0 to 2, 0 to 0):= zeros(3,1);
signal Uk_sinal : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
signal Uk_antigo : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
begin
MPC_gain : process(sp)
-- Matrizes do modelo
variable am : real_matrix (0 to 2, 0 to 2):=zeros(3,3);
variable cm : real_matrix (0 to 0, 0 to 2):=zeros(1,3);
variable bm : real_matrix (0 to 2, 0 to 0):=zeros(3,1);
-- Matrizes Aumentadas
variable Ae : real_matrix (0 to (Cm'length(1)+Cm'length(2)-1), 0 to (Cm'length(1)+Cm'length(2)-1
)):=eye((Cm'length(1)+Cm'length(2)),(Cm'length(1)+Cm'length(2)));
variable Ce : real_matrix (0 to (Cm'length(1)-1), 0 to (Cm'length(1)+Cm'length(2)-1)):=zeros((Cm'
length(1)),(Cm'length(1)+Cm'length(2)));
variable Be : real_matrix (0 to (Cm'length(1)+Cm'length(2)-1), 0 to (Bm'length(2)-1)):=zeros((Cm'
length(1)+Cm'length(2)),Bm'length(2));
variable Ce_transpose : real_matrix (0 to (Cm'length(1)+Cm'length(2)-1), 0 to (Cm'length(1)-1)):=
zeros((Cm'length(1)+Cm'length(2)),(Cm'length(1)));
variable um_mat: real_matrix (0 to (Cm'length(1)-1), 0 to (Cm'length(1)-1)):= eye(Cm'length(1),Cm
'length(1));
variable Q : real_matrix (0 to Ce'length(2)-1, 0 to Ce'length(2)-1):=zeros(Ce'length(2),Ce'length
(2));
variable R : real_matrix (0 to 0, 0 to 0):=zeros(1,1);
variable Omega_v : real_matrix (0 to N-1, 0 to N-1):=zeros(N,N);
variable Psi_v : real_matrix (0 to N-1, 0 to Be'length(1)-1):=zeros(N,Be'length(1));
variable Al : real_matrix(0 to N-1, 0 to N-1):=zeros(N,N);
variable L0 : real_matrix(0 to N-1, 0 to 0):=zeros(N,1);
variable cnt : integer := 0;
constant Tempo_amostragem : integer := 442;
variable XkMenosHum : real_matrix (0 to 2, 0 to 0):= zeros(3,1); -- X(k-1)
```

```
begin
am := ((-0.115892273831871, -1.33711207804134, -0.578470168753942), (0.0192823389584647,
0.154060471586635, -0.507971502827361), (0.0169323834275787, 0.256335706944567,
0.882152958972519
));
bm(0,0) := 0.0192823389584647;
bm(1,0) := 0.0169323834275787;
bm(2,0) := 0.00392823470091603;
cm(0,0) := 0.0;
cm(0,1) := 0.0;
cm(0,2) := 500.0;
--Matriz A
buildmatrix(Am, Ae, 0,0);
buildmatrix(Cm*Am, Ae, (Am'length(1)),0);
--Matriz B
buildmatrix(Bm, Be, 0,0);
buildmatrix(Cm*Bm, Be, Bm'length(1),0);
--Matriz C
buildmatrix(um_mat, Ce, 0,Am'length(2));
lagd(a_pole,N,Al,L0);
Al_s <= Al;
L0_s <= L0;
Ce_transpose := transpose(Ce);
Q := Ce_transpose*Ce;
R(0,0) := 0.1;
dmpc( Ae, Be, a_pole, N, Np, Q, R, Omega_v, Psi_v);
Omega <= Omega_v;
Psi <= Psi_v;
if (rst = '0') then
cnt := 0;
Mv_en <= '0';
XkMenosHum(0,0) := 0.0;
XkMenosHum(1,0) := 0.0;
XkMenosHum(2,0) := 0.0;
elsif (clk'event and clk = '1') then
cnt := cnt + 1;
if cnt = Tempo_amostragem then
cnt := 0;
Mv_en <= '1';
else
Mv_en <= '0';
end if;
if cnt = (Tempo_amostragem - 1) then
XkMenosHum(0,0) := xm1;
XkMenosHum(1,0) := xm2;
XkMenosHum(2,0) := xm3;
Uk_sinal <= Uk_antigo;
end if;
XkMenosHum_sinal <= XkMenosHum;
end if;
end process;
Control_law: process(rst, xm1,xm2,xm3, Yout)
variable Xf : real_matrix (0 to 3, 0 to 0):= zeros(4,1); -- delta_x(k) = X(k) - X(K-1)
variable vectorDeltaU : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
variable Uk : real_matrix (0 to 0, 0 to 0):= zeros(1,1);
variable Lzerot : real_matrix(0 to 0, 0 to N-1):=zeros(1,N);
variable inv_Omega : real_matrix(0 to N-1, 0 to N-1):=zeros(N,N);
variable eta : real_matrix(0 to 0, 0 to 0) := zeros(1,1);
begin
if (rst = '0') then
```

```
Xf(0,0) := 0.0;
Xf(1,0) := 0.0;
Xf(2,0) := 0.0;
Xf(3,0) := Yout - sp;
Lzerot := transpose(L0_s);
Mv <= 0.0;
elsif (xm1 = 0.0 and xm2 = 0.0 and xm3 = 0.0 and Yout = 0.0) then
inv_Omega := (inv(Omega));
eta := -(inv_Omega*Psi)*Xf;
vectorDeltaU := Lzerot*eta;
Uk := Uk_sinal + vectorDeltaU;
Mv <= Uk(0,0);
Uk_antigo <= Uk;
elsif (Mv_en = '1') then
Xf(0,0) := xm1;
Xf(1,0) := xm2;
Xf(2,0) := xm3;
Xf(3,0) := Yout - sp;
inv_Omega := (inv(Omega));
eta := -(inv_Omega*Psi)*Xf;
vectorDeltaU := Lzerot*eta;
-- integrador
Uk := Uk_sinal + vectorDeltaU;
--=========================================================
end if;
Uk_antigo <= Uk;
Mv <= Uk(0,0);
end process;
end architecture;
```