PLANEJAMENTO DE SINAIS PARA IDENTIFICAÇÃO DE MODELOS MULTIVARIÁVEIS COM RESTRIÇÃO

Cristiano Salah Mussoi

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Química, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Química.

Orientadores: Príamo Albuquerque Melo Junior
Argimiro Resende Secchi

Rio de Janeiro
Fevereiro de 2019

PLANEJAMENTO DE SINAIS PARA IDENTIFICAÇÃO DE MODELOS MULTIVARIÁVEIS COM RESTRIÇÃO

Cristiano Salah Mussoi

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA QUÍMICA.

Examinada por:

Prof. Príamo Albuquerque Melo Junior, D.Sc.

Prof. Argimiro Resende Secchi, D.Sc.

Prof. Bruno Didier Olivier Capron, D.Sc.

Dr. Mario Cesar Mello Massa de Campos, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2019

Mussoi, Cristiano Salah

Planejamento de Sinais para Identificação de Modelos Multivariáveis com Restrição/Cristiano Salah Mussoi. – Rio de Janeiro: UFRJ/COPPE, 2019.

XVIII, 154 p.: il.; 29, 7cm.

Orientadores: Príamo Albuquerque Melo Junior
Argimiro Resende Secchi

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Química, 2019.

Referências Bibliográficas: p. 59 – 61.

1. Identificação de sistemas. 2. Controle de processos. 3. MPC linear. I. Melo Junior, Príamo Albuquerque *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Química. III. Título.

Aos meus pais.

Agradecimentos

Agradeço, primeiramente, ao Programa de Engenharia Química da COPPE/UFRJ, por ter me dado a oportunidade de realizar este mestrado.

Também agradeço aos meus orientadores, Príamo e Argimiro, pelo auxílio na elaboração desta dissertação e a todos os professores que direta ou indiretamente contribuíram para o meu crescimento profissional ao longo destes dois anos.

O presente trabalho foi realizado com o apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) – Código de Financiamento 001. A estas duas instituições, o meu muito obrigado.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

PLANEJAMENTO DE SINAIS PARA IDENTIFICAÇÃO DE MODELOS MULTIVARIÁVEIS COM RESTRIÇÃO

Cristiano Salah Mussoi

Fevereiro/2019

Orientadores: Príamo Albuquerque Melo Junior
Argimiro Resende Secchi

Programa: Engenharia Química

Uma nova metodologia de identificação de modelos empíricos para controladores MPC (do inglês, *Model Predictive Control*), que considera tanto restrições operacionais quanto fenomenológicas, é proposta nesta dissertação. A partir do trabalho de ORENSTEIN (2013), foi desenvolvido um método de identificação do tipo "caixa cinza" capaz de gerar somente modelos fisicamente consistentes, ou seja, modelos nos quais o valor e o sinal dos ganhos estáticos têm sentido físico. A metodologia desenvolvida, que faz uso de perturbações de entrada do tipo degrau e do tipo GBN (do inglês, *Generalized Binary Noise*), envolve etapas tanto *on-line* como *off-line*, sendo que as etapas *off-line* são executadas por um pacote computacional composto de quatro algoritmos: os dois primeiros realizam a análise de dados do processo e os dois últimos resolvem problemas de otimização com restrição. A metodologia foi aplicada na identificação de sistemas dinâmicos lineares e em um problema clássico da área, que é um coluna de destilação da Shell, se mostrando rápida e robusta nas simulações realizadas. Foram utilizados os tempos de simulação como indicadores de rapidez e os parâmetros estatísticos MRSE (do inglês, *Mean Relative Squared Error*) e MVAF (do inglês, *Mean Variance-Accounted-For*) como indicadores de robustez da metodologia proposta.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

INPUT DESIGN FOR IDENTIFICATION OF CONSTRAINED MULTIVARIATE MODELS

Cristiano Salah Mussoi

February/2019

Advisors: Príamo Albuquerque Melo Junior
Argimiro Resende Secchi

Department: Chemical Engineering

A new methodology for the identification of empirical models for Model Predictive Control (MPC), which considers both operational and phenomenological constraints, is proposed in this dissertation. From the work of ORENSTEIN (2013), a "gray box" method of identification type was developed capable to generate only physically consistent models, i.e., models in which the value and the signal of the static gains have physical sense. The developed methodology, which makes use of step-type as well as GBN-type (Generalized Binary Noise) input disturbances, involves both online and offline steps, and the offline steps are performed by a computational package composed of four algorithms: the first two ones perform the data analysis of the process and the last two ones solve optimization problems with restriction. The methodology was applied in the identification of linear dynamic systems and of a classic problem in this field, namely, a Shell distillation column, showing to be fast and robust in the simulations presented. The simulation times were used as indicators of speed and the statistical parameters MRSE (Mean Relative Squared Error) and MVAF (Mean Variance-Accounted-For) as indicators of robustness of the proposed methodology.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

$\text{adj}(\cdot)$	Adjunta de uma matriz, p. 16
$\mathbb{C}$	Conjunto dos números complexos, p. 16
c_{eq}	Restrição de igualdade, p. 26
c_{in}	Restrição de desigualdade, p. 26
d	Tempo morto, p. 9
$\det(\cdot)$	Determinante de uma matriz, p. 16
$\mathbf{E}$	Matriz dos erros, p. 9
$\mathbf{e}$	Vetor dos erros, p. 9
f_s	Fator de segurança, p. 28
$\mathbf{G_p}$	Matriz de transferência do processo, p. 16
$\mathbf{I}$	Matriz identidade, p. 10
$\mathbf{K}$	Matriz de ganhos estáticos do processo, p. 16
$\mathbf{K}^{inf}$	Limite inferior para $\mathbf{K}$, p. 16
$\mathbf{K}^{sup}$	Limite superior para $\mathbf{K}$, p. 16
N	Número de medidas de $\mathbf{y}$ e de $\boldsymbol{\varphi}$ utilizadas no processo de identificação, p. 9
N_a	Ordem do modelo ARX em relação às saídas, p. 9
N_b	Ordem do modelo ARX em relação às entradas, p. 9
n_{col}	Número de colunas de $\mathbf{X}$, p. 11
n_{lin}	Número de linhas de $\mathbf{X}$, p. 11
n_p	Número de partes das sequências de degraus, p. 21

n_{par}	Número de parâmetros, p. 14
n_r	Número de polos do sistema, p. 17
n_u	Número de variáveis de entrada, p. 9
n_y	Número de variáveis de saída, p. 9
$\mathbf{p}$	Vetor das probabilidades de não mudança de nível dos sinais GBN, p. 6
R	Raio, p. 23
$\mathbb{R}$	Conjunto dos números reais, p. 9
$\mathbb{R}^m$	Espaço dos vetores de m componentes reais, p. 9
$\mathbb{R}^{m\times n}$	Espaço das matrizes de m linhas e n colunas cujos elementos são números reis, p. 9
$\mathbf{r}$	Vetor dos polos do sistema, p. 16
S_a	Função *softmax*, p. 24
T_{min}	Período de amostragem, p. 17
t^{ass}	Tempo de assentamento, p. 21
$\mathrm{tr}(\cdot)$	Traço de uma matriz, p. 10
t_{sim}	Tempo de simulação, p. 34
$\mathbf{u}$	Vetor das variáveis de entrada, p. 9
$\mathbf{W}^{LA}$	Matriz de pesos para o LASSO adaptativo, p. 13
$\mathbf{w}$	Vetor de pesos para a função *softmax*, p. 24
$\mathbf{X}$	Matriz de parâmetros do modelo ARX, p. 9
$\mathbf{Y}$	Matriz das saídas, p. 9
$\mathbf{y}$	Vetor das variáveis de saída, p. 9
$\mathcal{Z}\{\cdot\}$	Transformada Z de um sinal discreto, p. 16
δ^K	Delta de Kronecker, p. 65
$\boldsymbol{\delta}$	Vetor das amplitudes positivas dos sinais GBN, p. 6

$\boldsymbol{\Phi}$	Matriz de regressão, p. 9
$\boldsymbol{\varphi}$	Vetor de regressão, p. 9
λ	Parâmetro de regularização, p. 10
ν	Frequência, p. 17
$\boldsymbol{\rho}$	Vetor dos polos do sistema em módulo e ao quadrado, p. 24
τ	Constante de tempo, p. 17
$\|\cdot\|$	Norma vetorial, p. 11
$\|\cdot\|_F$	Norma de Frobenius de uma matriz, p. 9
$\lceil\cdot\rceil$	Teto de um número, p. 75
∞	Infinito, p. 11

Lista de Abreviaturas

AIC_c	*corrected Akaike Information Criterion*, p. 6
AIC	*Akaike Information Criterion*, p. 13
ARX	*AutoRegressive with eXogenous input*, p. 3
BIC	*Bayesian Information Criterion*, p. 13
DSR	*Deterministic and Stochastic subspace system identification and Realization*, p. 52
EBIC	*Extended Bayesian Information Criterion*, p. 14
FIR	*Finite Impulse Response*, p. 3
GBN	*Generalized Binary Noise*, p. 1
GCV	*Generalized Cross-Validation*, p. 10
IDENTIPHY	*system IDENTIfication using PHYsical knowledge*, p. 20
LARS	*Least Angle Regression*, p. 11
LASSO	*Least Absolute Selection and Shrinkage Operator*, p. 10
LP	*Linear Programming*, p. 30
MIMO	*Multiple-Input, Multiple-Output*, p. 3
MPC	*Model Predictive Control*, p. 1
MRSE	*Mean Relative Squared Error*, p. 27
MVAF	*Mean Variance-Accounted-For*, p. 27
NARX	*Nonlinear AutoRegressive with eXogenous input*, p. 3
NLP	*NonLinear Programming*, p. 26
OLS	*Ordinary Least Squares*, p. 9

PRBS *Pseudo-Random Binary Sequence*, p. 3

RLS *Regularized Least Squares*, p. 10

SISO *Single-Input, Single-Output*, p. 3

ST *Settling time Threshold*, p. 21

s.a. sujeito a, p. 23

Capítulo 1

Introdução

Identificação de sistemas é o ramo da ciência que se dedica a modelar o comportamento de processos a partir de dados experimentais: aplicam-se perturbações nas entradas do processo, medem-se suas saídas e, a partir desses dados, gera-se um modelo matemático. No projeto de um Controlador Preditivo Multivariável (MPC), a identificação de sistemas possui um papel de extrema importância, pois a qualidade do modelo de processo fornecido ao controlador tem influência direta sobre o seu desempenho. No entanto, identificar "bons" modelos (fiéis à realidade) consome muito tempo, além de que as perturbações aplicadas podem fazer o processo sair de sua região operacional, gerando assim riscos à segurança da planta, perda de especificação dos produtos e reduções na produção.

ORENSTEIN (2013) propôs em sua dissertação de mestrado uma metodologia de identificação baseada em oito etapas, com a utilização de perturbações do tipo degrau e do tipo Ruído Binário Generalizado (GBN). No entanto, a metodologia não considera os aspectos fenomenológicos do processo, levando muitas vezes a modelos fisicamente inconsistentes. Além disso, o potencial de identificação dos sinais GBN não foi amplamente explorado, uma vez que eles são projetados sem levar em consideração a influência individual de cada entrada sobre o processo.

O objetivo deste trabalho é o desenvolvimento de uma metodologia sistemática de identificação, capaz de gerar modelos de boa qualidade em um tempo relativamente curto, minimizando assim os efeitos adversos das perturbações aplicadas. Para tal, o método de Orenstein foi modificado de modo a incluir, entre outras coisas, informação fenomenológica e uma heurística mais eficiente para o planejamento de sinais GBN. A metodologia desenvolvida envolve etapas *on-line* e *off-line*, sendo que as etapas *off-line* são executadas por um pacote computacional composto de quatro algoritmos: os dois primeiros realizam análise de dados e os dois últimos resolvem problemas de otimização com restrição. A metodologia proposta foi aplicada na identificação de sistemas dinâmicos lineares e em um problema *benchmark* (coluna de destilação da Shell).

Esta dissertação é composta de cinco capítulos, incluindo este. No próximo capítulo é feita a revisão bibliográfica, no capítulo seguinte é apresentada a metodologia de identificação desenvolvida, no quarto capítulo são mostrados os resultados obtidos e no último capítulo é feita a conclusão do trabalho.

Capítulo 2

Revisão Bibliográfica

2.1 Estado da Arte

Os diferentes métodos de identificação existentes na literatura podem ser caracterizados de acordo com os seguintes aspectos:

- Tipo de sistema a ser identificado: SISO (*Single-Input, Single-Output*) ou MIMO (*Multiple-Input, Multiple-Output*);
- Domínio no qual é realizada a identificação: tempo ou frequência;
- Sinais de entrada utilizados: degrau, PRBS (*Pseudo-Random Binary Sequence*), GBN (*Generalized Binary Noise*), etc;
- Modelo matemático empregado no processo de identificação:
  - Linear: FIR (*Finite Impulse Response*), ARX (*AutoRegressive with eXogenous input*), espaço de estados, etc;
  - Não Linear: NARX (*Nonlinear AutoRegressive with eXogenous input*), Hammerstein, Wiener, etc.

Apesar dos métodos de identificação serem numerosos, poucos são aqueles que consideram restrições operacionais, ou seja, restrições sobre as variáveis de entrada e saída. A Tabela 2.1 mostra as características de alguns métodos de identificação com restrição existentes na literatura, todos muito recentes. Os métodos que não consideram restrições não são de interesse deste trabalho, por não serem muito apropriados para aplicações industriais.

Comparando os métodos da Tabela 2.1, verifica-se que o mais adequado é o de ORENSTEIN (2013), pois além de poder ser aplicado a sistemas MIMO, utilizar sinais de entrada do tipo degrau (fáceis de serem gerados) e um modelo matemático relativamente simples (ARX), o método considera restrições sobre as amplitudes

das entradas e saídas, o caso mais comum de ser encontrado em ambiente industrial. Assim sendo, o método de ORENSTEIN (2013) foi o ponto de partida para a nova metodologia de identificação desenvolvida neste trabalho.

Tabela 2.1: Métodos de identificação que consideram restrições operacionais.

Método	Tipo de sistema	Domínio	Sinais de entrada	Modelo	Restrições
ORENSTEIN (2013)	MIMO	Tempo	Degrau e GBN	Linear (ARX)	Amplitude das entradas e saídas
HÄGG *et al.* (2014)	MIMO	Tempo	Sinal com autocorrelação imposta	Linear (espaço de estados)	Amplitude das entradas e saídas
STOJANOVIC e FILIPOVIC (2014)	SISO	Tempo	PRBS e sinal adaptativo	Linear (ARX)	Variância das saídas
DARBY e NIKOLAOU (2014)	MIMO	Tempo	GBN	Linear (FIR)	Variância e covariância das entradas e saídas
WANG *et al.* (2015)	SISO	Frequência	GBN	Linear (ARX)	Amplitude das entradas

2.2 Método de Identificação de Orenstein

O método de ORENSTEIN (2013) é composto de 8 etapas. São elas:

1. Pré-Teste;
2. Teste Degrau;
3. Obtenção de Modelos Iniciais;
4. Projeto GBN;
5. Teste GBN;
6. Modelagem;
7. Verificação;
8. Geração de Modelos Finais.

2.2.1 Pré-Teste

Primeiramente, são escolhidas as variáveis-desvio de entrada (**u**) e saída (**y**) do processo. Em seguida, é aplicada uma perturbação positiva do tipo degrau em cada entrada (é necessário que o processo seja estável em malha aberta).

Nesta etapa, deve-se maximizar a amplitude dos degraus aplicados sem violar as restrições operacionais da planta, de modo a maximizar a relação sinal/ruído. Logo, é importante consultar o operador da planta para saber o valor da amplitude máxima permitida em cada caso.

A duração de cada degrau deve ser maior ou igual ao tempo de assentamento do sistema em relação à entrada em questão. A Figura 2.1 mostra um exemplo de pré-teste, em que k representa o número de tempos de amostragem.

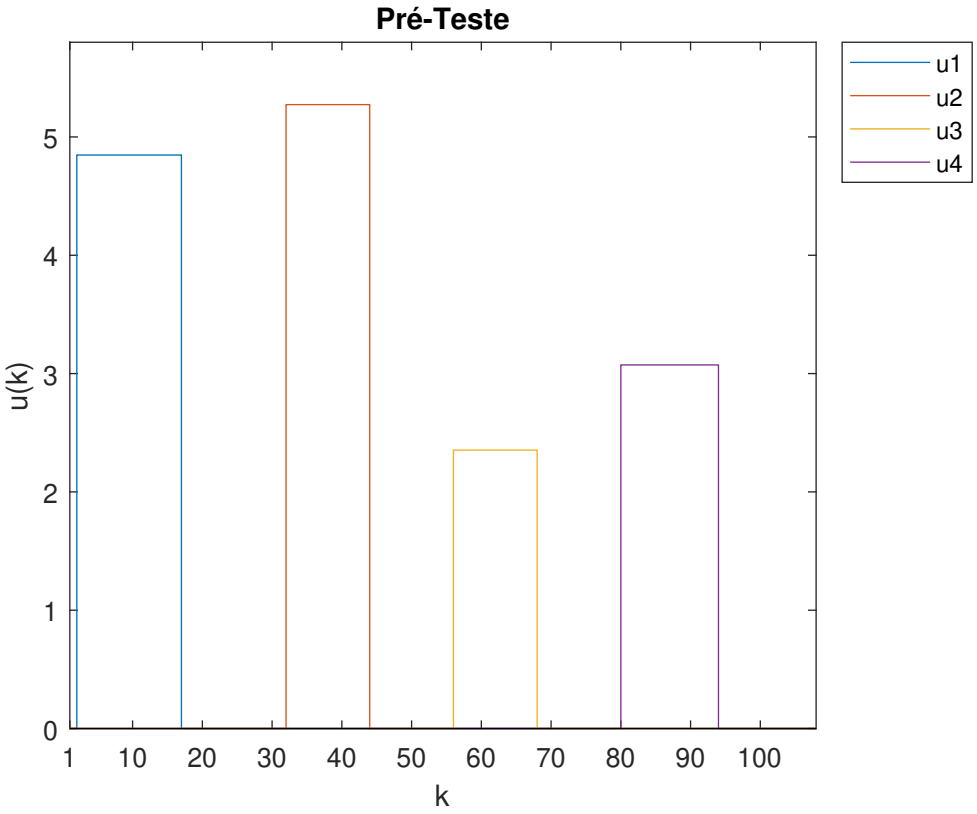

Figura 2.1: Exemplo de pré-teste (o processo em questão possui 4 entradas).

2.2.2 Teste Degrau

Nesta etapa, é aplicada uma sequência de degraus em cada entrada. Cada sequência é formada por 3 partes e cada parte, por sua vez, é formada por 2 degraus consecutivos de mesma duração e amplitudes opostas (de mesmo módulo mas com sentidos contrários).

Para a j-ésima entrada (u_j), o módulo da amplitude do teste degrau é igual à amplitude do pré-teste. A duração de cada degrau nas partes 1, 2 e 3 é igual a 1, 1,5 e 2 vezes o tempo de assentamento do sistema em relação a esta entrada, respectivamente.

As sequências são aleatórias em relação à ordem das entradas e à ordem das partes. A Figura 2.2 mostra um exemplo de teste degrau.

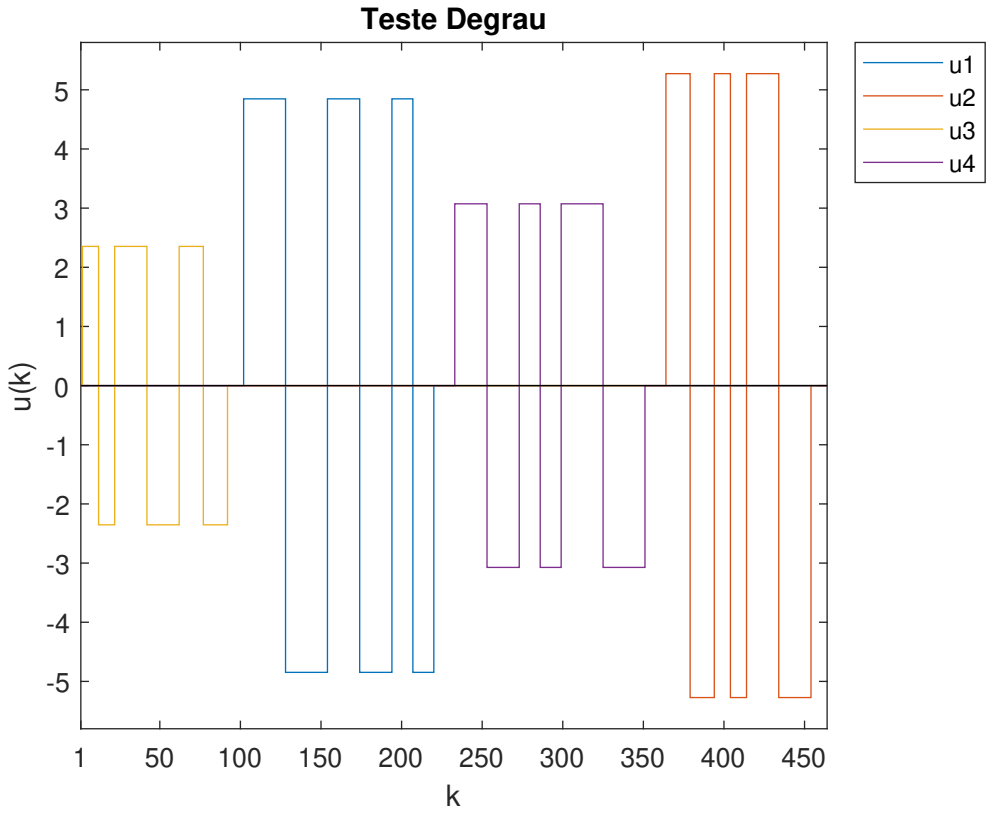

Figura 2.2: Exemplo de teste degrau (o processo em questão possui 4 entradas).

2.2.3 Obtenção de Modelos Iniciais

Diferentes modelos ARX são gerados a partir dos dados de entrada e saída obtidos no pré-teste e no teste degrau. Dentre os modelos gerados, seleciona-se aquele que apresenta o menor valor para o Critério de Informação de Akaike corrigido – AIC_c (HURVICH e TSAI, 1989). Os cálculos desta etapa são realizados com o auxílio da *toolbox* de identificação do MATLAB.

2.2.4 Projeto GBN

O GBN (*Generalized Binary Noise*) da entrada u_j é um sinal aleatório que pode assumir 2 valores: $+\delta_j$ (nível alto) ou $-\delta_j$ (nível baixo), sendo que a probabilidade de u_j não mudar de nível (p_j) é constante ao longo do tempo.

No projeto GBN, são determinados os pares (δ_j,p_j) para cada entrada u_j com base no modelo de processo obtido na etapa anterior. Quanto maior for δ_j, maior será a relação sinal/ruído. Logo, deve-se maximizar este parâmetro tomando o cuidado para não violar as restrições operacionais da planta. No método de Orenstein, todos os p_j's tem o mesmo valor, sendo este determinado pela heurística de ZHU e VAN DEN BOSCH (2000): escolhe-se p_j de modo que o tempo médio de troca de nível de u_j seja igual a 1/3 do maior tempo de assentamento do sistema.

2.2.5 Teste GBN

Todos os sinais GBN projetados são aplicados ao mesmo tempo na planta. A duração total deste teste é igual a 12 vezes o maior tempo de assentamento do sistema e o mesmo deve ser executado por computador. Nesta etapa, avaliam-se as interações entre as diferentes entradas. A Figura 2.3 mostra um exemplo de teste GBN.

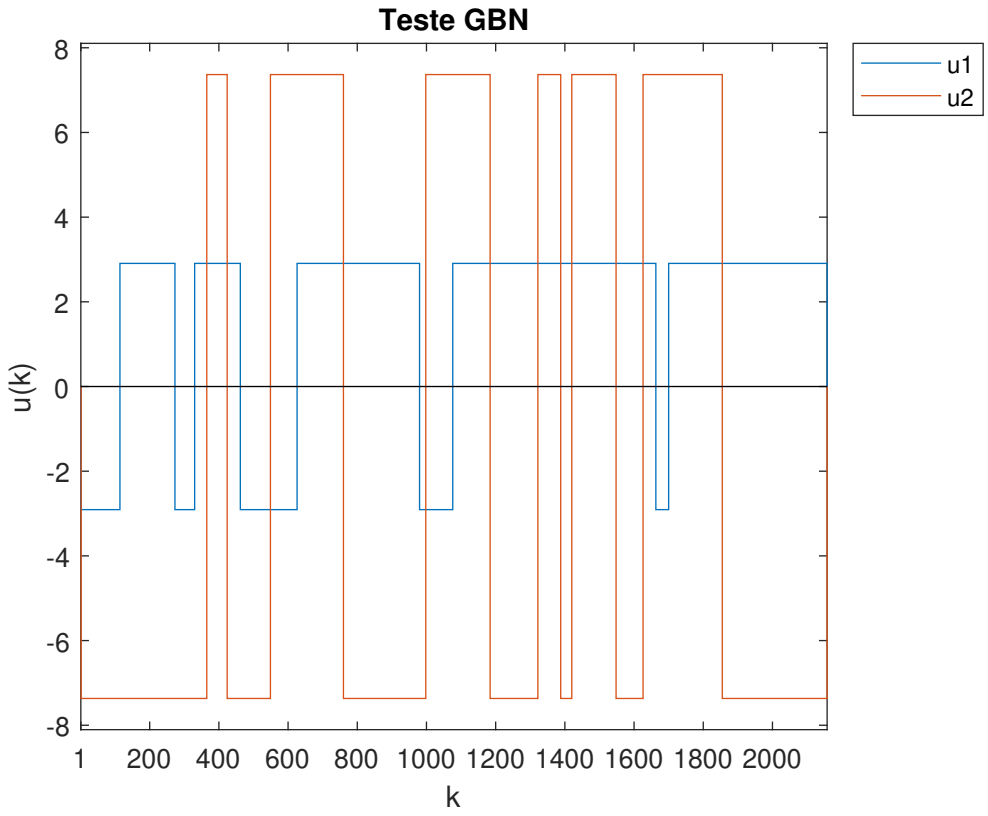

Figura 2.3: Exemplo de teste GBN (o processo em questão possui 2 entradas). Neste caso, $\delta_1 = 2,91$, $\delta_2 = 7,37$ e $p_1 = p_2 = 0,99$.

2.2.6 Modelagem, Verificação e Geração de Modelos Finais

Diferentes modelos ARX são gerados a partir dos dados de entrada e saída obtidos no teste GBN (o pré-teste e o teste degrau não são considerados nesta etapa). Como

anteriormente, seleciona-se o modelo que minimiza o AIC_c. Os cálculos desta etapa também são realizados com o auxílio da *toolbox* de identificação do MATLAB.

A consistência física do modelo selecionado é em seguida verificada por especialistas, como operadores e engenheiros. Se não forem encontradas inconsistências, o processo de identificação chega ao fim e tem-se o modelo final. Caso contrário, será necessário refazer algumas das etapas anteriores.

2.2.7 Pontos de Melhoria

Apesar de eficiente, verifica-se que o método de Orenstein pode ser melhorado. As propostas de melhoria são as seguintes:

1. Utilizar um procedimento mais simples e eficaz para a identificação de sistemas MIMO com modelos ARX;

2. Utilizar informação fenomenológica na identificação, fazendo com que os modelos gerados sejam todos fisicamente consistentes (desse modo, o etapa de verificação não será mais necessária);

3. Explorar o projeto ótimo de sinais GBN, estabelecendo uma nova rotina para o cálculo dos parâmetros p_j das entradas;

4. Utilizar simultaneamente os dados do pré-teste, teste degrau e teste GBN na geração de modelos finais.

A quarta proposta pode ser facilmente implementada. No entanto, o mesmo não é válido para as três primeiras. A seguir, são mostradas as soluções encontradas na literatura para estes três casos.

2.3 Identificação MIMO

2.3.1 Fundamentação Teórica

O modelo ARX para sistemas MIMO apresenta a seguinte forma:

$$\mathbf{y}(k) = \mathbf{A}_1\mathbf{y}(k-1)+\cdots+\mathbf{A}_{N_a}\mathbf{y}(k-N_a)+\mathbf{B}_d\mathbf{u}(k-d)+\cdots+\mathbf{B}_{N_b}\mathbf{u}(k-N_b)+\mathbf{e}(k) \quad (2.1)$$

em que $\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_{N_a} \in \mathbb{R}^{n_y \times n_y}$ e $\mathbf{B}_d, \mathbf{B}_{d+1}, ..., \mathbf{B}_{N_b} \in \mathbb{R}^{n_y \times n_u}$ são matrizes de coeficientes, $\mathbf{y}(k) \in \mathbb{R}^{n_y}$, $\mathbf{u}(k) \in \mathbb{R}^{n_u}$ e $\mathbf{e}(k) \in \mathbb{R}^{n_y}$ são os vetores das saídas, das entradas e dos erros no instante de amostragem k, respectivamente, N_a e N_b são as ordens do modelo em relação a $\mathbf{y}$ e a $\mathbf{u}$, respectivamente, d é o tempo morto, n_y é

o número de saídas e n_u é o número de entradas. A Equação 2.1 pode ser reescrita como:

$$\mathbf{y}(k)^T = \boldsymbol{\varphi}(k)^T \mathbf{X} + \mathbf{e}(k)^T \tag{2.2}$$

em que $\boldsymbol{\varphi}(k)$ (vetor de regressão no instante k) e $\mathbf{X}$ (matrix de parâmetros) são dados por:

$$\boldsymbol{\varphi}(k) = \begin{bmatrix} \mathbf{y}(k-1)^T & \cdots & \mathbf{y}(k-N_a)^T & \mathbf{u}(k)^T & \cdots & \mathbf{u}(k-N_b)^T \end{bmatrix}^T \in \mathbb{R}^{N_a n_y + (N_b+1) n_u}$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 & \cdots & \mathbf{A}_{N_a} & \mathbf{B}_0 & \mathbf{B}_1 & \cdots & \mathbf{B}_{N_b} \end{bmatrix}^T \in \mathbb{R}^{N_a n_y + (N_b+1) n_u \times n_y}$$

sendo que $\mathbf{B}_i = \mathbf{0}$ se $i < d$. Avaliando a Equação 2.2 em N instantes de tempo consecutivos e em seguida empilhando essas N novas equações, chega-se à seguinte expressão:

$$\mathbf{Y} = \boldsymbol{\Phi} \mathbf{X} + \mathbf{E} \tag{2.3}$$

em que $\mathbf{Y}$ (matriz das saídas), $\boldsymbol{\Phi}$ (matriz de regressão) e $\mathbf{E}$ (matriz dos erros) são dados por:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}(k_0) & \mathbf{y}(k_0+1) & \cdots & \mathbf{y}(k_0+N-1) \end{bmatrix}^T \in \mathbb{R}^{N \times n_y}$$

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\varphi}(k_0) & \boldsymbol{\varphi}(k_0+1) & \cdots & \boldsymbol{\varphi}(k_0+N-1) \end{bmatrix}^T \in \mathbb{R}^{N \times N_a n_y + (N_b+1) n_u}$$

$$\mathbf{E} = \begin{bmatrix} \mathbf{e}(k_0) & \mathbf{e}(k_0+1) & \cdots & \mathbf{e}(k_0+N-1) \end{bmatrix}^T \in \mathbb{R}^{N \times n_y}$$

sendo que $k_0 = \max(N_a, N_b) + 1$ é o instante de tempo inicial. Assim sendo, estando N_a, N_b e d especificados, o objetivo da identificação de sistemas é determinar a matriz de parâmetros $\mathbf{X}$ que minimize $\mathbf{E}$, usando para isso N medidas consecutivas de $\mathbf{y}$ e de $\boldsymbol{\varphi}$ (PEREPU e TANGIRALA, 2017).

No entanto, de modo geral, N_a, N_b e d não são conhecidos *a priori*. Logo, além da matriz $\mathbf{X}$, torna-se necessário identificar também estes três parâmetros. Para tal, deve-se primeiro superestimar o tamanho de $\mathbf{X}$, atribuindo valores suficientemente grandes para N_a e N_b e fazendo $d = 0$. Em seguida, utiliza-se um método matemático para determinar o valor de todos os seus elementos. Como o tamanho de $\mathbf{X}$ foi inicialmente superestimado, espera-se que esta matriz seja esparsa, ou seja, que ela apresente uma grande quantidade de elementos nulos. Finalmente, através da análise do padrão de esparsidade de $\mathbf{X}$, obtêm-se os valores de reais N_a, N_b e d.

Para determinar o valor dos elementos de $\mathbf{X}$, pode-se utilizar o método dos Mínimos Quadrados Ordinários (*Ordinary Least Squares* – OLS), que se baseia na resolução do seguinte problema de otimização:

$$\min_{\mathbf{X}} \|\mathbf{E}\|_F^2 = \min_{\mathbf{X}} \|\mathbf{Y} - \boldsymbol{\Phi}\mathbf{X}\|_F^2 \tag{2.4}$$

em que $\|\mathbf{E}\|_F$ é a norma de Frobenius da matriz $\mathbf{E}$, ou seja, é a soma do quadrado de todos os seus elementos. O problema de minimização da Equação 2.4 apresenta solução analítica:

$$\mathbf{X} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{Y} \tag{2.5}$$

Apesar de prático, o método dos Mínimos Quadrados Ordinários apresenta dois grandes inconvenientes: a matriz $\mathbf{X}$ assim determinada não é esparsa e a técnica só pode ser usada se a matriz $\mathbf{\Phi}^T\mathbf{\Phi}$ estiver bem condicionada.

2.3.2 Mínimos Quadrados Regularizados

Outro jeito de determinar $\mathbf{X}$ é utilizando as chamadas técnicas de Mínimos Quadrados Regularizados (*Regularized Least Squares* – RLS), como a regularização de Tikhonov, o LASSO (*Least Absolute Selection and Shrinkage Operator*) e o LASSO Adaptativo. Tais procedimentos são vistos a seguir.

Regularização de Tikhonov

Neste método, proposto por TIKHONOV (1963), resolve-se o seguinte problema de otimização para determinar $\mathbf{X}$:

$$\min_{\mathbf{X}} \|\mathbf{Y} - \mathbf{\Phi}\mathbf{X}\|_F^2 + \lambda\|\mathbf{X}\|_F^2 \tag{2.6}$$

em que λ (parâmetro de regularização) é um escalar positivo que deve ser escolhido de modo a fixar a importância relativa de $\|\mathbf{X}\|_F^2$ no problema da Equação 2.6. Este problema de minimização também tem solução analítica:

$$\mathbf{X} = (\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{\Phi}^T\mathbf{Y} \tag{2.7}$$

em que $\mathbf{I}$ é a matriz identidade de ordem $N_a n_y + (N_b + 1)n_u$. De maneira geral, a matriz $\mathbf{X}$ assim obtida não é esparsa. Porém, esta técnica pode ser usada mesmo se a matriz $\mathbf{\Phi}^T\mathbf{\Phi}$ estiver mal condicionada, desde que seja feita uma escolha adequada para λ.

GOLUB *et al.* (1979) propuseram um método bastante prático para a escolha de λ, chamado de Validação Cruzada Generalizada (*Generalized Cross-Validation* – GCV). Nesse método, tem-se que $\lambda = N\omega$, sendo ω a solução do problema de otimização abaixo:

$$\min_{\omega>0} f(\omega) = \min_{\omega>0} \frac{\frac{1}{N}\|(\mathbf{I} - \mathbf{M}(\omega))\mathbf{Y}\|_F^2}{\left[\frac{1}{N}\operatorname{tr}(\mathbf{I} - \mathbf{M}(\omega))\right]^2} \tag{2.8}$$

em que $\mathbf{M}(\omega) = \mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}\mathbf{\Phi}^T$ e $\text{tr}(\mathbf{I} - \mathbf{M}(\omega))$ é o traço da matriz $\mathbf{I} - \mathbf{M}(\omega)$, ou seja, é a soma dos elementos de sua diagonal principal.

LASSO (*Least Absolute Selection and Shrinkage Operator*)

Proposto por TIBSHIRANI (1996), o LASSO é um método para determinar $\mathbf{X}$ baseado na resolução do seguinte problema de otimização:

$$\min_{\mathbf{X}} \|\mathbf{Y} - \mathbf{\Phi X}\|_F^2 + \lambda \sum_{i=1}^{n_{lin}} \sum_{j=1}^{n_{col}} |x_{ij}| \tag{2.9}$$

em que $n_{lin} = N_a n_y + (N_b + 1)n_u$ e $n_{col} = n_y$ são o número de linhas e o número de colunas de $\mathbf{X}$, respectivamente, e x_{ij} é o seu elemento de índice (i,j). Este método tem a vantagem de gerar uma matriz $\mathbf{X}$ esparsa, cujo grau de esparsidade pode ser controlado pelo parâmetro λ: se $\lambda = 0$, então $\mathbf{X}$ é igual à solução de Mínimos Quadrados Ordinários (matriz cheia) e se $\lambda \to \infty$, então $\mathbf{X} \to \mathbf{0}$.

No entanto, o problema de minimização da Equação 2.9 não tem solução analítica e a função módulo não é diferenciável em zero. Logo, algoritmos de otimização que utilizam derivadas não podem ser empregados e o problema LASSO fica bastante complicado de ser resolvido. Além disso, diferentemente da regularização de Tikhonov, não é tão fácil encontrar um valor ótimo para o λ do LASSO.

Um jeito rápido e eficiente de resolver o problema LASSO é com a utilização do algoritmo LARS (*Least Angle Regression*) proposto por EFRON *et al.* (2004). Para utilizá-lo, é necessário decompor a equação matricial $\mathbf{Y} = \mathbf{\Phi X} + \mathbf{E}$ em n_y equações vetoriais do tipo:

$$\mathbf{y}_j = \mathbf{\Phi x}_j + \mathbf{e}_j, \qquad j = 1, ..., n_y \tag{2.10}$$

em que $\mathbf{y}_j$, $\mathbf{x}_j$ e $\mathbf{e}_j$ são as colunas de índice j de $\mathbf{Y}$, $\mathbf{X}$ e $\mathbf{E}$, respectivamente. Cada equação vetorial origina um problema LASSO da seguinte forma:

$$\min_{\mathbf{x}_j} \|\mathbf{y}_j - \mathbf{\Phi x}_j\|^2 + \lambda \sum_{i=1}^{n_{lin}} |x_{ij}|, \qquad j = 1, ..., n_y \tag{2.11}$$

O algoritmo LARS pode então ser utilizado para resolver esses problemas separadamente. O algoritmo funciona assim:

1. Primeiramente, o vetor $\mathbf{y}_j$ é centralizado e as colunas de $\mathbf{\Phi}$ são centralizadas e normalizadas (um vetor está centralizado quando a sua média é zero e normalizado quando a sua norma é um).

2. Seja $\widehat{\mathbf{y}}_j = \mathbf{\Phi x}_j$ um estimador de $\mathbf{y}_j$ que vai se deslocando a cada passo do algoritmo. Partindo de $\widehat{\mathbf{y}}_j = \mathbf{0}$ ($\mathbf{x}_j = \mathbf{0}$), toma-se a direção da coluna de $\mathbf{\Phi}$

que está mais correlacionada à $\mathbf{y}_j - \widehat{\mathbf{y}}_j$, ou seja, que está mais próxima de ser colinear à $\mathbf{y}_j - \widehat{\mathbf{y}}_j$. Desloca-se no sentido de se aproximar de $\mathbf{y}_j$.

3. Vai chegar o momento em que outra coluna de $\mathbf{\Phi}$ vai estar igualmente correlacionada à $\mathbf{y}_j - \widehat{\mathbf{y}}_j$. Nesse instante, registra-se o valor do vetor $\mathbf{x}_j$ correspondente a $\widehat{\mathbf{y}}_j$ e toma-se a direção equiangular a estas colunas (quando somente duas colunas estão envolvidas, essa direção é a bissetriz). Desloca-se novamente no sentido de se aproximar de $\mathbf{y}_j$.

4. Repete-se a etapa 3 até que a direção tomada seja equiangular a todas as colunas de $\mathbf{\Phi}$. Quando isso acontece, $\widehat{\mathbf{y}}_j$ se iguala à projeção ortogonal de $\mathbf{y}_j$ sobre o espaço das colunas de $\mathbf{\Phi}$, $\mathbf{x}_j$ se iguala à solução OLS da Equação 2.10 e a execução do algoritmo chega ao fim.

5. A cada passo do algoritmo, um novo vetor $\mathbf{x}_j$ é gerado. Esses vetores são na verdade modelos matemáticos que descrevem $\mathbf{y}_j$ como uma combinação linear de certas colunas de $\mathbf{\Phi}$. Ao final da execução do LARS, tem-se um conjunto de modelos.

6. Finalmente, os modelos $\mathbf{x}_j$ são escritos em termos das colunas de $\mathbf{\Phi}$ originais (não normalizadas):

$$x_{ij} \longrightarrow \frac{x_{ij}}{\|\boldsymbol{\varphi_i}\|}, \qquad i = 1, ..., n_{lin}$$

em que $\|\boldsymbol{\varphi_i}\|$ é a norma da i-ésima coluna de $\mathbf{\Phi}$.

Em sua forma original, o LARS gera alguns modelos que são soluções do problema LASSO vetorial e outros que não são. No entanto, uma pequena modificação do algoritmo acima – maiores detalhes podem ser vistos em EFRON *et al.* (2004) – faz com que o LARS seja capaz de gerar unicamente todas as soluções do problema da Equação 2.11. Em outras palavras, o LARS gera as soluções LASSO que seriam obtidas ao fazer λ variar de 0 até infinito. A escolha da solução mais adequada deve ser feita com o auxílio de um critério de seleção.

Assim, deve-se utilizar o algoritmo LARS para resolver o problema LASSO associado a cada uma das n_y equações vetoriais. Cada solução vetorial representa uma coluna de $\mathbf{X}$. O conjunto de soluções fornece toda a matriz $\mathbf{X}$.

LASSO Adaptativo

Este método, proposto por ZOU (2006), é na verdade uma versão melhorada do LASSO clássico. Deve-se, primeiramente, decompor a equação $\mathbf{Y} = \mathbf{\Phi X} + \mathbf{E}$ em n_y

equações vetoriais do tipo $\mathbf{y}_j = \mathbf{\Phi}\mathbf{x}_j + \mathbf{e}_j$. Em seguida, resolve-se para cada equação vetorial o seguinte problema de otimização:

$$\min_{\mathbf{x}_j} \|\mathbf{y}_j - \mathbf{\Phi}\mathbf{x}_j\|^2 + \lambda \sum_{i=1}^{n_{lin}} w_{ij}^{LA} |x_{ij}|, \qquad j = 1, ..., n_y \tag{2.12}$$

em que $w_{ij}^{LA} = 1/|\widehat{x}_{ij}|^\gamma$, sendo $\widehat{x}_{ij}$ uma estimativa inicial de x_{ij} (obtida, por exemplo, por Mínimos Quadrados Ordinários) e γ uma constante positiva. Uma boa escolha é fazer $\gamma = 1$, pois nesse caso o LASSO adaptativo é consistente para seleção de variáveis (ZOU, 2006). O conjunto de todas as soluções vetoriais $\mathbf{x}_j$ fornece a matriz de parâmetros $\mathbf{X}$.

A principal vantagem deste método é a de que ele consegue determinar com maior exatidão o padrão de esparsidade de $\mathbf{X}$. Além disso, fazendo as mudanças de variável $\widetilde{x}_{ij} = w_{ij}^{LA} x_{ij}$ e $\widetilde{\varphi}_{ij} = \varphi_{ij}/w_{ij}^{LA}$, temos o seguinte problema de otimização, que pode ser resolvido com o algoritmo LARS:

$$\min_{\widetilde{\mathbf{x}}_j} \|\mathbf{y}_j - \widetilde{\mathbf{\Phi}}\widetilde{\mathbf{x}}_j\|^2 + \lambda \sum_{i=1}^{n_{lin}} |\widetilde{x}_{ij}|, \qquad j = 1, ..., n_y \tag{2.13}$$

Assim, dentre os diferentes métodos disponíveis na literatura, verifica-se que o LASSO adaptativo (em conjunto com o algoritmo LARS) é o mais adequado para a identificação de sistemas MIMO com modelos ARX. No entanto, como visto anteriormente, para utilizar o algoritmo LARS é necessário escolher um critério de seleção de modelos. Este assunto é abordado a seguir.

2.3.3 Seleção de Modelos

ZOU *et al.* (2007) compararam os seguintes critérios de seleção de modelos no contexto da regularização LASSO com o algoritmo LARS: Cp de Mallows (MALLOWS, 1973), Critério de Informação de Akaike – AIC (AKAIKE, 1974) – e Critério de Informação Bayesiano – BIC (SCHWARZ, 1978). Os autores concluíram que o critério mais adequado é o BIC, pois este é o que melhor consegue evitar a superparametrização de modelos.

O BIC foi proposto por SCHWARZ (1978) e parte do pressuposto de que o modelo real se encontra no conjunto das soluções LASSO. Nesse cenário, o modelo real será aquele que apresentar o menor valor para o BIC. No entanto, este critério só é adequado quando a quantidade de pontos experimentais utilizada no processo de identificação for muito grande (tendendo ao infinito).

CHEN e CHEN (2008) modificaram o BIC original de modo a considerar os casos em que se tem poucos dados, ou seja, quando o número de pontos experimentais é muito pequeno em relação ao número de parâmetros do modelo. O novo critério,

denominado Critério de Informação Bayesiano Estendido (EBIC), pode ser definido da seguinte forma para o modelo $\mathbf{x}_j$ solução do problema da Equação 2.11:

$$EBIC = N\ln(S/N) + (n_{par}+1)\ln N + 2\sigma\ln\xi \tag{2.14}$$

em que:

$$S = \sum_{i=1}^{N} e_{ij}^2 \qquad \sigma = \max\left(0; 1 - \frac{1}{2\kappa}\right)$$
$$\xi = \binom{N_a n_y + (N_b+1)n_u}{n_{par}} \qquad \kappa = \frac{\ln(N_a n_y + (N_b+1)n_u)}{\ln N}$$

sendo n_{par} o número de parâmetros do modelo, ou seja, o número de elementos não nulos de $\mathbf{x}_j$. É importante observar que ξ é definido em termos de um binômio e que se o termo $2\sigma\ln\xi$ for removido da Equação 2.14, a expressão resultante será a definição original do BIC.

Semelhantemente ao que BANKS e JOYNER (2017) fizeram para o AIC, a Equação 2.14 está escrita na formulação OLS, na qual assume-se que os componentes de $\mathbf{e}_j$ têm distribuição normal com média 0 e igual variância (homocedasticidade).

2.4 Identificação Caixa Cinza

2.4.1 Técnicas de Modelagem

Os processos da indústria química podem ser modelados por meio de três técnicas distintas, chamadas de modelagem caixa preta, caixa branca e caixa cinza. A Tabela 2.2 sintetiza estas técnicas de modelagem.

Tabela 2.2: Técnicas de modelagem.

Técnica de modelagem	Natureza da modelagem	É fácil de ser realizada?	Traz informações sobre a física do processo?	Exemplo
Caixa preta	Puramente empírica	Sim	Não	ORENSTEIN (2013)
Caixa branca	Puramente fenomenológica	Não	Sim	Balanços de massa, energia e quantidade de movimento
Caixa cinza	Meio empírica, meio fenomenológica	Sim	Sim	TULLEKEN (1993)

Observa-se a partir da Tabela 2.2 que a modelagem caixa cinza junta os aspectos positivos das modelagens caixa preta e caixa branca, ou seja, é fácil de ser realizada e também traz informações sobre a física do processo que está sendo modelado.

2.4.2 Métodos de Modelagem Caixa Cinza

SOHLBERG e JACOBSEN (2008) separaram os métodos de modelagem caixa cinza em diferentes categorias. A Tabela 2.3 sintetiza o trabalho desses autores.

Tabela 2.3: Métodos de modelagem caixa cinza (SOHLBERG e JACOBSEN, 2008).

Método de modelagem caixa cinza	Descrição
Caixa preta com restrição	São impostas restrições físicas sobre os parâmetros empíricos do modelo.
Semi-física	Informação fenomenológica é utilizada para a criação de regressores não lineares nas entradas e saídas.
Mecanística	Modelos fenomenológicos são adaptados para se adequarem aos dados experimentais.
Híbrida	Separa o processo em um modelo caixa preta e outro caixa branca (ou cinza).

Como um dos objetivos do presente trabalho é modificar a metodologia de identificação caixa preta de ORENSTEIN (2013) de modo a incluir informação fenomenológica, o método de modelagem caixa cinza mais adequado é o do tipo caixa preta com restrição. Entretanto, esta modelagem é normalmente muito específica, tendo sua aplicação limitada a uma determinada classe de sistemas. O método de MURAKAMI e SEBORG (2000), por exemplo, pode ser aplicado somente a sistemas de mistura. Uma importante excessão é o método de TULLEKEN (1993), visto em detalhes a seguir.

2.4.3 Método de Tulleken

Segundo TULLEKEN (1993), os métodos de identificação caixa preta podem gerar modelos fisicamente inconsistentes. Isso é devido, principalmente, à escassez de dados experimentais e à presença de ruído. Contudo, tal problema pode ser resolvido por meio da incorporação de restrições sobre os parâmetros do modelo que levem em consideração a localização dos polos do sistema (estabilidade) e o valor de seus ganhos estáticos. Para exemplificar, considere o modelo ARX da Equação 2.1 com $\mathbf{e}(k) = \mathbf{0}$ e $d = 0$:

$$\mathbf{y}(k) = \mathbf{A}_1\mathbf{y}(k-1) + \cdots + \mathbf{A}_{N_a}\mathbf{y}(k-N_a) + \mathbf{B}_0\mathbf{u}(k) + \cdots + \mathbf{B}_{N_b}\mathbf{u}(k-N_b) \quad (2.15)$$

A transformada Z é a versão discreta da transformada de Laplace. Para um sinal

discreto $\mathbf{s}$(k), ela é calculada da seguinte forma:

$$\mathbf{S}(z) = \mathcal{Z}\{\mathbf{s}(k)\} = \sum_{k=0}^{\infty} \mathbf{s}(k) z^{-k} \tag{2.16}$$

em que $z \in \mathbb{C}$ é a variável do domínio transformado. Da Equação 2.16 decorre a seguinte propriedade:

$$\mathcal{Z}\{\mathbf{s}(k-m)\} = \mathbf{S}(z) z^{-m} \tag{2.17}$$

em que m é um número inteiro positivo. Aplicando a transformada Z nos dois lados da Equação 2.15 e usando a propriedade da Equação 2.17, chega-se à seguinte expressão:

$$\boldsymbol{\mathcal{A}}(z^{-1})\mathbf{Y}(z^{-1}) = \boldsymbol{\mathcal{B}}(z^{-1})\mathbf{U}(z^{-1}) \tag{2.18}$$

em que:

$$\boldsymbol{\mathcal{A}}(z^{-1}) = \mathbf{I} - \sum_{k=1}^{N_a} \mathbf{A}_k z^{-k}$$

$$\boldsymbol{\mathcal{B}}(z^{-1}) = \sum_{k=0}^{N_b} \mathbf{B}_k z^{-k}$$

sendo $\mathbf{I}$ a matriz identidade de ordem n_y. Isolando $\mathbf{Y}(z^{-1})$ no lado esquerdo da Equação 2.18 resulta em:

$$\mathbf{Y}(z^{-1}) = (\boldsymbol{\mathcal{A}}(z^{-1}))^{-1}\boldsymbol{\mathcal{B}}(z^{-1})\mathbf{U}(z^{-1}) \tag{2.19}$$

Analisando a Equação 2.19, verifica-se que a matriz de transferência do processo, simbolizada por $\mathbf{G_p}(z^{-1})$, é dada por:

$$\mathbf{G_p}(z^{-1}) = (\boldsymbol{\mathcal{A}}(z^{-1}))^{-1}\boldsymbol{\mathcal{B}}(z^{-1}) = \frac{1}{\det(\boldsymbol{\mathcal{A}}(z^{-1}))} \operatorname{adj}(\boldsymbol{\mathcal{A}}(z^{-1}))\boldsymbol{\mathcal{B}}(z^{-1}) \tag{2.20}$$

em que $\det(\boldsymbol{\mathcal{A}}(z^{-1}))$ e $\operatorname{adj}(\boldsymbol{\mathcal{A}}(z^{-1}))$ são o determinante e a adjunta de $\boldsymbol{\mathcal{A}}(z^{-1})$, respectivamente (vale lembrar que a adjunta de uma matriz é a transposta da matriz de seus cofatores).

Segundo a Equação 2.20, os elementos de $\mathbf{G_p}$ são funções racionais de z^{-1}, sendo que o grau máximo dos denominadores é $N_a n_y$ e dos numeradores é $N_a(n_y - 1) + N_b$. Além disso, os polos do sistema (r_n) são as soluções da seguinte equação polinomial:

$$\det(\boldsymbol{\mathcal{A}}(z^{-1})) = 0 \tag{2.21}$$

Para o sistema ser estável, os polos devem estar localizados no interior do círculo unitário centrado na origem do plano complexo, ou seja, as restrições sobre os polos

do sistema devem ser tais que:

$$|r_n| < 1, \qquad n = 1, ..., n_r \tag{2.22}$$

em que n_r é o número total de polos, sendo igual ao grau do polinômio $\det(\boldsymbol{\mathcal{A}}(z^{-1}))$.

As restrições sobre os elementos da matriz de ganhos estáticos do processo ($\mathbf{K}$) devem ser tais que:

$$\mathbf{K}^{inf} \leqslant \mathbf{K} \leqslant \mathbf{K}^{sup} \tag{2.23}$$

em que $\mathbf{K}^{inf}$ e $\mathbf{K}^{sup}$ são os limites inferior e superior para $\mathbf{K}$, respectivamente. Se o processo for estável, a matriz $\mathbf{K}$ pode ser calculada a partir de $\mathbf{G_p}$:

$$\mathbf{K} = \mathbf{G_p}(1) = (\boldsymbol{\mathcal{A}}(1))^{-1}\boldsymbol{\mathcal{B}}(1) \tag{2.24}$$

Substituindo a Equação 2.24 nas restrições dadas pela Equação 2.23:

$$\mathbf{K}^{inf} \leqslant (\boldsymbol{\mathcal{A}}(1))^{-1}\boldsymbol{\mathcal{B}}(1) \leqslant \mathbf{K}^{sup} \tag{2.25}$$

A Equação 2.25, que impõe restrições sobre os parâmetros do modelo ARX, pode ser utilizada para fazer com que os modelos gerados fiquem com os ganhos estáticos dentro de uma faixa aceitável de valores. Portanto, neste trabalho utilizou-se o método de Tulleken para introduzir informação fenomenológica na identificação caixa preta de Orenstein, assegurando assim que o valor e o sinal dos ganhos estáticos estimados tenham sentido físico.

2.5 Planejamento de Sinais GBN

No planejamento do sinal GBN da entrada u_j, a escolha adequada do parâmetro p_j (probabilidade de não mudança de nível) faz com que a energia do sinal fique concentrada em uma região específica de seu espectro de potência. Apesar de conhecida, esta característica não foi explorada por ORENSTEIN (2013) em seu trabalho.

CHEN e YU (1997) demonstraram analiticamente que, quando se deseja concentrar a potência do sinal GBN de u_j entre as frequências $\nu_{min,j}$ e $\nu_{max,j}$, deve-se adotar o seguinte valor ótimo para p_j:

$$p_j^* = \frac{1}{1 + \sqrt{\tan\left(\frac{\nu_{min,j}T_{min}}{2}\right)\tan\left(\frac{\nu_{max,j}T_{min}}{2}\right)}} \tag{2.26}$$

em que p_j^* é a probabilidade ótima de não mudança de nível de u_j e T_{min} é o período de amostragem.

A escolha do intervalo de frequências está relacionada à resposta do sistema a um degrau em u_j. Segundo GAIKWAD e RIVERA (1996), uma boa escolha é o seguinte intervalo:

$$\nu_{min,j} = \frac{1}{\beta\tau_{max,j}} \leqslant \nu_j \leqslant \frac{\alpha}{\tau_{min,j}} = \nu_{max,j} \tag{2.27}$$

em que $\alpha = 2$ e $\beta = 3$. $\tau_{min,j}$ e $\tau_{max,j}$ são a menor e a maior constante de tempo obtidas para a entrada u_j, respectivamente. Estas constantes podem ser determinadas utilizando-se o método de SMITH (1972), ilustrado na Figura 2.4.

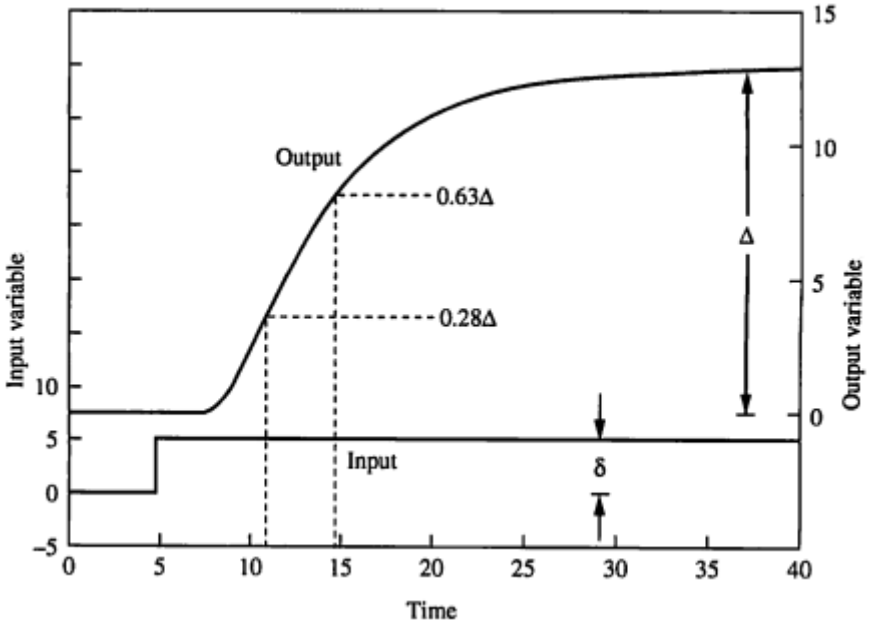

Figura 2.4: Método de Smith. O gráfico mostra a resposta de y_i para um degrau em u_j. Fonte: Adaptado de MARLIN (2015).

A partir da Figura 2.4, a constante de tempo τ_{ij} associada à resposta de y_i a um degrau em u_j pode ser calculada pela expressão abaixo:

$$\tau_{ij} = 1,5(t_{63\%} - t_{28\%}) \tag{2.28}$$

em que $t_{63\%}$ e $t_{28\%}$ são os tempos necessários para a saída y_i atingir 63% e 28% de seu valor estacionário final, respectivamente. Considerando todas as saídas, tem-se que:

$$\tau_{min,j} = \min_i(\tau_{ij}) \tag{2.29}$$

$$\tau_{max,j} = \max_i(\tau_{ij}) \tag{2.30}$$

HUNG *et al.* (2015) utilizaram a fórmula de CHEN e YU (1997) com a sugestão de GAIKWAD e RIVERA (1996) para identificar um sistema MIMO 2 x 2. O mesmo problema de identificação foi também resolvido com a heurística de ZHU e VAN DEN BOSCH (2000). Os sinais GBN foram aplicados durante 3600 segundos (cerca de 7,4 vezes o maior tempo de assentamento do sistema) e os modelos assim identificados foram implementados em um MPC. Os autores constataram que a fórmula de CHEN e YU (1997) dava melhores resultados.

Assim sendo, neste trabalho, os valores de p_j foram determinados a partir da fórmula de CHEN e YU (1997) e da sugestão de GAIKWAD e RIVERA (1996).

Capítulo 3

Metodologia Proposta

A metodologia de identificação proposta neste trabalho é o resultado da incorporação das melhorias sugeridas no capítulo anterior ao método de Orenstein. Ela apresenta etapas *on-line*, executadas diretamente na planta industrial, e *off-line*, executadas pelo pacote computacional IDENTIPHY (*system IDENTIfication using PHYsical knowledge*).

3.1 O Pacote Computacional IDENTIPHY

O pacote IDENTIPHY é composto de quatro algoritmos, cujos códigos e instruções de uso são mostrados no Apêndice B. Para executá-los, é necessário possuir o *software* MATLAB (versão R2008a ou mais recente) com a *Control System Toolbox* e a *Optimisation Toolbox*. Uma descrição de cada algoritmo é apresentada a seguir.

3.1.1 Algoritmo 1

O Algoritmo 1 faz a análise dos dados obtidos na etapa do pré-teste. Mais precisamente, a partir da resposta de y_i ao degrau em u_j, o algoritmo calcula:

1. O ganho estático K_{ij}, definido pela equação abaixo:

$$K_{ij} = \frac{\Delta y_i^{ee}}{\Delta u_j^{ee}} \tag{3.1}$$

   em que Δy_i^{ee} e Δu_j^{ee} são as variações nos valores de estado estacionário de y_i e u_j, respectivamente.

2. A constante de tempo τ_{ij}, definida pela Equação 2.28.

3. O tempo de assentamento t_{ij}^{ass}, definido como sendo o tempo necessário para a resposta de y_i atingir 100(1−ST)% de seu valor estacionário final. O pa-

râmetro ST (*Settling time Threshold*) deve ser fornecido pelo usuário, sendo normalmente utilizado o valor de 0,02 (tempo de assentamento a 98%).

Como no pré-teste há uma resposta para y_i quando o degrau em u_j é aplicado e outra quando este mesmo degrau é removido, o algoritmo gera duas medidas para cada um dos parâmetros K_{ij}, τ_{ij} e t_{ij}^{ass}. Assim sendo, tem-se o seguinte conjunto total de medidas: $\{K_{ij1}, K_{ij2}, \tau_{ij1}, \tau_{ij2}, t_{ij1}^{ass}, t_{ij2}^{ass}\}$. Este procedimento é realizado para todos os pares (y_i, u_j), em que $i = 1, ..., n_y$ e $j = 1, ..., n_u$.

Considerando todas as saídas e todas as medidas, tem-se que o tempo de assentamento do sistema em relação à u_j ($t_{max,j}^{ass}$) é dado por:

$$t_{max,j}^{ass} = \max_{i,k}(t_{ijk}^{ass}) \tag{3.2}$$

em que i é o índice da variável de saída e k é o índice da medida. Assim, a partir das amplitudes dos degraus do pré-teste e da Equação 3.2, o algoritmo faz também o planejamento de sinais para a etapa do teste degrau, utilizando para isso uma generalização do procedimento descrito na Seção 2.2.2, na qual é permitido ao usuário escolher o número de partes das sequências de degraus (n_p) e a duração destas perturbações.

3.1.2 Algoritmo 2

Primeiramente, o Algoritmo 2 faz a análise dos dados obtidos na etapa do teste degrau utilizando a mesma metodologia do Algoritmo 1. No entanto, como nesta etapa é aplicada uma sequência de $2n_p$ degraus em u_j, o algoritmo gera $2n_p + 1$ medidas para cada um dos parâmetros K_{ij}, τ_{ij} e t_{ij}^{ass} do par (y_i, u_j). Em seguida, é realizado o tratamento conjunto dos dados obtidos no pré-teste e no teste degrau. Ao todo, tem-se $2n_p + 3$ medidas de cada um destes três parâmetros.

Para cada entrada u_j, considerando todas as saídas e todas as medidas, o algoritmo atualiza o valor de $t_{max,j}^{ass}$ com a Equação 3.2 e calcula os valores de $\tau_{min,j}$ e $\tau_{max,j}$ por meio das seguintes expressões:

$$\tau_{min,j} = \min_{i,k}(\tau_{ijk}) \tag{3.3}$$

$$\tau_{max,j} = \max_{i,k}(\tau_{ijk}) \tag{3.4}$$

em que k é o índice da medida.

Para cada par (y_i, u_j), o algoritmo assume que as medidas de K_{ij} são variáveis aleatórias contínuas com distribuição normal e calcula o intervalo de confiança de Student a 99,99%. Os limites inferior e superior deste intervalo são os limites aceitáveis para o verdadeiro valor de K_{ij}, ou seja, são K_{ij}^{inf} e K_{ij}^{sup}, respectivamente.

Se K_{ij}^{inf} e K_{ij}^{sup} tiverem sinais opostos, o sinal de K_{ij} fica indeterminado. Isso acontece, por exemplo, quando o ruído em y_i for muito intenso. Nessa situação, ilustrada na Figura 3.1, o algoritmo pergunta ao usuário se ele sabe qual é o sinal de K_{ij}. Caso ele saiba, deve digitar 1 se o ganho for positivo, 0 se for nulo ou -1 se for negativo. Caso não saiba, não deve digitar nada. Assim, se o sinal for fornecido, K_{ij}^{inf} e/ou K_{ij}^{sup} serão modificados de modo a assegurar que K_{ij} tenha o sinal informado: se o usuário digitar 1, então $K_{ij}^{inf} = 0$; se digitar -1, então $K_{ij}^{sup} = 0$ e se digitar 0, então $K_{ij}^{inf} = K_{ij}^{sup} = 0$.

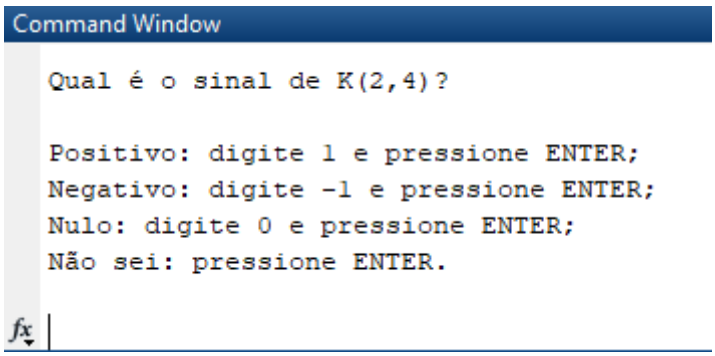

Figura 3.1: Execução do Algoritmo 2 no MATLAB. Neste exemplo, como K_{24}^{inf} e K_{24}^{sup} tem sinais opostos, o algoritmo pergunta ao usuário se ele sabe o sinal de K_{24}.

Caso o processo esteja sendo identificado pela primeira vez, deve-se utilizar informação fenomenológica para determinar os sinais desconhecidos de $\mathbf{K}$, ou seja, conhecimento físico acerca das interações entre as entradas e as saídas. Caso contrário, recomenda-se fazer uso da tabela de direções do processo, na qual estão inseridos os sinais dos ganhos estáticos obtidos na última identificação. Exemplos desse tipo de tabela são encontrados em ORENSTEIN (2013).

Às vezes, por razões de segurança ou pelo fato dos valores de $t_{max,j}^{ass}$ apresentarem ordens de grandeza muito diferentes, é conveniente separar as entradas u_j em grupos para a etapa do projeto GBN. O Algoritmo 2 também pode ser usado nesse contexto, permitindo a criação, por exemplo, do grupo das entradas de dinâmica rápida (baixos valores de $t_{max,j}^{ass}$) e do grupo das entradas de dinâmica lenta (altos valores de $t_{max,j}^{ass}$). Mais detalhes podem ser vistos na Seção B.2 do Apêndice B.

3.1.3 Algoritmo 3

A partir dos dados de entrada e saída e dos valores de N_a e N_b fornecidos pelo usuário, o Algoritmo 3 identifica a melhor matriz de transferência para o processo. A Figura 3.2 mostra a estrutura deste algoritmo em diagrama de blocos.

Primeiramente, é verificado se a matriz $\mathbf{\Phi}^T\mathbf{\Phi}$ está bem condicionada. Para tal, é aplicado o critério de condicionamento do MATLAB, segundo o qual uma matriz está bem condicionada se o recíproco de seu número de condicionamento for superior

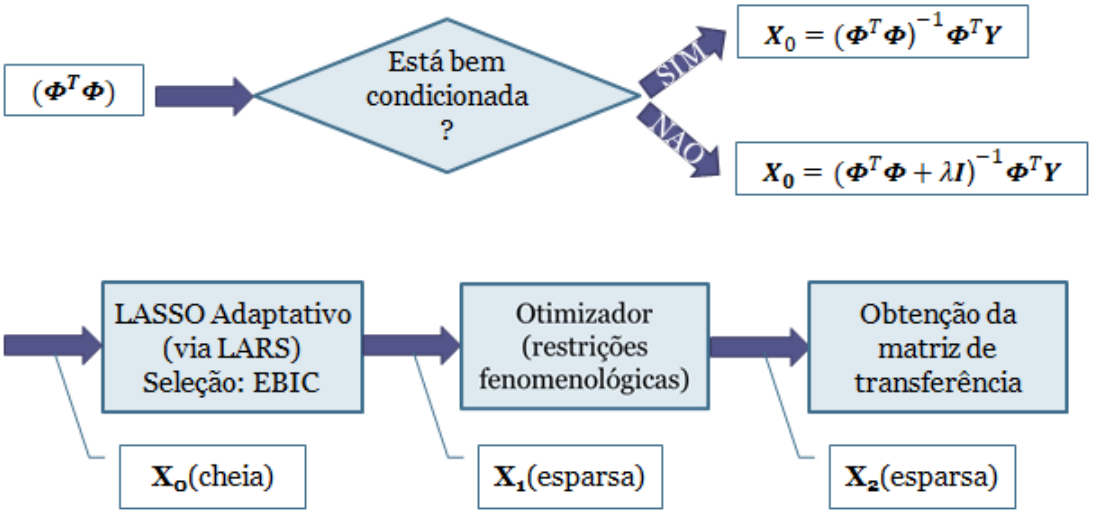

Figura 3.2: Estrutura do Algoritmo 3.

a $eps = 2,22 \times 10^{-16}$. Se ela estiver, calcula-se uma estimativa para a matriz $\mathbf{X}$ utilizando o método dos Mínimos Quadrados Ordinários (OLS). Se não estiver, utiliza-se a regularização de Tikhonov, na qual λ é obtido por Validação Cruzada Generalizada (GCV). Essa estimativa, denominada $\mathbf{X}_0$, é normalmente uma matriz cheia.

Em seguida, a estimativa $\mathbf{X}_0$ é empregada no cálculo da matriz de pesos $\mathbf{W}^{LA}$ do problema LASSO adaptativo, cuja resolução é feita com o algoritmo LARS. Durante sua execução, o LARS seleciona os melhores modelos com base no Critério de Informação Bayesiano Estendido (EBIC). A solução obtida nesta etapa é a matriz esparsa $\mathbf{X}_1$.

Utilizando $\mathbf{X}_1$ como estimativa inicial, o otimizador resolve o seguinte problema:

$$\begin{aligned} \min_{\mathbf{X}} \quad & F(\mathbf{X}) = \|\mathbf{Y} - \mathbf{\Phi X}\|_F^2 \\ \text{s.a.} \quad & |r_n(\mathbf{X})| < R, \qquad n = 1, ..., n_r \\ & \mathbf{K}^{inf} \leqslant \mathbf{K}(\mathbf{X}) \leqslant \mathbf{K}^{sup} \end{aligned} \tag{3.5}$$

no qual as variáveis de decisão são os elementos não nulos de $\mathbf{X}_1$ (o seu padrão de esparsidade é preservado ao longo da otimização). O raio $R \in (0, 1]$ deve ser fornecido pelo usuário e, caso não estejam disponíveis informações mais detalhadas sobre a localização dos polos do sistema, deve-se fazer $R = 1$ para assegurar a estabilidade. As matrizes $\mathbf{K}^{inf}$ e $\mathbf{K}^{sup}$ são fornecidas pelo Algoritmo 2, sendo importante observar que se $K_{ij}^{inf} = K_{ij}^{sup} = 0$, então há uma restrição de igualdade sobre o ganho estático K_{ij}. A relação entre os r_n's e $\mathbf{X}$ é dada implicitamente pela Equação 2.21, cuja resolução é feita com o algoritmo baseado na transformada de Fourier discreta proposto por HROMČÍK e ŠEBEKT (1999). Já a relação entre $\mathbf{K}$ e $\mathbf{X}$, para sistemas

estáveis, é dada explicitamente pela Equação 2.24.

A solução $\mathbf{X}_2$ do problema da Equação 3.5 tem sua consistência física assegurada e, a partir dela, obtém-se a matriz de transferência do processo por meio da Equação 2.20. No entanto, do jeito que as restrições estão formuladas, é bastante complicado de se resolver este problema com algoritmos clássicos de otimização. A seguir, elas são reescritas em uma forma mais adequada.

Reformulação do Problema de Otimização com Restrições Fenomenológicas

Restrições sobre os polos De acordo com o problema da Equação 3.5, as restrições sobre os polos do sistema são tais que:

$$|r_n(\mathbf{X})| < R, \qquad n = 1, ..., n_r \tag{3.6}$$

Entretanto, a derivada de $|r_n(\mathbf{X})|$ em relação à $\mathbf{X}$ não está definida nos pontos onde $r_n(\mathbf{X}) = 0$, o que é inadequado para certos métodos de otimização. Para resolver tal problema, deve-se primeiro observar que a restrição da Equação 3.6 é equivalente a:

$$|r_n(\mathbf{X})|^2 < R^2, \qquad n = 1, ..., n_r \tag{3.7}$$

Em seguida, como $|r_n(\mathbf{X})|^2 = r_n(\mathbf{X})\overline{r}_n(\mathbf{X})$, em que $\overline{r}_n$ é o complexo conjugado de r_n, tem-se que:

$$r_n(\mathbf{X})\overline{r}_n(\mathbf{X}) < R^2, \qquad n = 1, ..., n_r \tag{3.8}$$

Finalmente, o conjunto das restrições da Equação 3.8 pode ser substituído por:

$$\max_n(r_n(\mathbf{X})\overline{r}_n(\mathbf{X})) < R^2 \tag{3.9}$$

ou seja, as n_r restrições iniciais envolvendo a função módulo podem ser substituídas por uma única restrição que não faz uso dessa função. Utilizando a definição $\rho_n(\mathbf{X}) = r_n(\mathbf{X})\overline{r}_n(\mathbf{X})$ na Equação 3.9, chega-se à seguinte expressão:

$$\max_n(\rho_n(\mathbf{X})) < R^2 \tag{3.10}$$

A função $\max_n(\rho_n(\mathbf{X}))$ também tem o incoveniente de, normalmente, não ser diferenciável em toda parte. Assim, ao invés de $\max_n(\rho_n(\mathbf{X}))$, é preferível utilizar uma aproximação diferenciável desta função, como por exemplo a *softmax* – S_a (LANGE *et al.*, 2014). A função $S_a(\mathbf{X})$ é definida da seguinte forma:

$$S_a(\mathbf{X}) = \sum_{n=1}^{n_r} w_n(\mathbf{X})\rho_n(\mathbf{X}) \tag{3.11}$$

em que os pesos $w_n(\mathbf{X})$ são dados por:

$$w_n(\mathbf{X}) = \frac{e^{a\rho_n(\mathbf{X})}}{\sum_{i=1}^{n_r} e^{a\rho_i(\mathbf{X})}} \tag{3.12}$$

sendo $a > 0$. Quanto maior for o valor de a, mais acurada será a aproximação $S_a(\mathbf{X})$. Logo, neste trabalho, escolheu-se utilizar $a = 1 \times 10^6$.

No MATLAB, o cálculo de $w_n(\mathbf{X})$ com a Equação 3.12 gera *overflow*. Apesar de cada peso estar compreendido entre 0 e 1, as funções exponencias da Equação 3.12 geram valores muito elevados quando computadas individualmente, valores estes que são superiores ao maior número que o MATLAB é capaz de registrar. Assim, ao calcular primeiro o numerador de $w_n(\mathbf{X})$, depois o seu denominador e por final fazer a divisão dos dois, o MATLAB obtém os resultados ∞, ∞ e ∞/∞ = indeterminado, respectivamente. Segundo COOK (2011), o *overflow* pode ser evitado por meio de uma alteração na fórmula de cálculo de $w_n(\mathbf{X})$. Isto é feito da seguinte forma:

$$\begin{aligned}
w_n(\mathbf{X}) &= \frac{e^{a\rho_n(\mathbf{X})}}{\sum_{i=1}^{n_r} e^{a\rho_i(\mathbf{X})}} \\
&= \exp\left(\ln\left(\frac{e^{a\rho_n(\mathbf{X})}}{\sum_{i=1}^{n_r} e^{a\rho_i(\mathbf{X})}}\right)\right) \\
&= \exp\left(\ln(e^{a\rho_n(\mathbf{X})}) - \ln\left(\sum_{i=1}^{n_r} e^{a\rho_i(\mathbf{X})}\right)\right) \\
&= \exp\left(a\rho_n(\mathbf{X}) - \ln\left(\sum_{i=1}^{n_r} e^{a\rho_i(\mathbf{X})}\right)\right) \\
&= \exp\left(a\rho_n(\mathbf{X}) - \ln\left(e^b \sum_{i=1}^{n_r} e^{a\rho_i(\mathbf{X})-b}\right)\right) \\
&= \exp\left(a\rho_n(\mathbf{X}) - \ln(e^b) - \ln\left(\sum_{i=1}^{n_r} e^{a\rho_i(\mathbf{X})-b}\right)\right) \\
&= \exp\left(a\rho_n(\mathbf{X}) - b - \ln\left(\sum_{i=1}^{n_r} e^{a\rho_i(\mathbf{X})-b}\right)\right)
\end{aligned} \tag{3.13}$$

em que $b = a\max_n(\rho_n(\mathbf{X}))$. Assim, se o cálculo de $w_n(\mathbf{X})$ for realizado com a Equação 3.13, o *overflow* nunca acontecerá, pois todos os valores de $e^{a\rho_i(\mathbf{X})-b}$ estão compreendidos entre 0 e 1. Fazendo $\max_n(\rho_n(\mathbf{X})) = S_a(\mathbf{X})$ na restrição da Equação 3.10, tem-se:

$$S_a(\mathbf{X}) - R^2 < 0 \tag{3.14}$$

Portanto, a restrição de desigualdade sobre os polos do sistema ($c_{in,0}$) é dada por:

$$c_{in,0}(\mathbf{X}) = S_a(\mathbf{X}) - R^2 \tag{3.15}$$

Restrições sobre os ganhos estáticos De acordo com o problema da Equação 3.5, as restrições sobre os ganhos estáticos do processo são tais que:

$$\mathbf{K}^{inf} \leqslant \mathbf{K}(\mathbf{X}) \leqslant \mathbf{K}^{sup} \tag{3.16}$$

Para poder ser melhor tratada por algoritmos de otimização, as restrições da Equação 3.16 devem ser reformuladas da seguinte maneira:

$$K_{ij}^{inf} \leqslant K_{ij}(\mathbf{X}) \leqslant K_{ij}^{sup}, \qquad \forall (i,j) \text{ tal que } K_{ij}^{inf} < K_{ij}^{sup} \tag{3.17}$$

$$K_{ij}(\mathbf{X}) = K_{ij}^{sup}, \qquad \forall (i,j) \text{ tal que } K_{ij}^{inf} = K_{ij}^{sup} \tag{3.18}$$

em que as igualdades são analisadas separadamente. As restrições das Equações 3.17 e 3.18 são equivalentes a:

$$K_{ij}(\mathbf{X}) - K_{ij}^{sup} \leqslant 0, \qquad \forall (i,j) \text{ tal que } K_{ij}^{inf} < K_{ij}^{sup} \tag{3.19}$$

$$K_{ij}^{inf} - K_{ij}(\mathbf{X}) \leqslant 0, \qquad \forall (i,j) \text{ tal que } K_{ij}^{inf} < K_{ij}^{sup} \tag{3.20}$$

$$K_{ij}(\mathbf{X}) - K_{ij}^{sup} = 0, \qquad \forall (i,j) \text{ tal que } K_{ij}^{inf} = K_{ij}^{sup} \tag{3.21}$$

Portanto, as restrições de desigualdade ($c_{in,ij}^{sup}$ e $c_{in,ij}^{inf}$) e igualdade ($c_{eq,ij}$) sobre os ganhos estáticos do processo são dadas por:

$$c_{in,ij}^{sup}(\mathbf{X}) = K_{ij}(\mathbf{X}) - K_{ij}^{sup}, \qquad \forall (i,j) \text{ tal que } K_{ij}^{inf} < K_{ij}^{sup} \tag{3.22}$$

$$c_{in,ij}^{inf}(\mathbf{X}) = K_{ij}^{inf} - K_{ij}(\mathbf{X}), \qquad \forall (i,j) \text{ tal que } K_{ij}^{inf} < K_{ij}^{sup} \tag{3.23}$$

$$c_{eq,ij}(\mathbf{X}) = K_{ij}(\mathbf{X}) - K_{ij}^{sup}, \qquad \forall (i,j) \text{ tal que } K_{ij}^{inf} = K_{ij}^{sup} \tag{3.24}$$

Problema reformulado O problema de otimização da Equação 3.5, em sua nova formulação, é mostrado abaixo:

$$\begin{aligned}
\min_{\mathbf{X}} \quad & F(\mathbf{X}) = \|\mathbf{Y} - \mathbf{\Phi X}\|_F^2 \\
\text{s.a.} \quad & c_{in,0}(\mathbf{X}) \leqslant 0 \\
& c_{in,ij}^{sup}(\mathbf{X}) \leqslant 0, \qquad \forall (i,j) \text{ tal que } K_{ij}^{inf} < K_{ij}^{sup} \\
& c_{in,ij}^{inf}(\mathbf{X}) \leqslant 0, \qquad \forall (i,j) \text{ tal que } K_{ij}^{inf} < K_{ij}^{sup} \\
& c_{eq,ij}(\mathbf{X}) = 0, \qquad \forall (i,j) \text{ tal que } K_{ij}^{inf} = K_{ij}^{sup}
\end{aligned} \tag{3.25}$$

Por se tratar de um problema de Programação Não Linear (NLP), neste trabalho, sua resolução é feita com o algoritmo do ponto interior do MATLAB. Foram utilizados os critérios de tolerâncias de 1×10^{-10} para as variáveis de decisão e de 1×10^{-6} para a função objetivo e restrições.

Velocidade e Robustez

O Algoritmo 3, como um todo, é bastante rápido: o LASSO adaptativo é executado em pouco tempo, há poucas variáveis de decisão no problema da Equação 3.25 (a matriz $\mathbf{X}_1$ é esparsa) e os gradientes e Hessianas são calculados analiticamente nas etapas de otimização. Além disso, como somente modelos fisicamente consistentes são gerados, independentemente da intensidade do ruído, pode-se afirmar que o algoritmo também é robusto. Assim, o processo de identificação é rápido e robusto ao mesmo tempo (uma análise quantitativa é realizada no Capítulo 4).

As expressões analíticas dos gradientes e das Hessianas, tanto para a Validação Cruzada Generalizada (GCV) quanto para a otimização com restrições fenomenológicas, são deduzidas no Apêndice A.

Qualidade do Modelo Identificado

A qualidade do modelo identificado pode ser verificada tanto visualmente quanto numericamente. Além de comparar em gráficos as respostas medida e calculada para cada variável de saída, o algoritmo também determina o valor dos seguintes parâmetros estatísticos: Erro Médio Quadrático Relativo (*Mean Relative Squared Error* – MRSE) e Variância Média Contabilizada (*Mean Variance-Accounted-For* – MVAF). Estes parâmatros são definidos da seguinte forma:

$$\text{MRSE} = \frac{1}{n_y} \sum_{j=1}^{n_y} \frac{\|\mathbf{e}_j\|}{\|\mathbf{y}_j\|} \times 100\% \tag{3.26}$$

$$\text{MVAF} = \frac{1}{n_y} \sum_{j=1}^{n_y} \left(1 - \frac{\text{var}(\mathbf{e}_j)}{\text{var}(\mathbf{y}_j)}\right) \times 100\% \tag{3.27}$$

em que $\text{var}(\mathbf{y}_j)$ e $\text{var}(\mathbf{e}_j)$ são as variâncias da j-ésima coluna de $\mathbf{Y}$ e $\mathbf{E}$, respectivamente. Quanto menor for MRSE e maior for MVAF, maior será a qualidade do modelo identificado.

O algoritmo também determina a região de validade do modelo, definida pelos intervalos abaixo:

$$\begin{aligned} \mathbf{y}_{min} \leqslant \mathbf{y} \leqslant \mathbf{y}_{max} \\ \mathbf{u}_{min} \leqslant \mathbf{u} \leqslant \mathbf{u}_{max} \end{aligned} \tag{3.28}$$

em que $y_{min,i}$ e $y_{max,i}$ são o menor e o maior valor medidos para a saída y_i, respectivamente ($u_{min,j}$ e $u_{max,j}$ são definidos de forma análoga para a entrada u_j).

3.1.4 Algoritmo 4

O Algoritmo 4 faz o planejamento de sinais para a etapa do teste GBN. Primeiramente, os valores de p_j são calculados por meio das Equações 2.26 e 2.27, nas quais

as constantes de tempo $\tau_{min,j}$ e $\tau_{max,j}$ são dadas pelo Algoritmo 2. Em seguida, utilizando-se uma versão modificada do código MATLAB *gbngen.m* de ZHU (2001), é projetado para cada entrada do processo um sinal GBN unitário ($u_j^\star$), ou seja, um sinal GBN de amplitude 1 e probabilidade de não mudança de nível p_j (a modificação em questão permite que sejam considerados os grupos de entradas criados com o Algoritmo 2). Finalmente, tem-se que o sinal GBN da entrada u_j é dado por:

$$u_j(k) = \delta_j u_j^\star(k), \qquad j = 1, ..., n_u \tag{3.29}$$

em que as amplitudes δ_j das entradas são determinadas por meio da resolução do seguinte problema de otimização:

$$\begin{aligned} \max_{\boldsymbol{\delta}} \quad & \sum_{j=1}^{n_u} \delta_j \\ \text{s.a.} \quad & f_s\, \mathbf{y}_{min} \leqslant \widehat{\mathbf{y}}(k) \leqslant f_s\, \mathbf{y}_{max}, \qquad k = 1, ..., N_{GBN} \\ & \mathbf{0} \leqslant \boldsymbol{\delta} \leqslant \boldsymbol{\delta}_{max} \end{aligned} \tag{3.30}$$

sendo N_{GBN} a duração total do teste GBN, $\widehat{\mathbf{y}}(k)$ a resposta estimada de $\mathbf{y}$ aos sinais GBN projetados no instante k, $\boldsymbol{\delta}_{max}$ o vetor das amplitudes das entradas no pré-teste e $f_s \in (0, 1]$ um fator de segurança. O cálculo de $\widehat{\mathbf{y}}(k)$ é realizado utilizando-se um modelo preliminar do processo ($\widehat{\mathbf{G}}_\mathbf{p}$), de modo que, quanto mais acurado for este modelo, maior deve ser o valor de f_s. Tanto $\widehat{\mathbf{G}}_\mathbf{p}$ quanto f_s devem ser fornecidos pelo usuário.

Para poder ser resolvido no MATLAB, o problema de maximização da Equação 3.30 deve ser reformulado. Inicialmente, ele é reescrito em termos de uma minimização:

$$\begin{aligned} \max_{\boldsymbol{\delta}} \sum_{j=1}^{n_u} \delta_j &= \min_{\boldsymbol{\delta}} \sum_{j=1}^{n_u} (-\delta_j) \\ &= \min_{\boldsymbol{\delta}} \underbrace{\begin{bmatrix} -1 & -1 & \cdots & -1 \end{bmatrix}}_{\mathbf{c}^T} \underbrace{\begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_{n_u} \end{bmatrix}}_{\boldsymbol{\delta}} \\ &= \min_{\boldsymbol{\delta}} \mathbf{c}^T \boldsymbol{\delta} \end{aligned} \tag{3.31}$$

Para cada variável de saída há uma restrição da forma:

$$f_s\, y_{min,i} \leqslant \widehat{y}_i(k) \leqslant f_s\, y_{max,i}, \qquad k = 1, ..., N_{GBN} \tag{3.32}$$

Calculando $\widehat{y}_i(k)$:

$$\begin{aligned}
\widehat{y}_i(k) &= \mathcal{Z}^{-1}\left\{\widehat{Y}_i(z^{-1})\right\} \\
&= \mathcal{Z}^{-1}\left\{\sum_{j=1}^{n_u}\widehat{G}_{p,ij}(z^{-1})U_j(z^{-1})\right\} \\
&= \mathcal{Z}^{-1}\left\{\sum_{j=1}^{n_u}\delta_j\widehat{G}_{p,ij}(z^{-1})U_j^{\star}(z^{-1})\right\} \\
&= \sum_{j=1}^{n_u}\delta_j\mathcal{Z}^{-1}\left\{\widehat{G}_{p,ij}(z^{-1})U_j^{\star}(z^{-1})\right\} \\
&= \sum_{j=1}^{n_u}\delta_j\mathcal{Z}^{-1}\left\{\widehat{S}_{ij}(z^{-1})\right\} \\
&= \sum_{j=1}^{n_u}\delta_j\widehat{s}_{ij}(k) \\
&= \begin{bmatrix}\widehat{s}_{i1}(k) & \widehat{s}_{i2}(k) & \cdots & \widehat{s}_{in_u}(k)\end{bmatrix}\begin{bmatrix}\delta_1\\ \delta_2\\ \vdots\\ \delta_{n_u}\end{bmatrix}
\end{aligned} \tag{3.33}$$

em que o sinal $\widehat{s}_{ij}(k)$ é a resposta estimada de y_i à perturbação $u_j^{\star}$ no instante k. Substituindo a Equação 3.33 nas restrições da Equação 3.32:

$$f_s\, y_{min,i} \leqslant \begin{bmatrix}\widehat{s}_{i1}(k) & \widehat{s}_{i2}(k) & \cdots & \widehat{s}_{in_u}(k)\end{bmatrix}\begin{bmatrix}\delta_1\\ \delta_2\\ \vdots\\ \delta_{n_u}\end{bmatrix} \leqslant f_s\, y_{max,i}, \qquad k = 1, ..., N_{GBN} \tag{3.34}$$

Considerando todos os instantes de tempo:

$$f_s \underbrace{\begin{bmatrix}y_{min,i}\\ y_{min,i}\\ \vdots\\ y_{min,i}\end{bmatrix}}_{\mathbf{y_{min,i}}} \leqslant \underbrace{\begin{bmatrix}\widehat{s}_{i1}(1) & \widehat{s}_{i2}(1) & \cdots & \widehat{s}_{in_u}(1)\\ \widehat{s}_{i1}(2) & \widehat{s}_{i2}(2) & \cdots & \widehat{s}_{in_u}(2)\\ \vdots & \vdots & \ddots & \vdots\\ \widehat{s}_{i1}(N_{GBN}) & \widehat{s}_{i2}(N_{GBN}) & \cdots & \widehat{s}_{in_u}(N_{GBN})\end{bmatrix}}_{\widehat{\mathbf{S}}_i}\underbrace{\begin{bmatrix}\delta_1\\ \delta_2\\ \vdots\\ \delta_{n_u}\end{bmatrix}}_{\boldsymbol{\delta}} \leqslant f_s \underbrace{\begin{bmatrix}y_{max,i}\\ y_{max,i}\\ \vdots\\ y_{max,i}\end{bmatrix}}_{\mathbf{y_{max,i}}}$$

$$f_s\, \mathbf{y_{min,i}} \leqslant \widehat{\mathbf{S}}_i\boldsymbol{\delta} \leqslant f_s\, \mathbf{y_{max,i}} \tag{3.35}$$

Considerando todas as variáveis de saída:

$$\underbrace{f_s \begin{bmatrix} \mathbf{y_{min,1}} \\ \mathbf{y_{min,2}} \\ \vdots \\ \mathbf{y_{min,n_y}} \end{bmatrix}}_{\mathbf{b}^{inf}} \leqslant \underbrace{\begin{bmatrix} \widehat{\mathbf{S}}_1 \\ \widehat{\mathbf{S}}_2 \\ \vdots \\ \widehat{\mathbf{S}}_{n_y} \end{bmatrix}}_{\widetilde{\mathbf{A}}} \boldsymbol{\delta} \leqslant \underbrace{f_s \begin{bmatrix} \mathbf{y_{max,1}} \\ \mathbf{y_{max,2}} \\ \vdots \\ \mathbf{y_{max,n_y}} \end{bmatrix}}_{\mathbf{b}^{sup}}$$

$$\mathbf{b}^{inf} \leqslant \widetilde{\mathbf{A}}\boldsymbol{\delta} \leqslant \mathbf{b}^{sup} \tag{3.36}$$

Reescrevendo as restrições da Equação 3.36:

$$\begin{aligned} \widetilde{\mathbf{A}}\boldsymbol{\delta} &\leqslant \mathbf{b}^{sup} \\ -\widetilde{\mathbf{A}}\boldsymbol{\delta} &\leqslant -\mathbf{b}^{inf} \end{aligned} \tag{3.37}$$

E, finalmente, tem-se que:

$$\underbrace{\begin{bmatrix} \widetilde{\mathbf{A}} \\ -\widetilde{\mathbf{A}} \end{bmatrix}}_{\mathbf{A}} \boldsymbol{\delta} \leqslant \underbrace{\begin{bmatrix} \mathbf{b}^{sup} \\ -\mathbf{b}^{inf} \end{bmatrix}}_{\mathbf{b}}$$

$$\mathbf{A}\boldsymbol{\delta} \leqslant \mathbf{b} \tag{3.38}$$

Portanto, o problema da Equação 3.30 reformulado tem a seguinte estrutura:

$$\begin{aligned} \min_{\boldsymbol{\delta}} \quad & \mathbf{c}^T\boldsymbol{\delta} \\ \text{s.a.} \quad & \mathbf{A}\boldsymbol{\delta} \leqslant \mathbf{b} \\ & \mathbf{0} \leqslant \boldsymbol{\delta} \leqslant \boldsymbol{\delta}_{max} \end{aligned} \tag{3.39}$$

Por se tratar de um problema de Programação Linear (LP), sua solução pode ser facilmente encontrada com a função *linprog* do MATLAB.

3.2 Procedimento Proposto

A junção das etapas *on-line* e *off-line* dá origem à metodologia de identificação proposta nesta dissertação. O procedimento a ser seguido, composto de 8 etapas, é mostrado abaixo:

1. Realizar o pré-teste conforme descrito na Seção 2.2.1;
2. Fazer o planejamento de sinais para o teste degrau com o Algoritmo 1;
3. Realizar o teste degrau;

4. Fazer o tratamento conjunto dos dados obtidos no pré-teste e no teste degrau com o Algoritmo 2;

5. Gerar um modelo preliminar do processo com o Algoritmo 3, utilizando para isso os dados obtidos no pré-teste e no teste degrau;

6. Projetar os sinais GBN com o Algoritmo 4;

7. Realizar o teste GBN;

8. Gerar o modelo final do processo com o Algoritmo 3, utilizando para isso os dados obtidos no pré-teste, no teste degrau e no teste GBN.

Capítulo 4

Resultados e Discussões

A metodologia desenvolvida foi aplicada, primeiramente, na identificação de sistemas dinâmicos lineares e, em seguida, em um problema *benchmark* (coluna de destilação da Shell). Em todas as simulações foram utilizados os valores *default* dos parâmetros do pacote IDENTIPHY (estes valores são mostrados no Apêndice B, nas instruções de uso dos algoritmos), foi assumido que os sinais dos ganhos estáticos eram conhecidos e foram atribuídos valores suficientemente grandes a N_a e N_b, ou seja, estes parâmetros foram aumentados gradualmente até não terem sido mais observadas mudanças significativas em MRSE e MVAF. A duração do teste GBN em cada simulação foi de 7 vezes o maior tempo de assentamento do sistema.

4.1 Sistemas Dinâmicos Lineares

Foram identificados sistemas dinâmicos de diferentes ordens, com e sem a presença de tempo morto.

4.1.1 Sistemas de 1ª Ordem

Sem Tempo Morto

Seja o sistema MIMO 4 x 4 cuja matriz de transferência ($\mathbf{G_p}$) é mostrada abaixo:

$$\mathbf{G_p}(z^{-1}) = \begin{bmatrix} \dfrac{0,9481 - 1,961z^{-1}}{1 - 0,7583z^{-1}} & 0 & \dfrac{2,068 + 0,5063z^{-1}}{1 - 0,7583z^{-1}} & 1,798 \\ \dfrac{1,987 + 0,397z^{-1}}{1 - 0,7583z^{-1}} & 0 & \dfrac{0,4892 - 0,4735z^{-1}}{1 - 0,7583z^{-1}} & -0,639 \\ \dfrac{0,123 - 0,2839z^{-1}}{1 - 0,7583z^{-1}} & -1,241 & \dfrac{0,6708 + 0,07328z^{-1}}{1 - 0,7583z^{-1}} & 0 \\ \dfrac{0,5941 + 2,586z^{-1}}{1 - 0,7583z^{-1}} & 0 & \dfrac{0,2437 - 0,8523z^{-1}}{1 - 0,7583z^{-1}} & 0 \end{bmatrix} \quad (4.1)$$

Assume-se que esta matriz não é conhecida. Assim, a metodologia proposta foi utilizada para identificá-la. Foram executados cada um dos 8 passos descritos na Seção 3.2:

1. As Figuras 4.1 e 4.2 mostram os resultados obtidos no pré-teste.

2. A partir dos resultados do pré-teste, foi feito um planejamento de sinais para a etapa do teste degrau. Tal planejamento pode ser visto na Figura 4.3.

3. Os degraus projetados foram então aplicados no sistema. A resposta obtida é mostrada na Figura 4.4.

4. O tratamento conjunto dos resultados do pré-teste e do teste degrau forneceu os limites inferior ($\mathbf{K}^{inf}$) e superior ($\mathbf{K}^{sup}$) para a matriz de ganhos estáticos do processo:

$$\mathbf{K}^{inf} = \begin{bmatrix} -4,2748 & 0 & 9,7821 & 1,7977 \\ 9,2795 & 0 & 0,0493 & -0,6390 \\ -0,6783 & -1,2414 & 2,8352 & 0 \\ 12,2228 & 0 & -2,6219 & 0 \end{bmatrix} \tag{4.2}$$

$$\mathbf{K}^{sup} = \begin{bmatrix} -3,8146 & 0 & 10,9738 & 1,7981 \\ 9,9912 & 0 & 0,1081 & -0,6390 \\ -0,6081 & -1,2412 & 3,1695 & 0 \\ 13,3565 & 0 & -2,2385 & 0 \end{bmatrix} \tag{4.3}$$

   A matriz real de ganhos estáticos ($\mathbf{K}$) é mostrada abaixo:

$$\mathbf{K} = \mathbf{G_p}(1) = \begin{bmatrix} -4,1933 & 0 & 10,6517 & 1,7979 \\ 9,8666 & 0 & 0,0652 & -0,6390 \\ -0,6657 & -1,2413 & 3,0792 & 0 \\ 13,1577 & 0 & -2,5183 & 0 \end{bmatrix} \tag{4.4}$$

   Assim, verifica-se claramente que $\mathbf{K}^{inf} \leqslant \mathbf{K} \leqslant \mathbf{K}^{sup}$. Também foram calculados nesta etapa os vetores $\boldsymbol{\tau}_{min}$ e $\boldsymbol{\tau}_{max}$:

$$\boldsymbol{\tau}_{min} = \begin{bmatrix} 3,4187 & 3,3701 & 3,4187 & 0 \end{bmatrix}^T \tag{4.5}$$

$$\boldsymbol{\tau}_{max} = \begin{bmatrix} 3,6105 & 4,5000 & 3,6105 & 3,6155 \end{bmatrix}^T \tag{4.6}$$

5. Os resultados do pré-teste e do teste degrau foram utilizados para gerar uma primeira aproximação da matriz de transferência do processo. Na identificação,

utilizou-se $N_a = N_b = 1$. A matriz identificada ($\widehat{\mathbf{G}}_{\mathbf{p}}^{\star}$) é mostrada abaixo:

$$\widehat{\mathbf{G}}_{\mathbf{p}}^{\star}(z^{-1}) = \begin{bmatrix} \dfrac{0,9481 - 1,961z^{-1}}{1 - 0,7583z^{-1}} & 0 & \dfrac{2,068 + 0,5063z^{-1}}{1 - 0,7583z^{-1}} & \widehat{G}_{p,14}^{\star} \\ \dfrac{1,987 + 0,397z^{-1}}{1 - 0,7583z^{-1}} & 0 & \dfrac{0,4892 - 0,4735z^{-1}}{1 - 0,7583z^{-1}} & \widehat{G}_{p,24}^{\star} \\ \dfrac{0,123 - 0,2839z^{-1}}{1 - 0,7583z^{-1}} & \widehat{G}_{p,32}^{\star} & \dfrac{0,6708 + 0,07328z^{-1}}{1 - 0,7583z^{-1}} & 0 \\ \dfrac{0,5941 + 2,586z^{-1}}{1 - 0,7583z^{-1}} & 0 & \dfrac{0,2437 - 0,8523z^{-1}}{1 - 0,7583z^{-1}} & 0 \end{bmatrix} \tag{4.7}$$

em que:

$$\widehat{G}_{p,14}^{\star} = \frac{1,798 - 1,363z^{-1}}{1 - 0,7583z^{-1}} = \frac{1,798\cancel{(1 - 0,7583z^{-1})}}{\cancel{1 - 0,7583z^{-1}}} = 1,798$$

$$\widehat{G}_{p,24}^{\star} = \frac{-0,639 + 0,4846z^{-1}}{1 - 0,7583z^{-1}} = \frac{-0,639\cancel{(1 - 0,7583z^{-1})}}{\cancel{1 - 0,7583z^{-1}}} = -0,639$$

$$\widehat{G}_{p,32}^{\star} = \frac{-1,241 + 0,9413z^{-1}}{1 - 0,7583z^{-1}} = \frac{-1,241\cancel{(1 - 0,7583z^{-1})}}{\cancel{1 - 0,7583z^{-1}}} = -1,241$$

O tempo de execução do Algoritmo 3, t_{sim}, foi de apenas 6,01 segundos[1]. Apesar do MATLAB não ter conseguido simplificar alguns elementos de $\widehat{\mathbf{G}}_{\mathbf{p}}^{\star}$, verifica-se que a matriz de transferência identificada é igual à original. De fato, tem-se que MRSE $= 3,0982 \times 10^{-8}\%$ e MVAF $= 100\%$, ambos resultados excelentes. As Figuras 4.5, 4.6, 4.7 e 4.8 comparam os valores medidos e calculados com o modelo inicial para as saídas. Como já era esperado, as duas curvas estão sobrepostas.

6. A partir do modelo preliminar $\widehat{\mathbf{G}}_{\mathbf{p}}^{\star}$ e dos vetores $\boldsymbol{\tau}_{min}$ e $\boldsymbol{\tau}_{max}$, foi feito um planejamento de sinais para a etapa do teste GBN. Tal planejamento pode ser visto na Figura 4.9.

7. Os sinais GBN projetados foram então aplicados no sistema. A resposta obtida é mostrada na Figura 4.10.

8. Finalmente, a partir dos resultados do pré-teste, do teste degrau e do teste GBN, foi gerado o modelo final do processo ($\widehat{\mathbf{G}}_{\mathbf{p}}$). Nesta identificação também utilizou-se $N_a = N_b = 1$. As Figuras 4.11, 4.12, 4.13 e 4.14 comparam os valores medidos e calculados com o modelo final para as saídas.

[1]Foi utilizado um computador portátil HP Pavilion dm4–1075br.

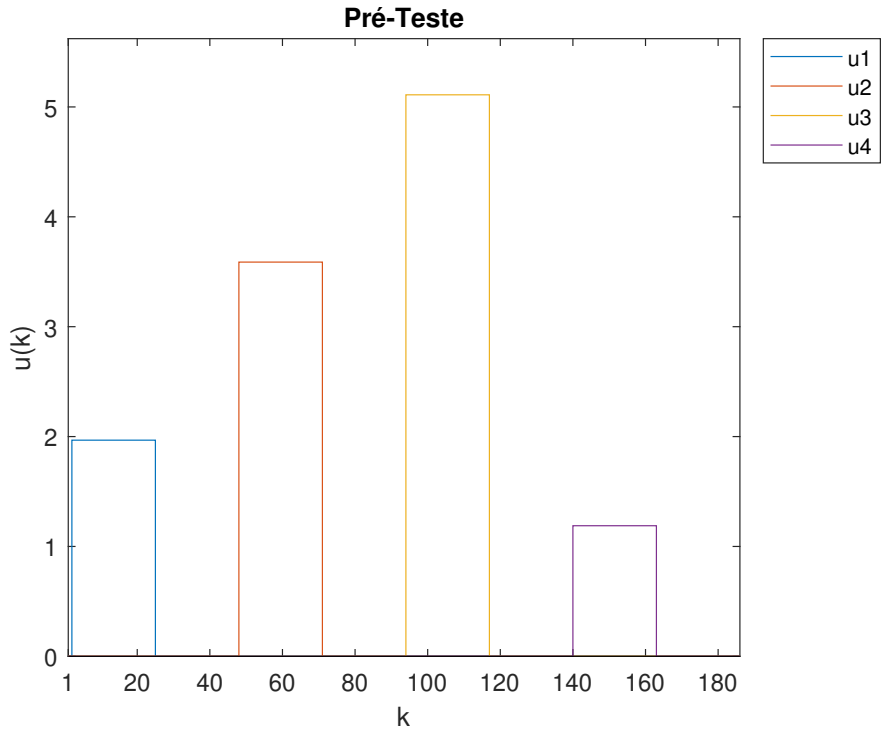

Figura 4.1: Simulação do pré-teste (variáveis de entrada) para o caso linear sem tempo morto.

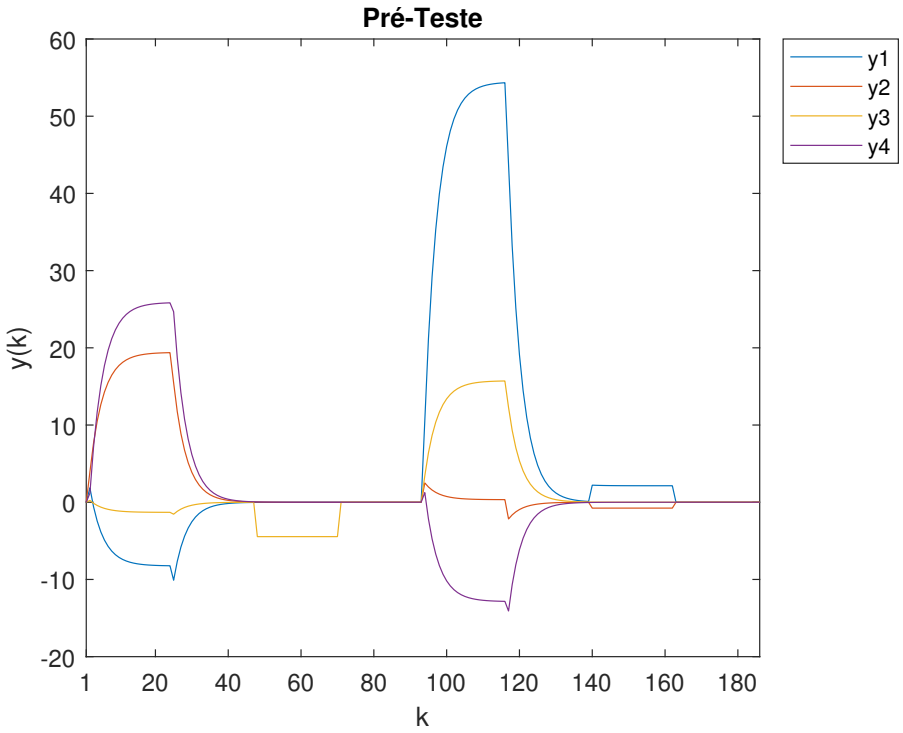

Figura 4.2: Simulação do pré-teste (variáveis de saída) para o caso linear sem tempo morto.

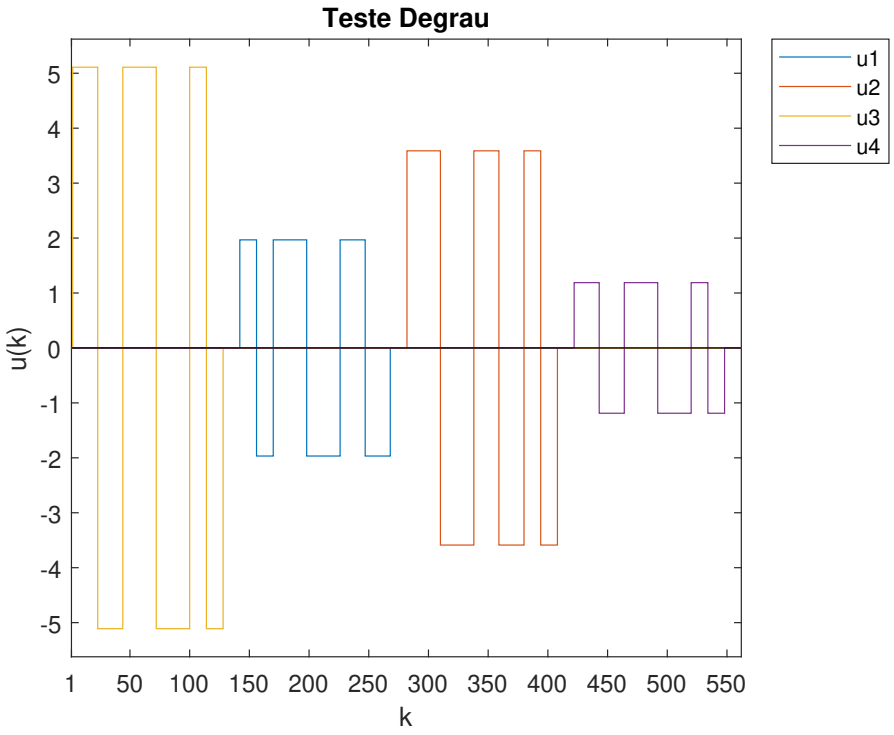

Figura 4.3: Simulação do teste degrau (variáveis de entrada) para o caso linear sem tempo morto.

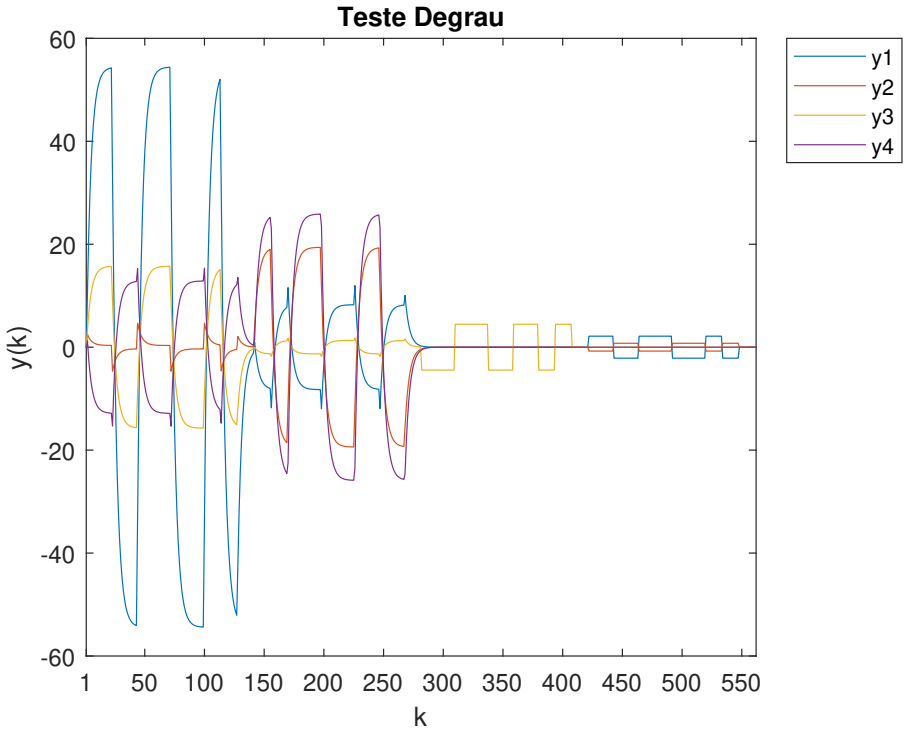

Figura 4.4: Simulação do teste degrau (variáveis de saída) para o caso linear sem tempo morto.

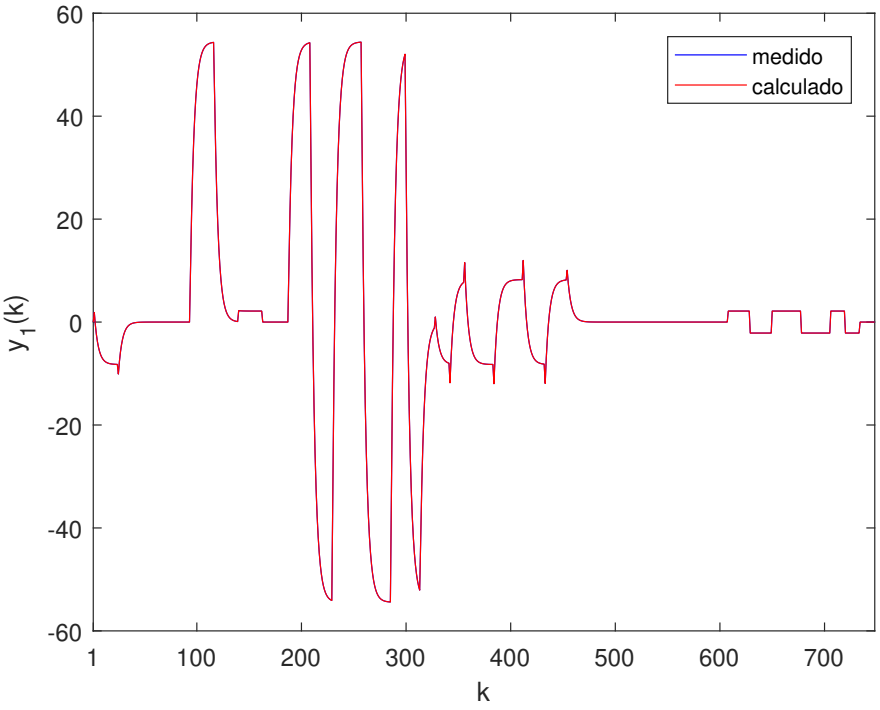

Figura 4.5: Valores medidos e calculados com o modelo inicial para y_1 (sistema de 1ª ordem sem tempo morto).

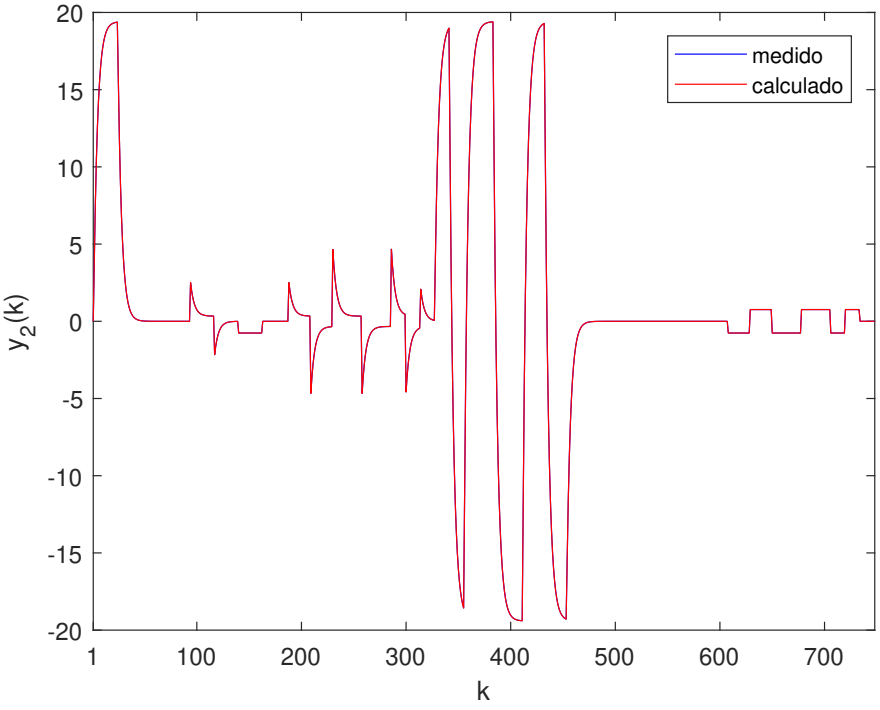

Figura 4.6: Valores medidos e calculados com o modelo inicial para y_2 (sistema de 1ª ordem sem tempo morto).

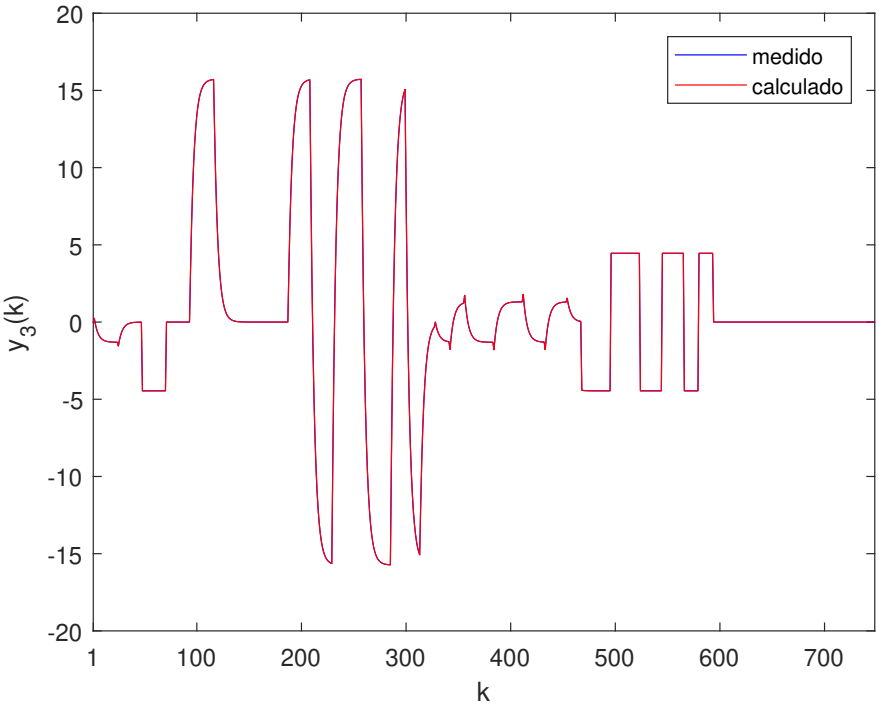

Figura 4.7: Valores medidos e calculados com o modelo inicial para y_3 (sistema de 1ª ordem sem tempo morto).

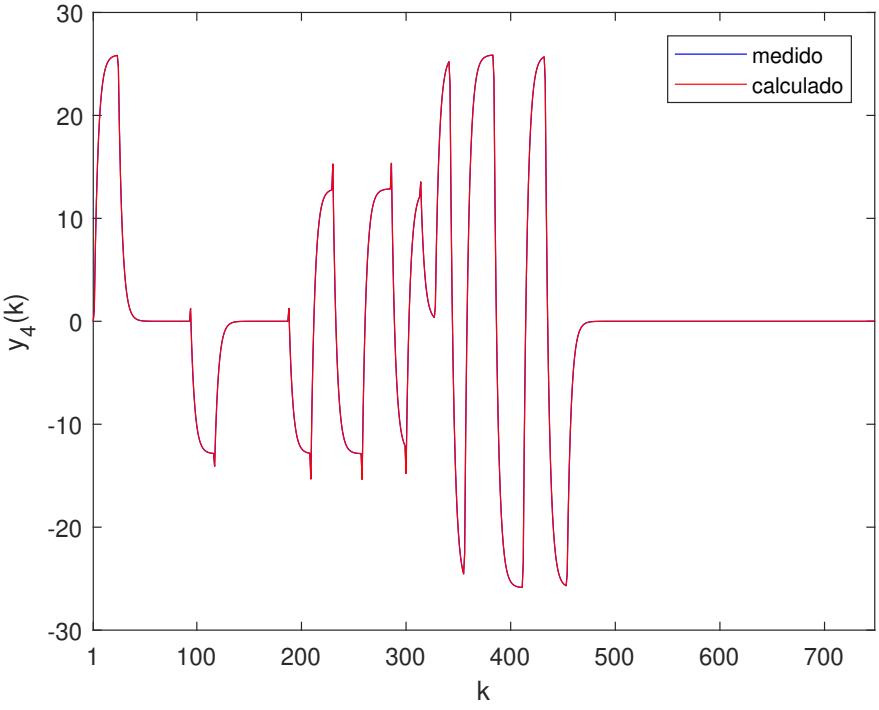

Figura 4.8: Valores medidos e calculados com o modelo inicial para y_4 (sistema de 1ª ordem sem tempo morto).

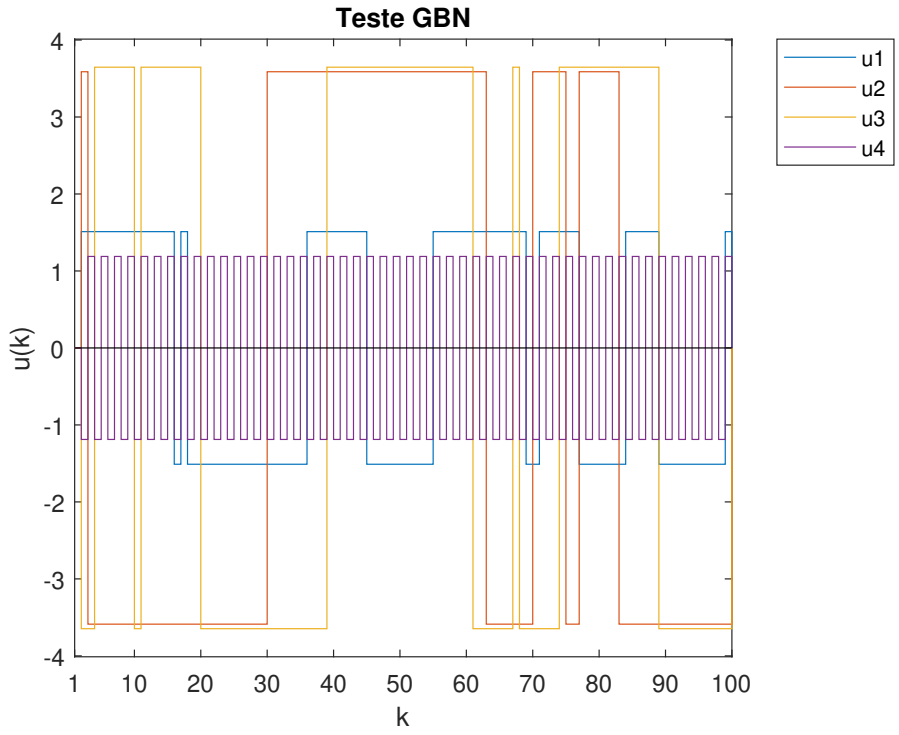

Figura 4.9: Simulação do teste GBN (variáveis de entrada) para o caso linear sem tempo morto.

Figura 4.10: Simulação do teste GBN (variáveis de saída) para o caso linear sem tempo morto.

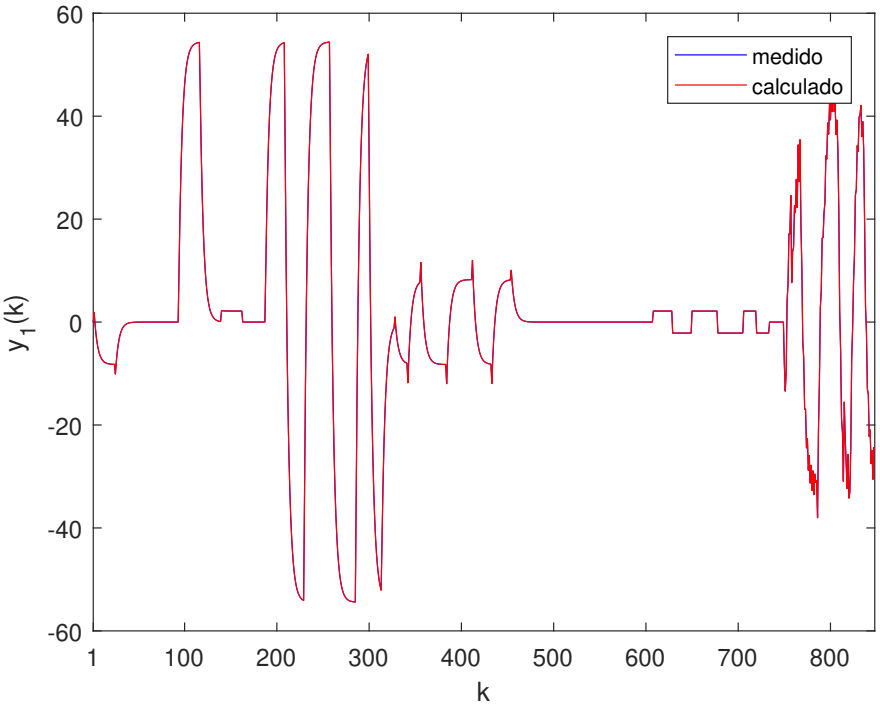

Figura 4.11: Valores medidos e calculados com o modelo final para y_1 (sistema de 1ª ordem sem tempo morto).

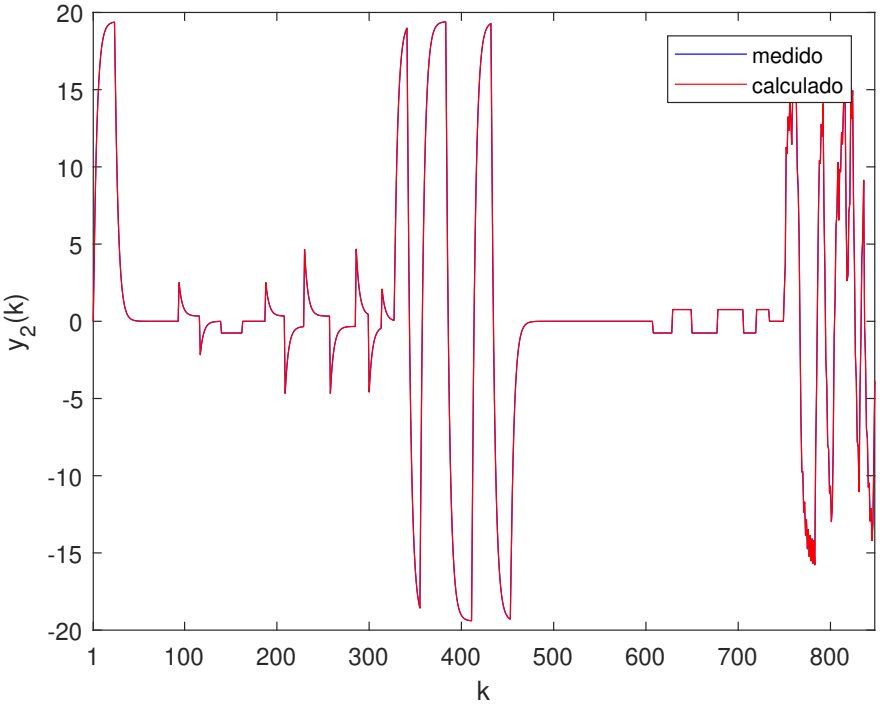

Figura 4.12: Valores medidos e calculados com o modelo final para y_2 (sistema de 1ª ordem sem tempo morto).

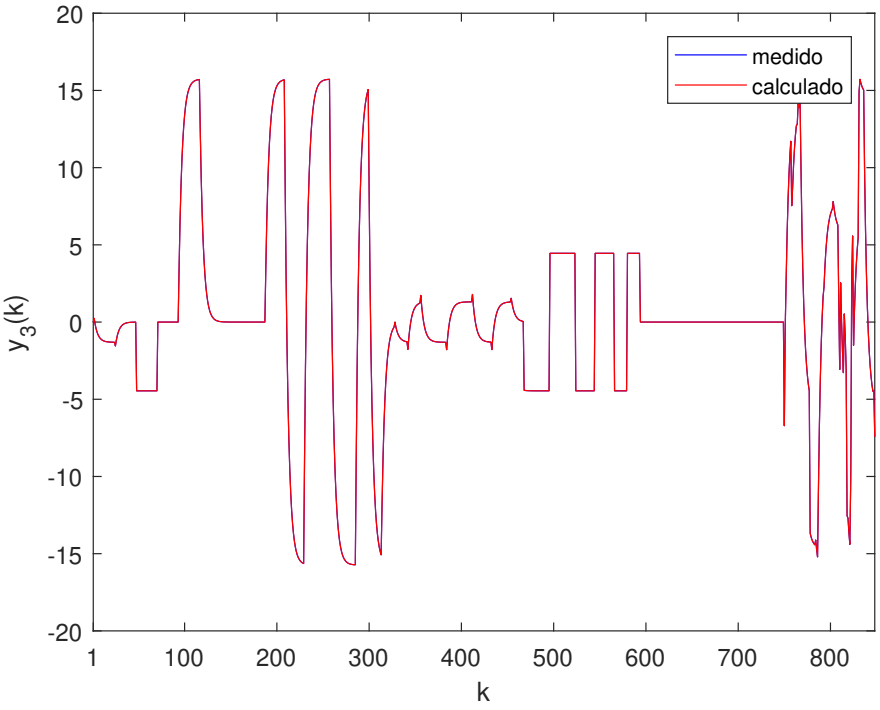

Figura 4.13: Valores medidos e calculados com o modelo final para y_3 (sistema de 1ª ordem sem tempo morto).

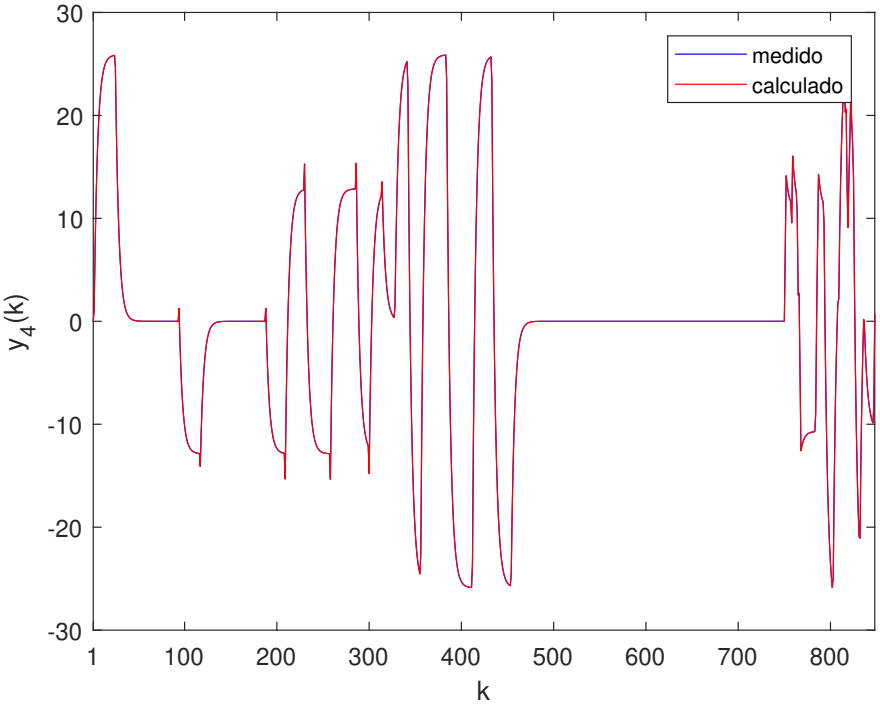

Figura 4.14: Valores medidos e calculados com o modelo final para y_4 (sistema de 1ª ordem sem tempo morto).

Constatou-se, como já era esperado, que $\widehat{\mathbf{G}}_{\mathbf{p}} = \widehat{\mathbf{G}}_{\mathbf{p}}^{\star}$. Obteve-se $t_{sim} = 7,28\,s$, MRSE $= 2,7905 \times 10^{-8}\%$ e MVAF $= 100\%$. Por se tratar de um sistema relativamente simples (1ª ordem sem tempo morto), o teste GBN não acrescentou novas informações ao processo de identificação. No entanto, ele é bastante útil para sistemas mais complexos.

Com Tempo Morto

Seja o sistema MIMO 4 x 3 cuja matriz de transferência é mostrada abaixo:

$$\mathbf{G_p}(z^{-1}) = \begin{bmatrix} \dfrac{-0,742z^{-9} - 0,2009z^{-10}}{1 - 0,4405z^{-1}} & \dfrac{1,587z^{-4} - 0,3176z^{-5}}{1 - 0,4405z^{-1}} & \dfrac{-0,834z^{-8} + 0,4381z^{-9}}{1 - 0,4405z^{-1}} \\ \dfrac{-0,486z^{-9} + 1,279z^{-10}}{1 - 0,4405z^{-1}} & \dfrac{0,08802z^{-6} - 2,468z^{-7}}{1 - 0,4405z^{-1}} & \dfrac{-0,2222z^{-1} - 0,3521z^{-2}}{1 - 0,4405z^{-1}} \\ \dfrac{0,1606z^{-9} + 0,5715z^{-10}}{1 - 0,4405z^{-1}} & \dfrac{0,7291z^{-8} - 1,085z^{-9}}{1 - 0,4405z^{-1}} & \dfrac{0,155z^{-8} - 0,2693z^{-9}}{1 - 0,4405z^{-1}} \\ -0,5444z^{-7} & 0 & 1,411z^{-8} \end{bmatrix} \tag{4.8}$$

Utilizando o procedimento descrito na Seção 3.2 com $N_a = N_b = 10$, chegou-se ao seguinte modelo identificado:

$$\widehat{\mathbf{G}}_{\mathbf{p}}(z^{-1}) = \begin{bmatrix} \dfrac{-0,742z^{-9} - 0,2009z^{-10}}{1 - 0,4405z^{-1}} & \dfrac{1,587z^{-4} - 0,3176z^{-5}}{1 - 0,4405z^{-1}} & \dfrac{-0,834z^{-8} + 0,4381z^{-9}}{1 - 0,4405z^{-1}} \\ \dfrac{-0,486z^{-9} + 1,279z^{-10}}{1 - 0,4405z^{-1}} & \dfrac{0,08802z^{-6} - 2,468z^{-7}}{1 - 0,4405z^{-1}} & \dfrac{-0,2222z^{-1} - 0,3521z^{-2}}{1 - 0,4405z^{-1}} \\ \dfrac{0,1606z^{-9} + 0,5715z^{-10}}{1 - 0,4405z^{-1}} & \dfrac{0,7291z^{-8} - 1,085z^{-9}}{1 - 0,4405z^{-1}} & \dfrac{0,155z^{-8} - 0,2693z^{-9}}{1 - 0,4405z^{-1}} \\ -0,5444z^{-7} & 0 & 1,411z^{-8} \end{bmatrix} \tag{4.9}$$

sendo $t_{sim} = 12,58\,s$, MRSE $= 0,0053\%$ e MVAF $= 100\%$. As Figuras 4.15, 4.16, 4.17 e 4.18 comparam os valores medidos e calculados para as saídas do processo.

Observa-se, mais uma vez, que a matriz de transferência identificada é igual à original. No entanto, quando há tempo morto, é necessário atribuir valores mais elevados aos parâmetros N_a e N_b para se obter um modelo de boa qualidade. Verifica-se também que as saídas respondem rapidamente ao sinal GBN (oscilações de alta frequência).

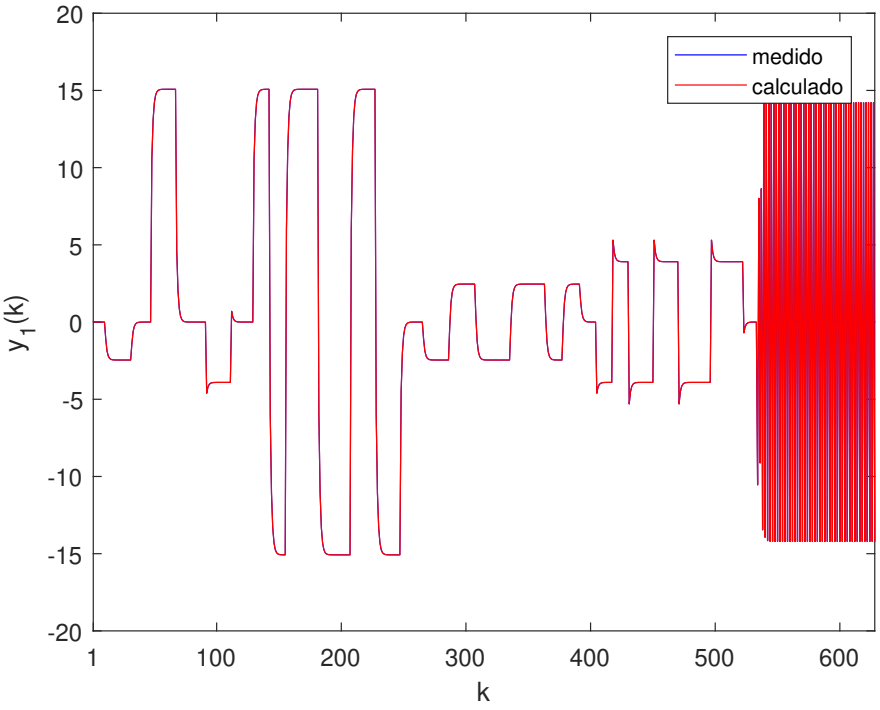

Figura 4.15: Valores medidos e calculados para y_1 (sistema de 1ª ordem com tempo morto).

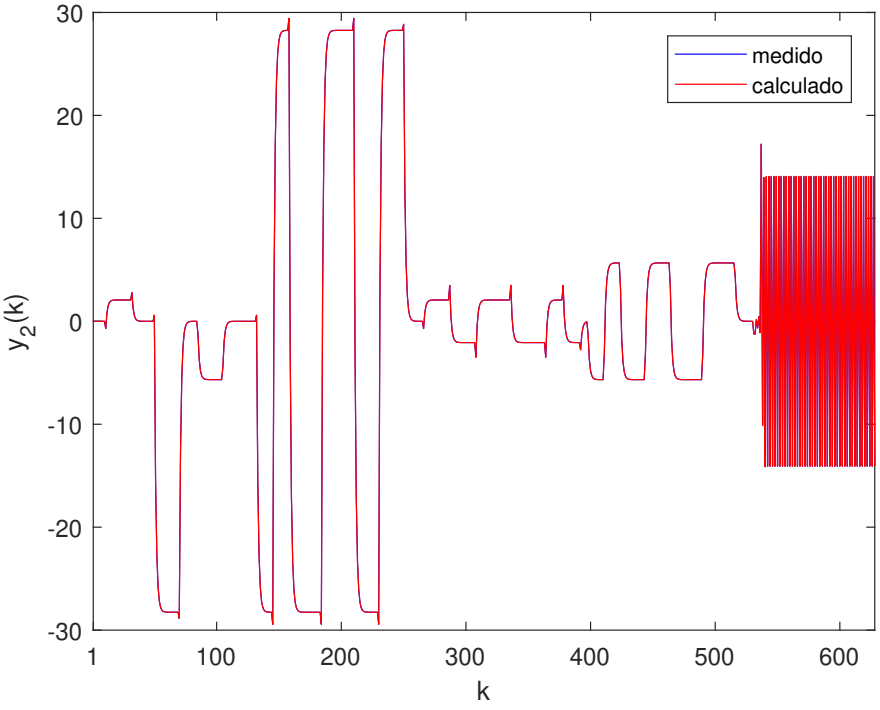

Figura 4.16: Valores medidos e calculados para y_2 (sistema de 1ª ordem com tempo morto).

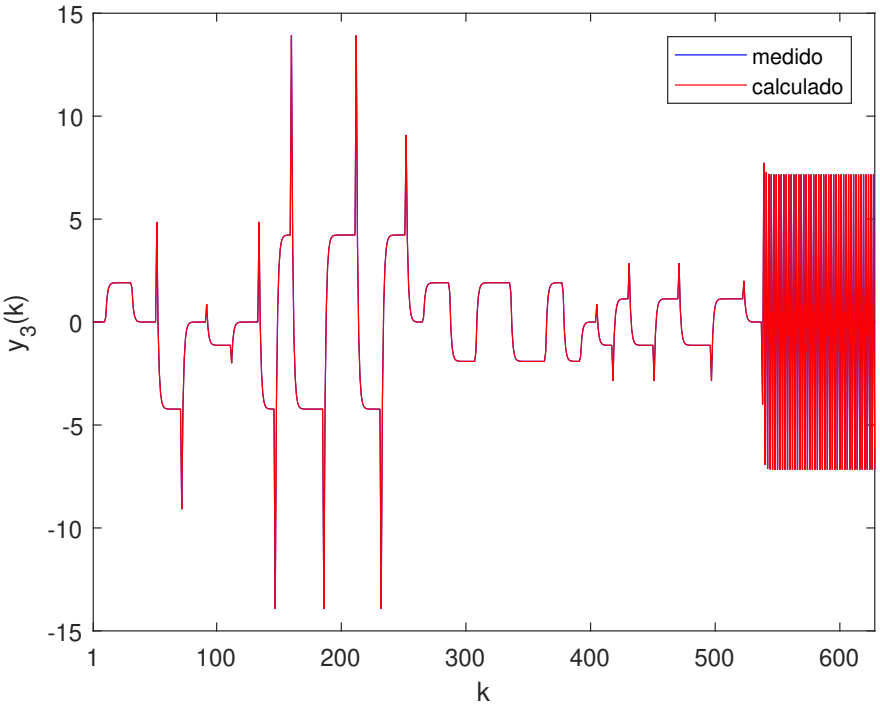

Figura 4.17: Valores medidos e calculados para y_3 (sistema de 1ª ordem com tempo morto).

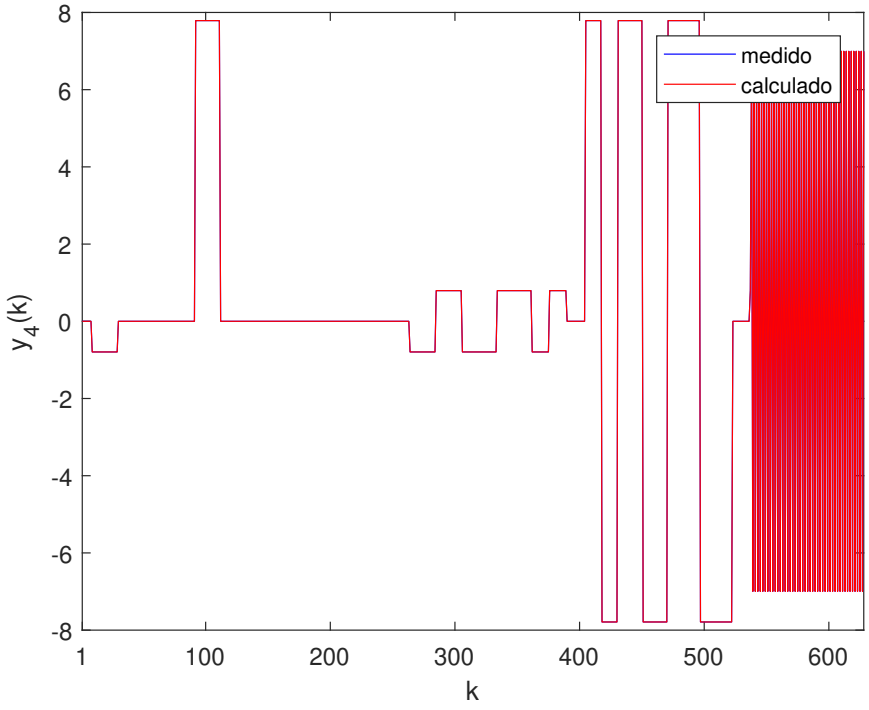

Figura 4.18: Valores medidos e calculados para y_4 (sistema de 1ª ordem com tempo morto).

4.1.2 Sistemas de 2ª Ordem

Sem Tempo Morto

Seja o sistema MIMO 5 x 1 cuja matriz de transferência é mostrada abaixo:

$$\mathbf{G_p}(z^{-1}) = \begin{bmatrix} \dfrac{0,7308 + 1,663z^{-1} + 0,9409z^{-2}}{1 + 1,265z^{-1} + 0,3922z^{-2}} \\ \dfrac{0,7713 + 0,07559z^{-1} + 0,03741z^{-2}}{1 + 1,265z^{-1} + 0,3922z^{-2}} \\ \dfrac{-1,652 + 0,008588z^{-1} + 0,519z^{-2}}{1 + 1,265z^{-1} + 0,3922z^{-2}} \\ \dfrac{-0,1179 - 0,8666z^{-1} - 0,4288z^{-2}}{1 + 1,265z^{-1} + 0,3922z^{-2}} \\ \dfrac{0,66 - 0,61z^{-1} - 0,3859z^{-2}}{1 + 1,265z^{-1} + 0,3922z^{-2}} \end{bmatrix} \tag{4.10}$$

Utilizando o procedimento descrito na Seção 3.2 com $N_a = N_b = 1$, chegou-se ao seguinte modelo identificado:

$$\widehat{\mathbf{G}}_{\mathbf{p}}(z^{-1}) = \begin{bmatrix} \dfrac{0,7308 + 1,671z^{-1} + 0,9612z^{-2} + 0,01151z^{-3}}{1 + 1,277z^{-1} + 0,4076z^{-2} + 0,004797z^{-3}} \\ \dfrac{0,7713 + 0,08502z^{-1} + 0,03833z^{-2}}{1 + 1,277z^{-1} + 0,4076z^{-2} + 0,004797z^{-3}} \\ \dfrac{-1,652 - 0,01162z^{-1} + 0,5191z^{-2} + 0,006349z^{-3}}{1 + 1,277z^{-1} + 0,4076z^{-2} + 0,004797z^{-3}} \\ \dfrac{-0,1179 - 0,868z^{-1} - 0,4394z^{-2} - 0,005246z^{-3}}{1 + 1,277z^{-1} + 0,4076z^{-2} + 0,004797z^{-3}} \\ \dfrac{0,66 - 0,6019z^{-1} - 0,3934z^{-2} - 0,004721z^{-3}}{1 + 1,277z^{-1} + 0,4076z^{-2} + 0,004797z^{-3}} \end{bmatrix} \tag{4.11}$$

sendo $t_{sim} = 5,15\,s$, MRSE $= 0,0375\%$ e MVAF $= 100\%$. As Figuras 4.19, 4.20, 4.21, 4.22 e 4.23 comparam os valores medidos e calculados para as saídas do processo.

Neste caso, apesar de $\widehat{\mathbf{G}}_{\mathbf{p}}$ não ser igual à matriz $\mathbf{G_p}$, verifica-se claramente que o modelo identificado consegue representar muito bem a dinâmica do sistema estudado.

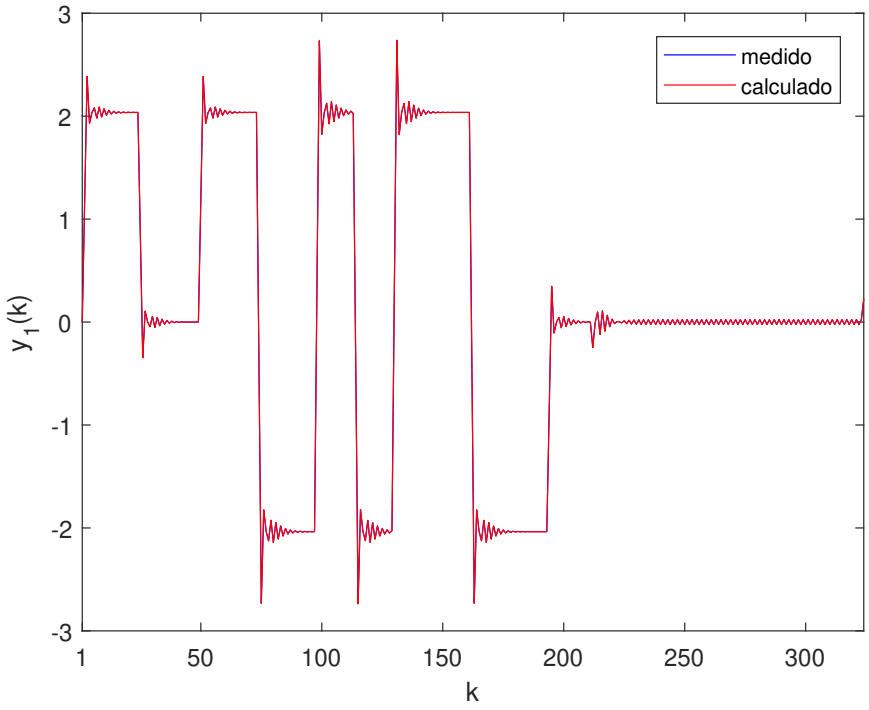

Figura 4.19: Valores medidos e calculados para y_1 (sistema de 2ª ordem sem tempo morto).

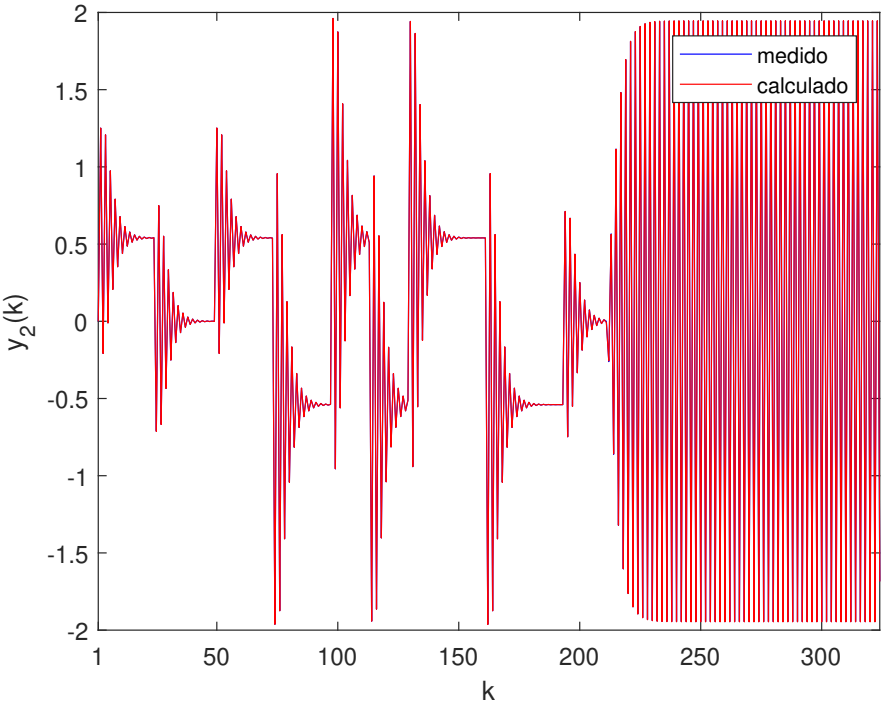

Figura 4.20: Valores medidos e calculados para y_2 (sistema de 2ª ordem sem tempo morto).

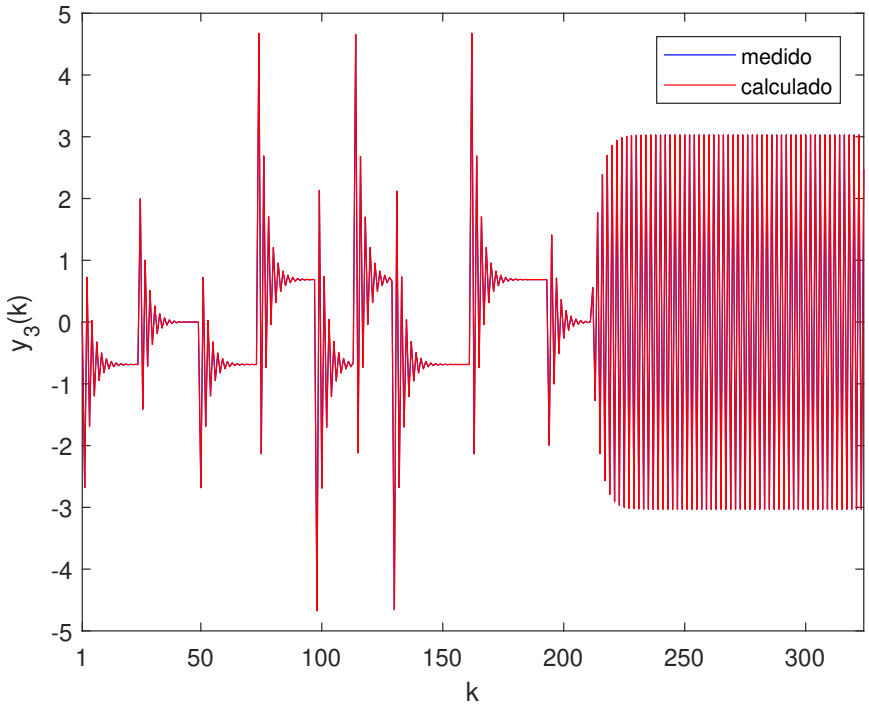

Figura 4.21: Valores medidos e calculados para y_3 (sistema de 2ª ordem sem tempo morto).

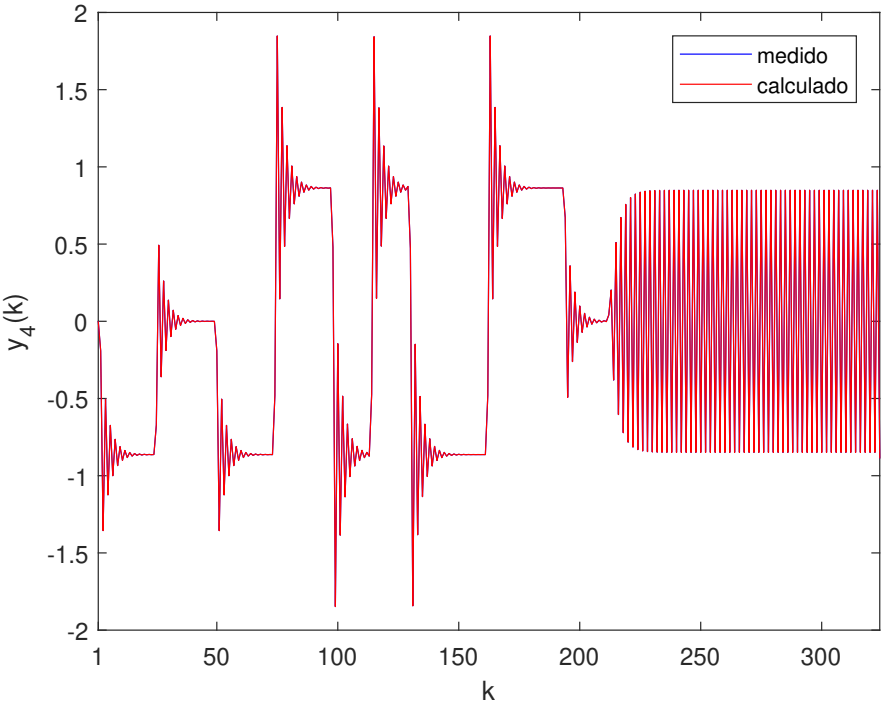

Figura 4.22: Valores medidos e calculados para y_4 (sistema de 2ª ordem sem tempo morto).

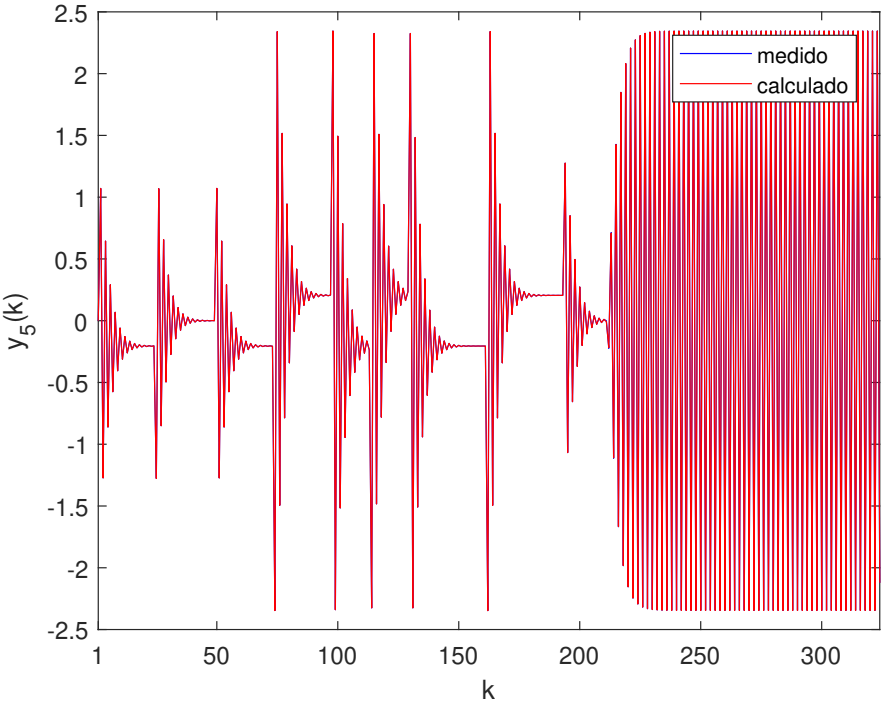

Figura 4.23: Valores medidos e calculados para y_5 (sistema de 2ª ordem sem tempo morto).

Com Tempo Morto

Seja o sistema MIMO 5 x 1 cuja matriz de transferência é mostrada abaixo:

$$\mathbf{G_p}(z^{-1}) = \begin{bmatrix} \dfrac{-0,107z^{-8} + 0,04487z^{-9} + 0,01985z^{-10}}{1 + 0,3438z^{-1} - 0,05841z^{-2}} \\ \dfrac{0,02318z^{-3} - 0,1542z^{-4} + 0,08496z^{-5}}{1 + 0,3438z^{-1} - 0,05841z^{-2}} \\ \dfrac{0,04098z^{-4} - 0,6971z^{-5} - 0,2372z^{-6}}{1 + 0,3438z^{-1} - 0,05841z^{-2}} \\ \dfrac{-0,115z^{-4} - 0,7557z^{-5} - 0,09702z^{-6}}{1 + 0,3438z^{-1} - 0,05841z^{-2}} \\ \dfrac{-0,2945z^{-9} - 0,1012z^{-10} + 0,04983z^{-11}}{1 + 0,3438z^{-1} - 0,05841z^{-2}} \end{bmatrix} \tag{4.12}$$

Utilizando o procedimento descrito na Seção 3.2 com $N_a = N_b = 10$, chegou-se ao seguinte modelo identificado:

$$\widehat{\mathbf{G}}_{\mathbf{p}}(z^{-1}) = \begin{bmatrix} \dfrac{-0,107z^{-8} + 0,06771z^{-9} + 0,009325z^{-10} - 0,003633z^{-11}}{1 + 0,1304z^{-1} - 0,1228z^{-2} + 0,01363z^{-3} - 0,001176z^{-4}} \\ \dfrac{0,02318z^{-3} - 0,1542z^{-4} + 0,08496z^{-5}}{1 + 0,3438z^{-1} - 0,05841z^{-2}} \\ \dfrac{0,04098z^{-4} - 0,7058z^{-5} - 0,08807z^{-6} + 0,04432z^{-7} - 0,0008z^{-8} + 0,0004z^{-9}}{1 + 0,1304z^{-1} - 0,1228z^{-2} + 0,01363z^{-3} - 0,001176z^{-4}} \\ \dfrac{-0,115z^{-4} - 0,7311z^{-5} + 0,06424z^{-6} + 0,0207z^{-7}}{1 + 0,1304z^{-1} - 0,1318z^{-2} + 0,01246z^{-3}} \\ \dfrac{-0,2945z^{-9} - 0,07974z^{-10} + 0,06341z^{-11} - 0,0013z^{-12} - 0,0009z^{-13}}{1 + 0,2708z^{-1} - 0,1045z^{-2} - 0,003619z^{-3}} \end{bmatrix} \tag{4.13}$$

sendo $t_{sim} = 12,04\,s$, MRSE $= 0,0420\%$ e MVAF $= 100\%$. As Figuras 4.24, 4.25, 4.26, 4.27 e 4.28 comparam os valores medidos e calculados para as saídas do processo.

Como no caso anterior, $\widehat{\mathbf{G}}_{\mathbf{p}}$ não é igual à matriz $\mathbf{G}_{\mathbf{p}}$. No entanto, além do modelo identificado ser bastante satisfatório (baixo MRSE e alto MVAF), os valores de tempo morto foram determinados com exatidão.

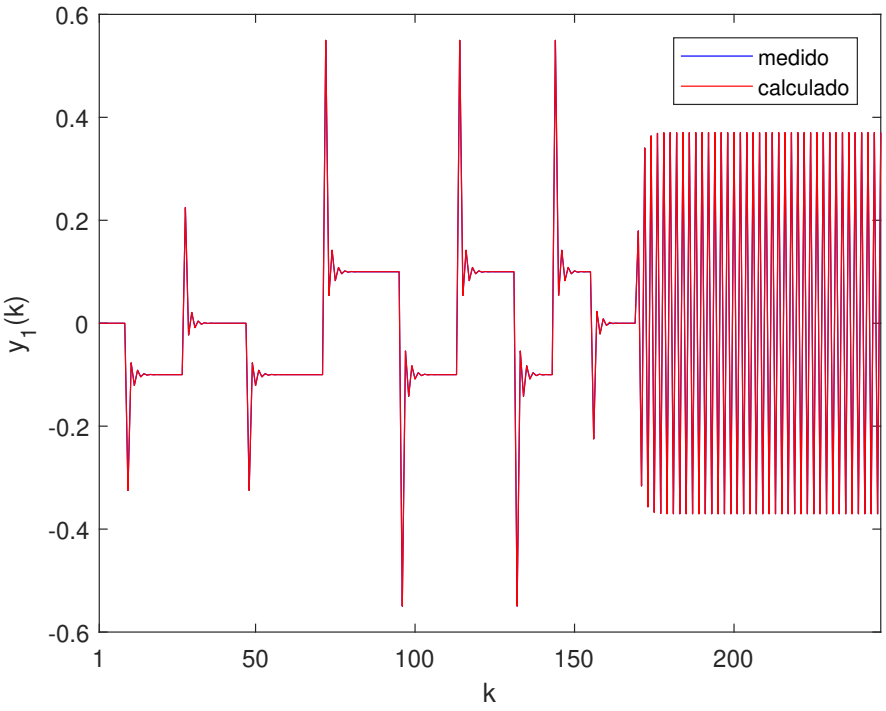

Figura 4.24: Valores medidos e calculados para y_1 (sistema de 2ª ordem com tempo morto).

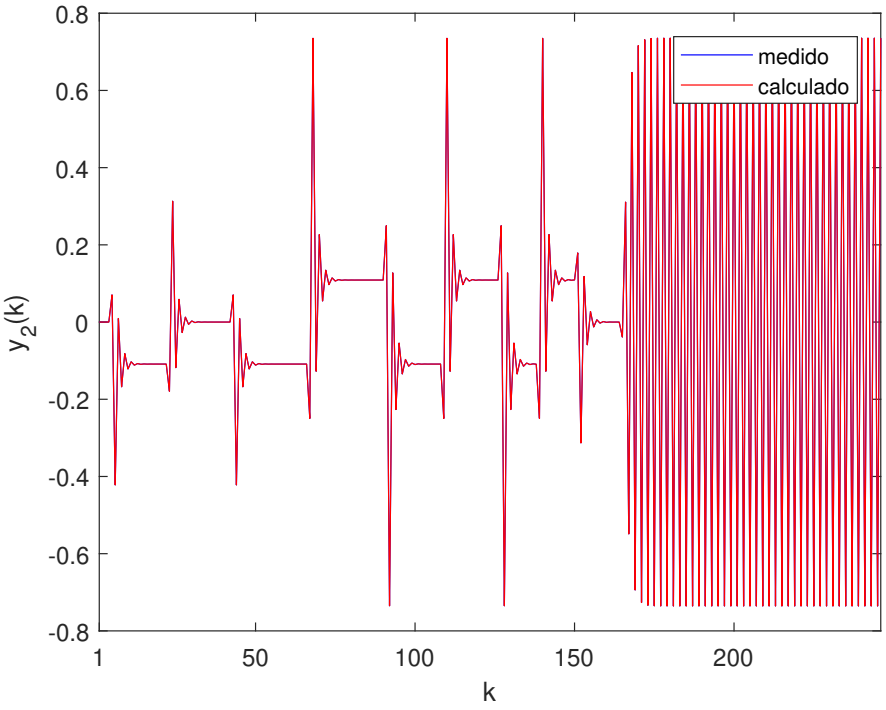

Figura 4.25: Valores medidos e calculados para y_2 (sistema de 2ª ordem com tempo morto).

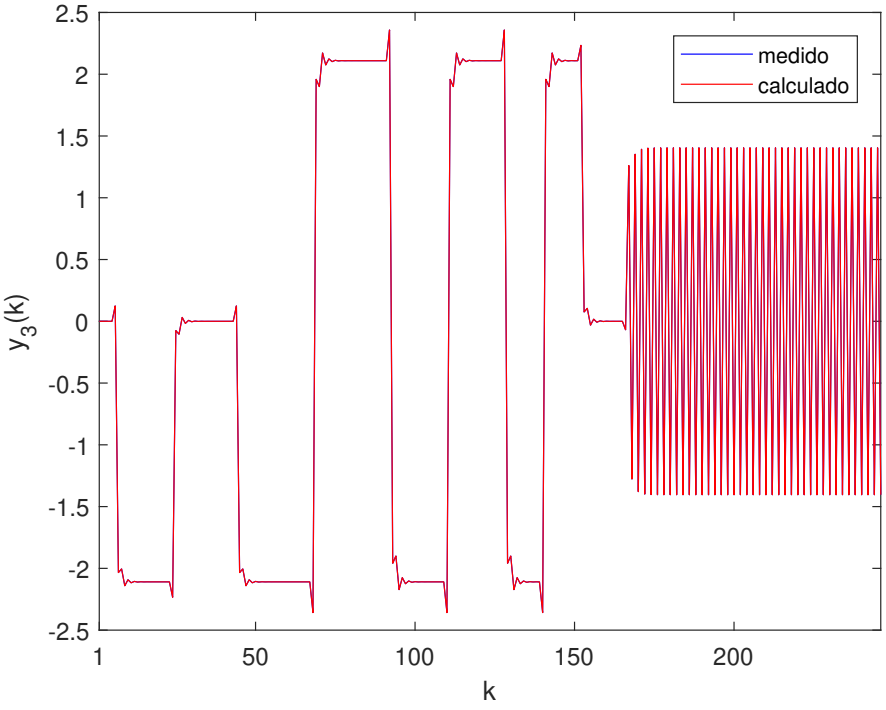

Figura 4.26: Valores medidos e calculados para y_3 (sistema de 2ª ordem com tempo morto).

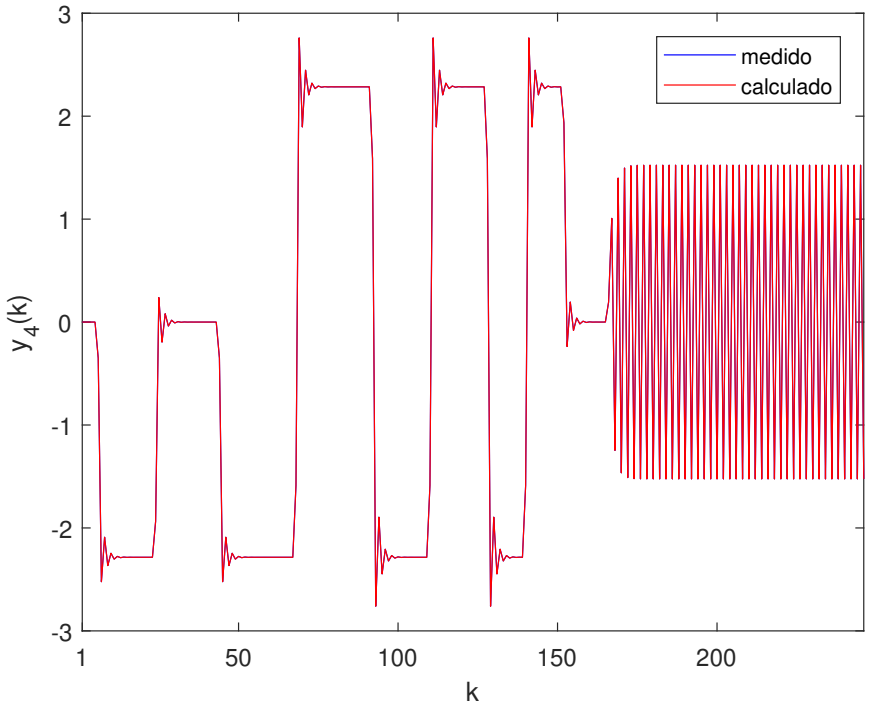

Figura 4.27: Valores medidos e calculados para y_4 (sistema de 2ª ordem com tempo morto).

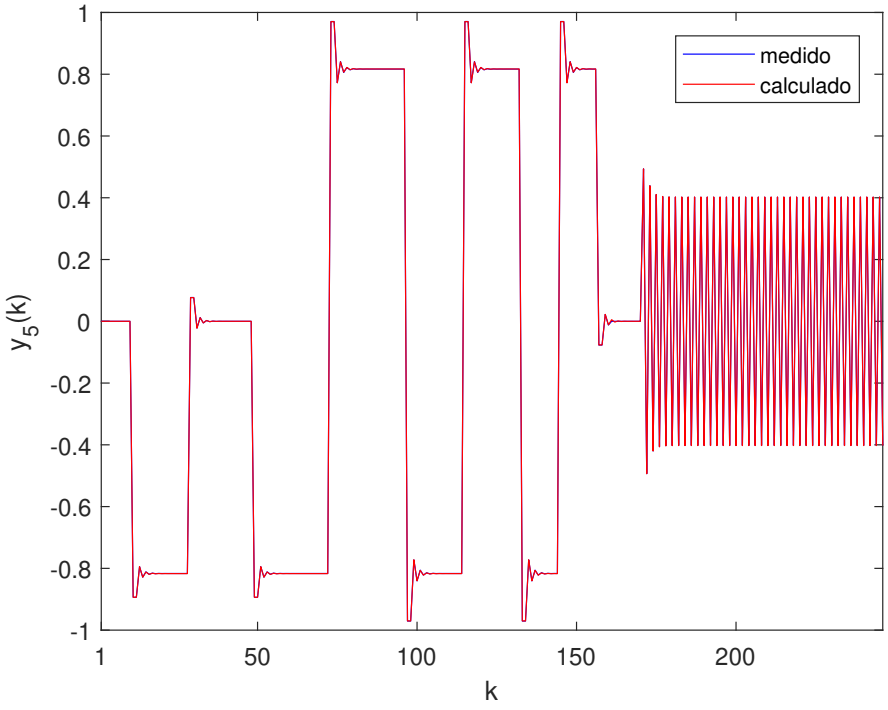

Figura 4.28: Valores medidos e calculados para y_5 (sistema de 2ª ordem com tempo morto).

4.1.3 Comparações com a Literatura

SOTOMAYOR *et al.* (2003) compararam a performance de 6 diferentes métodos de identificação baseados em modelo de subespaço de estados. Foi adicionado um ruído branco de média 0 e desvio padrão 0,05 a cada saída do sistema estudado. O método DSR (*Deterministic and Stochastic subspace system identification and Realization*) foi aquele que apresentou melhores resultados.

ORENSTEIN (2013), visando avaliar a desempenho de seu método, simulou a identificação de 75 sistemas lineares aleatórios. Foi adicionado a cada variável de saída um ruído branco de média 0 e desvio padrão igual a 5% do maior valor obtido para essa saída durante o teste degrau.

O método de identificação proposto neste trabalho foi avaliado com uma metodologia semelhante à de Orenstein (mesmo ruído branco), onde simulou-se a identificação de 40 sistemas lineares aleatórios com tempo morto (d) variando de 1 a 10. As comparações com os trabalhos de SOTOMAYOR *et al.* (2003) e ORENSTEIN (2013) são mostradas na Tabela 4.1.

Tabela 4.1: Desempenho de diferentes métodos de identificação.

Método de identificação	Sistemas estudados	Natureza dos dados utilizados no cálculo de MRSE e MVAF	MRSE (%)	MVAF (%)
DSR (SOTOMAYOR *et al.*, 2003)	2 x 5 (ordem = 3)	Validação Identificação	50,99 34,24	84,43 88,22
ORENSTEIN (2013)	4 x 3 (ordem $\leqslant$ 10)	Validação	70,87 (médio)	52,00 (médio)
Proposto neste trabalho	4 x 3 (ordem = 10)	Identificação	18,29 (médio)	95,88 (médio)

Percebe-se a partir da Tabela 4.1 que a metodologia proposta neste trabalho foi a que gerou os melhores resultados, ou seja, foi a que apresentou o menor valor para MRSE e o maior valor para MVAF, sendo portanto a mais robusta. Para chegar a esta conclusão, utilizou-se $N_a = 3$ e $N_b = 1$. O tempo médio de cada simulação foi de 27,30 segundos.

4.2 *Benchmark* da Shell

Neste problema *benchmark*, deve-se identificar a matriz de transferência da coluna de destilação mostrada na Figura 4.29, onde as variáveis de entrada são a vazão de vapor do destilado (D) e a carga térmica do refervedor (Q), e as variáveis de saída

são a pressão da coluna (P) e a composição do produto de fundo (X). A Tabela 4.2 mostra os limites operacionais destas variáveis.

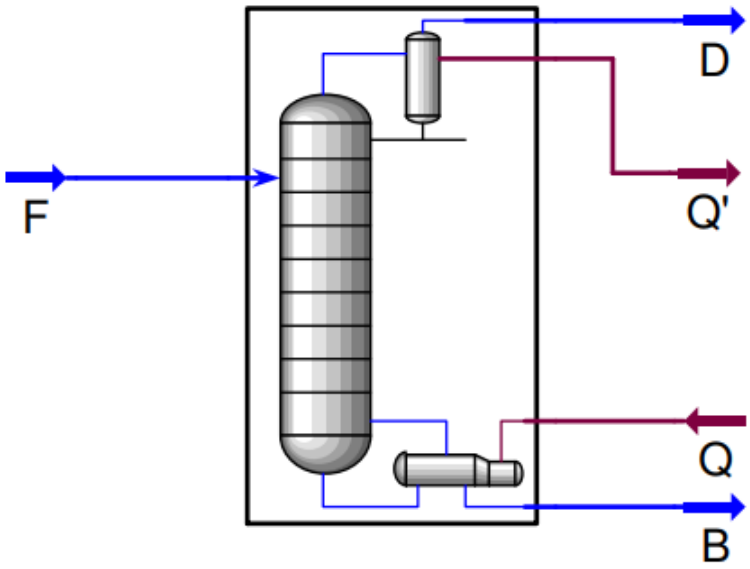

Figura 4.29: Fluxograma de processo para a coluna de destilação do *benchmark* da Shell. Estão representadas na figura as correntes de alimentação (F), do destilado (D) e do produto de fundo (B), assim como as cargas térmicas do refervedor (Q) e do condensador (Q').

Tabela 4.2: Condições de operação típicas da coluna de destilação. Fonte: Adaptado de COTT (1995a).

Variável	Valor nominal	Operação normal
Pressão da coluna (P)	2800	$2700 < P < 2900$
Composição do produto de fundo (X)	500	$250 < X < 1000$
Vazão de vapor do destilado (D)	20	$10 < D < 30$
Carga térmica do refervedor (Q)	2500	$2000 < Q < 3000$

As equações que relacionam as saídas com as entradas do processo são mostradas abaixo:

$$\begin{aligned} P^d(z^{-1}) = & \frac{-0,6096 + 0,4022z^{-1}}{1 - 1,5298z^{-1} + 0,5740z^{-2}} D^d(z^{-1}) \\ & + \frac{0,1055 - 0,0918z^{-1}}{1 - 1,5298z^{-1} + 0,5740z^{-2}} Q^d(z^{-1}) \\ & + \frac{N_s}{1 - 1,5945z^{-1} + 0,5945z^{-2}} e_P(z^{-1}) \end{aligned} \tag{4.14}$$

$$X(k) = 0,0765 \frac{500\,000}{Q(k-7) - 1500} + 0,9235X(k-1) + N_X(k) \tag{4.15}$$

$$N_X(z^{-1}) = \frac{N_s}{1 - 1,6595z^{-1} + 0,6595z^{-2}} e_X(z^{-1}) \tag{4.16}$$

em que P^d, D^d e Q^d são as variáveis-desvio de P, D e Q, respectivamente, e_P é um ruído branco de média 0 e desvio padrão 1,231, e_X é um ruído branco de média 0 e desvio padrão 0,677 e N_s é a intensidade dos ruídos (COTT, 1995b). Neste trabalho, foram considerados os valores de 0,02, 0,05 e 0,10 para N_s.

Os vetores **u** e **y** são definidos da seguinte forma:

$$u_1 = D^d = D - 20 \tag{4.17}$$

$$u_2 = Q^d = Q - 2500 \tag{4.18}$$

$$y_1 = P^d = P - 2800 \tag{4.19}$$

$$y_2 = X^d = X - 500 \tag{4.20}$$

Logo, os limites operacionais destes vetores são dados por:

$$-10 \leqslant u_1 \leqslant 10 \tag{4.21}$$

$$-500 \leqslant u_2 \leqslant 500 \tag{4.22}$$

$$-100 \leqslant y_1 \leqslant 100 \tag{4.23}$$

$$-250 \leqslant y_2 \leqslant 500 \tag{4.24}$$

Primeiramente, foi aplicado no pré-teste um degrau de magnitude 5 em u_1 e outro de magnitude 250 em u_2. Em seguida, utilizando o procedimento descrito na Seção 3.2 com $N_a = N_b = 2$, foram obtidos os resultados mostrados nas Figuras 4.30, 4.31, 4.32, 4.33, 4.34 e 4.35. A Tabela 4.3 sintetiza estes resultados.

Tabela 4.3: Resultados obtidos para o *benchmark* da Shell.

N_s	MRSE(%)	MVAF(%)	$t_{sim}(s)$
0,02	14,44	98,41	2,91
0,05	30,13	85,63	3,12
0,10	34,35	93,17	2,95

Por causa do ruído intenso e da não linearidade entre X e Q, o modelo identificado para a coluna de destilação não é tão acurado. No entanto, verifica-se que os resultados mostrados na Tabela 4.3 são melhores do que aqueles apresentados por ORENSTEIN (2013) em sua dissertação.

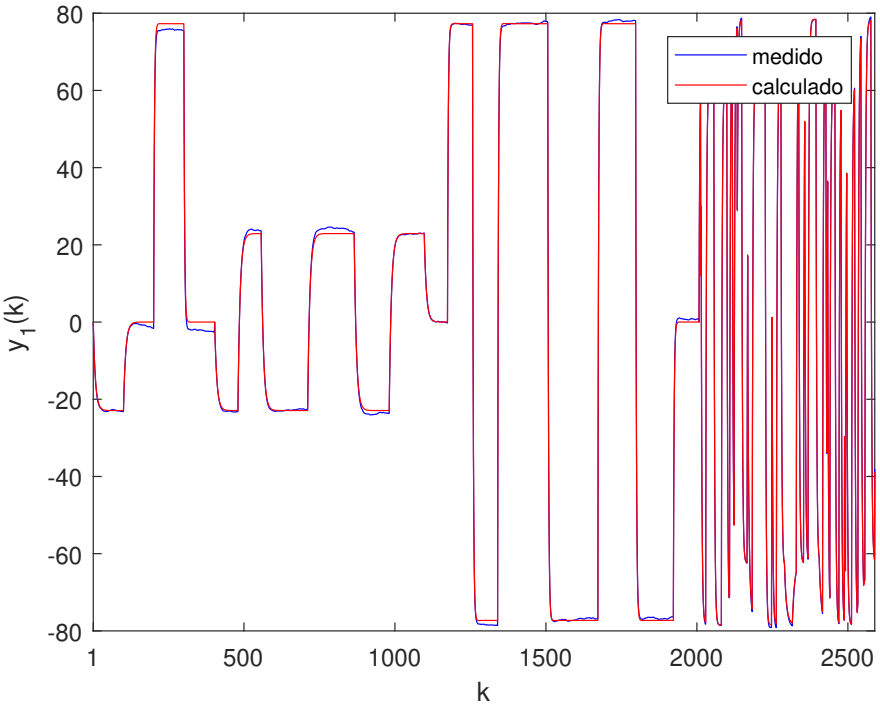

Figura 4.30: Valores medidos e calculados para y_1 no *benchmark* da Shell ($N_s = 0,02$).

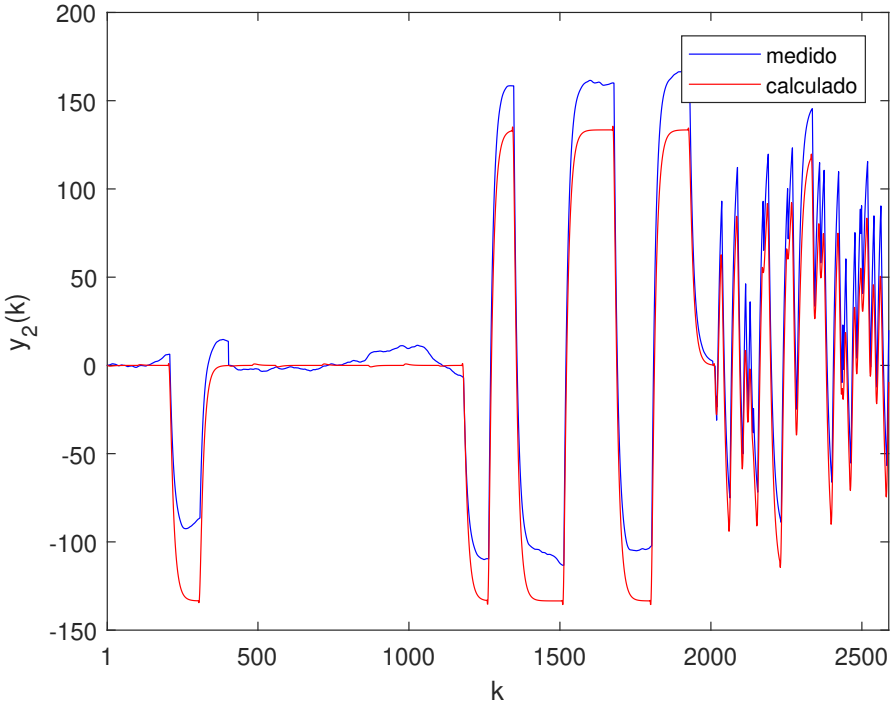

Figura 4.31: Valores medidos e calculados para y_2 no *benchmark* da Shell ($N_s = 0,02$).

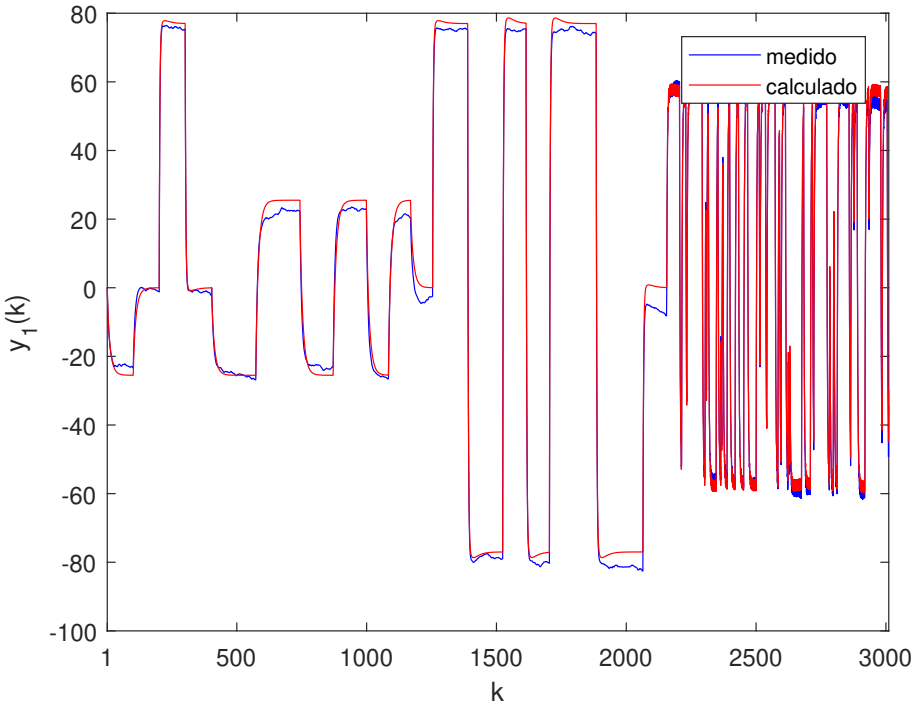

Figura 4.32: Valores medidos e calculados para y_1 no *benchmark* da Shell ($N_s = 0,05$).

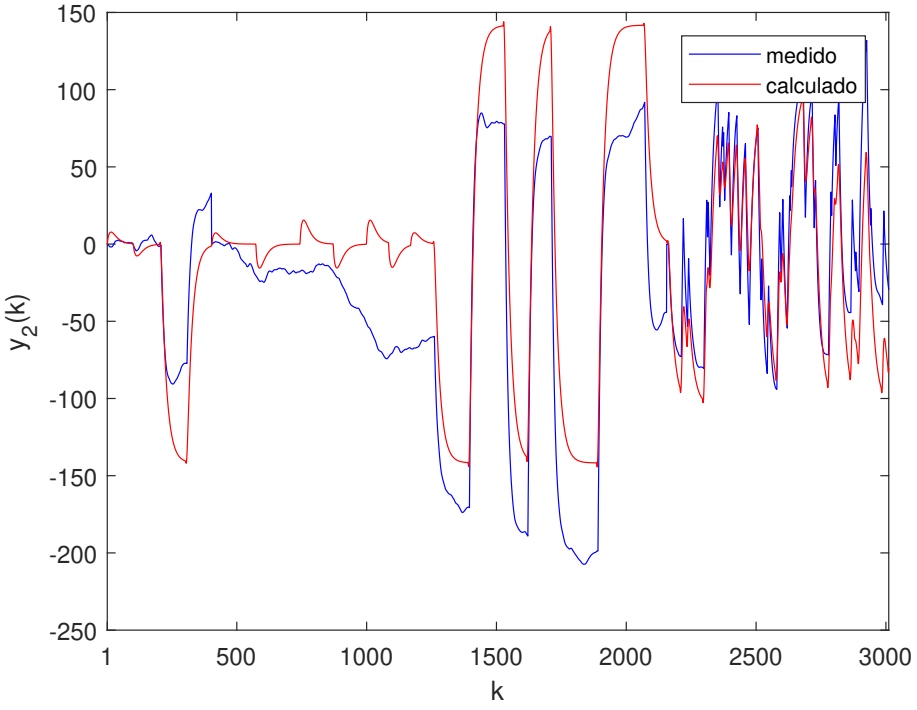

Figura 4.33: Valores medidos e calculados para y_2 no *benchmark* da Shell ($N_s = 0,05$).

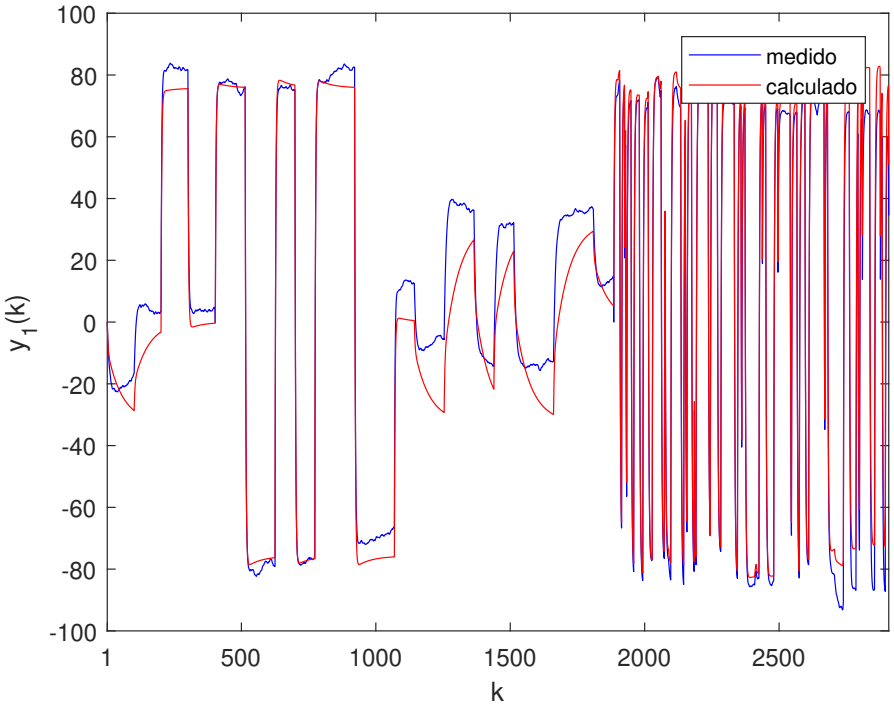

Figura 4.34: Valores medidos e calculados para y_1 no *benchmark* da Shell ($N_s = 0, 10$).

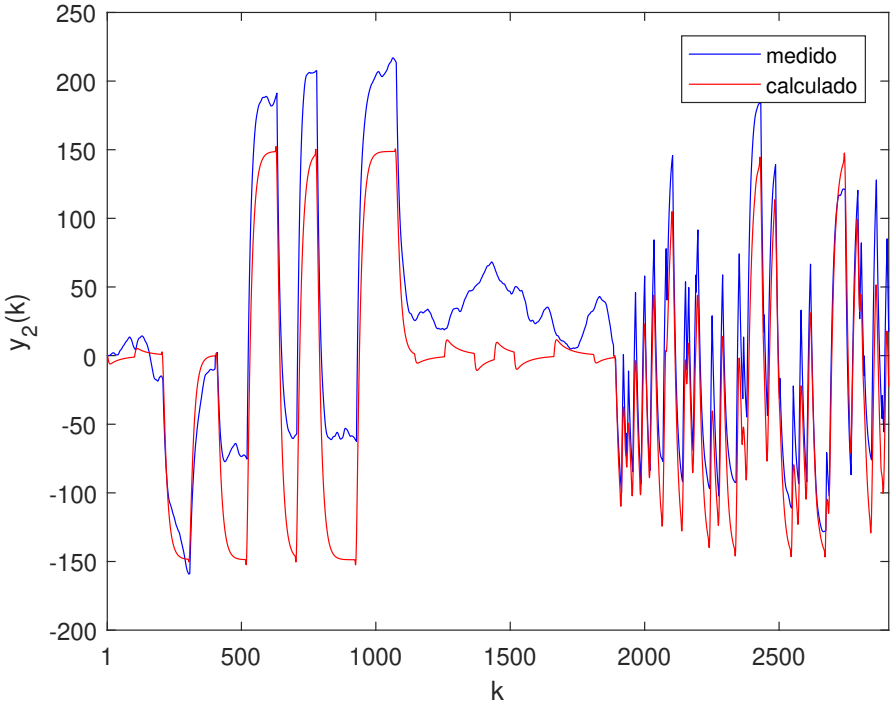

Figura 4.35: Valores medidos e calculados para y_2 no *benchmark* da Shell ($N_s = 0, 10$).

Capítulo 5

Conclusão

A metodologia de identificação desenvolvida neste trabalho apresentou resultados bastante satisfatórios, uma vez que foi capaz de gerar modelos fiéis à realidade em tempos de simulação relativamente curtos (da ordem de segundos). Além disso, ela se mostrou melhor do que o método de Orenstein em todas as comparações realizadas.

O uso de informação fenomenológica no processo de identificação, principal característica desta metodologia, assegura a consistência física dos modelos gerados. Em outras palavras, não é mais necessário refazer etapas como o pré-teste e o teste degrau com tanta frequência. Desse modo, como o teste GBN também está mais curto (7 vezes o maior tempo de assentamento do sistema ao invés de 12), há uma redução no tempo total de identificação em relação a Orenstein.

Como sugestão de um trabalho futuro, a metodologia aqui desenvolvida poderia ser estendida para o contexto da identificação de sistemas com modelos não lineares. Assim sendo, os mesmos princípios utilizados neste trabalho poderiam ser empregados na identificação não linear, onde o modelo ARX seria substituído, por exemplo, pelo NARX.

Referências Bibliográficas

AKAIKE, H., 1974, "A New Look at the Statistical Model Identification", *IEEE Transactions on Automatic Control*, v. 19, n. 6, pp. 716–723.

BANKS, H., JOYNER, M. L., 2017, "AIC under the framework of least squares estimation", *Applied Mathematics Letters*, v. 74, pp. 33–45.

CHEN, J., CHEN, Z., 2008, "Extended Bayesian information criteria for model selection with large model spaces", *Biometrika*, v. 95, n. 3, pp. 759–771.

CHEN, J.-K., YU, C.-C., 1997, "Optimal Input Design Using Generalized Binary Sequence", *Automatica*, v. 33, n. 11, pp. 2081–2084.

COOK, J., 2011, *Basic properties of the soft maximum*. Relatório técnico, The University of Texas M. D. Anderson Cancer Center, Houston, Texas, USA, September.

COTT, B. J., 1995a, "Introduction to the Process Identification Workshop at the 1992 Canadian Chemical Engineering Conference", *Journal of Process Control*, v. 5, n. 2, pp. 67–69.

COTT, B. J., 1995b, "Summary of the Process Identification Workshop at the 1992 Canadian Chemical Engineering Conference", *Journal of Process Control*, v. 5, n. 2, pp. 109–113.

DARBY, M. L., NIKOLAOU, M., 2014, "Identification test design for multivariable model-based control: An industrial perspective", *Control Engineering Practice*, v. 22, pp. 165–180.

EFRON, B., HASTIE, T., JOHNSTONE, I., et al., 2004, "Least Angle Regression", *The Annals of Statistics*, v. 32, n. 2, pp. 407–499.

GAIKWAD, S., RIVERA, D., 1996, "Control-Relevant Input Signal Design for Multivariable System Identification: Application to High-Purity Distillation", *IFAC Proceedings Volumes*, v. 29, n. 1, pp. 6143–6148.

GOLUB, G. H., HEATH, M., WAHBA, G., 1979, "Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter", *Technometrics*, v. 21, n. 2, pp. 215–223.

HÄGG, P., LARSSON, C. A., EBADAT, A., et al., 2014, "Input Signal Generation for Constrained Multiple-Input Multple-Output Systems", *IFAC Proceedings Volumes*, v. 47, n. 3, pp. 1410–1415.

HROMČÍK, M., ŠEBEKT, M., 1999, "New Algorithm for Polynomial Matrix Determinant Based on FFT". In: *1999 European Control Conference (ECC)*, pp. 4173–4177, Karlsruhe, Germany, August.

HUNG, N. T., ISMAIL, I., SAAD, N. B., et al., 2015, "Design of Optimal GBN Sequences for Identification of MIMO Systems". In: *The 10th Asian Control Conference (ASCC)*, pp. 1–6, Kota Kinabalu, Malaysia, May.

HURVICH, C. M., TSAI, C.-L., 1989, "Regression and Time Series Model Selection in Small Samples", *Biometrika*, v. 76, n. 2, pp. 297–307.

LANGE, M., ZÜHLKE, D., HOLZ, O., et al., 2014, "Applications of l_p-Norms and their Smooth Approximations for Gradient Based Learning Vector Quantization". In: *ESANN 2014 proceedings*, pp. 271–276, Bruges, Belgium, April.

MALLOWS, C. L., 1973, "Some Comments on Cp", *Technometrics*, v. 15, n. 4, pp. 661–675.

MARLIN, T. E., 2015, "Process Control: Designing Processes and Control Systems for Dynamic Performance". 2 ed., pp. 179–181. Disponível em: <http://pc-textbook.mcmaster.ca/>.

MURAKAMI, K., SEBORG, D. E., 2000, "Constrained parameter estimation with applications to blending operations", *Journal of Process Control*, v. 10, pp. 195–202.

ORENSTEIN, L. P., 2013, *Procedimento para identificação de sistemas dinâmicos em ambiente industrial*. Dissertação de mestrado, UFRJ/COPPE/Programa de Engenharia Elétrica, Rio de Janeiro, RJ, Brasil.

PEREPU, S. K., TANGIRALA, A. K., 2017, "Identification of MIMO ARX models from small samples using sparse matrix optimization". In: *2017 Indian Control Conference (ICC)*, pp. 47–52, Guwahati, India, January.

SCHWARZ, G., 1978, "Estimating the Dimension of a Model", *The Annals of Statistics*, v. 6, n. 2, pp. 461–464.

SMITH, C., 1972, *Digital Computer Process Control*. Scranton, PA, Intext Education Publishers.

SOHLBERG, B., JACOBSEN, E., 2008, "Grey Box Modelling – Branches and Experiences", *IFAC Proceedings Volumes*, v. 41, n. 2, pp. 11415–11420.

SOTOMAYOR, O. A., PARK, S. W., GARCIA, C., 2003, "Multivariable identification of an activated sludge process with subspace-based algorithms", *Control Engineering Practice*, v. 11, n. 8, pp. 961–969.

STOJANOVIC, V., FILIPOVIC, V., 2014, "Adaptive Input Design for Identification of Output Error Model with Constrained Output", *Circuits, Systems, and Signal Processing*, v. 33, n. 1, pp. 97–113.

TIBSHIRANI, R., 1996, "Regression Shrinkage and Selection via the Lasso", *Journal of the Royal Statistical Society. Series B (Methodological)*, v. 58, n. 1, pp. 267–288.

TIKHONOV, A. N., 1963, "Solution of incorrectly formulated problems and the regularization method", *Soviet Mathematics*, v. 4, pp. 1035–1038.

TULLEKEN, H. J. A. F., 1993, "Grey-box Modelling and Identification Using Physical Knowledge and Bayesian Techniques", *Automatica*, v. 29, n. 2, pp. 285–308.

WANG, D., SHEN, J., LI, Y., et al., 2015, "An Excitation Signal Design for Identification with the Amplitude Constraints". In: *The 34th Chinese Control Conference*, pp. 1941–1946, Hangzhou, China, July.

ZHU, Y., 2001, "Multivariable System Identification for Process Control". 1 ed., pp. 45–48, Elsevier Science.

ZHU, Y., VAN DEN BOSCH, P. P., 2000, "Optimal closed-loop identification test design for internal model control", *Automatica*, v. 36, n. 8, pp. 1237–1241.

ZOU, H., 2006, "The Adaptive Lasso and Its Oracle Properties", *Journal of the American Statistical Association*, v. 101, n. 476, pp. 1418–1429.

ZOU, H., HASTIE, T., TIBSHIRANI, R., 2007, "On the "Degrees of Freedom"of the Lasso", *The Annals of Statistics*, v. 35, n. 5, pp. 2173–2192.

Apêndice A

Gradientes e Hessianas

Neste apêndice são deduzidas as fórmulas matemáticas utilizadas pelo Algoritmo 3 para calcular determinados gradientes e Hessianas. O código do algoritmo é mostrado no Apêndice B.

A.1 Teoremas Úteis

Os teoremas apresentados a seguir são utilizados nas deduções matemáticas das próximas seções.

A.1.1 Propriedades do Traço de uma Matriz

Teorema 1. *Sejam* $\boldsymbol{A} = [a_{ij}]_{n\times n}$, $\boldsymbol{B} = [b_{ij}]_{n\times n}$ *e* $c_1, c_2 \in \mathbb{R}$. *Então:*

1. $\operatorname{tr}(c_1\boldsymbol{A} + c_2\boldsymbol{B}) = c_1 \operatorname{tr}(\boldsymbol{A}) + c_2 \operatorname{tr}(\boldsymbol{B})$
2. $\operatorname{tr}(\boldsymbol{A}^T) = \operatorname{tr}(\boldsymbol{A})$

Demonstração.

1. O traço é um operador linear:

$$\begin{aligned}\operatorname{tr}(c_1\mathbf{A} + c_2\mathbf{B}) &= \sum_{i=1}^{n}(c_1\mathbf{A} + c_2\mathbf{B})_{ii} = \sum_{i=1}^{n}(c_1 a_{ii} + c_2 b_{ii}) \\ &= \sum_{i=1}^{n} c_1 a_{ii} + \sum_{i=1}^{n} c_2 b_{ii} = c_1 \sum_{i=1}^{n} a_{ii} + c_2 \sum_{i=1}^{n} b_{ii} \\ &= c_1 \operatorname{tr}(\mathbf{A}) + c_2 \operatorname{tr}(\mathbf{B})\end{aligned}$$

2. Quando uma matriz é transposta, a posição dos elementos de sua diagonal principal permanece inalterada. Consequentemente, $\operatorname{tr}(\mathbf{A}^T) = \operatorname{tr}(\mathbf{A})$.

□

Teorema 2. *Sejam* $\boldsymbol{A} = [a_{ij}]_{m\times n}$, $\boldsymbol{B} = [b_{ij}]_{m\times n}$ *e* $\boldsymbol{C} = [c_{ij}]_{n\times m}$. *Então:*

1. $\mathrm{tr}(\boldsymbol{AC}) = \mathrm{tr}(\boldsymbol{CA})$
2. $\mathrm{tr}(\boldsymbol{A}^T\boldsymbol{B}) = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}b_{ij}$
3. $\mathrm{tr}(\boldsymbol{A}^T\boldsymbol{A}) = \|\boldsymbol{A}\|_F^2$ *se* $a_{ij} \in \mathbb{R}$

Demonstração.

1. Seja $(\mathbf{AC})_{ik}$ o componente (i,k) de $\mathbf{AC}$. A partir da definição do produto de duas matrizes, tem-se que:

$$(\mathbf{AC})_{ik} = \sum_{j=1}^{n} a_{ij}c_{jk}$$
$$\mathrm{tr}(\mathbf{AC}) = \sum_{i=1}^{m}(\mathbf{AC})_{ii} = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}c_{ji}$$

   Utilizando um raciocínio análogo para a matriz $\mathbf{CA}$, tem-se que:

$$(\mathbf{CA})_{jk} = \sum_{i=1}^{m} c_{ji}a_{ik}$$
$$\mathrm{tr}(\mathbf{CA}) = \sum_{j=1}^{n}(\mathbf{CA})_{jj} = \sum_{j=1}^{n}\sum_{i=1}^{m} c_{ji}a_{ij} = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}c_{ji}$$

   Ou seja, $\mathrm{tr}(\mathbf{AC}) = \mathrm{tr}(\mathbf{CA})$.

2. A partir da definição do produto de duas matrizes, verifica-se que:

$$(\mathbf{A}^T\mathbf{B})_{jk} = \sum_{i=1}^{m}(\mathbf{A}^T)_{ji}b_{ik} = \sum_{i=1}^{m} a_{ij}b_{ik}$$
$$\mathrm{tr}(\mathbf{A}^T\mathbf{B}) = \sum_{j=1}^{n}(\mathbf{A}^T\mathbf{B})_{jj} = \sum_{j=1}^{n}\sum_{i=1}^{m} a_{ij}b_{ij} = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}b_{ij}$$

3. Fazendo $\mathbf{B} = \mathbf{A}$ no item 2 deste teorema, chega-se ao seguinte resultado:

$$\mathrm{tr}(\mathbf{A}^T\mathbf{A}) = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}a_{ij} = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2$$
$$= \left(\underbrace{\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}}_{\|\mathbf{A}\|_F}\right)^2 = \|\mathbf{A}\|_F^2$$

□

A.1.2 Derivada do Traço de uma Matriz

Teorema 3. *Seja* $\boldsymbol{A}(x) = [a_{ij}(x)]_{n\times n}$*, em que* $x \in \mathbb{R}$*. Se* $\boldsymbol{A}$ *for diferenciável em um ponto* x*, então o seu traço,* $\mathrm{tr}(\boldsymbol{A})$*, também será diferenciável nesse ponto, sendo a derivada dada por:*

$$\frac{d(\mathrm{tr}(\boldsymbol{A}))}{dx} = \mathrm{tr}\left(\frac{d\boldsymbol{A}}{dx}\right)$$

Demonstração.

$$\frac{d(\mathrm{tr}(\mathbf{A}))}{dx} = \frac{d}{dx}\left(\sum_{i=1}^{n} a_{ii}\right) = \sum_{i=1}^{n} \frac{da_{ii}}{dx} = \sum_{i=1}^{n} \left(\frac{d\mathbf{A}}{dx}\right)_{ii} = \mathrm{tr}\left(\frac{d\mathbf{A}}{dx}\right)$$

□

A.1.3 Derivada da Inversa de uma Matriz

Teorema 4. *Seja* $\boldsymbol{A}(x) = [a_{ij}(x)]_{n\times n}$*, em que* $x \in \mathbb{R}$*. Se* $\boldsymbol{A}$ *for diferenciável e inversível em um ponto* x*, então a sua inversa,* $\boldsymbol{A}^{-1}$*, também será diferenciável nesse ponto, sendo a derivada dada por:*

$$\frac{d\boldsymbol{A}^{-1}}{dx} = -\boldsymbol{A}^{-1}\frac{d\boldsymbol{A}}{dx}\boldsymbol{A}^{-1}$$

Demonstração.

$$\begin{aligned}
\mathbf{A}\mathbf{A}^{-1} &= \mathbf{I} \\
\frac{d(\mathbf{A}\mathbf{A}^{-1})}{dx} &= \mathbf{0} \\
\mathbf{A}\frac{d\mathbf{A}^{-1}}{dx} + \frac{d\mathbf{A}}{dx}\mathbf{A}^{-1} &= \mathbf{0} \\
\mathbf{A}\frac{d\mathbf{A}^{-1}}{dx} &= -\frac{d\mathbf{A}}{dx}\mathbf{A}^{-1} \\
\underbrace{\mathbf{A}^{-1}\mathbf{A}}_{\mathbf{I}}\frac{d\mathbf{A}^{-1}}{dx} &= -\mathbf{A}^{-1}\frac{d\mathbf{A}}{dx}\mathbf{A}^{-1} \\
\frac{d\mathbf{A}^{-1}}{dx} &= -\mathbf{A}^{-1}\frac{d\mathbf{A}}{dx}\mathbf{A}^{-1}
\end{aligned}$$

□

A.1.4 Derivada do Determinante de uma Matriz

Teorema 5 (fórmula de Jacobi)**.** *Seja* $\boldsymbol{A}(x) = [a_{ij}(x)]_{n\times n}$*, em que* $x \in \mathbb{R}$*. Se* $\boldsymbol{A}$ *for diferenciável em um ponto* x*, então o seu determinante,* $\det(\boldsymbol{A})$*, também será*

diferenciável nesse ponto, sendo a derivada dada por:

$$\frac{d(\det(\boldsymbol{A}))}{dx} = \operatorname{tr}\left(\operatorname{adj}(\boldsymbol{A})\frac{d\boldsymbol{A}}{dx}\right)$$

em que $\operatorname{adj}(\boldsymbol{A})$ *é a adjunta da matriz* $\boldsymbol{A}$*.*

Demonstração. Seja C_{ij} o cofator do elemento (i,j) de $\mathbf{A}$ e δ_{ij}^K o delta de Kronecker, definido como segue:

$$\delta_{ij}^K = \begin{cases} 1, & \text{se } i = j \\ 0, & \text{se } i \neq j \end{cases}$$

Desse modo:

$$\begin{aligned}
\frac{d(\det(\mathbf{A}))}{dx} &= \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial(\det(\mathbf{A}))}{\partial a_{ij}}\frac{da_{ij}}{dx} && \text{Regra da cadeia} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial}{\partial a_{ij}}\left(\sum_{k=1}^{n} a_{ik}C_{ik}\right)\frac{da_{ij}}{dx} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\sum_{k=1}^{n}\frac{\partial(a_{ik}C_{ik})}{\partial a_{ij}}\right)\frac{da_{ij}}{dx} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\sum_{k=1}^{n} a_{ik}\cancelto{0}{\frac{\partial C_{ik}}{\partial a_{ij}}} + \sum_{k=1}^{n} C_{ik}\underbrace{\frac{\partial a_{ik}}{\partial a_{ij}}}_{\delta_{kj}^K}\right)\frac{da_{ij}}{dx} && C_{ik} \text{ não depende de } a_{ij} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\sum_{k=1}^{n} C_{ik}\delta_{kj}^K\right)\frac{da_{ij}}{dx} = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}\frac{da_{ij}}{dx} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}(\operatorname{adj}(\mathbf{A})^T)_{ij}\left(\frac{d\mathbf{A}}{dx}\right)_{ij} \\
&= \operatorname{tr}\left(\operatorname{adj}(\mathbf{A})\frac{d\mathbf{A}}{dx}\right) && \text{Teorema 2, item 2}
\end{aligned}$$

□

A.1.5 Derivada do conjugado de um número complexo

Teorema 6. *Seja* $z\colon \mathbb{R} \to \mathbb{C}$ *tal que* $z(t) = x(t) + iy(t)$*, em que* i *é a unidade imaginária. Se* z *for diferenciável em um ponto* t*, então o seu conjugado,* $\overline{z}$*, também será diferenciável nesse ponto, sendo a derivada dada por:*

$$\frac{d\overline{z}}{dt} = \overline{\frac{dz}{dt}}$$

Demonstração. Se z é diferenciável em t, então x e y também são diferenciáveis nesse ponto. Assim:

$$\begin{aligned}\overline{z}(t) &= x(t) - iy(t)\\ \frac{d\overline{z}}{dt} &= \frac{dx}{dt} - i\frac{dy}{dt} = \overline{\frac{dx}{dt} + i\frac{dy}{dt}} = \overline{\frac{dz}{dt}}\end{aligned}$$

□

A.2 Validação Cruzada Generalizada (GCV)

A.2.1 Gradiente da Função Objetivo

Sejam as matrizes $\mathbf{M}_1 = \mathbf{M}(\omega)$, $\mathbf{M}_2 = (\mathbf{I} - \mathbf{M}_1)\mathbf{Y}$ e $\mathbf{M}_3 = \mathbf{I} - \mathbf{M}_1$. Utilizando estas definições, a função objetivo $f(\omega)$ pode ser reescrita da seguinte forma:

$$\begin{aligned}f(\omega) &= \frac{\frac{1}{N}\|(\mathbf{I} - \mathbf{M}(\omega))\mathbf{Y}\|_F^2}{\left[\frac{1}{N}\operatorname{tr}(\mathbf{I} - \mathbf{M}(\omega))\right]^2} = \frac{\frac{1}{N}\|\overbrace{(\mathbf{I} - \mathbf{M}_1)\mathbf{Y}}^{\text{matriz } \mathbf{M}_2}\|_F^2}{\left[\frac{1}{N}\operatorname{tr}(\underbrace{\mathbf{I} - \mathbf{M}_1}_{\text{matriz } \mathbf{M}_3})\right]^2}\\ &= \frac{\frac{1}{N}\|\mathbf{M}_2\|_F^2}{\left[\frac{1}{N}\operatorname{tr}(\mathbf{M}_3)\right]^2} = \frac{\|\mathbf{M}_2\|_F^2}{\cancel{N}} \cdot \frac{\cancel{N^2}}{\operatorname{tr}(\mathbf{M}_3)^2} = N\left[\frac{\|\mathbf{M}_2\|_F^2}{\operatorname{tr}(\mathbf{M}_3)^2}\right]\\ &= N\left[\frac{\operatorname{tr}(\mathbf{M}_2^T\mathbf{M}_2)}{\operatorname{tr}(\mathbf{M}_3)^2}\right] \qquad \text{Teorema 2, item 3}\end{aligned}$$

A derivada de f em relação à ω é dada por:

$$\begin{aligned}\frac{df}{d\omega} &= N\left[\frac{\operatorname{tr}(\mathbf{M}_3)^2\frac{d(\operatorname{tr}(\mathbf{M}_2^T\mathbf{M}_2))}{d\omega} - \operatorname{tr}(\mathbf{M}_2^T\mathbf{M}_2)\frac{d(\operatorname{tr}(\mathbf{M}_3)^2)}{d\omega}}{\operatorname{tr}(\mathbf{M}_3)^4}\right]\\ &= N\left[\frac{\operatorname{tr}(\mathbf{M}_3)^2\frac{d(\operatorname{tr}(\mathbf{M}_2^T\mathbf{M}_2))}{d\omega} - \operatorname{tr}(\mathbf{M}_2^T\mathbf{M}_2)\cdot 2\operatorname{tr}(\mathbf{M}_3)\frac{d(\operatorname{tr}(\mathbf{M}_3))}{d\omega}}{\operatorname{tr}(\mathbf{M}_3)^4}\right]\\ &= N\left[\frac{\operatorname{tr}(\mathbf{M}_3)^2\operatorname{tr}\left(\frac{d(\mathbf{M}_2^T\mathbf{M}_2)}{d\omega}\right) - 2\operatorname{tr}(\mathbf{M}_2^T\mathbf{M}_2)\operatorname{tr}(\mathbf{M}_3)\operatorname{tr}\left(\frac{d\mathbf{M}_3}{d\omega}\right)}{\operatorname{tr}(\mathbf{M}_3)^4}\right]\end{aligned}$$

Teorema 3

$$
\begin{aligned}
&= N \left[\frac{\operatorname{tr}(\mathbf{M}_3)^2 \operatorname{tr}\left(\mathbf{M}_2^T \dfrac{d\mathbf{M}_2}{d\omega} + \left(\dfrac{d\mathbf{M}_2}{d\omega} \right)^T \mathbf{M}_2 \right) - 2 \operatorname{tr}(\mathbf{M}_2^T \mathbf{M}_2) \operatorname{tr}(\mathbf{M}_3) \operatorname{tr}\left(\dfrac{d\mathbf{M}_3}{d\omega} \right)}{\operatorname{tr}(\mathbf{M}_3)^4} \right] \\
&= N \left[\frac{2 \operatorname{tr}(\mathbf{M}_3)^2 \operatorname{tr}\left(\mathbf{M}_2^T \dfrac{d\mathbf{M}_2}{d\omega} \right) - 2 \operatorname{tr}(\mathbf{M}_2^T \mathbf{M}_2) \operatorname{tr}(\mathbf{M}_3) \operatorname{tr}\left(\dfrac{d\mathbf{M}_3}{d\omega} \right)}{\operatorname{tr}(\mathbf{M}_3)^4} \right] \quad \text{Teorema 1} \\
&= 2N \underbrace{\left[\frac{\operatorname{tr}\left(\mathbf{M}_2^T \dfrac{d\mathbf{M}_2}{d\omega} \right)}{\operatorname{tr}(\mathbf{M}_3)^2} \right]}_{\text{escalar } E_1} - 2N \underbrace{\left[\frac{\operatorname{tr}(\mathbf{M}_2^T \mathbf{M}_2) \operatorname{tr}\left(\dfrac{d\mathbf{M}_3}{d\omega} \right)}{\operatorname{tr}(\mathbf{M}_3)^3} \right]}_{\text{escalar } E_2} \\
&= 2NE_1 - 2NE_2 \\
&= 2N(E_1 - E_2) \qquad \text{(A.1)}
\end{aligned}
$$

Para determinar o valor dos escalares E_1 e E_2, é necessário conhecer primeiro o valor das derivadas de $\mathbf{M}_2$ e $\mathbf{M}_3$ em relação à ω. Assim sendo:

$$
\begin{aligned}
\mathbf{M}_2 &= (\mathbf{I} - \mathbf{M}_1)\mathbf{Y} = \mathbf{Y} - \mathbf{M}_1\mathbf{Y} & \qquad \mathbf{M}_3 &= \mathbf{I} - \mathbf{M}_1 \\
\frac{d\mathbf{M}_2}{d\omega} &= -\frac{d\mathbf{M}_1}{d\omega}\mathbf{Y} \qquad \text{(A.2)} & \qquad \frac{d\mathbf{M}_3}{d\omega} &= -\frac{d\mathbf{M}_1}{d\omega} \qquad \text{(A.3)}
\end{aligned}
$$

Calculando a derivada de $\mathbf{M}_1$ em relação à ω:

$$
\begin{aligned}
\mathbf{M}_1 &= \mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}\mathbf{\Phi}^T \\
\frac{d\mathbf{M}_1}{d\omega} &= \mathbf{\Phi}\frac{d}{d\omega}[(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}\mathbf{\Phi}^T] = \mathbf{\Phi}\frac{d}{d\omega}[(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}]\mathbf{\Phi}^T \\
&= -\mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}\frac{d}{d\omega}[\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I}](\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}\mathbf{\Phi}^T \qquad \text{Teorema 4} \\
&= -\mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}(N\mathbf{I})(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}\mathbf{\Phi}^T \\
&= -N\mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}\mathbf{\Phi}^T \\
&= -N\mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-2}\mathbf{\Phi}^T \qquad \text{(A.4)}
\end{aligned}
$$

Portanto, o gradiente de f, simbolizado por $\nabla f = df/d\omega$, pode ser calculado com o auxílio das Equações A.1, A.2, A.3 e A.4.

A.2.2 Hessiana da Função Objetivo

A derivada segunda de f em relação à ω é obtida da seguinte forma:

$$\frac{d^2 f}{d\omega^2} = 2N\left(\frac{dE_1}{d\omega} - \frac{dE_2}{d\omega}\right) \tag{A.5}$$

Calculando as derivadas de E_1 e E_2 em relação à ω:

$$\begin{aligned}
E_1 &= \frac{\operatorname{tr}\left(\mathbf{M}_2^T \frac{d\mathbf{M}_2}{d\omega}\right)}{\operatorname{tr}(\mathbf{M}_3)^2} \\
\frac{dE_1}{d\omega} &= \frac{\operatorname{tr}(\mathbf{M}_3)^2 \frac{d}{d\omega}\left(\operatorname{tr}\left(\mathbf{M}_2^T \frac{d\mathbf{M}_2}{d\omega}\right)\right) - \operatorname{tr}\left(\mathbf{M}_2^T \frac{d\mathbf{M}_2}{d\omega}\right) \frac{d(\operatorname{tr}(\mathbf{M}_3)^2)}{d\omega}}{\operatorname{tr}(\mathbf{M}_3)^4} \\
&= \frac{\operatorname{tr}(\mathbf{M}_3)^2 \frac{d}{d\omega}\left(\operatorname{tr}\left(\mathbf{M}_2^T \frac{d\mathbf{M}_2}{d\omega}\right)\right) - \operatorname{tr}\left(\mathbf{M}_2^T \frac{d\mathbf{M}_2}{d\omega}\right) \cdot 2\operatorname{tr}(\mathbf{M}_3) \frac{d(\operatorname{tr}(\mathbf{M}_3))}{d\omega}}{\operatorname{tr}(\mathbf{M}_3)^4} \\
&= \frac{\operatorname{tr}(\mathbf{M}_3)^2 \operatorname{tr}\left(\frac{d}{d\omega}\left(\mathbf{M}_2^T \frac{d\mathbf{M}_2}{d\omega}\right)\right) - 2\operatorname{tr}\left(\mathbf{M}_2^T \frac{d\mathbf{M}_2}{d\omega}\right) \operatorname{tr}(\mathbf{M}_3) \operatorname{tr}\left(\frac{d\mathbf{M}_3}{d\omega}\right)}{\operatorname{tr}(\mathbf{M}_3)^4} \\
&= \frac{\operatorname{tr}(\mathbf{M}_3)^2 \operatorname{tr}\left(\mathbf{M}_2^T \frac{d^2\mathbf{M}_2}{d\omega^2} + \left(\frac{d\mathbf{M}_2}{d\omega}\right)^T \frac{d\mathbf{M}_2}{d\omega}\right) - 2\operatorname{tr}\left(\mathbf{M}_2^T \frac{d\mathbf{M}_2}{d\omega}\right) \operatorname{tr}(\mathbf{M}_3) \operatorname{tr}\left(\frac{d\mathbf{M}_3}{d\omega}\right)}{\operatorname{tr}(\mathbf{M}_3)^4}
\end{aligned} \tag{A.6}$$

$$\begin{aligned}
E_2 &= \frac{\operatorname{tr}(\mathbf{M}_2^T \mathbf{M}_2) \operatorname{tr}\left(\frac{d\mathbf{M}_3}{d\omega}\right)}{\operatorname{tr}(\mathbf{M}_3)^3} \\
\frac{dE_2}{d\omega} &= \frac{\operatorname{tr}(\mathbf{M}_3)^3 \frac{d}{d\omega}\left[\operatorname{tr}(\mathbf{M}_2^T \mathbf{M}_2) \operatorname{tr}\left(\frac{d\mathbf{M}_3}{d\omega}\right)\right] - \operatorname{tr}(\mathbf{M}_2^T \mathbf{M}_2) \operatorname{tr}\left(\frac{d\mathbf{M}_3}{d\omega}\right) \frac{d(\operatorname{tr}(\mathbf{M}_3)^3)}{d\omega}}{\operatorname{tr}(\mathbf{M}_3)^6} \\
&= \frac{\operatorname{tr}(\mathbf{M}_3)^3 \left[\operatorname{tr}(\mathbf{M}_2^T \mathbf{M}_2) \operatorname{tr}\left(\frac{d^2\mathbf{M}_3}{d\omega^2}\right) + \operatorname{tr}\left(\mathbf{M}_2^T \frac{d\mathbf{M}_2}{d\omega} + \left(\frac{d\mathbf{M}_2}{d\omega}\right)^T \mathbf{M}_2\right) \operatorname{tr}\left(\frac{d\mathbf{M}_3}{d\omega}\right)\right]}{\operatorname{tr}(\mathbf{M}_3)^6} \\
&\quad - \frac{\operatorname{tr}(\mathbf{M}_2^T \mathbf{M}_2) \operatorname{tr}\left(\frac{d\mathbf{M}_3}{d\omega}\right) \cdot 3\operatorname{tr}(\mathbf{M}_3)^2 \operatorname{tr}\left(\frac{d\mathbf{M}_3}{d\omega}\right)}{\operatorname{tr}(\mathbf{M}_3)^6} \\
&= \frac{\operatorname{tr}(\mathbf{M}_3)^3 \left[\operatorname{tr}(\mathbf{M}_2^T \mathbf{M}_2) \operatorname{tr}\left(\frac{d^2\mathbf{M}_3}{d\omega^2}\right) + 2\operatorname{tr}\left(\mathbf{M}_2^T \frac{d\mathbf{M}_2}{d\omega}\right) \operatorname{tr}\left(\frac{d\mathbf{M}_3}{d\omega}\right)\right]}{\operatorname{tr}(\mathbf{M}_3)^6} \\
&\quad - \frac{3\operatorname{tr}(\mathbf{M}_2^T \mathbf{M}_2) \operatorname{tr}(\mathbf{M}_3)^2 \operatorname{tr}\left(\frac{d\mathbf{M}_3}{d\omega}\right)^2}{\operatorname{tr}(\mathbf{M}_3)^6}
\end{aligned} \tag{A.7}$$

Para determinar o valor das derivadas $dE_1/d\omega$ e $dE_2/d\omega$, é necessário conhecer primeiro o valor das derivadas segundas de $\mathbf{M}_2$ e $\mathbf{M}_3$ em relação à ω. Assim sendo:

$$\frac{d^2\mathbf{M}_2}{d\omega^2} = -\frac{d^2\mathbf{M}_1}{d\omega^2}\mathbf{Y} \qquad \text{(A.8)} \qquad\qquad \frac{d^2\mathbf{M}_3}{d\omega^2} = -\frac{d^2\mathbf{M}_1}{d\omega^2} \qquad \text{(A.9)}$$

Calculando a derivada segunda de $\mathbf{M}_1$ em relação à ω:

$$\begin{aligned}
\frac{d\mathbf{M}_1}{d\omega} &= -N\mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}\mathbf{\Phi}^T \\
\frac{d^2\mathbf{M}_1}{d\omega^2} &= -N\mathbf{\Phi}\frac{d}{d\omega}[(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}]\mathbf{\Phi}^T \\
&= -N\mathbf{\Phi}\left\{(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}\frac{d}{d\omega}[(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}]\right. \\
&\quad \left. + \frac{d}{d\omega}[(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}](\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}\right\}\mathbf{\Phi}^T \\
&= -N\mathbf{\Phi}\left\{-(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-2}\frac{d}{d\omega}[\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I}](\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}\right. \\
&\quad \left. -(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-1}\frac{d}{d\omega}[\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I}](\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-2}\right\}\mathbf{\Phi}^T \quad \text{Teorema 4} \\
&= -N\mathbf{\Phi}\left\{-N(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-3} - N(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-3}\right\}\mathbf{\Phi}^T \\
&= +N\mathbf{\Phi}\left\{+2N(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-3}\right\}\mathbf{\Phi}^T \\
&= 2N^2\mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{\Phi} + N\omega\mathbf{I})^{-3}\mathbf{\Phi}^T \qquad \text{(A.10)}
\end{aligned}$$

Portanto, a Hessiana de f, simbolizada por $H(f) = d^2f/d\omega^2$, pode ser calculada com o auxílio das Equações A.5, A.6, A.7, A.8, A.9 e A.10. Também são necessárias as Equações A.2, A.3 e A.4, deduzidas na Seção A.2.1.

A.3 Otimização com Restrições Fenomenológicas

A.3.1 Gradiente da Função Objetivo

Seja $\mathbf{X} = [x_{ij}]_{n_{lin}\times n_{col}}$ a matriz de parâmetros do modelo ARX, em que $x_{ij} \in \mathbb{R}$, $n_{lin} = N_a n_y + (N_b + 1)n_u$ e $n_{col} = n_y$. A função objetivo $F(\mathbf{X})$ pode ser reescrita da seguinte forma:

$$\begin{aligned}
F(\mathbf{X}) &= \|\mathbf{Y} - \mathbf{\Phi}\mathbf{X}\|_F^2 \\
&= \operatorname{tr}((\mathbf{Y} - \mathbf{\Phi}\mathbf{X})^T(\mathbf{Y} - \mathbf{\Phi}\mathbf{X})) \qquad \text{Teorema 2, item 3} \\
&= \operatorname{tr}((\mathbf{Y}^T - \mathbf{X}^T\mathbf{\Phi}^T)(\mathbf{Y} - \mathbf{\Phi}\mathbf{X})) \\
&= \operatorname{tr}(\mathbf{Y}^T\mathbf{Y} - \mathbf{Y}^T\mathbf{\Phi}\mathbf{X} - \mathbf{X}^T\mathbf{\Phi}^T\mathbf{Y} + \mathbf{X}^T\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{X}) \\
&= \operatorname{tr}(\mathbf{Y}^T\mathbf{Y}) - 2\operatorname{tr}(\mathbf{Y}^T\mathbf{\Phi}\mathbf{X}) + \operatorname{tr}(\mathbf{X}^T\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{X}) \qquad \text{Teorema 1}
\end{aligned}$$

Calculando a derivada parcial de F em relação à x_{ij}:

$$\begin{aligned}
\frac{\partial F}{\partial x_{ij}} &= -2\,\mathrm{tr}\left(\frac{\partial(\mathbf{Y}^T\mathbf{\Phi}\mathbf{X})}{\partial x_{ij}}\right) + \mathrm{tr}\left(\frac{\partial(\mathbf{X}^T\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{X})}{\partial x_{ij}}\right) && \text{Teorema 3}\\
&= -2\,\mathrm{tr}\left(\mathbf{Y}^T\mathbf{\Phi}\frac{\partial\mathbf{X}}{\partial x_{ij}}\right) + \mathrm{tr}\left(\mathbf{X}^T\mathbf{\Phi}^T\frac{\partial(\mathbf{\Phi}\mathbf{X})}{\partial x_{ij}} + \frac{\partial(\mathbf{X}^T\mathbf{\Phi}^T)}{\partial x_{ij}}\mathbf{\Phi}\mathbf{X}\right)\\
&= -2\,\mathrm{tr}\left(\mathbf{Y}^T\mathbf{\Phi}\frac{\partial\mathbf{X}}{\partial x_{ij}}\right) + \mathrm{tr}\left(\mathbf{X}^T\mathbf{\Phi}^T\mathbf{\Phi}\frac{\partial\mathbf{X}}{\partial x_{ij}} + \left(\frac{\partial\mathbf{X}}{\partial x_{ij}}\right)^T\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{X}\right)\\
&= -2\,\mathrm{tr}\left(\left(\frac{\partial\mathbf{X}}{\partial x_{ij}}\right)^T\mathbf{\Phi}^T\mathbf{Y}\right) + 2\,\mathrm{tr}\left(\left(\frac{\partial\mathbf{X}}{\partial x_{ij}}\right)^T\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{X}\right) && \text{Teorema 1}\\
&= -2\,\mathrm{tr}\left(\left(\frac{\partial\mathbf{X}}{\partial x_{ij}}\right)^T\mathbf{\Phi}^T\mathbf{Y} - \left(\frac{\partial\mathbf{X}}{\partial x_{ij}}\right)^T\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{X}\right) && \text{Teorema 1, item 1}\\
&= -2\,\mathrm{tr}\left(\left(\frac{\partial\mathbf{X}}{\partial x_{ij}}\right)^T(\mathbf{\Phi}^T\mathbf{Y} - \mathbf{\Phi}^T\mathbf{\Phi}\mathbf{X})\right)\\
&= -2\sum_{k=1}^{n_{lin}}\sum_{l=1}^{n_{col}}\left(\frac{\partial\mathbf{X}}{\partial x_{ij}}\right)_{kl}(\mathbf{\Phi}^T\mathbf{Y} - \mathbf{\Phi}^T\mathbf{\Phi}\mathbf{X})_{kl} && \text{Teorema 2, item 2}\\
&= -2\sum_{k=1}^{n_{lin}}\sum_{l=1}^{n_{col}}\frac{\partial x_{kl}}{\partial x_{ij}}(\mathbf{\Phi}^T\mathbf{Y} - \mathbf{\Phi}^T\mathbf{\Phi}\mathbf{X})_{kl}\\
&= -2\sum_{k=1}^{n_{lin}}\sum_{l=1}^{n_{col}}\delta^K_{ki}\delta^K_{lj}(\mathbf{\Phi}^T\mathbf{Y} - \mathbf{\Phi}^T\mathbf{\Phi}\mathbf{X})_{kl} && \delta^K = \text{delta de Kronecker}\\
&= -2(\mathbf{\Phi}^T\mathbf{Y} - \mathbf{\Phi}^T\mathbf{\Phi}\mathbf{X})_{ij} && \text{(A.11)}
\end{aligned}$$

Portanto, o gradiente de F é dado por:

$$\begin{aligned}
\nabla\mathbf{F} &= -2(\mathbf{\Phi}^T\mathbf{Y} - \mathbf{\Phi}^T\mathbf{\Phi}\mathbf{X})\\
&= -2\mathbf{\Phi}^T(\mathbf{Y} - \mathbf{\Phi}\mathbf{X}) && \text{(A.12)}
\end{aligned}$$

A.3.2 Hessiana da Função Objetivo

Calculando a derivada segunda de F em relação à x_{ij} e à x_{pq}:

$$\begin{aligned}
\frac{\partial F}{\partial x_{ij}} &= -2(\mathbf{\Phi}^T\mathbf{Y} - \mathbf{\Phi}^T\mathbf{\Phi}\mathbf{X})_{ij} = -2(\mathbf{\Phi}^T\mathbf{Y})_{ij} + 2(\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{X})_{ij}\\
\frac{\partial^2 F}{\partial x_{pq}\partial x_{ij}} &= 2\frac{\partial}{\partial x_{pq}}[(\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{X})_{ij}]\\
&= 2\frac{\partial}{\partial x_{pq}}\left[\sum_{k=1}^{n_{lin}}(\mathbf{\Phi}^T\mathbf{\Phi})_{ik}x_{kj}\right]\\
&= 2\sum_{k=1}^{n_{lin}}(\mathbf{\Phi}^T\mathbf{\Phi})_{ik}\frac{\partial x_{kj}}{\partial x_{pq}}
\end{aligned}$$

$$
\begin{aligned}
&= 2\sum_{k=1}^{n_{lin}} (\mathbf{\Phi}^T\mathbf{\Phi})_{ik}\delta_{kp}^K\delta_{jq}^K \\
&= 2(\mathbf{\Phi}^T\mathbf{\Phi})_{ip}\delta_{jq}^K
\end{aligned} \tag{A.13}
$$

Portanto, a Hessiana de F é dada por:

$$
(\mathbf{H}(F))_{pqij} = \frac{\partial^2 F}{\partial x_{pq}\partial x_{ij}} = \begin{cases} 2(\mathbf{\Phi}^T\mathbf{\Phi})_{ip}, & \text{se } j = q \\ 0, & \text{se } j \neq q \end{cases} \tag{A.14}
$$

A.3.3 Gradiente da Restrição sobre os Polos

Seja $\boldsymbol{\rho} = \begin{bmatrix} \rho_1 & \rho_2 & \cdots & \rho_n & \cdots & \rho_{n_r-1} & \rho_{n_r} \end{bmatrix}^T$ tal que $\rho_n = |r_n|^2 = r_n\overline{r}_n$, em que r_n é o n-ésimo polo do sistema e n_r é o número total de polos. A derivada parcial da restrição $c_{in,0}$ em relação à x_{ij} é obtida da seguinte forma:

$$
\begin{aligned}
c_{in,0} &= S_a - R^2 \\
\frac{\partial c_{in,0}}{\partial x_{ij}} &= \frac{\partial S_a}{\partial x_{ij}} \\
&= \sum_{n=1}^{n_r} \frac{\partial S_a}{\partial \rho_n}\frac{\partial \rho_n}{\partial x_{ij}} && \text{Regra da cadeia} \\
&= \underbrace{\begin{bmatrix} \frac{\partial S_a}{\partial \rho_1} & \cdots & \frac{\partial S_a}{\partial \rho_{n_r}} \end{bmatrix}}_{\left(\frac{\partial S_a}{\partial \boldsymbol{\rho}}\right)^T} \underbrace{\begin{bmatrix} \frac{\partial \rho_1}{\partial x_{ij}} \\ \vdots \\ \frac{\partial \rho_{n_r}}{\partial x_{ij}} \end{bmatrix}}_{\frac{\partial \boldsymbol{\rho}}{\partial x_{ij}}} \\
&= \left(\frac{\partial S_a}{\partial \boldsymbol{\rho}}\right)^T \frac{\partial \boldsymbol{\rho}}{\partial x_{ij}}
\end{aligned} \tag{A.15}
$$

Calculando a derivada parcial de S_a em relação à ρ_n:

$$
\begin{aligned}
S_a &= \sum_{i=1}^{n_r} w_i\rho_i \\
\frac{\partial S_a}{\partial \rho_n} &= \sum_{i=1}^{n_r} \frac{\partial(w_i\rho_i)}{\partial \rho_n} = \sum_{i=1}^{n_r} \left(w_i \underbrace{\frac{\partial \rho_i}{\partial \rho_n}}_{\delta_{in}^K} + \rho_i \frac{\partial w_i}{\partial \rho_n} \right) \\
&= \sum_{i=1}^{n_r} \left(w_i\delta_{in}^K + \rho_i\frac{\partial w_i}{\partial \rho_n} \right) = \sum_{i=1}^{n_r} w_i\delta_{in}^K + \sum_{i=1}^{n_r} \rho_i\frac{\partial w_i}{\partial \rho_n}
\end{aligned}
$$

$$= w_n + \sum_{i=1}^{n_r} \rho_i \frac{\partial w_i}{\partial \rho_n} \tag{A.16}$$

Calculando a derivada parcial de w_i em relação à ρ_n:

$$\begin{aligned}
w_i &= \frac{e^{a\rho_i}}{\sum_{j=1}^{n_r} e^{a\rho_j}} \\
\frac{\partial w_i}{\partial \rho_n} &= \frac{\sum_{j=1}^{n_r} e^{a\rho_j} \frac{\partial (e^{a\rho_i})}{\partial \rho_n} - e^{a\rho_i} \sum_{j=1}^{n_r} \frac{\partial (e^{a\rho_j})}{\partial \rho_n}}{\left(\sum_{j=1}^{n_r} e^{a\rho_j}\right)^2} \\
&= \frac{\sum_{j=1}^{n_r} e^{a\rho_j} a \frac{\partial \rho_i}{\partial \rho_n} e^{a\rho_i} - e^{a\rho_i} a \sum_{j=1}^{n_r} \frac{\partial \rho_j}{\partial \rho_n} e^{a\rho_j}}{\left(\sum_{j=1}^{n_r} e^{a\rho_j}\right)^2} \\
&= \frac{\sum_{j=1}^{n_r} e^{a\rho_j} a \delta_{in}^K e^{a\rho_i} - e^{a\rho_i} a \sum_{j=1}^{n_r} \delta_{jn}^K e^{a\rho_j}}{\left(\sum_{j=1}^{n_r} e^{a\rho_j}\right)^2} \\
&= \frac{\sum_{j=1}^{n_r} e^{a\rho_j} a \delta_{in}^K e^{a\rho_i} - e^{a\rho_i} a e^{a\rho_n}}{\left(\sum_{j=1}^{n_r} e^{a\rho_j}\right)^2} \\
&= \frac{\cancel{\sum_{j=1}^{n_r} e^{a\rho_j}} a \delta_{in}^K e^{a\rho_i}}{\left(\sum_{j=1}^{n_r} e^{a\rho_j}\right)^{\cancel{2}}} - \frac{a e^{a\rho_i} e^{a\rho_n}}{\left(\sum_{j=1}^{n_r} e^{a\rho_j}\right)^2} \\
&= a\delta_{in}^K \underbrace{\left(\frac{e^{a\rho_i}}{\sum_{j=1}^{n_r} e^{a\rho_j}}\right)}_{w_i} - a \underbrace{\left(\frac{e^{a\rho_i}}{\sum_{j=1}^{n_r} e^{a\rho_j}}\right)}_{w_i} \underbrace{\left(\frac{e^{a\rho_n}}{\sum_{j=1}^{n_r} e^{a\rho_j}}\right)}_{w_n} \\
&= a\delta_{in}^K w_i - a w_i w_n \\
&= a w_i (\delta_{in}^K - w_n)
\end{aligned} \tag{A.17}$$

Substituindo a Equação A.17 na Equação A.16:

$$\begin{aligned}
\frac{\partial S_a}{\partial \rho_n} &= w_n + \sum_{i=1}^{n_r} \rho_i \frac{\partial w_i}{\partial \rho_n} \\
&= w_n + \sum_{i=1}^{n_r} \rho_i a w_i (\delta_{in}^K - w_n) \\
&= w_n + \sum_{i=1}^{n_r} (\rho_i a w_i \delta_{in}^K - \rho_i a w_i w_n) \\
&= w_n + a \sum_{i=1}^{n_r} \delta_{in}^K w_i \rho_i - a w_n \underbrace{\sum_{i=1}^{n_r} w_i \rho_i}_{S_a} \\
&= w_n + a w_n \rho_n - a w_n S_a
\end{aligned}$$

$$
\begin{aligned}
&= w_n(1 + a\rho_n - aS_a) \\
&= w_n[1 + a(\rho_n - S_a)] \qquad \text{(A.18)}
\end{aligned}
$$

Calculando a derivada parcial de ρ_n em relação à x_{ij}:

$$
\begin{aligned}
\rho_n &= r_n \overline{r}_n \\
\frac{\partial \rho_n}{\partial x_{ij}} &= \frac{\partial (r_n \overline{r}_n)}{\partial x_{ij}} \\
&= r_n \frac{\partial \overline{r}_n}{\partial x_{ij}} + \overline{r}_n \frac{\partial r_n}{\partial x_{ij}} \\
&= r_n \overline{\frac{\partial r_n}{\partial x_{ij}}} + \overline{r}_n \frac{\partial r_n}{\partial x_{ij}} && \text{Teorema 6} \\
&= r_n \overline{\frac{\partial r_n}{\partial x_{ij}}} + \frac{\partial r_n}{\partial x_{ij}} \overline{r}_n && \text{(A.19)}
\end{aligned}
$$

Se r_n é um dos polos do sistema, então:

$$
\det(\boldsymbol{\mathcal{A}}_{r_n}) = 0 \qquad \text{(A.20)}
$$

em que $\boldsymbol{\mathcal{A}}_{r_n} = \boldsymbol{\mathcal{A}}(r_n^{-1})$. Derivando em relação à x_{ij} os dois lados da Equação A.20 :

$$
\begin{aligned}
\frac{\partial(\det(\boldsymbol{\mathcal{A}}_{r_n}))}{\partial x_{ij}} &= 0 \\
\operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}) \frac{\partial \boldsymbol{\mathcal{A}}_{r_n}}{\partial x_{ij}}\right) &= 0 \quad \text{Teorema 5} \\
\operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}) \frac{\partial}{\partial x_{ij}} \left(\mathbf{I} - \sum_{k=1}^{N_a} \mathbf{A}_k r_n^{-k}\right)\right) &= 0 \\
\operatorname{tr}\left(-\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}) \sum_{k=1}^{N_a} \frac{\partial(\mathbf{A}_k r_n^{-k})}{\partial x_{ij}}\right) &= 0 \\
\operatorname{tr}\left(-\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}) \sum_{k=1}^{N_a} \left(\mathbf{A}_k \frac{\partial(r_n^{-k})}{\partial x_{ij}} + r_n^{-k} \frac{\partial \mathbf{A}_k}{\partial x_{ij}}\right)\right) &= 0 \\
\operatorname{tr}\left(-\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}) \sum_{k=1}^{N_a} \left(\mathbf{A}_k (-k) r_n^{-k-1} \frac{\partial r_n}{\partial x_{ij}} + r_n^{-k} \frac{\partial \mathbf{A}_k}{\partial x_{ij}}\right)\right) &= 0 \\
\operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}) \sum_{k=1}^{N_a} \left(\mathbf{A}_k k r_n^{-k-1} \frac{\partial r_n}{\partial x_{ij}} - r_n^{-k} \frac{\partial \mathbf{A}_k}{\partial x_{ij}}\right)\right) &= 0 \\
\operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}) \left(\frac{\partial r_n}{\partial x_{ij}} \sum_{k=1}^{N_a} \mathbf{A}_k k r_n^{-k-1} - \sum_{k=1}^{N_a} r_n^{-k} \frac{\partial \mathbf{A}_k}{\partial x_{ij}}\right)\right) &= 0 \\
\operatorname{tr}\left(\frac{\partial r_n}{\partial x_{ij}} \operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}) \sum_{k=1}^{N_a} \mathbf{A}_k k r_n^{-k-1} - \operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}) \sum_{k=1}^{N_a} r_n^{-k} \frac{\partial \mathbf{A}_k}{\partial x_{ij}}\right) &= 0
\end{aligned}
$$

$$\frac{\partial r_n}{\partial x_{ij}} \operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}) \sum_{k=1}^{N_a} \mathbf{A}_k k r_n^{-k-1}\right) - \operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}) \sum_{k=1}^{N_a} r_n^{-k} \frac{\partial \mathbf{A}_k}{\partial x_{ij}}\right) = 0 \quad \text{Teorema 1}$$

$$\frac{\partial r_n}{\partial x_{ij}} = \frac{\operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}) \sum_{k=1}^{N_a} \left(\frac{\partial \mathbf{A}_k^T}{\partial x_{ij}}\right)^T r_n^{-k}\right)}{\operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}) \sum_{k=1}^{N_a} k \mathbf{A}_k r_n^{-(k+1)}\right)} \tag{A.21}$$

Sabe-se que a matriz $\mathbf{X}$ tem a seguinte forma:

$$\mathbf{X} = \begin{bmatrix} \mathbf{A}_1 & \cdots & \mathbf{A}_{N_a} & \mathbf{B}_0 & \cdots & \mathbf{B}_{N_b} \end{bmatrix}^T = \begin{bmatrix} \mathbf{A}_1^T \\ \vdots \\ \mathbf{A}_{N_a}^T \\ \mathbf{B}_0^T \\ \vdots \\ \mathbf{B}_{N_b}^T \end{bmatrix} \tag{A.22}$$

em que as submatrizes $\mathbf{A}_k^T$ são do tipo $n_y \times n_y$ e as $\mathbf{B}_k^T$ do tipo $n_u \times n_y$. Logo:

$$\mathbf{A}_k^T = \begin{bmatrix} x_{(k-1)n_y+1,1} & \cdots & x_{(k-1)n_y+1,n_y} \\ \vdots & \ddots & \vdots \\ x_{kn_y,1} & \cdots & x_{kn_y,n_y} \end{bmatrix}, \qquad k = 1, \cdots, N_a \tag{A.23}$$

$$\mathbf{B}_k^T = \begin{bmatrix} x_{N_a n_y+kn_u+1,1} & \cdots & x_{N_a n_y+kn_u+1,n_y} \\ \vdots & \ddots & \vdots \\ x_{N_a n_y+(k+1)n_u,1} & \cdots & x_{N_a n_y+(k+1)n_u,n_y} \end{bmatrix}, \qquad k = 0, \cdots, N_b \tag{A.24}$$

Analisando a Equação A.23, verifica-se que $\dfrac{\partial \mathbf{A}_k^T}{\partial x_{ij}} = \mathbf{0}$ se x_{ij} não for um elemento de $\mathbf{A}_k^T$. Desse modo, a Equação A.21 pode ser simplificada:

$$\frac{\partial r_n}{\partial x_{ij}} = \begin{cases} \dfrac{\operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}) \left(\dfrac{\partial \mathbf{A}_{k_0}^T}{\partial x_{ij}}\right)^T r_n^{-k_0}\right)}{\operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}) \sum_{k=1}^{N_a} k \mathbf{A}_k r_n^{-(k+1)}\right)}, & \text{se } i \leqslant N_a n_y \\ 0, & \text{se } i > N_a n_y \end{cases} \tag{A.25}$$

em que $k_0 = \left\lceil \dfrac{i}{n_y} \right\rceil$ é o índice da submatriz $\mathbf{A}_k^T$ que contém x_{ij}. Além disso:

$$\operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}) \left(\frac{\partial \mathbf{A}_{k_0}^T}{\partial x_{ij}}\right)^T r_n^{-k_0}\right) = r_n^{-k_0} \operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}) \left(\frac{\partial \mathbf{A}_{k_0}^T}{\partial x_{ij}}\right)^T\right)$$

$$
\begin{aligned}
&= r_n^{-k_0} \operatorname{tr}\left(\left(\frac{\partial \mathbf{A}_{k_0}^T}{\partial x_{ij}}\right)^T \operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\right) \\
&= r_n^{-k_0} \sum_{u=1}^{n_y}\sum_{v=1}^{n_y}\left(\frac{\partial \mathbf{A}_{k_0}^T}{\partial x_{ij}}\right)_{u,v} (\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{u,v} \\
&= r_n^{-k_0} \sum_{u=1}^{n_y}\sum_{v=1}^{n_y}\delta^K_{u,i-(k_0-1)n_y}\delta^K_{v,j}(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{u,v} \\
&= r_n^{-k_0}(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{i-(k_0-1)n_y,j} \qquad \text{(A.26)}
\end{aligned}
$$

Substituindo a Equação A.26 na Equação A.25, chega-se à expressão final para a derivada parcial de r_n em relação à x_{ij}:

$$
\frac{\partial r_n}{\partial x_{ij}} = \begin{cases} \dfrac{r_n^{-k_0}(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{i-(k_0-1)n_y,j}}{\operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k\mathbf{A}_k r_n^{-(k+1)}\right)}, & \text{se } i \leqslant N_a n_y \\ 0, & \text{se } i > N_a n_y \end{cases} \qquad \text{(A.27)}
$$

Portanto, os componentes do gradiente de $c_{in,0}$, simbolizados por $(\nabla \mathbf{c_{in,0}})_{ij} = \dfrac{\partial c_{in,0}}{\partial x_{ij}}$, podem ser calculados com o auxílio das Equações A.15, A.18, A.19 e A.27.

A.3.4 Hessiana da Restrição sobre os Polos

A derivada segunda de $c_{in,0}$ em relação à x_{ij} e à x_{pq} é obtida da seguinte forma:

$$
\begin{aligned}
\frac{\partial c_{in,0}}{\partial x_{ij}} &= \sum_{n=1}^{n_r}\frac{\partial S_a}{\partial \rho_n}\frac{\partial \rho_n}{\partial x_{ij}} \\
\frac{\partial^2 c_{in,0}}{\partial x_{pq}\partial x_{ij}} &= \sum_{n=1}^{n_r}\frac{\partial}{\partial x_{pq}}\left(\frac{\partial S_a}{\partial \rho_n}\frac{\partial \rho_n}{\partial x_{ij}}\right) \\
&= \sum_{n=1}^{n_r}\left(\frac{\partial S_a}{\partial \rho_n}\frac{\partial^2 \rho_n}{\partial x_{pq}\partial x_{ij}} + \frac{\partial \rho_n}{\partial x_{ij}}\frac{\partial}{\partial x_{pq}}\left(\frac{\partial S_a}{\partial \rho_n}\right)\right) \\
&= \sum_{n=1}^{n_r}\frac{\partial S_a}{\partial \rho_n}\frac{\partial^2 \rho_n}{\partial x_{pq}\partial x_{ij}} + \sum_{n=1}^{n_r}\frac{\partial \rho_n}{\partial x_{ij}}\frac{\partial}{\partial x_{pq}}\left(\frac{\partial S_a}{\partial \rho_n}\right) \\
&= \sum_{n=1}^{n_r}\frac{\partial S_a}{\partial \rho_n}\frac{\partial^2 \rho_n}{\partial x_{pq}\partial x_{ij}} + \sum_{n=1}^{n_r}\frac{\partial \rho_n}{\partial x_{ij}}\sum_{m=1}^{n_r}\frac{\partial}{\partial \rho_m}\left(\frac{\partial S_a}{\partial \rho_n}\right)\frac{\partial \rho_m}{\partial x_{pq}} \\
&= \sum_{n=1}^{n_r}\frac{\partial S_a}{\partial \rho_n}\frac{\partial^2 \rho_n}{\partial x_{pq}\partial x_{ij}} + \sum_{n=1}^{n_r}\sum_{m=1}^{n_r}\frac{\partial \rho_n}{\partial x_{ij}}\frac{\partial^2 S_a}{\partial \rho_m \partial \rho_n}\frac{\partial \rho_m}{\partial x_{pq}} \\
&= \sum_{n=1}^{n_r}\frac{\partial S_a}{\partial \rho_n}\frac{\partial^2 \rho_n}{\partial x_{pq}\partial x_{ij}} + \sum_{m=1}^{n_r}\sum_{n=1}^{n_r}\frac{\partial \rho_m}{\partial x_{pq}}\frac{\partial^2 S_a}{\partial \rho_m \partial \rho_n}\frac{\partial \rho_n}{\partial x_{ij}}
\end{aligned}
$$

$$
\begin{aligned}
&= \underbrace{\begin{bmatrix} \dfrac{\partial S_a}{\partial \rho_1} & \cdots & \dfrac{\partial S_a}{\partial \rho_{n_r}} \end{bmatrix}}_{\left(\frac{\partial S_a}{\partial \boldsymbol{\rho}}\right)^T} \underbrace{\begin{bmatrix} \dfrac{\partial^2 \rho_1}{\partial x_{pq} \partial x_{ij}} \\ \vdots \\ \dfrac{\partial^2 \rho_{n_r}}{\partial x_{pq} \partial x_{ij}} \end{bmatrix}}_{\frac{\partial^2 \boldsymbol{\rho}}{\partial x_{pq} \partial x_{ij}}} \\
&\quad + \underbrace{\begin{bmatrix} \dfrac{\partial \rho_1}{\partial x_{pq}} & \cdots & \dfrac{\partial \rho_{n_r}}{\partial x_{pq}} \end{bmatrix}}_{\left(\frac{\partial \boldsymbol{\rho}}{\partial x_{pq}}\right)^T} \underbrace{\begin{bmatrix} \dfrac{\partial^2 S_a}{\partial \rho_1^2} & \cdots & \dfrac{\partial^2 S_a}{\partial \rho_1 \partial \rho_{n_r}} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial^2 S_a}{\partial \rho_{n_r} \partial \rho_1} & \cdots & \dfrac{\partial^2 S_a}{\partial \rho_{n_r}^2} \end{bmatrix}}_{\frac{\partial^2 S_a}{\partial \boldsymbol{\rho}^2}} \underbrace{\begin{bmatrix} \dfrac{\partial \rho_1}{\partial x_{ij}} \\ \vdots \\ \dfrac{\partial \rho_{n_r}}{\partial x_{ij}} \end{bmatrix}}_{\frac{\partial \boldsymbol{\rho}}{\partial x_{ij}}} \\
&= \left(\frac{\partial S_a}{\partial \boldsymbol{\rho}}\right)^T \frac{\partial^2 \boldsymbol{\rho}}{\partial x_{pq} \partial x_{ij}} + \left(\frac{\partial \boldsymbol{\rho}}{\partial x_{pq}}\right)^T \frac{\partial^2 S_a}{\partial \boldsymbol{\rho}^2} \frac{\partial \boldsymbol{\rho}}{\partial x_{ij}} \qquad \text{(A.28)}
\end{aligned}
$$

Calculando a derivada segunda de S_a em relação à ρ_n e à ρ_m:

$$
\begin{aligned}
\frac{\partial S_a}{\partial \rho_n} &= w_n[1 + a(\rho_n - S_a)] \\
\frac{\partial^2 S_a}{\partial \rho_m \partial \rho_n} &= w_n \frac{\partial}{\partial \rho_m}[1 + a(\rho_n - S_a)] + [1 + a(\rho_n - S_a)] \frac{\partial w_n}{\partial \rho_m} \\
&= a w_n \left[\frac{\partial \rho_n}{\partial \rho_m} - \frac{\partial S_a}{\partial \rho_m}\right] + [1 + a(\rho_n - S_a)] \frac{\partial w_n}{\partial \rho_m} \\
&= a w_n \left\{\delta_{nm}^K - w_m[1 + a(\rho_m - S_a)]\right\} + [1 + a(\rho_n - S_a)] a w_n (\delta_{nm}^K - w_m) \\
&= a w_n \left\{\delta_{nm}^K - w_m[1 + a(\rho_m - S_a)] + (\delta_{nm}^K - w_m)[1 + a(\rho_n - S_a)]\right\} \\
&= a w_n \left\{\delta_{nm}^K [2 + a(\rho_n - S_a)] - w_m[2 + a(\rho_m - S_a) + a(\rho_n - S_a)]\right\} \\
&= a w_n \left\{\delta_{mn}^K [2 + a(\rho_n - S_a)] - w_m[2 + a(\rho_m + \rho_n - 2S_a)]\right\} \qquad \text{(A.29)}
\end{aligned}
$$

Calculando a derivada segunda de ρ_n em relação à x_{ij} e à x_{pq}:

$$
\begin{aligned}
\frac{\partial \rho_n}{\partial x_{ij}} &= r_n \overline{\frac{\partial r_n}{\partial x_{ij}}} + \frac{\partial r_n}{\partial x_{ij}} \overline{r}_n \\
\frac{\partial^2 \rho_n}{\partial x_{pq} \partial x_{ij}} &= \frac{\partial}{\partial x_{pq}} \left(r_n \overline{\frac{\partial r_n}{\partial x_{ij}}}\right) + \frac{\partial}{\partial x_{pq}} \left(\frac{\partial r_n}{\partial x_{ij}} \overline{r}_n\right) \\
&= r_n \frac{\partial}{\partial x_{pq}} \left(\overline{\frac{\partial r_n}{\partial x_{ij}}}\right) + \overline{\frac{\partial r_n}{\partial x_{ij}}} \frac{\partial r_n}{\partial x_{pq}} + \frac{\partial r_n}{\partial x_{ij}} \frac{\partial \overline{r}_n}{\partial x_{pq}} + \overline{r}_n \frac{\partial^2 r_n}{\partial x_{pq} \partial x_{ij}} \\
&= r_n \overline{\frac{\partial}{\partial x_{pq}} \left(\frac{\partial r_n}{\partial x_{ij}}\right)} + \overline{\frac{\partial r_n}{\partial x_{ij}}} \frac{\partial r_n}{\partial x_{pq}} + \frac{\partial r_n}{\partial x_{ij}} \overline{\frac{\partial r_n}{\partial x_{pq}}} + \overline{r}_n \frac{\partial^2 r_n}{\partial x_{pq} \partial x_{ij}} \qquad \text{Teorema 6} \\
&= r_n \overline{\frac{\partial^2 r_n}{\partial x_{pq} \partial x_{ij}}} + \frac{\partial r_n}{\partial x_{pq}} \overline{\frac{\partial r_n}{\partial x_{ij}}} + \frac{\partial r_n}{\partial x_{ij}} \overline{\frac{\partial r_n}{\partial x_{pq}}} + \frac{\partial^2 r_n}{\partial x_{pq} \partial x_{ij}} \overline{r}_n \qquad \text{(A.30)}
\end{aligned}
$$

Calculando a derivada segunda de r_n em relação à x_{ij} e à x_{pq}:

$$\frac{\partial r_n}{\partial x_{ij}} = \begin{cases} \dfrac{r_n^{-k_0}(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{i-(k_0-1)n_y,j}}{\mathrm{tr}\left(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k\mathbf{A}_k r_n^{-(k+1)}\right)}, & \text{se } i \leqslant N_a n_y \\ 0, & \text{se } i > N_a n_y \end{cases}$$

$$\frac{\partial^2 r_n}{\partial x_{pq}\partial x_{ij}} = \begin{cases} \dfrac{\partial}{\partial x_{pq}}\left[\dfrac{r_n^{-k_0}(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{i-(k_0-1)n_y,j}}{\mathrm{tr}\left(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k\mathbf{A}_k r_n^{-(k+1)}\right)}\right] = \psi, & \text{se } i \leqslant N_a n_y \\ 0, & \text{se } i > N_a n_y \end{cases} \tag{A.31}$$

Desenvolvendo a expressão para ψ na Equação A.31:

$$\begin{aligned} \psi &= \frac{\partial}{\partial x_{pq}}\left[\frac{r_n^{-k_0}(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{i-(k_0-1)n_y,j}}{\mathrm{tr}\left(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k\mathbf{A}_k r_n^{-(k+1)}\right)}\right] \\ &= \frac{\mathrm{tr}\left(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k\mathbf{A}_k r_n^{-(k+1)}\right)\dfrac{\partial}{\partial x_{pq}}\left[r_n^{-k_0}(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{i-(k_0-1)n_y,j}\right]}{\mathrm{tr}\left(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k\mathbf{A}_k r_n^{-(k+1)}\right)^2} \\ &\quad - \frac{r_n^{-k_0}(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{i-(k_0-1)n_y,j}\dfrac{\partial}{\partial x_{pq}}\left[\mathrm{tr}\left(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k\mathbf{A}_k r_n^{-(k+1)}\right)\right]}{\mathrm{tr}\left(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k\mathbf{A}_k r_n^{-(k+1)}\right)^2} \\ &= \frac{f_1 f_2 - f_3 f_4}{f_1^2} \end{aligned} \tag{A.32}$$

em que as variáveis f_1, f_2, f_3 e f_4 são dadas por:

$$f_1 = \mathrm{tr}\left(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k\mathbf{A}_k r_n^{-(k+1)}\right) \tag{A.33}$$

$$\begin{aligned} f_2 &= \frac{\partial}{\partial x_{pq}}\left[r_n^{-k_0}(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{i-(k_0-1)n_y,j}\right] \\ &= r_n^{-k_0}\frac{\partial}{\partial x_{pq}}\left[(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{i-(k_0-1)n_y,j}\right] + (\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{i-(k_0-1)n_y,j}\frac{\partial}{\partial x_{pq}}\left[r_n^{-k_0}\right] \\ &= r_n^{-k_0}\left(\frac{\partial(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}))}{\partial x_{pq}}\right)_{i-(k_0-1)n_y,j} + (\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{i-(k_0-1)n_y,j}(-k_0)r_n^{-k_0-1}\frac{\partial r_n}{\partial x_{pq}} \\ &= r_n^{-k_0}\left(\frac{\partial(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}))}{\partial x_{pq}}\right)_{i-(k_0-1)n_y,j} - k_0 r_n^{-(k_0+1)}\frac{\partial r_n}{\partial x_{pq}}(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{i-(k_0-1)n_y,j} \end{aligned} \tag{A.34}$$

$$f_3 = r_n^{-k_0}(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{i-(k_0-1)n_y,j} \tag{A.35}$$

$$f_4 = \frac{\partial}{\partial x_{pq}}\left[\mathrm{tr}\left(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k\mathbf{A}_k r_n^{-(k+1)}\right)\right]$$

$$
\begin{aligned}
&= \operatorname{tr}\left(\frac{\partial}{\partial x_{pq}}\left[\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k\mathbf{A}_k r_n^{-(k+1)}\right]\right) && \text{Teorema 3}\\
&= \operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k\frac{\partial(\mathbf{A}_k r_n^{-(k+1)})}{\partial x_{pq}} + \frac{\partial\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\right)}{\partial x_{pq}}\sum_{k=1}^{N_a} k\mathbf{A}_k r_n^{-(k+1)}\right)\\
&= \operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k\left(\mathbf{A}_k\frac{\partial(r_n^{-(k+1)})}{\partial x_{pq}} + \frac{\partial\mathbf{A}_k}{\partial x_{pq}}r_n^{-(k+1)}\right)\right.\\
&\quad \left. + \frac{\partial\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\right)}{\partial x_{pq}}\sum_{k=1}^{N_a} k\mathbf{A}_k r_n^{-(k+1)}\right)\\
&= \operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a}\left(k\frac{\partial\mathbf{A}_k}{\partial x_{pq}}r_n^{-(k+1)} - k(k+1)\mathbf{A}_k r_n^{-(k+2)}\frac{\partial r_n}{\partial x_{pq}}\right)\right.\\
&\quad \left. + \frac{\partial\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\right)}{\partial x_{pq}}\sum_{k=1}^{N_a} k\mathbf{A}_k r_n^{-(k+1)}\right)\\
&= \operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k\frac{\partial\mathbf{A}_k}{\partial x_{pq}}r_n^{-(k+1)} - \frac{\partial r_n}{\partial x_{pq}}\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k(k+1)\mathbf{A}_k r_n^{-(k+2)}\right.\\
&\quad \left. + \frac{\partial\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\right)}{\partial x_{pq}}\sum_{k=1}^{N_a} k\mathbf{A}_k r_n^{-(k+1)}\right)\\
&= \operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k\left(\frac{\partial\mathbf{A}_k^T}{\partial x_{pq}}\right)^T r_n^{-(k+1)}\right)\\
&\quad - \frac{\partial r_n}{\partial x_{pq}}\operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k(k+1)\mathbf{A}_k r_n^{-(k+2)}\right)\\
&\quad + \operatorname{tr}\left(\frac{\partial\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\right)}{\partial x_{pq}}\sum_{k=1}^{N_a} k\mathbf{A}_k r_n^{-(k+1)}\right) && \text{(A.36)}
\end{aligned}
$$

As Equações A.34 e A.36 podem ser simplificadas considerando-se que:

$$
\frac{\partial r_n}{\partial x_{pq}} = \begin{cases} \dfrac{r_n^{-k_1}(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{p-(k_1-1)n_y,q}}{\operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k\mathbf{A}_k r_n^{-(k+1)}\right)}, & \text{se } p \leqslant N_a n_y \\ 0, & \text{se } p > N_a n_y \end{cases} \qquad \text{(A.37)}
$$

em que $k_1 = \left\lceil \dfrac{p}{n_y} \right\rceil$ é o índice da submatriz $\mathbf{A}_k^T$ que contém o elemento x_{pq}. Também pode ser utilizada a seguinte simplificação na Equação A.36:

$$
\sum_{k=1}^{N_a} k\left(\frac{\partial\mathbf{A}_k^T}{\partial x_{pq}}\right)^T r_n^{-(k+1)} = \begin{cases} k_1\left(\dfrac{\partial\mathbf{A}_{k_1}^T}{\partial x_{pq}}\right)^T r_n^{-(k_1+1)}, & \text{se } p \leqslant N_a n_y \\ \mathbf{0}, & \text{se } p > N_a n_y \end{cases} \qquad \text{(A.38)}
$$

Consequentemente, quando $p \leqslant N_a n_y$:

$$\begin{aligned}
\operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\sum_{k=1}^{N_a} k\left(\frac{\partial \mathbf{A}_k^T}{\partial x_{pq}}\right)^T r_n^{-(k+1)}\right) &= \operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})k_1\left(\frac{\partial \mathbf{A}_{k_1}^T}{\partial x_{pq}}\right)^T r_n^{-(k_1+1)}\right) \\
&= k_1 r_n^{-(k_1+1)}\operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\left(\frac{\partial \mathbf{A}_{k_1}^T}{\partial x_{pq}}\right)^T\right) \\
&= k_1 r_n^{-(k_1+1)}\operatorname{tr}\left(\left(\frac{\partial \mathbf{A}_{k_1}^T}{\partial x_{pq}}\right)^T \operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})\right) \\
&= k_1 r_n^{-(k_1+1)}\sum_{u=1}^{n_y}\sum_{v=1}^{n_y}\left(\frac{\partial \mathbf{A}_{k_1}^T}{\partial x_{pq}}\right)_{u,v}(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{u,v} \\
&= k_1 r_n^{-(k_1+1)}\sum_{u=1}^{n_y}\sum_{v=1}^{n_y}\delta^K_{u,p-(k_1-1)n_y}\delta^K_{v,q}(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{u,v} \\
&= k_1 r_n^{-(k_1+1)}(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{p-(k_1-1)n_y,q} \qquad \text{(A.39)}
\end{aligned}$$

Para determinar o valor das variáveis f_2 e f_4, é necessário conhecer primeiro o valor da derivada parcial da adjunta de $\boldsymbol{\mathcal{A}}_{r_n}$ em relação à x_{pq}. Por definição, o elemento (u,v) de $\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n})$ é dado por:

$$(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}))_{uv} = \begin{cases} 1, & \text{se } n_y = 1 \\ C_{vu}, & \text{se } n_y > 1 \end{cases}, \qquad u, v = 1, \cdots, n_y \qquad \text{(A.40)}$$

em que C_{vu} é o cofator do elemento (v,u) de $\boldsymbol{\mathcal{A}}_{r_n}$. Derivando em relação à x_{pq} os dois lados da Equação A.40:

$$\left(\frac{\partial(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}))}{\partial x_{pq}}\right)_{uv} = \begin{cases} 0, & \text{se } n_y = 1 \\ \dfrac{\partial C_{vu}}{\partial x_{pq}}, & \text{se } n_y > 1 \end{cases} \qquad \text{(A.41)}$$

Sabe-se que $C_{vu} = (-1)^{v+u}\det(\boldsymbol{\mathcal{A}}_{r_n}^{vu})$, onde $\boldsymbol{\mathcal{A}}_{r_n}^{vu}$ é a matriz obtida após a remoção da v-ésima linha e da u-ésima coluna de $\boldsymbol{\mathcal{A}}_{r_n}$. Calculando a derivada parcial de C_{vu} em relação à x_{pq}:

$$\begin{aligned}
C_{vu} &= (-1)^{v+u}\det(\boldsymbol{\mathcal{A}}_{r_n}^{vu}) \\
\frac{\partial C_{vu}}{\partial x_{pq}} &= (-1)^{v+u}\frac{\partial(\det(\boldsymbol{\mathcal{A}}_{r_n}^{vu}))}{\partial x_{pq}} \\
&= (-1)^{v+u}\operatorname{tr}\left(\operatorname{adj}(\boldsymbol{\mathcal{A}}_{r_n}^{vu})\frac{\partial \boldsymbol{\mathcal{A}}_{r_n}^{vu}}{\partial x_{pq}}\right) \qquad \text{Teorema 5}
\end{aligned}$$

$$
\begin{aligned}
&= (-1)^{v+u}\,\mathrm{tr}\left(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}^{vu})\frac{\partial}{\partial x_{pq}}\left(\mathbf{I}-\sum_{k=1}^{N_a}\mathbf{A}_k r_n^{-k}\right)^{vu}\right)\\
&= (-1)^{v+u}\,\mathrm{tr}\left(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}^{vu})\frac{\partial}{\partial x_{pq}}\left(\mathbf{I}^{vu}-\sum_{k=1}^{N_a}\mathbf{A}_k^{vu} r_n^{-k}\right)\right)\\
&= (-1)^{v+u}\,\mathrm{tr}\left(-\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}^{vu})\sum_{k=1}^{N_a}\frac{\partial(\mathbf{A}_k^{vu} r_n^{-k})}{\partial x_{pq}}\right)\\
&= (-1)^{v+u}\,\mathrm{tr}\left(-\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}^{vu})\sum_{k=1}^{N_a}\left(\mathbf{A}_k^{vu}\frac{\partial(r_n^{-k})}{\partial x_{pq}}+r_n^{-k}\frac{\partial\mathbf{A}_k^{vu}}{\partial x_{pq}}\right)\right)\\
&= (-1)^{v+u}\,\mathrm{tr}\left(-\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}^{vu})\sum_{k=1}^{N_a}\left(\mathbf{A}_k^{vu}(-k)r_n^{-k-1}\frac{\partial r_n}{\partial x_{pq}}+r_n^{-k}\frac{\partial\mathbf{A}_k^{vu}}{\partial x_{pq}}\right)\right)\\
&= (-1)^{v+u}\,\mathrm{tr}\left(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}^{vu})\sum_{k=1}^{N_a}\left(\mathbf{A}_k^{vu}kr_n^{-k-1}\frac{\partial r_n}{\partial x_{pq}}-r_n^{-k}\frac{\partial\mathbf{A}_k^{vu}}{\partial x_{pq}}\right)\right)\\
&= (-1)^{v+u}\,\mathrm{tr}\left(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}^{vu})\left(\frac{\partial r_n}{\partial x_{pq}}\sum_{k=1}^{N_a}\mathbf{A}_k^{vu}kr_n^{-k-1}-\sum_{k=1}^{N_a}r_n^{-k}\frac{\partial\mathbf{A}_k^{vu}}{\partial x_{pq}}\right)\right)\\
&= (-1)^{v+u}\,\mathrm{tr}\left(\frac{\partial r_n}{\partial x_{pq}}\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}^{vu})\sum_{k=1}^{N_a}\mathbf{A}_k^{vu}kr_n^{-k-1}-\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}^{vu})\sum_{k=1}^{N_a}r_n^{-k}\frac{\partial\mathbf{A}_k^{vu}}{\partial x_{pq}}\right)\\
&= (-1)^{v+u}\left[\frac{\partial r_n}{\partial x_{pq}}\,\mathrm{tr}\left(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}^{vu})\sum_{k=1}^{N_a}\mathbf{A}_k^{vu}kr_n^{-k-1}\right)-\mathrm{tr}\left(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}^{vu})\sum_{k=1}^{N_a}r_n^{-k}\frac{\partial\mathbf{A}_k^{vu}}{\partial x_{pq}}\right)\right]\\
&= (-1)^{v+u}\left[\frac{\partial r_n}{\partial x_{pq}}\,\mathrm{tr}\left(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}^{vu})\sum_{k=1}^{N_a}\mathbf{A}_k^{vu}kr_n^{-(k+1)}\right)\right.\\
&\qquad\left.-\,\mathrm{tr}\left(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}^{vu})\sum_{k=1}^{N_a}r_n^{-k}\left(\left(\frac{\partial\mathbf{A}_k^{T}}{\partial x_{pq}}\right)^{uv}\right)^{T}\right)\right]
\end{aligned}
\tag{A.42}
$$

A Equação A.42 pode ser simplificada com o auxílio da Equação A.37 e da seguinte relação:

$$
\sum_{k=1}^{N_a}r_n^{-k}\left(\left(\frac{\partial\mathbf{A}_k^{T}}{\partial x_{pq}}\right)^{uv}\right)^{T}=\begin{cases}r_n^{-k_1}\left(\left(\dfrac{\partial\mathbf{A}_{k_1}^{T}}{\partial x_{pq}}\right)^{uv}\right)^{T}, & \text{se } p\leqslant N_a n_y\\ \mathbf{0}, & \text{se } p>N_a n_y\end{cases}
\tag{A.43}
$$

Nesse momento, é importante observar que se $p > N_a n_y$, então $\dfrac{\partial r_n}{\partial x_{pq}} = 0$, $\dfrac{\partial \rho_n}{\partial x_{pq}} = 0$, $\dfrac{\partial(\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}))}{\partial x_{pq}} = \mathbf{0}$, $f_2 = 0$, $f_4 = 0$, $\psi = 0$, $\dfrac{\partial^2 r_n}{\partial x_{pq}\partial x_{ij}} = 0$, $\dfrac{\partial^2 \rho_n}{\partial x_{pq}\partial x_{ij}} = 0$ e $\dfrac{\partial^2 c_{in,0}}{\partial x_{pq}\partial x_{ij}} = 0$. Além disso, se $\boldsymbol{\mathcal{A}}_{r_n}^{vu}$ for inversível:

$$
\mathrm{adj}(\boldsymbol{\mathcal{A}}_{r_n}^{vu}) = \det(\boldsymbol{\mathcal{A}}_{r_n}^{vu})(\boldsymbol{\mathcal{A}}_{r_n}^{vu})^{-1}
\tag{A.44}
$$

Portanto, os componentes da Hessiana de $c_{in,0}$, simbolizados por $(\mathbf{H}(c_{in,0}))_{pqij} = \dfrac{\partial^2 c_{in,0}}{\partial x_{pq} \partial x_{ij}}$, podem ser calculados com o auxílio das Equações A.28 a A.44. Também são necessárias as Equações A.18, A.19 e A.27, deduzidas na Seção A.3.3.

A.3.5 Gradientes das Restrições sobre os Ganhos Estáticos

Seja K_{uv} o elemento (u,v) da matriz de ganhos estáticos $\mathbf{K}$. A derivada parcial das restrições $c^{sup}_{in,uv}$, $c^{inf}_{in,uv}$ e $c_{eq,uv}$ em relação à x_{ij} é obtida da seguinte forma:

$$\begin{aligned} c^{sup}_{in,uv} &= K_{uv} - K^{sup}_{uv}, \qquad \forall (u,v) \text{ tal que } K^{inf}_{uv} < K^{sup}_{uv} \\ \frac{\partial c^{sup}_{in,uv}}{\partial x_{ij}} &= \frac{\partial K_{uv}}{\partial x_{ij}} = \left(\frac{\partial \mathbf{K}}{\partial x_{ij}} \right)_{uv} \end{aligned} \tag{A.45}$$

$$\begin{aligned} c^{inf}_{in,uv} &= K^{inf}_{uv} - K_{uv}, \qquad \forall (u,v) \text{ tal que } K^{inf}_{uv} < K^{sup}_{uv} \\ \frac{\partial c^{inf}_{in,uv}}{\partial x_{ij}} &= -\frac{\partial K_{uv}}{\partial x_{ij}} = -\left(\frac{\partial \mathbf{K}}{\partial x_{ij}} \right)_{uv} \end{aligned} \tag{A.46}$$

$$\begin{aligned} c_{eq,uv} &= K_{uv} - K^{sup}_{uv}, \qquad \forall (u,v) \text{ tal que } K^{inf}_{uv} = K^{sup}_{uv} \\ \frac{\partial c_{eq,uv}}{\partial x_{ij}} &= \frac{\partial K_{uv}}{\partial x_{ij}} = \left(\frac{\partial \mathbf{K}}{\partial x_{ij}} \right)_{uv} \end{aligned} \tag{A.47}$$

Sabe-se que $\mathbf{K} = \boldsymbol{\mathcal{A}}_1^{-1} \boldsymbol{\mathcal{B}}_1$, onde $\boldsymbol{\mathcal{A}}_1 = \boldsymbol{\mathcal{A}}(1)$ e $\boldsymbol{\mathcal{B}}_1 = \boldsymbol{\mathcal{B}}(1)$. Calculando a derivada parcial de $\mathbf{K}$ em relação à x_{ij}:

$$\begin{aligned} \mathbf{K} &= \boldsymbol{\mathcal{A}}_1^{-1} \boldsymbol{\mathcal{B}}_1 \\ \frac{\partial \mathbf{K}}{\partial x_{ij}} &= \frac{\partial (\boldsymbol{\mathcal{A}}_1^{-1} \boldsymbol{\mathcal{B}}_1)}{\partial x_{ij}} \\ &= \boldsymbol{\mathcal{A}}_1^{-1} \frac{\partial \boldsymbol{\mathcal{B}}_1}{\partial x_{ij}} + \frac{\partial \boldsymbol{\mathcal{A}}_1^{-1}}{\partial x_{ij}} \boldsymbol{\mathcal{B}}_1 \\ &= \boldsymbol{\mathcal{A}}_1^{-1} \frac{\partial \boldsymbol{\mathcal{B}}_1}{\partial x_{ij}} - \boldsymbol{\mathcal{A}}_1^{-1} \frac{\partial \boldsymbol{\mathcal{A}}_1}{\partial x_{ij}} \boldsymbol{\mathcal{A}}_1^{-1} \boldsymbol{\mathcal{B}}_1 \qquad \text{Teorema 4} \\ &= \boldsymbol{\mathcal{A}}_1^{-1} \frac{\partial}{\partial x_{ij}} \left(\sum_{k=0}^{N_b} \mathbf{B}_k \right) - \boldsymbol{\mathcal{A}}_1^{-1} \frac{\partial}{\partial x_{ij}} \left(\mathbf{I} - \sum_{k=1}^{N_a} \mathbf{A}_k \right) \boldsymbol{\mathcal{A}}_1^{-1} \boldsymbol{\mathcal{B}}_1 \\ &= \boldsymbol{\mathcal{A}}_1^{-1} \sum_{k=0}^{N_b} \frac{\partial \mathbf{B}_k}{\partial x_{ij}} + \boldsymbol{\mathcal{A}}_1^{-1} \sum_{k=1}^{N_a} \frac{\partial \mathbf{A}_k}{\partial x_{ij}} \boldsymbol{\mathcal{A}}_1^{-1} \boldsymbol{\mathcal{B}}_1 \\ &= \boldsymbol{\mathcal{A}}_1^{-1} \left(\sum_{k=0}^{N_b} \frac{\partial \mathbf{B}_k}{\partial x_{ij}} + \sum_{k=1}^{N_a} \frac{\partial \mathbf{A}_k}{\partial x_{ij}} \boldsymbol{\mathcal{A}}_1^{-1} \boldsymbol{\mathcal{B}}_1 \right) \end{aligned}$$

$$= \boldsymbol{\mathcal{A}}_1^{-1} \left[\sum_{k=0}^{N_b} \left(\frac{\partial \mathbf{B}_k^T}{\partial x_{ij}} \right)^T + \sum_{k=1}^{N_a} \left(\frac{\partial \mathbf{A}_k^T}{\partial x_{ij}} \right)^T \boldsymbol{\mathcal{A}}_1^{-1} \boldsymbol{\mathcal{B}}_1 \right] \tag{A.48}$$

Simplificando a Equação A.48:

$$\frac{\partial \mathbf{K}}{\partial x_{ij}} = \begin{cases} \boldsymbol{\mathcal{A}}_1^{-1} \left(\dfrac{\partial \mathbf{A}_{k_0}^T}{\partial x_{ij}} \right)^T \boldsymbol{\mathcal{A}}_1^{-1} \boldsymbol{\mathcal{B}}_1, & \text{se } i \leqslant N_a n_y \\ \boldsymbol{\mathcal{A}}_1^{-1} \left(\dfrac{\partial \mathbf{B}_{k_0^*}^T}{\partial x_{ij}} \right)^T, & \text{se } i > N_a n_y \end{cases} \tag{A.49}$$

em que $k_0 = \left\lceil \dfrac{i}{n_y} \right\rceil$ é o índice da submatriz $\mathbf{A}_k^T$ que contém o elemento x_{ij} e $k_0^* = \left\lceil \dfrac{i - N_a n_y}{n_u} \right\rceil - 1$ é o índice da submatriz $\mathbf{B}_k^T$ que contém este elemento. Portanto, os gradientes das restrições sobre os ganhos estáticos podem ser calculados com o auxílio das Equações A.45, A.46, A.47 e A.49.

A.3.6 Hessianas das Restrições sobre os Ganhos Estáticos

A derivada segunda das restrições $c_{in,uv}^{sup}$, $c_{in,uv}^{inf}$ e $c_{eq,uv}$ em relação à x_{ij} e à x_{pq} é obtida da seguinte forma:

$$\begin{aligned} \frac{\partial c_{in,uv}^{sup}}{\partial x_{ij}} &= \frac{\partial K_{uv}}{\partial x_{ij}}, \qquad \forall (u,v) \text{ tal que } K_{uv}^{inf} < K_{uv}^{sup} \\ \frac{\partial^2 c_{in,uv}^{sup}}{\partial x_{pq} \partial x_{ij}} &= \frac{\partial^2 K_{uv}}{\partial x_{pq} \partial x_{ij}} = \left(\frac{\partial^2 \mathbf{K}}{\partial x_{pq} \partial x_{ij}} \right)_{uv} \end{aligned} \tag{A.50}$$

$$\begin{aligned} \frac{\partial c_{in,uv}^{inf}}{\partial x_{ij}} &= -\frac{\partial K_{uv}}{\partial x_{ij}}, \qquad \forall (u,v) \text{ tal que } K_{uv}^{inf} < K_{uv}^{sup} \\ \frac{\partial^2 c_{in,uv}^{inf}}{\partial x_{pq} \partial x_{ij}} &= -\frac{\partial^2 K_{uv}}{\partial x_{pq} \partial x_{ij}} = -\left(\frac{\partial^2 \mathbf{K}}{\partial x_{pq} \partial x_{ij}} \right)_{uv} \end{aligned} \tag{A.51}$$

$$\begin{aligned} \frac{\partial c_{eq,uv}}{\partial x_{ij}} &= \frac{\partial K_{uv}}{\partial x_{ij}}, \qquad \forall (u,v) \text{ tal que } K_{uv}^{inf} = K_{uv}^{sup} \\ \frac{\partial^2 c_{eq,uv}}{\partial x_{pq} \partial x_{ij}} &= \frac{\partial^2 K_{uv}}{\partial x_{pq} \partial x_{ij}} = \left(\frac{\partial^2 \mathbf{K}}{\partial x_{pq} \partial x_{ij}} \right)_{uv} \end{aligned} \tag{A.52}$$

Calculando a derivada segunda de $\mathbf{K}$ em relação à x_{ij} e à x_{pq}:

$$
\begin{aligned}
\frac{\partial \mathbf{K}}{\partial x_{ij}} &= \boldsymbol{\mathcal{A}}_1^{-1}\left[\sum_{k=0}^{N_b}\left(\frac{\partial \mathbf{B}_k^T}{\partial x_{ij}}\right)^T + \sum_{k=1}^{N_a}\left(\frac{\partial \mathbf{A}_k^T}{\partial x_{ij}}\right)^T \boldsymbol{\mathcal{A}}_1^{-1}\boldsymbol{\mathcal{B}}_1\right] \\
\frac{\partial^2 \mathbf{K}}{\partial x_{pq}\partial x_{ij}} &= \frac{\partial}{\partial x_{pq}}\left\{\boldsymbol{\mathcal{A}}_1^{-1}\left[\sum_{k=0}^{N_b}\left(\frac{\partial \mathbf{B}_k^T}{\partial x_{ij}}\right)^T + \sum_{k=1}^{N_a}\left(\frac{\partial \mathbf{A}_k^T}{\partial x_{ij}}\right)^T \boldsymbol{\mathcal{A}}_1^{-1}\boldsymbol{\mathcal{B}}_1\right]\right\} \\
&= \boldsymbol{\mathcal{A}}_1^{-1}\frac{\partial}{\partial x_{pq}}\left[\sum_{k=0}^{N_b}\left(\frac{\partial \mathbf{B}_k^T}{\partial x_{ij}}\right)^T + \sum_{k=1}^{N_a}\left(\frac{\partial \mathbf{A}_k^T}{\partial x_{ij}}\right)^T \boldsymbol{\mathcal{A}}_1^{-1}\boldsymbol{\mathcal{B}}_1\right] \\
&\quad + \frac{\partial \boldsymbol{\mathcal{A}}_1^{-1}}{\partial x_{pq}}\left[\sum_{k=0}^{N_b}\left(\frac{\partial \mathbf{B}_k^T}{\partial x_{ij}}\right)^T + \sum_{k=1}^{N_a}\left(\frac{\partial \mathbf{A}_k^T}{\partial x_{ij}}\right)^T \boldsymbol{\mathcal{A}}_1^{-1}\boldsymbol{\mathcal{B}}_1\right] \\
&= \boldsymbol{\mathcal{A}}_1^{-1}\sum_{k=0}^{N_b}\cancelto{\mathbf{0}}{\left(\frac{\partial^2 \mathbf{B}_k^T}{\partial x_{pq}\partial x_{ij}}\right)^T} + \boldsymbol{\mathcal{A}}_1^{-1}\frac{\partial}{\partial x_{pq}}\left[\sum_{k=1}^{N_a}\left(\frac{\partial \mathbf{A}_k^T}{\partial x_{ij}}\right)^T \boldsymbol{\mathcal{A}}_1^{-1}\boldsymbol{\mathcal{B}}_1\right] \\
&\quad - \boldsymbol{\mathcal{A}}_1^{-1}\frac{\partial \boldsymbol{\mathcal{A}}_1}{\partial x_{pq}}\boldsymbol{\mathcal{A}}_1^{-1}\left[\sum_{k=0}^{N_b}\left(\frac{\partial \mathbf{B}_k^T}{\partial x_{ij}}\right)^T + \sum_{k=1}^{N_a}\left(\frac{\partial \mathbf{A}_k^T}{\partial x_{ij}}\right)^T \boldsymbol{\mathcal{A}}_1^{-1}\boldsymbol{\mathcal{B}}_1\right] \\
&= \boldsymbol{\mathcal{A}}_1^{-1}\frac{\partial}{\partial x_{pq}}\left[\sum_{k=1}^{N_a}\left(\frac{\partial \mathbf{A}_k^T}{\partial x_{ij}}\right)^T \boldsymbol{\mathcal{A}}_1^{-1}\boldsymbol{\mathcal{B}}_1\right] \\
&\quad - \boldsymbol{\mathcal{A}}_1^{-1}\frac{\partial}{\partial x_{pq}}\left(\mathbf{I} - \sum_{k=1}^{N_a}\mathbf{A}_k\right)\boldsymbol{\mathcal{A}}_1^{-1}\left[\sum_{k=0}^{N_b}\left(\frac{\partial \mathbf{B}_k^T}{\partial x_{ij}}\right)^T + \sum_{k=1}^{N_a}\left(\frac{\partial \mathbf{A}_k^T}{\partial x_{ij}}\right)^T \boldsymbol{\mathcal{A}}_1^{-1}\boldsymbol{\mathcal{B}}_1\right] \\
&= \boldsymbol{\mathcal{A}}_1^{-1}\sum_{k=1}^{N_a}\left(\frac{\partial \mathbf{A}_k^T}{\partial x_{ij}}\right)^T \frac{\partial \overbrace{\left(\boldsymbol{\mathcal{A}}_1^{-1}\boldsymbol{\mathcal{B}}_1\right)}^{\mathbf{K}}}{\partial x_{pq}} + \boldsymbol{\mathcal{A}}_1^{-1}\sum_{k=1}^{N_a}\cancelto{\mathbf{0}}{\left(\frac{\partial^2 \mathbf{A}_k^T}{\partial x_{pq}\partial x_{ij}}\right)^T}\boldsymbol{\mathcal{A}}_1^{-1}\boldsymbol{\mathcal{B}}_1 \\
&\quad + \boldsymbol{\mathcal{A}}_1^{-1}\sum_{k=1}^{N_a}\frac{\partial \mathbf{A}_k}{\partial x_{pq}}\boldsymbol{\mathcal{A}}_1^{-1}\left[\sum_{k=0}^{N_b}\left(\frac{\partial \mathbf{B}_k^T}{\partial x_{ij}}\right)^T + \sum_{k=1}^{N_a}\left(\frac{\partial \mathbf{A}_k^T}{\partial x_{ij}}\right)^T \boldsymbol{\mathcal{A}}_1^{-1}\boldsymbol{\mathcal{B}}_1\right] \\
&= \boldsymbol{\mathcal{A}}_1^{-1}\sum_{k=1}^{N_a}\left(\frac{\partial \mathbf{A}_k^T}{\partial x_{ij}}\right)^T \frac{\partial \mathbf{K}}{\partial x_{pq}} \\
&\quad + \boldsymbol{\mathcal{A}}_1^{-1}\sum_{k=1}^{N_a}\left(\frac{\partial \mathbf{A}_k^T}{\partial x_{pq}}\right)^T \boldsymbol{\mathcal{A}}_1^{-1}\left[\sum_{k=0}^{N_b}\left(\frac{\partial \mathbf{B}_k^T}{\partial x_{ij}}\right)^T + \sum_{k=1}^{N_a}\left(\frac{\partial \mathbf{A}_k^T}{\partial x_{ij}}\right)^T \boldsymbol{\mathcal{A}}_1^{-1}\boldsymbol{\mathcal{B}}_1\right] \\
&= \boldsymbol{\mathcal{A}}_1^{-1}\sum_{k=1}^{N_a}\left(\frac{\partial \mathbf{A}_k^T}{\partial x_{ij}}\right)^T \boldsymbol{\mathcal{A}}_1^{-1}\left[\sum_{k=0}^{N_b}\left(\frac{\partial \mathbf{B}_k^T}{\partial x_{pq}}\right)^T + \sum_{k=1}^{N_a}\left(\frac{\partial \mathbf{A}_k^T}{\partial x_{pq}}\right)^T \boldsymbol{\mathcal{A}}_1^{-1}\boldsymbol{\mathcal{B}}_1\right] \\
&\quad + \boldsymbol{\mathcal{A}}_1^{-1}\sum_{k=1}^{N_a}\left(\frac{\partial \mathbf{A}_k^T}{\partial x_{pq}}\right)^T \boldsymbol{\mathcal{A}}_1^{-1}\left[\sum_{k=0}^{N_b}\left(\frac{\partial \mathbf{B}_k^T}{\partial x_{ij}}\right)^T + \sum_{k=1}^{N_a}\left(\frac{\partial \mathbf{A}_k^T}{\partial x_{ij}}\right)^T \boldsymbol{\mathcal{A}}_1^{-1}\boldsymbol{\mathcal{B}}_1\right]
\end{aligned}
\tag{A.53}
$$

Simplificando a Equação A.53:

$$\frac{\partial^2 \mathbf{K}}{\partial x_{pq} \partial x_{ij}} = \begin{cases} \boldsymbol{\mathcal{A}}_1^{-1} \left(\dfrac{\partial \mathbf{A}_{k_0}^T}{\partial x_{ij}} \right)^T \boldsymbol{\mathcal{A}}_1^{-1} \left(\dfrac{\partial \mathbf{A}_{k_1}^T}{\partial x_{pq}} \right)^T \boldsymbol{\mathcal{A}}_1^{-1} \boldsymbol{\mathcal{B}}_1 & \\ + \boldsymbol{\mathcal{A}}_1^{-1} \left(\dfrac{\partial \mathbf{A}_{k_1}^T}{\partial x_{pq}} \right)^T \boldsymbol{\mathcal{A}}_1^{-1} \left(\dfrac{\partial \mathbf{A}_{k_0}^T}{\partial x_{ij}} \right)^T \boldsymbol{\mathcal{A}}_1^{-1} \boldsymbol{\mathcal{B}}_1, & \text{se } i \leqslant N_a n_y \text{ e } p \leqslant N_a n_y \\ \boldsymbol{\mathcal{A}}_1^{-1} \left(\dfrac{\partial \mathbf{A}_{k_0}^T}{\partial x_{ij}} \right)^T \boldsymbol{\mathcal{A}}_1^{-1} \left(\dfrac{\partial \mathbf{B}_{k_1^*}^T}{\partial x_{pq}} \right)^T, & \text{se } i \leqslant N_a n_y \text{ e } p > N_a n_y \\ \boldsymbol{\mathcal{A}}_1^{-1} \left(\dfrac{\partial \mathbf{A}_{k_1}^T}{\partial x_{pq}} \right)^T \boldsymbol{\mathcal{A}}_1^{-1} \left(\dfrac{\partial \mathbf{B}_{k_0^*}^T}{\partial x_{ij}} \right)^T, & \text{se } i > N_a n_y \text{ e } p \leqslant N_a n_y \\ \mathbf{0}, & \text{se } i > N_a n_y \text{ e } p > N_a n_y \end{cases} \tag{A.54}$$

em que $k_0 = \left\lceil \dfrac{i}{n_y} \right\rceil$, $k_0^* = \left\lceil \dfrac{i - N_a n_y}{n_u} \right\rceil - 1$, $k_1 = \left\lceil \dfrac{p}{n_y} \right\rceil$ e $k_1^* = \left\lceil \dfrac{p - N_a n_y}{n_u} \right\rceil - 1$. Portanto, as Hessianas das restrições sobre os ganhos estáticos podem ser calculadas com o auxílio das Equações A.50, A.51, A.52 e A.54.

Apêndice B

Códigos dos Algoritmos do Pacote IDENTIPHY

Neste apêndice são mostrados os códigos comentados e as instruções de uso dos quatro algoritmos que compõem o pacote computacional IDENTIPHY. Nos códigos a seguir, quando um argumento de entrada possui um valor *default*, este pode ser acessado atribuindo-se vazio ([ ]) ao argumento em questão ou omitindo-o, caso se trate de um argumento final.

B.1 Algoritmo 1: *pre_teste.m*

```
1  function [U1,dt1,P] = pre_teste(U0,Y0,ST,np,f,g)
2  %
3  %   A partir dos dados obtidos no pré-teste, a função PRE_TESTE faz o
4  %planejamento de sinais para a etapa do teste degrau.
5  %
6  %************************************************************************
7  %
8  %SINTAXE:
9  %
10 %   [U1,dt1,P] = pre_teste(U0,Y0,ST,np,f,g)
11 %
12 %************************************************************************
13 %
14 %ARGUMENTOS DE ENTRADA:
15 %
16 %   U0(nu x N) = [u1(1:N)';u2(1:N)';u3(1:N)';...;unu(1:N)'], onde
17 %   u1,...,unu são os degraus aplicados em cada uma das nu entradas do
18 %   sistema durante o pré-teste, em ordem cronológica, e N é a duração
```

```
%   total do pré-teste, expressa em tempos de amostragem (k). O primeiro
%   degrau (u1) deve ser aplicado a partir do instante k=2.
%
%   Y0(ny x N) = [y1(1:N)';y2(1:N)';y3(1:N)';...;yny(1:N)'], onde
%   y1,...,yny são as ny saídas do sistema medidas durante o pré-teste e N
%   é a duração total do pré-teste, expressa em tempos de amostragem (k).
%
%   ST(Settling Time Threshold) = Número compreendido entre 0 e 1 (0<ST<1)
%   que descreve a tolerância utilizada nos cálculos de tempo de
%   assentamento. Normalmente é utilizado o valor de 0.02 (2%), mas quando
%   os sinais de saída apresentam muito ruído, recomenda-se a utilização de
%   um valor maior, sendo sugerida uma tolerância de 0.05 (5%).
%   (Default: ST=0.02)
%
%   np = Número de partes das entradas projetadas para a próxima etapa
%   (teste degrau). Cada entrada ui será composta de np partes; cada parte,
%   por sua vez, será formada por 2 degraus consecutivos de mesma duração,
%   o primeiro de amplitude A(i) e o segundo de amplitude -A(i). np deve
%   ser especificado como um número inteiro positivo. (Default: np=3)
%
%   f(1 x np) = Fatores multiplicativos para o cálculo das durações dos
%   degraus. A duração de cada degrau da j-ésima parte de ui é dada por
%   ceil(f(j)*dtmax(i)), onde dtmax(i) é o tempo de assentamento do sistema
%   em relação a ui, expresso em tempos de amostragem (k). f deve ser
%   especificado como um vetor-linha contendo np números maiores ou iguais
%   a 1. (Default: f(j)=1+0.5*(j-1))
%
%   g = Comando gráfico. Fazer g=1 para o algoritmo plotar o gráfico das
%   entradas-desvio projetadas em função do tempo. (Default: g=1)
%
%*************************************************************************
%
%ARGUMENTOS DE SAÍDA:
%
%   U1(nu x N) = [u1(1:N)';u2(1:N)';u3(1:N)';...;unu(1:N)'], onde
%   u1,...,unu são as entradas projetadas para a próxima etapa
%   (teste degrau), expressas em termos de variáveis-desvio. U1 é uma
%   matriz esparsa.
%
%   dt1 = Duração mínima da próxima etapa (teste degrau), expressa em
%   tempos de amostragem (k).
```

```
60 %
61 %   P = Estrutura formada pelos parâmetros ST, Ka(ny x nu), Kb(ny x nu),
62 %   taua(ny x nu), taub(ny x ny) e dtmax(nu x 1), onde:
63 %       -> P.ST é o "Settling Time Threshold" utilizado;
64 %       -> P.Ka(i,j) é o ganho estático da saída yi em relação à entrada uj
65 %          calculado quando o degrau em uj está sendo aplicado;
66 %       -> P.Kb(i,j) é o ganho estático da saída yi em relação à entrada uj
67 %          calculado após o degrau em uj ter sido removido;
68 %       -> P.taua(i,j) é a constante de tempo da saída yi em relação à
69 %          entrada uj calculada quando o degrau em uj está sendo aplicado,
70 %          expressa em tempos de amostragem (k);
71 %       -> P.taub(i,j) é a constante de tempo da saída yi em relação à
72 %          entrada uj calculada após o degrau em uj ter sido removido,
73 %          expressa em tempos de amostragem (k);
74 %       -> P.dtmax(j) é o tempo de assentamento do sistema em relação a uj,
75 %          expresso em tempos de amostragem (k).
76 %
77 %*************************************************************************
78 %
79 %AUTOR: Cristiano Salah Mussoi
80 %DATA DE CRIAÇÃO: 05/01/2019
81
82 %% Verificação dos argumentos de entrada:
83
84 switch nargin %Número de argumentos de entrada
85     case 1
86         disp(['Erro: os argumentos de entrada U0 e Y0 '...
87             'devem ser fornecidos']);
88         return; %Encerra a execução da função
89     case 2
90         if isempty(U0)||isempty(Y0)
91             disp(['Erro: os argumentos de entrada U0 e Y0 '...
92             'devem ser fornecidos']);
93             return;
94         end
95         ST = 2/100;
96         np = 3;
97         f = [1 1.5 2];
98         g = 1;
99     case 3
100        if isempty(U0)||isempty(Y0)
```

```
101             disp(['Erro: os argumentos de entrada U0 e Y0 '...
102             'devem ser fornecidos']);
103             return;
104         end
105         if isempty(ST) %Caso ST seja vazio
106             ST = 2/100; %Default
107         elseif isscalar(ST)==0||ST<=0||ST>=1 %ST deve ser um número
                compreendido entre 0 e 1 (0<ST<1)
108             disp('Erro: valor de ST inválido');
109             return;
110         end
111         np = 3;
112         f = [1 1.5 2];
113         g = 1;
114     case 4
115         if isempty(U0)||isempty(Y0)
116             disp(['Erro: os argumentos de entrada U0 e Y0 '...
117             'devem ser fornecidos']);
118             return;
119         end
120         if isempty(ST)
121             ST = 2/100;
122         elseif isscalar(ST)==0||ST<=0||ST>=1
123             disp('Erro: valor de ST inválido');
124             return;
125         end
126         if isempty(np) %Caso np seja vazio
127             np = 3; %Default
128         elseif isscalar(np)==0||np<=0||np~=round(np) %np deve ser um número
                inteiro positivo
129             disp('Erro: valor de np inválido');
130             return;
131         end
132         for i=1:np
133             f(i) = 1+0.5*(i-1);
134         end
135         g = 1;
136     case 5
137         if isempty(U0)||isempty(Y0)
138             disp(['Erro: os argumentos de entrada U0 e Y0 '...
139             'devem ser fornecidos']);
```

```
            return;
        end
        if isempty(ST)
            ST = 2/100;
        elseif isscalar(ST)==0||ST<=0||ST>=1
            disp('Erro: valor de ST inválido');
            return;
        end
        if isempty(np)
            np = 3;
        elseif isscalar(np)==0||np<=0||np~=round(np)
            disp('Erro: valor de np inválido');
            return;
        end
        if isempty(f) %Caso f seja vazio
            for i=1:np
                f(i) = 1+0.5*(i-1); %Default
            end
        elseif length(f)~=np
            disp(['Erro: f deve conter ',num2str(np),' elementos']);
            return;
        elseif length(f)~=length(f(f>=1))
            disp(['Erro: f deve conter somente números ',...
                'maiores ou iguais a 1']);
            return;
        end
        g = 1;
    case 6
        if isempty(U0)||isempty(Y0)
            disp(['Erro: os argumentos de entrada U0 e Y0 '...
            'devem ser fornecidos']);
            return;
        end
        if isempty(ST)
            ST = 2/100;
        elseif isscalar(ST)==0||ST<=0||ST>=1
            disp('Erro: valor de ST inválido');
            return;
        end
        if isempty(np)
            np = 3;
```

```
        elseif isscalar(np)==0||np<=0||np~=round(np)
            disp('Erro: valor de np inválido');
            return;
        end
        if isempty(f)
            for i=1:np
                f(i) = 1+0.5*(i-1);
            end
        elseif length(f)~=np
            disp(['Erro: f deve conter ',num2str(np),' elementos']);
            return;
        elseif length(f)~=length(f(f>=1))
            disp(['Erro: f deve conter somente números ',...
                'maiores ou iguais a 1']);
            return;
        end
        if isempty(g) %Caso g seja vazio
            g = 1; %Default
        end
    otherwise %Caso nargin seja 0
        disp('Erro: número de argumentos de entrada inválido');
        return;
end

%% Verificação dos argumentos de saída:

if nargout~=3 %O número de argumentos de saída deve ser igual a 3
    disp('Erro: devem ser fornecidos 3 argumentos de saída para a função');
    return;
end

%% Verificação das dimensões de U0 e Y0:

if size(U0,2)==size(Y0,2) %Caso U0 e Y0 tenham o mesmo número de colunas
    nu = size(U0,1); %Número de variáveis de entrada
    ny = size(Y0,1); %Número de variáveis de saída
    N = size(Y0,2); %Número total de tempos de amostragem
else
    disp('Erro: U0 e Y0 devem ter o mesmo número de colunas');
    return;
end
```

```
222
223 %% Obtenção de U0 e Y0 em termos de variáveis-desvio:
224
225 L = zeros(nu,1); %Alocação de memória para o vetor L
226 for i=1:nu
227     L(i) = 0.5*(max(U0(i,:)) + min(U0(i,:))); %Média entre o maior e o menor
          valor de ui
228     if U0(i,1)>L(i) %Caso o degrau em ui tenha sido aplicado em k=1
229         disp(['Erro: o degrau em u',num2str(i),...
230             ' deve ser aplicado a partir do instante k=2']);
231         return;
232     else
233         U0(i,:) = U0(i,:) - U0(i,1); %Cálculo de ui-desvio
234         L(i) = 0.5*(max(U0(i,:)) + min(U0(i,:))); %Cálculo de L(i) em termos
              de ui-desvio
235     end
236 end
237
238 for i=1:ny
239     Y0(i,:) = Y0(i,:) - Y0(i,1); %Cálculo de yi-desvio
240 end
241
242 %% Obtenção das amplitudes dos degraus aplicados no pré-teste:
243
244 Iua = cell(nu,1); %Alocação de memória para a célula Iua
245 for i=1:nu
246     k = 1; %Contador
247     for j=1:N
248         if U0(i,j)>L(i)
249             Iua{i}(k) = j; %Iua{i} é um vetor-linha que armazena o intervalo
                  "alto" de ui (correspondente ao seu patamar)
250             k = k + 1;
251         end
252     end
253     if isempty(Iua{i})
254         disp(['Erro: não foi dado um degrau em u',num2str(i),...
255             ' no pré-teste']);
256         return;
257     end
258     ai = Iua{i}(1);
259     af = Iua{i}(end);
```

```
260     if length(Iua{i})<length(linspace(ai,af,af-ai+1))
261         disp(['Erro: foi dado mais de um degrau em u',num2str(i),...
262             ' no pré-teste']);
263         return;
264     end
265 end
266 
267 for i=1:nu-1
268     if Iua{i}(end)>Iua{i+1}(1) %Configuração inválida da matriz U0
269         if Iua{i}(1)>Iua{i+1}(end) %Caso o degrau em ui seja aplicado depois
                do degrau em ui+1
270             disp(['Erro: as linhas ',num2str(i),' e ',num2str(i+1),...
271                 ' de U0 devem ser trocadas de lugar']);
272         else
273             disp(['Erro: os degraus em u',num2str(i),...
274                 ' e u',num2str(i+1),' estão parcialmente sobrepostos']);
275         end
276         return;
277     end
278 end
279 
280 ua = cell(nu,1); %Alocação de memória para a célula ua
281 A = zeros(nu,1); %Alocação de memória para o vetor A
282 ub = cell(nu,1); %Alocação de memória para a célula ub
283 A0 = zeros(nu,1); %Alocação de memória para o vetor A0
284 for i=1:nu
285     ua{i} = U0(i,Iua{i}); %Valores "altos" de ui (patamar alto)
286     A(i) = mean(ua{i}); %Amplitude do degrau em ui
287     if i~=nu
288         ub{i} = U0(i,Iua{i}(end)+1:Iua{i+1}(1)-1); %Valores "baixos" de ui (
                patamar baixo) para i<nu
289     else
290         ub{i} = U0(i,Iua{i}(end)+1:N); %Valores "baixos" de ui para i=nu
291     end
292     A0(i) = mean(ub{i}); %A0 deve ser um vetor de amplitudes nulas
293 end
294 A_max = max(A); %Maior amplitude aplicada
295 
296 %% Cálculo dos ganhos estáticos e das constantes de tempo:
297 
298 ya = cell(ny,nu); %Alocação de memória para a célula ya
```

```
299 yb = cell(ny,nu); %Alocação de memória para a célula yb
300 ya_ee = zeros(ny,nu); %Alocação de memória para a matriz ya_ee
301 yb_ee = zeros(ny,nu); %Alocação de memória para a matriz yb_ee
302 Ka = zeros(ny,nu); %Alocação de memória para a matriz Ka
303 Kb = zeros(ny,nu); %Alocação de memória para a matriz Kb
304 taua = zeros(ny,nu); %Alocação de memória para a matriz taua
305 taub = zeros(ny,nu); %Alocação de memória para a matriz taub
306 dta = zeros(ny,nu); %Alocação de memória para a matriz dta
307 dtb = zeros(ny,nu); %Alocação de memória para a matriz dtb
308 for i=1:ny
309     for j=1:nu
310     ya{i,j} = Y0(i,Iua{j}); %yi "alto": valores de yi quando o degrau em uj
            está sendo aplicado
311     ya_ee(i,j) = mean(ya{i,j}(ceil(0.9*end):end)); %Valor de ya{i,j} no
            estado estacionário
312     Ka(i,j) = ya_ee(i,j)/A(j); %Ganho estático "alto" de yi em relação a uj
313     S = stepinfo(ya{i,j},0:length(ya{i,j})-1,ya_ee(i,j),...
314         'RiseTimeThreshold',[0.283 0.632],'SettlingTimeThreshold',ST);
315     taua(i,j) = 1.5*S.RiseTime; %Constante de tempo "alta" de yi em relação
            a uj (método de Smith)
316     dta(i,j) = S.SettlingTime; %Tempo de assentamento "alto" de yi em
            relação a uj
317     if j~=nu
318         yb{i,j} = Y0(i,Iua{j}(end)+1:Iua{j+1}(1)-1); %yi "baixo": valores de
                yi após o degrau em uj ter sido removido e antes do degrau em
               uj+1 ser aplicado
319     else
320         yb{i,j} = Y0(i,Iua{j}(end)+1:N); %yi "baixo" após o último degrau (
               unu) ter sido removido
321     end
322     yb_ee(i,j) = mean(yb{i,j}(ceil(0.9*end):end)); %Valor de yb{i,j} no
            estado estacionário
323     Kb(i,j) = (yb_ee(i,j) - ya_ee(i,j))/(A0(j)-A(j)); %Ganho estático "baixo
            " de yi em relação a uj
324     S = stepinfo(yb{i,j},0:length(yb{i,j})-1,yb_ee(i,j),...
325         'RiseTimeThreshold',[0.283 0.632],'SettlingTimeThreshold',ST);
326     taub(i,j) = 1.5*S.RiseTime; %Constante de tempo "baixa" de yi em relação
             a uj
327     dtb(i,j) = S.SettlingTime; %Tempo de assentamento "baixo" de yi em
            relação a uj
328     end
```

```
329 end
330 P = struct; %Inicialização da estrutura de parâmetros P
331 P.ST = ST;
332 P.Ka = Ka;
333 P.Kb = Kb;
334 P.taua = taua;
335 P.taub = taub;
336
337 %% Cálculo do tempo de assentamento do sistema em relação a cada entrada:
338
339 dtmax = zeros(nu,1); %Alocação de memória para o vetor dtmax
340 for j=1:nu
341     dtmax(j) = ceil(max([dta(:,j);dtb(:,j)])); %Tempo de assentamento do
          sistema em relação a uj (número inteiro)
342 end
343 for j=1:nu
344     if isnan(dtmax(j)) %Caso não tenha sido possível calcular dtmax(j) por
          causa do ruído
345         dtmax(j) = max(dtmax); %dtmax(j) recebe o valor do maior tempo de
              assentamento obtido (considerando todas as entradas)
346     end
347     if dtmax(j)==0 %Caso dtmax(j) seja nulo
348         dtmax(j) = min(dtmax(dtmax~=0)); %dtmax(j) recebe o valor do menor
              tempo de assentamento não nulo obtido (considerando todas as
              entradas)
349     end
350 end
351 P.dtmax = dtmax;
352
353 %% Cálculo da matriz U1:
354
355 u = cell(nu,np); %Alocação de memória para a célula u
356 tm = zeros(nu,np); %Alocação de memória para a matriz tm
357 for i=1:nu
358     for j=1:np %np é o número de partes das entradas que serão projetadas
359         tm(i,j) = 1+ceil(f(j)*dtmax(i)); %Em tm (tempo de mudança) a
              amplitude do degrau da j-ésima parte de ui muda de A(i) para -A(
              i)
360         for k=1:tm(i,j)-1
361             u{i,j}(k) = A(i); %u{i,j} é a j-ésima parte da entrada projetada
                  ui
```

```
362         end
363         for k=tm(i,j):2*(tm(i,j)-1)
364             u{i,j}(k) = -A(i);
365         end
366     end
367 end
368 
369 dt1 = 1+sum(dtmax)+2*sum(sum(tm-1)); %Duração mínima da próxima etapa (teste
     degrau)
370 U1 = zeros(nu,dt1+1); %Alocação de memória para a matriz U1
371 
372 k0 = 2; %Contador (a primeira coluna de U1 é composta somente de zeros)
373 I = randperm(nu); %Vetor formado pelos números de 1 até nu distribuídos de
    forma aleatória
374 for i=I
375     J = randperm(np); %Vetor formado pelos números de 1 até np distribuídos
        de forma aleatória
376     for j=J
377         U1(i,k0:k0+length(u{i,j})-1) = u{i,j}; %U1 é a matriz das entradas
            projetadas para a próxima etapa (teste degrau)
378         k0 = k0+length(u{i,j});
379     end
380     k0 = k0+dtmax(i); %É deixado um espaço de dtmax(i) após a aplicação da
        última parte de ui
381 end
382 U1 = sparse(U1); %U1 é convertida em uma matriz esparsa
383 
384 %% Gráfico das entradas-desvio projetadas para o teste degrau:
385 
386 if g==1
387     figure(3)
388     stairs(U1');
389     hold on;
390     plot(zeros(dt1+1,1),'k');
391     hold off;
392     title('Teste Degrau');
393     xlabel('k'); %Tempo de amostragem
394     ylabel('u(k)'); %Variáveis-desvio de entrada
395     xlim([1,dt1+1]);
396     ylim([-1.1*A_max,1.1*A_max]);
397     set(gca,'XTick',unique([1,get(gca,'XTick')]));
```

```
398     leg = cell(1,nu); %Alocação de memória para a célula leg
399     for i=1:nu
400         leg{i}=['u',num2str(i)]; %Legendas
401     end
402     legend(leg,'Orientation','vertical','Location','northeastoutside');
403 end
404 
405 end
```

B.2 Algoritmo 2: *teste_degrau.m*

```
1  function [K,tau,dtmax,dtg,dt2] = teste_degrau(U1,Y1,P,MS,G,w)
2  %
3  %   A função TESTE_DEGRAU realiza o tratamento conjunto dos dados obtidos
4  %no teste degrau e no pré-teste. Ela também separa os sinais de entrada em
5  %grupos GBN (Generalized Binary Noise).
6  %
7  %*************************************************************************
8  %
9  %SINTAXE:
10 %
11 %   [K,tau,dtmax,dtg,dt2] = teste_degrau(U1,Y1,P,MS,G,w)
12 %
13 %*************************************************************************
14 %
15 %ARGUMENTOS DE ENTRADA:
16 %
17 %   U1(nu x N) = [u1(1:N)';u2(1:N)';u3(1:N)';...;unu(1:N)'], onde
18 %   u1,...,unu são as sequências de degraus aplicadas em cada uma das nu
19 %   entradas do sistema durante o teste degrau e N é a duração total do
20 %   teste, expressa em tempos de amostragem (k). O primeiro degrau deve ser
21 %   aplicado a partir do instante k=2.
22 %
23 %   Y1(ny x N) = [y1(1:N)';y2(1:N)';y3(1:N)';...;yny(1:N)'], onde
24 %   y1,...,yny são as ny saídas do sistema medidas durante o teste degrau e
25 %   N é a duração total do teste, expressa em tempos de amostragem (k).
26 %
27 %   P = Estrutura de parâmetros gerada pela função pre_teste.
28 %
29 %   MS(ny x nu) = Matriz de sinais dos ganhos estáticos. O valor de MS(i,j)
30 %   informa o sinal do ganho estático de yi em relação a uj, podendo ser
31 %   1 (ganho positivo), -1 (ganho negativo), 0 (ganho nulo) ou NaN (o sinal
```

```
32 %   do ganho é desconhecido). Como esta matriz é calculada pela própria
33 %   função TESTE_DEGRAU, deve-se atribuir inicialmente o valor de sqrt(-1)
34 %   a todos os elementos de MS. (Default: MS=sqrt(-1)*ones(ny,nu))
35 %
36 %   G = {G{1},G{2},...,G{ng}} é uma célula, onde G{i} é um vetor-linha que
37 %   contém os índices das entradas pertencentes ao i-ésimo grupo GBN e ng é
38 %   o número total de grupos. (Default: G={1:nu})
39 %
40 %   w = Fator multiplicativo para o cálculo das durações dos grupos GBN. w
41 %   deve ser especificado como um número maior ou igual a 1. (Default: w=7)
42 %
43 %*************************************************************************
44 %
45 %ARGUMENTOS DE SAÍDA:
46 %
47 %   K = Estrutura formada pelos parâmetros K.inf(ny x nu), K.sup(ny x nu) e
48 %   K.MS(ny x nu), onde K.inf(i,j) e K.sup(i,j) são os limites inferior e
49 %   superior para o ganho estático de yi em relação a uj, respectivamente,
50 %   e K.MS é a matriz de sinais dos ganhos estáticos.
51 %
52 %   tau = Estrutura formada pelos parâmetros tau.min(nu x 1) e
53 %   tau.max(nu x 1), onde tau.min(j) e tau.max(j) são a menor e a maior
54 %   constante de tempo obtidas para a entrada uj, respectivamente. Os
55 %   valores de tau são expressos em tempos de amostragem (k).
56 %
57 %   dtmax(2 x nu) = Tempos de assentamento do sistema. A primeira linha da
58 %   matriz informa o índice da entrada (j) e a segunda o tempo de
59 %   assentamento em relação a uj, expresso em tempos de amostragem (k). Os
60 %   tempos de assentamento estão dispostos em ordem decrescente.
61 %
62 %   dtg(1 x ng) = Durações dos grupos GBN, expressas em tempos de
63 %   amostragem (k). A duração do i-ésimo grupo (dtg(i)) é igual ao maior
64 %   tempo de assentamento no grupo multiplicado por w.
65 %
66 %   dt2 = Duração mínima da próxima etapa (teste GBN), expressa em tempos
67 %   de amostragem (k).
68 %
69 %*************************************************************************
70 %
71 %OBSERVAÇÃO:
72 %
```

```
%   Inicialmente, deve-se executar a função apenas com os 3 primeiros
%   argumentos de entrada. Em seguida, se for necessário separar os sinais
%   de entrada em grupos GBN, é preciso criar a célula G e executar a
%   função uma segunda vez, agora fazendo MS=K.MS (a célula G pode ser
%   criada com base em dtmax).
%
%*************************************************************************
%
%AUTOR: Cristiano Salah Mussoi
%DATA DE CRIAÇÃO: 05/01/2019

%% Definição das variáveis globais:

global t2;
t2 = [];

%% Verificação dos argumentos de entrada:

switch nargin %Número de argumentos de entrada
    case 1
        disp(['Erro: os argumentos de entrada U1, Y1 e P ',...
            'devem ser fornecidos']);
        return; %Encerra a execução da função
    case 2
        disp(['Erro: os argumentos de entrada U1, Y1 e P ',...
            'devem ser fornecidos']);
        return;
    case 3
        if isempty(U1)||isempty(Y1)||isempty(P)
            disp(['Erro: os argumentos de entrada U1, Y1 e P ',...
            'devem ser fornecidos']);
            return;
        end
        MS = sqrt(-1)*ones(size(Y1,1),size(U1,1));
        G = {1:size(U1,1)};
        w = 7;
    case 4
        if isempty(U1)||isempty(Y1)||isempty(P)
            disp(['Erro: os argumentos de entrada U1, Y1 e P ',...
            'devem ser fornecidos']);
            return;
```

```
114         end
115         if isempty(MS) %Caso MS seja vazio
116             MS = sqrt(-1)*ones(size(Y1,1),size(U1,1)); %Default
117         elseif size(MS,1)~=size(Y1,1)||size(MS,2)~=size(U1,1)
118             disp(['Erro: MS deve ser especificado como uma matriz ',...
119                 'com ny linhas e nu colunas']);
120             return;
121         else
122             for i=1:size(Y1,1)
123                 for j=1:size(U1,1)
124                     if isnan(MS(i,j))==0 && MS(i,j)~=-1 && MS(i,j)~=0 &&...
125                             MS(i,j)~=1 && MS(i,j)~=sqrt(-1)
126                         disp(['Erro: valor de MS('...
127                             num2str(i) ',' num2str(j) ') inválido']);
128                         return;
129                     end
130                 end
131             end
132         end
133         G = {1:size(U1,1)};
134         w = 7;
135     case 5
136         if isempty(U1)||isempty(Y1)||isempty(P)
137             disp(['Erro: os argumentos de entrada U1, Y1 e P ',...
138             'devem ser fornecidos']);
139             return;
140         end
141         if isempty(MS)
142             MS = sqrt(-1)*ones(size(Y1,1),size(U1,1));
143         elseif size(MS,1)~=size(Y1,1)||size(MS,2)~=size(U1,1)
144             disp(['Erro: MS deve ser especificado como uma matriz ',...
145                 'com ny linhas e nu colunas']);
146             return;
147         else
148             for i=1:size(Y1,1)
149                 for j=1:size(U1,1)
150                     if isnan(MS(i,j))==0 && MS(i,j)~=-1 && MS(i,j)~=0 &&...
151                             MS(i,j)~=1 && MS(i,j)~=sqrt(-1)
152                         disp(['Erro: valor de MS('...
153                             num2str(i) ',' num2str(j) ') inválido']);
154                         return;
```

```
                    end
                end
            end
        end
        if isempty(G) %Caso G seja vazio
            G = {1:size(U1,1)}; %Default
        elseif iscell(G)==0
            disp('Erro: G deve ser especificado como uma célula');
            return;
        else
            for i=1:length(G)
                if isvector(G{i})==0
                    disp('Erro: os elementos de G devem ser vetores');
                    return;
                end
            end
        end
        w = 7;
    case 6
        if isempty(U1)||isempty(Y1)||isempty(P)
            disp(['Erro: os argumentos de entrada U1, Y1 e P ',...
            'devem ser fornecidos']);
            return;
        end
        if isempty(MS)
            MS = sqrt(-1)*ones(size(Y1,1),size(U1,1));
        elseif size(MS,1)~=size(Y1,1)||size(MS,2)~=size(U1,1)
            disp(['Erro: MS deve ser especificado como uma matriz ',...
                'com ny linhas e nu colunas']);
            return;
        else
            for i=1:size(Y1,1)
                for j=1:size(U1,1)
                    if isnan(MS(i,j))==0 && MS(i,j)~=-1 && MS(i,j)~=0 &&...
                            MS(i,j)~=1 && MS(i,j)~=sqrt(-1)
                        disp(['Erro: valor de MS('...
                            num2str(i) ',' num2str(j) ') inválido']);
                        return;
                    end
                end
            end
```

```
196         end
197         if isempty(G)
198             G = {1:size(U1,1)};
199         elseif iscell(G)==0
200             disp('Erro: G deve ser especificado como uma célula');
201             return;
202         else
203             for i=1:length(G)
204                 if isvector(G{i})==0
205                     disp('Erro: os elementos de G devem ser vetores');
206                     return;
207                 end
208             end
209         end
210         if isempty(w) %Caso w seja vazio
211             w = 7; %Default
212         elseif isscalar(w)==0||w<1 %w deve ser especificado como um número
                maior ou igual a 1
213             disp('Erro: valor de w inválido');
214             return;
215         end
216     otherwise %Caso nargin seja 0
217         disp(['Erro: devem ser fornecidos argumentos de entrada ',...
218             'para a função']);
219         return;
220 end
221
222 %% Verificação dos argumentos de saída:
223
224 if nargout~=5 %O número de argumentos de saída deve ser igual a 5
225     disp('Erro: devem ser fornecidos 5 argumentos de saída para a função');
226     return;
227 end
228
229 %% Extração dos dados da estrutura P (provenientes do pré-teste):
230
231 ST = P.ST; %"Settling Time Threshold" utilizado no pré-teste
232 Ka = P.Ka;
233 Kb = P.Kb;
234 taua = P.taua;
235 taub = P.taub;
```

```
236 dtmax_pt = P.dtmax;
237
238 %% Verificação das dimensões de U1 e Y1:
239
240 if size(U1,2)==size(Y1,2) %Caso U1 e Y1 tenham o mesmo número de colunas
241     nu = size(U1,1); %Número de variáveis de entrada
242     ny = size(Y1,1); %Número de variáveis de saída
243     N = size(Y1,2); %Número total de tempos de amostragem
244 else
245     disp('Erro: U1 e Y1 devem ter o mesmo número de colunas');
246     return;
247 end
248
249 %% Obtenção de U1 e Y1 em termos de variáveis-desvio:
250
251 L0 = zeros(nu,1); %Alocação de memória para o vetor L0
252 LS = zeros(nu,1); %Alocação de memória para o vetor LS
253 LI = zeros(nu,1); %Alocação de memória para o vetor LI
254 for i=1:nu
255     L0(i) =  0.5*(max(U1(i,:)) + min(U1(i,:))); %Média entre o maior e o
        menor valor de ui
256     LS(i) = 0.5*(max(U1(i,:)) + L0(i)); %Média entre o maior valor de ui e
        L0(i)
257     LI(i) = 0.5*(min(U1(i,:)) + L0(i)); %Média entre o menor valor de ui e
        L0(i)
258     if U1(i,1)>LS(i) || U1(i,1)<LI(i) %Caso o primeiro degrau em ui tenha
        sido aplicado em k=1
259         disp(['Erro: o primeiro degrau em u',num2str(i),...
260             ' deve ser aplicado a partir do instante k=2']);
261         return;
262     else
263         U1(i,:) = U1(i,:) - U1(i,1); %Cálculo de ui-desvio
264         L0(i) =  0.5*(max(U1(i,:)) + min(U1(i,:))); %Cálculo de L0(i) em
            termos de ui-desvio
265         LS(i) = 0.5*(max(U1(i,:)) + L0(i)); %Cálculo de LS(i) em termos de
            ui-desvio
266         LI(i) = 0.5*(min(U1(i,:)) + L0(i)); %Cálculo de LI(i) em termos de
            ui-desvio
267     end
268 end
269
```

```
for i=1:ny
    Y1(i,:) = Y1(i,:) - Y1(i,1); %Cálculo de yi-desvio
end

%% Obtenção do número de partes dos sinais de entrada:

np = zeros(nu,1); %Alocação de memória para o vetor np
for i=1:nu
    for j=2:N
        if U1(i,j)>LS(i) && U1(i,j-1)<LS(i)
            np(i) = np(i)+1;
        end
    end
end

if (np-np(1))~=zeros(nu,1)
    disp(['Erro: o números de partes de cada sinal de entrada ',...
        'deve ser igual']);
    return;
else
    np = np(1); %np passa a ser um escalar igual ao número de partes de cada
        sinal
end

%% Obtenção das amplitudes dos degraus aplicados:

Iu = cell(nu,np,2); %Alocação de memória para a célula Iu
Iu0 = cell(nu,1); %Alocação de memória para a célula Iu0
n0 = zeros(nu,1); %Alocação de memória para o vetor n0
for i=1:nu %Índice da entrada
    j = 1; %Índice da parte
    k = 1; %Índice do degrau
    n = 2; %Instante de tempo inicial
    while n<=N
        if U1(i,n)>LS(i) && U1(i,n-1)<LS(i)
            q = 1; %Contador
            while U1(i,n)>LS(i)
                Iu{i,j,k}(q) = n; %Intervalo de tempo correspondente ao 1º
                    degrau da parte j de ui
                q = q+1;
                n = n+1; %Próximo instante de tempo
```

```
309             end
310             k = k+1; %Mudança de degrau
311             continue; %Retorna ao início do loop while (n<=N)
312         end
313         if U1(i,n)<LI(i) && U1(i,n-1)>LI(i)
314             q = 1; %Contador
315             while U1(i,n)<LI(i)
316                 Iu{i,j,k}(q) = n; %Intervalo de tempo correspondente ao 2º
                        degrau da parte j de ui
317                 q = q+1;
318                 n = n+1; %Próximo instante de tempo
319             end
320             j = j+1; %Mudança de parte
321             k = k-1; %Mudança de degrau
322             continue; %Retorna ao início do loop while (n<=N)
323         end
324         if j==np+1 && k==1 %Fim da sequência de degraus aplicada em ui
325             n0(i) = n; %Instante de tempo em que acabou o último degrau em
                    ui
326             break; %Sai do loop while (n<=N)
327         end
328         n = n+1; %Caso n não satifaça nenhum condicional if -> próximo
                instante de tempo
329     end
330 end
331
332 for i=1:nu
333     for j=1:np
334         for k=1:2
335             a = Iu{i,j,k}(1);
336             b = Iu{i,j,k}(end);
337             if isempty(Iu{i,j,k})||...
338                     length(Iu{i,j,k})<length(linspace(a,b,b-a+1)) %
                        Verificação dos intervalos Iu{i,j,k} obtidos
339                 disp(['Erro: os degraus em u',num2str(i),...
340                     ' não foram aplicados corretamente']);
341                 return;
342             end
343         end
344     end
345 end
```

```
346
347 u = cell(nu,np,2); %Alocação de memória para a célula u
348 A = zeros(nu,np,2); %Alocação de memória para a matriz A
349 Lu = zeros(nu,np,2); %Alocação de memória para a matriz Lu
350 u0 = cell(nu,1); %Alocação de memória para a célula u0
351 A0 = zeros(nu,1); %Alocação de memória para o vetor A0
352 Lu0 = zeros(nu,1); %Alocação de memória para o vetor Lu0
353 for i=1:nu
354     for j=1:np
355         for k=1:2
356         u{i,j,k} = U1(i,Iu{i,j,k}); %k-ésimo degrau da parte j de ui
357         A(i,j,k) = mean(u{i,j,k}); %Amplitude do k-ésimo degrau da parte j
                de ui
358         Lu(i,j,k) = length(u{i,j,k}); %Comprimento do k-ésimo degrau da
                parte j de ui
359         end
360     end
361     Iu0{i} = n0(i):min(n0(i)+dtmax_pt(i)-1,N); %Intervalo de tempo após a
            aplicação dos degraus em ui em que o sistema não é perturbado
362     u0{i} = U1(i,Iu0{i}); %Degrau nulo
363     A0(i) = mean(u0{i}); %Amplitude nula
364     Lu0(i) = length(u0{i}); %Comprimento do degrau nulo
365 end
366
367 f = zeros(nu,np,2); %Alocação de memória para a matriz f
368 f0 = zeros(nu,1); %Alocação de memória para o vetor f0
369 for i=1:nu
370     for j=1:np
371         for k=1:2
372             f(i,j,k) = Lu(i,j,k)/dtmax_pt(i); %Fator multiplicativo de u{i,j
                    ,k}
373         end
374     end
375     f0(i) = Lu0(i)/dtmax_pt(i); %Fator multiplicativo de u0{i}
376 end
377
378 %% Cálculo dos parâmetros K, tau e dtmax:
379
380 y = cell(ny,nu,np,2); %Alocação de memória para a célula y
381 y_ee = zeros(ny,nu,np,2); %Alocação de memória para a matriz y_ee
382 K = zeros(ny,nu,np,2); %Alocação de memória para a matriz K
```

```
383 tau = zeros(ny,nu,np,2); %Alocação de memória para a matriz tau
384 dt = zeros(ny,nu,np,2); %Alocação de memória para a matriz dt
385 y0 = cell(ny,nu); %Alocação de memória para a célula y0
386 y0_ee = zeros(ny,nu); %Alocação de memória para a matriz y0_ee
387 K0 = zeros(ny,nu); %Alocação de memória para a matriz K0
388 tau0 = zeros(ny,nu); %Alocação de memória para a matriz tau0
389 dt0 = zeros(ny,nu); %Alocação de memória para a matriz dt0
390 for n=1:ny
391     for i=1:nu
392         for j=1:np
393             for k=1:2
394                 y{n,i,j,k} = Y1(n,Iu{i,j,k}); %Resposta de yn ao k-ésimo
                        degrau da parte j de ui
395                 y_ee(n,i,j,k) = mean(y{n,i,j,k}...
396                     (ceil(0.9*end/f(i,j,k)):end)); %Valor de y{n,i,j,k} no
                            estado estacionário
397                 if j==1 && k==1 %Primeiro degrau da parte 1 de ui
398                     K(n,i,j,k) = y_ee(n,i,j,k)/A(i,j,k); %Ganho estático de
                            yn em relação ao k-ésimo degrau da parte j de ui
399                 elseif k==1 %Primeiro degrau da parte j de ui (j>=2)
400                     K(n,i,j,k) = (y_ee(n,i,j,k) - y_ee(n,i,j-1,k+1))...
401                         /(A(i,j,k) - A(i,j-1,k+1));
402                 else %Segundo degrau da parte j de ui
403                     K(n,i,j,k) = (y_ee(n,i,j,k) - y_ee(n,i,j,k-1))...
404                         /(A(i,j,k) - A(i,j,k-1));
405                 end
406                 S = stepinfo(y{n,i,j,k},0:length(y{n,i,j,k})-1,...
407                     y_ee(n,i,j,k),'RiseTimeThreshold',[0.283 0.632],...
408                     'SettlingTimeThreshold',ST);
409                 tau(n,i,j,k) = 1.5*S.RiseTime; %Constante de tempo de yn em
                        relação ao k-ésimo degrau da parte j de ui (método de
                        Smith)
410                 dt(n,i,j,k) = S.SettlingTime; %Tempo de assentamento de yn
                        em relação ao k-ésimo degrau da parte j de ui
411             end
412         end
413         y0{n,i} = Y1(n,Iu0{i}); %Resposta de yn à remoção do último degrau
                em ui
414         y0_ee(n,i) = mean(y0{n,i}(ceil(0.9*end/f0(i)):end)); %Valor de y0{n,
                i} no estado estacionário
415         K0(n,i) = (y0_ee(n,i) - y_ee(n,i,np,2))/(A0(i) - A(i,np,2)); %Último
```

```
            ganho estático de yn em relação a ui
416         S = stepinfo(y0{n,i},0:length(y0{n,i})-1,y0_ee(n,i),...
417             'RiseTimeThreshold',[0.283 0.632],'SettlingTimeThreshold',ST);
418         tau0(n,i) = 1.5*S.RiseTime; %Última constante de tempo de yn em
                relação a ui
419         dt0(n,i) = S.SettlingTime; %Último tempo de assentamento de yn em
                relação a ui
420     end
421 end
422
423 Kinf = zeros(ny,nu); %Alocação de memória para a matriz Kinf
424 Ksup = zeros(ny,nu); %Alocação de memória para a matriz Ksup
425 for n=1:ny
426     for i=1:nu
427         M1 = K(n,i,:,:);
428         x = [M1(:); K0(n,i); Ka(n,i); Kb(n,i)]; %Conjunto de valores obtidos
                para o ganho estático de yn em relação a ui
429         NC = 99.99; %Nível de confiança (%)
430         IC = Student(x,NC); %Intervalo de confiança de Student
431         Kinf(n,i) = IC(1); %Limite inferior para o ganho estático de yn em
                relação a ui (considerando o teste degrau e o pré-teste)
432         Ksup(n,i) = IC(2); %Limite superior para o ganho estático de yn em
                relação a ui (considerando o teste degrau e o pré-teste)
433         if Kinf(n,i)<0 && Ksup(n,i)<=0 %Caso o ganho estático de yn em
                relação a ui seja negativo
434             MS(n,i) = -1;
435         end
436         if Kinf(n,i)==0 && Ksup(n,i)==0 %Caso o ganho estático de yn em
                relação a ui seja nulo
437             MS(n,i) = 0;
438         end
439         if Kinf(n,i)>=0 && Ksup(n,i)>0 %Caso o ganho estático de yn em
                relação a ui seja positivo
440             MS(n,i) = 1;
441         end
442         if Kinf(n,i)<0 && Ksup(n,i)>0 %Caso o sinal do ganho estático de yn
                em relação a ui esteja indefinido
443             while MS(n,i)==sqrt(-1) %O programa pergunta ao usuário qual é o
                    sinal do ganho estático
444                 prompt = ['\nQual é o sinal de K('...
445                     num2str(n) ',' num2str(i) ')?\n'...
```

```
446                     '\nPositivo: digite 1 e pressione ENTER;'...
447                     '\nNegativo: digite -1 e pressione ENTER;'...
448                     '\nNulo: digite 0 e pressione ENTER;'...
449                     '\nNão sei: pressione ENTER.\n\n'];
450                 sinal = input(prompt); %O sinal do ganho estático é
                        fornecido pelo usuário
451                 if isempty(sinal) %Caso o usuário não saiba o sinal
452                     MS(n,i) = NaN;
453                 elseif isscalar(sinal)==0 ||...
454                         sinal~=-1 && sinal~=0 && sinal~=1 %Caso o usuário
                            forneça um valor inválido para o sinal
455                     disp('Entrada inválida');
456                     MS(n,i) = sqrt(-1); %A pergunta é refeita
457                 else
458                     MS(n,i) = sinal;
459                 end
460             end
461             switch MS(n,i)
462                 case -1 %Sinal negativo
463                     Ksup(n,i) = 0;
464                 case 0 %Sinal nulo
465                     Kinf(n,i) = 0;
466                     Ksup(n,i) = 0;
467                 case 1 %Sinal positivo
468                     Kinf(n,i) = 0;
469             end
470         end
471     end
472 end
473 K = struct; %Inicialização da estrutura K
474 K.inf = Kinf;
475 K.sup = Ksup;
476 K.MS = MS;
477
478 taumin = zeros(nu,1); %Alocação de memória para o vetor taumin
479 taumax = zeros(nu,1); %Alocação de memória para o vetor taumax
480 dtmax = zeros(nu,1); %Alocação de memória para o vetor dtmax
481 for i=1:nu
482     M2 = tau(:,i,:,:);
483     taumin(i) = min([M2(:); tau0(:,i); taua(:,i); taub(:,i)]); %Menor
          constante de tempo obtida para a entrada ui (considerando o teste
```

```
        degrau e o pré-teste)
484     taumax(i) = max([M2(:); tau0(:,i); taua(:,i); taub(:,i)]); %Maior
        constante de tempo obtida para a entrada ui (considerando o teste
        degrau e o pré-teste)
485     M3 = dt(:,i,:,:);
486     dtmax(i) = max([M3(:); dt0(:,i); dtmax_pt(i)]);
487     dtmax(i) = ceil(dtmax(i)); %Tempo de assentamento do sistema em relação
        a ui (considerando o teste degrau e o pré-teste)
488 end
489 tau = struct; %Inicialização da estrutura tau
490 tau.min = taumin;
491 tau.max = taumax;
492
493 %% Cálculo dos parâmetros dtg e dt2:
494
495 ng = length(G); %Número de grupos GBN
496 dtg = zeros(1,ng); %Inicialização do vetor-linha dtg
497 for i=1:ng
498     dtg(i) = ceil(w*max(dtmax(G{i}(:)))); %Duração do grupo i
499 end
500 dt2 = 1+sum(dtg); %Duração mínima da próxima etapa (teste GBN)
501
502 [dtmax,Id] = sort(dtmax,'descend');
503 dtmax = [Id dtmax]'; %Os tempos de assentamento contidos no vetor dtmax são
    postos em ordem decrescente
504
505 end
506
507 function IC = Student(x,NC)
508 %
509 %Calcula o intervalo de confiança para variáveis aleatórias contínuas com
510 %distribuição normal.
511 %
512 %SINTAXE:
513 %
514 %IC = Student(x,NC)
515 %
516 %ARGUMENTOS DE ENTRADA:
517 %
518 %x(N x 1) = Amostra de tamanho N.
519 %
```

```
%NC = Nível de confiança, expresso em porcentagem (%).
%
%ARGUMENTOS DE SAÍDA:
%
%IC(1 x 2) = Intervalo de confiança de Student.

%% Cálculo do intervalo de confiança de Student (IC):

global t2;

%Tamanho da amostra (N):
N = length(x);

%Média amostral (xmed):
xmed = mean(x);

%Desvio padrão amostral (s):
s = std(x);

if isempty(t2)

    %Número de graus de liberdade (ni):
    ni = N-1;

    %Função densidade de probabilidade (p):
    p = @(t) 1/sqrt(pi*ni)*gamma((ni+1)/2)/gamma(ni/2)*...
        (1+t.^2/ni).^(-(ni+1)/2);

    %Função probabilidade acumulada (P):
    P = @(t) quadl(p,-1e10,t);

    %Cálculo de t1 e t2:
    P1 = (1-NC/100)/2;
    f1 = @(t) P(t)-P1;
    t1 = fzero(f1,0);
    t2 = -t1;

end

%Intervalo de confiança (IC):
IC(1) = xmed-t2*s/sqrt(N);
```

```
561 IC(2) = xmed+t2*s/sqrt(N);
562
563 end
```

B.3 Algoritmo 3: *identificador.m*

```
1  function [Gpi,MRSE,MVAF,u,y] = identificador(U0,Y0,U1,Y1,U2,Y2,Na,Nb,K,R)
2  %
3  %   A partir de dados de entrada e saída, a função IDENTIFICADOR identifica
4  %a melhor matriz de transferência para o processo.
5  %
6  %************************************************************************
7  %
8  %SINTAXE:
9  %
10 %   [Gpi,MRSE,MVAF,u,y] = identificador(U0,Y0,U1,Y1,U2,Y2,Na,Nb,K,R)
11 %
12 %************************************************************************
13 %
14 %ARGUMENTOS DE ENTRADA:
15 %
16 %   U0(nu x N0) = [u1(1:N0)';u2(1:N0)';u3(1:N0)';...;unu(1:N0)'], onde
17 %   u1,...,unu  são os degraus aplicados em cada uma das nu entradas do
18 %   sistema durante o  pré-teste, em ordem cronológica, e N0 é a duração
19 %   total do pré-teste, expressa em tempos de amostragem (k). O primeiro
20 %   degrau (u1) deve ser aplicado a partir do instante k=2.
21 %
22 %   Y0(ny x N0) = [y1(1:N0)';y2(1:N0)';y3(1:N0)';...;yny(1:N0)'], onde
23 %   y1,...,yny são as ny saídas do sistema medidas durante o pré-teste e N0
24 %   é a duração total do pré-teste, expressa em tempos de amostragem (k).
25 %
26 %   U1(nu x N1) = [u1(1:N1)';u2(1:N1)';u3(1:N1)';...;unu(1:N1)'], onde
27 %   u1,...,unu são as sequências de degraus aplicadas em cada uma das nu
28 %   entradas do sistema durante o teste degrau e N1 é a duração total do
29 %   teste, expressa em tempos de amostragem (k). O primeiro degrau deve ser
30 %   aplicado a partir do instante k=2. Se o teste degrau ainda não tiver
31 %   sido realizado, fazer U1=[].
32 %
33 %   Y1(ny x N1) = [y1(1:N1)';y2(1:N1)';y3(1:N1)';...;yny(1:N1)'], onde
34 %   y1,...,yny são as ny saídas do sistema medidas durante o teste degrau e
35 %   N1 é a duração total do teste, expressa em tempos de amostragem (k). Se
36 %   o teste degrau ainda não tiver sido realizado, fazer Y1=[].
```

```
37 %
38 %   U2(nu x N2) = [u1(1:N2)';u2(1:N2)';u3(1:N2)';...;unu(1:N2)'], onde
39 %   u1,...,unu são as sequências de perturbações aplicadas em cada uma das
40 %   nu entradas do sistema durante o teste GBN e N2 é a duração total do
41 %   teste, expressa em tempos de amostragem (k). As primeiras perturbações
42 %   devem ser aplicadas a partir do instante k=2. Se o teste GBN ainda não
43 %   tiver sido realizado, fazer U2=[].
44 %
45 %   Y2(ny x N2) = [y1(1:N2)';y2(1:N2)';y3(1:N2)';...;yny(1:N2)'], onde
46 %   y1,...,yny são as ny saídas do sistema medidas durante o teste GBN e N2
47 %   é a duração total do teste, expressa em tempos de amostragem (k). Se o
48 %   teste GBN ainda não tiver sido realizado, fazer Y2=[].
49 %
50 %   Na = Número inteiro maior ou igual a 1 que determina o número máximo de
51 %   polos (Np) da matriz de transferência. Tem-se que Np = Na*ny, onde ny é
52 %   o número de saídas do sistema.
53 %
54 %   Nb = Número inteiro maior ou igual a 0 que, juntamente com Na,
55 %   determina o número máximo de zeros (Nz) dos elementos da matriz de
56 %   transferência. Tem-se que Nz = Na*(ny-1)+Nb, onde ny é o número de
57 %   saídas do sistema.
58 %
59 %   K = Estrutura formada pelos parâmetros K.inf(ny x nu), K.sup(ny x nu) e
60 %   K.MS(ny x nu), onde K.inf(i,j) e K.sup(i,j) são os limites inferior e
61 %   superior para o ganho estático de yi em relação a uj, respectivamente,
62 %   e K.MS é a matriz de sinais dos ganhos estáticos. Esta estrutura é
63 %   gerada pela função teste_degrau.
64 %
65 %   R = Raio do menor círculo centrado na origem do plano complexo que,
66 %   sabidamente, contém todos os polos da matriz de transferência do
67 %   sistema. Para ser estável, os polos devem estar localizados no interior
68 %   do círculo unitário. R deve ser especificado como um número positivo
69 %   menor ou igual a 1. (Default: R=1)
70 %
71 %*************************************************************************
72 %
73 %ARGUMENTOS DE SAÍDA:
74 %
75 %   Gpi = Matriz de transferência do processo identificada (na variável
76 %   z^-1).
77 %
```

```
%   MRSE = Erro médio quadrático relativo (Mean Relative Squared Error),
%   expresso em porcentagem (%). Quanto menor for este valor, maior será a
%   qualidade da matriz de transferência identificada.
%
%   MVAF = Variância média contabilizada (Mean Variance-Accounted-For),
%   expressa em porcentagem (%). Quanto maior for este valor, maior será a
%   qualidade da matriz de transferência identificada.
%
%   u = Estrutura formada pelos parâmetros u.min(nu x 1) e u.max(nu x 1),
%   onde u.min(j) e u.max(j) são o menor e o maior valor obtidos para a
%   entrada uj, respectivamente. Os valores de u são dados em termos de
%   variáveis-desvio.
%
%   y = Estrutura formada pelos parâmetros y.min(ny x 1) e y.max(ny x 1),
%   onde y.min(i) e y.max(i) são o menor e o maior valor obtidos para a
%   saída yi, respectivamente. Os valores de y são dados em termos de
%   variáveis-desvio.
%
%*************************************************************************
%
%OBSERVAÇÃO:
%
%   Os números atribuídos a Na e Nb devem ser suficientemente grandes, de
%   modo que se forem aumentados ainda mais, não serão observadas mudanças
%   significativas nos valores calculados para MRSE e MVAF.
%
%*************************************************************************
%
%AUTOR: Cristiano Salah Mussoi
%DATA DE CRIAÇÃO: 05/01/2019

%% Definição das variáveis globais:

clear global;
global ny nu Nin Neq der_AT der_BT norma_PHI PHI Y X0;

%% Verificação dos argumentos de entrada:

if nargin<9
    disp('Erro: o argumento de entrada K deve ser fornecido');
    return;
```

```
elseif nargin==9
    if isempty([U0 U1 U2])==1 && isempty([Y0 Y1 Y2])==1
        disp(['Erro: os argumentos de entrada U0, Y0, U1, Y1, U2 e Y2 ',...
            'não podem ser todos vazios']);
        return;
    end
    if isempty(U0)==1 && isempty(Y0)==0 || isempty(U0)==0 && isempty(Y0)==1
        disp('Erro: U0 e Y0 devem ser ambos vazios ou ambos não vazios');
        return;
    end
    if isempty(U1)==1 && isempty(Y1)==0 || isempty(U1)==0 && isempty(Y1)==1
        disp('Erro: U1 e Y1 devem ser ambos vazios ou ambos não vazios');
        return;
    end
    if isempty(U2)==1 && isempty(Y2)==0 || isempty(U2)==0 && isempty(Y2)==1
        disp('Erro: U2 e Y2 devem ser ambos vazios ou ambos não vazios');
        return;
    end
    if isempty(Na)==1 || isscalar(Na)==0 || Na<=0 || Na~=round(Na)
        disp('Erro: valor de Na inválido');
        return;
    end
    if isempty(Nb)==1 || isscalar(Nb)==0 || Nb<0 || Nb~=round(Nb)
        disp('Erro: valor de Nb inválido');
        return;
    end
    if isempty(K)==1
        disp('Erro: K não pode ser vazio');
        return;
    end
    R = 1;
elseif nargin==10
    if isempty([U0 U1 U2])==1 && isempty([Y0 Y1 Y2])==1
        disp(['Erro: os argumentos de entrada U0, Y0, U1, Y1, U2 e Y2 ',...
            'não podem ser todos vazios']);
        return;
    end
    if isempty(U0)==1 && isempty(Y0)==0 || isempty(U0)==0 && isempty(Y0)==1
        disp('Erro: U0 e Y0 devem ser ambos vazios ou ambos não vazios');
        return;
    end
```

```
160     if isempty(U1)==1 && isempty(Y1)==0 || isempty(U1)==0 && isempty(Y1)==1
161         disp('Erro: U1 e Y1 devem ser ambos vazios ou ambos não vazios');
162         return;
163     end
164     if isempty(U2)==1 && isempty(Y2)==0 || isempty(U2)==0 && isempty(Y2)==1
165         disp('Erro: U2 e Y2 devem ser ambos vazios ou ambos não vazios');
166         return;
167     end
168     if isempty(Na)==1 || isscalar(Na)==0 || Na<=0 || Na~=round(Na)
169         disp('Erro: valor de Na inválido');
170         return;
171     end
172     if isempty(Nb)==1 || isscalar(Nb)==0 || Nb<0 || Nb~=round(Nb)
173         disp('Erro: valor de Nb inválido');
174         return;
175     end
176     if isempty(K)==1
177         disp('Erro: K não pode ser vazio');
178         return;
179     end
180     if isempty(R)==1
181         R = 1;
182     elseif isscalar(R)==0 || R<=0 || R>1
183         disp('Erro: valor de R inválido');
184         return;
185     end
186 else
187     disp('Erro: número de argumentos de entrada inválido');
188     return;
189 end
190 
191 %% Verificação dos argumentos de saída:
192 
193 if nargout~=5 %O número de argumentos de saída deve ser igual a 5
194     disp('Erro: devem ser fornecidos 5 argumentos de saída para a função');
195     return;
196 end
197 
198 %% Obtenção de U0 e Y0 em termos de variáveis-desvio:
199 
200 L = zeros(size(U0,1),1); %Alocação de memória para o vetor L
```

```
201
202 if isempty(U0)==0
203     for i=1:size(U0,1)
204         L(i) = 0.5*(max(U0(i,:)) + min(U0(i,:))); %Média entre o maior e o
                menor valor de ui
205         if U0(i,1)>L(i) %Caso o degrau em ui tenha sido aplicado em k=1
206             disp(['Erro: no pré-teste, o degrau em u',num2str(i),...
207                 ' deve ser aplicado a partir do instante k=2']);
208             return;
209         else
210             U0(i,:) = U0(i,:) - U0(i,1); %Cálculo de ui-desvio
211         end
212     end
213 end
214
215 if isempty(Y0)==0
216     for i=1:size(Y0,1)
217         Y0(i,:) = Y0(i,:) - Y0(i,1); %Cálculo de yi-desvio
218     end
219 end
220
221 %% Obtenção de U1 e Y1 em termos de variáveis-desvio:
222
223 L0 = zeros(size(U1,1),1); %Alocação de memória para o vetor L0
224 LS = zeros(size(U1,1),1); %Alocação de memória para o vetor LS
225 LI = zeros(size(U1,1),1); %Alocação de memória para o vetor LI
226
227 if isempty(U1)==0
228     for i=1:size(U1,1)
229         L0(i) = 0.5*(max(U1(i,:)) + min(U1(i,:))); %Média entre o maior e o
                menor valor de ui
230         LS(i) = 0.5*(max(U1(i,:)) + L0(i)); %Média entre o maior valor de ui
                 e L0(i)
231         LI(i) = 0.5*(min(U1(i,:)) + L0(i)); %Média entre o menor valor de ui
                 e L0(i)
232         if U1(i,1)>LS(i) || U1(i,1)<LI(i) %Caso o primeiro degrau em ui
                tenha sido aplicado em k=1
233             disp(['Erro: no teste degrau, o primeiro degrau em u',...
234                 num2str(i),' deve ser aplicado a partir do instante k=2']);
235             return;
236         else
```

```
237             U1(i,:) = U1(i,:) - U1(i,1); %Cálculo de ui-desvio
238         end
239     end
240 end
241
242 if isempty(Y1)==0
243     for i=1:size(Y1,1)
244         Y1(i,:) = Y1(i,:) - Y1(i,1); %Cálculo de yi-desvio
245     end
246 end
247
248 %% Obtenção de U2 e Y2 em termos de variáveis-desvio:
249
250 if isempty(U2)==0
251     for i=1:size(U2,1)
252         L0(i) = 0.5*(max(U2(i,:)) + min(U2(i,:))); %Média entre o maior e o
              menor valor de ui
253         LS(i) = 0.5*(max(U2(i,:)) + L0(i)); %Média entre o maior valor de ui
               e L0(i)
254         LI(i) = 0.5*(min(U2(i,:)) + L0(i)); %Média entre o menor valor de ui
               e L0(i)
255         if U2(i,1)>LS(i) || U2(i,1)<LI(i) %Caso a primeira perturbação em ui
               tenha sido aplicada em k=1
256             disp(['Erro: no teste GBN, a primeira perturbação em u',...
257                 num2str(i),' deve ser aplicada a partir do instante k=2']);
258             return;
259         else
260             U2(i,:) = U2(i,:) - U2(i,1); %Cálculo de ui-desvio
261         end
262     end
263 end
264
265 if isempty(Y2)==0
266     for i=1:size(Y2,1)
267         Y2(i,:) = Y2(i,:) - Y2(i,1); %Cálculo de yi-desvio
268     end
269 end
270
271 %% Obtenção das matrizes U e Y:
272
273 U = [U0 U1 U2]; %Matriz das entradas
```

```
274 Y = [Y0 Y1 Y2]; %Matriz das saídas
275
276 %% Verificação das dimensões de U e Y:
277
278 if size(U,2)==size(Y,2) %Caso U e Y tenham o mesmo número de colunas
279     nu = size(U,1); %Número de variáveis de entrada
280     ny = size(Y,1); %Número de variáveis de saída
281     N = size(Y,2); %Número total de tempos de amostragem
282 else
283     disp('Erro: U e Y devem ter o mesmo número de colunas');
284     return;
285 end
286
287 %% Obtenção das estruturas u e y:
288
289 umin = zeros(nu,1); %Alocação de memória para o vetor umin
290 umax = zeros(nu,1); %Alocação de memória para o vetor umax
291 for i=1:nu
292     umin(i) = min(U(i,:)); %Menor valor obtido para a entrada ui
293     umax(i) = max(U(i,:)); %Maior valor obtido para a entrada ui
294 end
295 u = struct; %Inicialização da estrutura u
296 u.min = umin;
297 u.max = umax;
298
299 ymin = zeros(ny,1); %Alocação de memória para o vetor ymin
300 ymax = zeros(ny,1); %Alocação de memória para o vetor ymax
301 for i=1:ny
302     ymin(i) = min(Y(i,:)); %Menor valor obtido para a saída yi
303     ymax(i) = max(Y(i,:)); %Maior valor obtido para a saída yi
304 end
305 y = struct; %Inicialização da estrutura y
306 y.min = ymin;
307 y.max = ymax;
308
309 %% Cálculo de Nin e Neq:
310
311 Nin = 1; %Número de restrições de desigualdade
312 Neq = 0; %Número de restrições de igualdade
313 for i=1:ny
314     for j=1:nu
```

```
315         if K.inf(i,j)<K.sup(i,j)
316             Nin = Nin+2; %Atualização do valor de Nin
317         else %K.inf(i,j)==K.sup(i,j)
318             Neq = Neq+1; %Atualização do valor de Neq
319         end
320     end
321 end
322
323 %% Cálculo de der_AT e der_BT:
324
325 der_AT = cell(Na,Na*ny+(Nb+1)*nu,ny); %Alocação de memória para a célula
        der_AT
326 Z1 = zeros(ny,ny,Na,Na*ny+(Nb+1)*nu,ny); %Alocação de memória para a matriz
        Z1
327 for k=1:Na
328     for i=(k-1)*ny+1:k*ny
329         q = i-(k-1)*ny; %q varia de 1 até ny
330         for j=1:ny
331             Z1(q,j,k,i,j) = 1; %Elemento (q,j) da derivada primeira de Ak'
                    em relação a xij
332         end
333     end
334 end
335 for k=1:Na
336     for i=1:Na*ny+(Nb+1)*nu
337         for j=1:ny
338             der_AT{k,i,j} = sparse(Z1(:,:,k,i,j)); %Derivada primeira de Ak'
                     em relação a xij (matriz esparsa)
339         end
340     end
341 end
342
343 der_BT = cell(Nb+1,Na*ny+(Nb+1)*nu,ny); %Alocação de memória para a célula
        der_BT
344 Z2 = zeros(nu,ny,Nb+1,Na*ny+(Nb+1)*nu,ny); %Alocação de memória para a
        matriz Z2
345 for k=0:Nb
346     for i=Na*ny+k*nu+1:Na*ny+(k+1)*nu
347         q = i-Na*ny-k*nu; %q varia de 1 até nu
348         for j=1:ny
349             Z2(q,j,k+1,i,j) = 1; %Elemento (q,j) da derivada primeira de Bk'
```

```
                em relação a xij
350         end
351     end
352 end
353 for k=0:Nb
354     for i=1:Na*ny+(Nb+1)*nu
355         for j=1:ny
356             der_BT{k+1,i,j} = sparse(Z2(:,:,k+1,i,j)); %Derivada primeira de
                    Bk' em relação a xij (matriz esparsa)
357         end
358     end
359 end
360
361 %% Cálculo de PHI e da nova matriz Y:
362
363 k0 = max(Na,Nb)+1; %Tempo de amostragem inicial
364 if k0>N %max(Na,Nb)>(N-1)
365     disp(['Erro: para este problema de identificação, Na e Nb ',...
366         'não podem ser maiores que ',num2str(N-1)]);
367     return;
368 end
369
370 PHI = zeros(Na*ny+(Nb+1)*nu,N-k0+1);
371 for k=k0:N
372     for i=1:Na
373         PHI((i-1)*ny+1:i*ny,k-k0+1) = Y(:,k-i);
374     end
375     for i=0:Nb
376         PHI(Na*ny+i*nu+1:Na*ny+(i+1)*nu,k-k0+1) = U(:,k-i);
377     end
378 end
379 PHI = PHI'; %Matriz de regressão (PHI)
380 norma_PHI = zeros(Na*ny+(Nb+1)*nu,1);
381 for j=1:Na*ny+(Nb+1)*nu
382     PHI(:,j) = PHI(:,j) - mean(PHI(:,j)); %Centralização da j-ésima coluna
        de PHI
383     norma_PHI(j) = norm(PHI(:,j)); %Norma da j-ésima coluna de PHI
384     if norma_PHI(j)~=0 %Caso a j-ésima coluna de PHI não seja o vetor nulo
385         PHI(:,j) = PHI(:,j)/norma_PHI(j); %A j-ésima coluna de PHI é
                normalizada (Matriz X original -> Matriz X modificada)
386     end
```

```
387 end
388
389 Y = Y(:,k0:N);
390 Y = Y'; %Nova matriz Y
391 for j=1:ny
392     Y(:,j) = Y(:,j) - mean(Y(:,j)); %Centralização da j-ésima coluna de Y
393 end
394
395 %% Cálculo de X0:
396
397 %X0 = estimativa inicial dos parâmetros do modelo
398 if rcond(PHI'*PHI)>eps %Caso a matriz (PHI'*PHI) esteja bem condicionada
399     X0 = (PHI'*PHI)\(PHI'*Y); %Mínimos quadrados ordinários (Ordinary Least
        Squares - OLS)
400 else
401     options = optimset('Algorithm','interior-point',...
402         'SubproblemAlgorithm','ldl-factorization','Display','off',...
403         'GradObj','on','Hessian','user-supplied',...
404         'HessFcn',@(omega,lambda)GCV_hess(omega,lambda,Na,Nb),...
405         'TolFun',1e-6,'TolCon',1e-6,'TolX',eps,...
406         'MaxIter',Inf,'MaxFunEvals',Inf);
407     omega = fmincon(@(omega)GCV(omega,Na,Nb),1/size(Y,1),...
408         [],[],[],[],0,Inf,[],options);
409     omega = omega*size(Y,1); %Parâmetro de regularização
410     X0 = (omega*eye(Na*ny+(Nb+1)*nu)+PHI'*PHI)\(PHI'*Y); %Regularização de
        Tikhonov
411 end
412
413 %% Cálculo de X1:
414
415 warning('off','all');
416 X1 = lasso_adaptativo(Na,Nb); %Regularização do tipo LASSO adaptativo -> X1
417 warning('on','all');
418 X1 = X1(:); %A matriz X1 é convertida em um vetor-coluna
419 Id = find(X1); %Vetor dos índices dos componentes não nulos de X1
420 X1 = X1(Id); %Os componentes nulos de X1 são apagados
421
422 %% Cálculo de X2:
423
424 options = optimset('Algorithm','interior-point',...
425     'SubproblemAlgorithm','ldl-factorization','Display','iter',...
```

```
    'GradObj','on','GradConstr','on','Hessian','user-supplied',...
    'HessFcn',@(X,lambda)hess(X,lambda,Id,Na,Nb,K),...
    'TolFun',1e-6,'TolCon',1e-6,'TolX',1e-10,...
    'MaxIter',Inf,'MaxFunEvals',Inf);

[X2,fval,exitflag,output] = fmincon(@(X)obj(X,Id,Na,Nb),X1,...
    [],[],[],[],[],[],@(X)restr(X,Id,Na,Nb,K,R),options);
delta = output.constrviolation; %Maior violação absoluta das restrições

vetor = zeros((Na*ny+(Nb+1)*nu)*ny,1);
vetor(Id) = X2;
X2 = vetor; %Estimativa final dos parâmetros do modelo (vetor-coluna)

%% Cálculo de Gpi:

z = tf('z',-1); %Definição da variável z

X = X2;
X = reshape(X,[Na*ny+(Nb+1)*nu ny]); %Reconstrução da matriz X
for i=1:Na*ny+(Nb+1)*nu
    if norma_PHI(i)~=0
        X(i,:) = X(i,:)/norma_PHI(i); %Recuperação da matriz X original
    end
end
XT = X'; %Transposição de X

A = zeros(ny,ny,Na);
for i=1:Na
    A(:,:,i) = XT(:,(i-1)*ny+1:i*ny); %Obtenção das matrizes Ai
end

B = zeros(ny,nu,Nb+1);
for i=0:Nb
    B(:,:,i+1) = XT(:,Na*ny+i*nu+1:Na*ny+(i+1)*nu); %Obtenção das matrizes
        Bi
end

Az = eye(ny);
for i=1:Na
    Az = Az - A(:,:,i)*z^-i; %Cálculo da matriz polinomial A(z)
end
```

```
466 set(Az,'Variable','z^-1'); %A(z) -> A(z^-1)
467
468 Bz = zeros(ny,nu);
469 for i=0:Nb
470     Bz = Bz + B(:,:,i+1)*z^-i; %Cálculo da matriz polinomial B(z)
471 end
472 set(Bz,'Variable','z^-1'); %B(z) -> B(z^-1)
473
474 Gpi = Az\Bz; %Matriz de transferência do processo identificada
475 [Num,Den] = tfdata(Gpi); %Obtenção das células Num (numeradores de Gpi) e
        Den (denominadores de Gpi)
476 for i=1:ny
477     for j=1:nu
478         if isscalar(Den{i,j})==0 %Caso o denominador de Gpi(i,j) seja um
                polinômio de grau maior ou igual a 1
479             n = 100; %Inicialização da variável n
480             ganho = Inf; %Inicialização da variável ganho
481             max_rho = Inf; %Inicialização da variável max_rho
482             while ganho<K.inf(i,j)-delta || ganho>K.sup(i,j)+delta ||...
483                     max_rho>R^2+delta %Caso as restrições sobre Gpi(i,j) não
                         sejam satisfeitas
484                 n=10*n; %n=1000 na primeira iteração do loop while
485                 D = Den{i,j}; %Novo denominador de Gpi(i,j)
486                 for k=1:length(D)
487                     if abs(D(k))<max(abs(D))/n
488                         D(k) = 0; %Anulação dos menores componentes de D
489                     end
490                 end
491                 if D==Den{i,j} %Caso nenhum componente de D tenha sido
                        anulado (n ficou muito grande)
492                     break; %Sair do loop while
493                 end
494                 ganho = sum(Num{i,j})/sum(D); %Novo ganho estático de Gpi(i,
                        j)
495                 r = roots(D); %Novos polos de Gpi(i,j)
496                 rho = r.*conj(r); %abs(r).^2
497                 max_rho = max(rho); %max(abs(r).^2)
498             end
499             Den{i,j} = D; %Atualização de Den{i,j}
500         end
501         if isscalar(Num{i,j})==0 %Caso o numerador de Gpi(i,j) seja um
```

```
            polinômio de grau maior ou igual a 1
502             n = 100; %Inicialização da variável n
503             ganho = Inf; %Inicialização da variável ganho
504             while ganho<K.inf(i,j)-delta || ganho>K.sup(i,j)+delta %Caso as
                    restrições sobre o ganho estático de Gpi(i,j) não sejam
                    satisfeitas
505                 n = 10*n; %n=1000 na primeira iteração do loop while
506                 N = Num{i,j}; %Novo numerador de Gpi(i,j)
507                 for k=1:length(N)
508                     if abs(N(k))<max(abs(N))/n
509                         N(k) = 0; %Anulação dos menores componentes de N
510                     end
511                 end
512                 if N==Num{i,j} %Caso nenhum componente de N tenha sido
                        anulado (n ficou muito grande)
513                     break; %Sair do loop while
514                 end
515                 ganho = sum(N)/sum(Den{i,j}); %Novo ganho estático de Gpi(i,
                        j)
516             end
517             Num{i,j} = N; %Atualização de Num{i,j}
518         end
519     end
520 end
521 set(Gpi,'num',Num,'den',Den); %Atualização de Gpi
522
523 %% Cálculo de MRSE e MVAF:
524
525 Ym = [Y0 Y1 Y2]; %Matriz das saídas medidas
526 Yc = (lsim(Gpi,U'))'; %Matriz das saídas calculadas
527 E = Ym - Yc; %Matriz de erros
528
529 MRSE = 0;
530 for i=1:ny
531     MRSE = MRSE + sqrt(sum(E(i,:).^2)/sum(Ym(i,:).^2));
532 end
533 MRSE = (MRSE/ny)*100; %Erro médio quadrático relativo (Mean Relative Squared
        Error - MRSE)
534
535 MVAF = 0;
536 for i=1:ny
```

```
537     MVAF = MVAF + 1 - var(E(i,:))/var(Ym(i,:));
538 end
539 MVAF = (MVAF/ny)*100; %Variância média contabilizada (Mean Variance-
        Accounted-For - MVAF)
540
541 %% Gráficos:
542
543 [status,msg] = mkdir('identificação_resultados'); %Criação da pasta
        identificação_resultados
544 if status==0
545     disp('Erro: não foi possível criar a pasta identificação_resultados');
546     return;
547 elseif status==1 && isempty(msg)==0 %A pasta identificação_resultados já
        existe
548     warning('off','all');
549     rmdir identificação_resultados s; %Remoção da pasta existente e de todo
            o seu conteúdo
550     mkdir identificação_resultados; %Criação de uma nova pasta
551     warning('on','all');
552 end
553
554 figure(5)
555 stairs(U');
556 hold on;
557 plot(zeros(size(U',1),1),'k');
558 hold off;
559 xlabel('k'); %Tempo de amostragem
560 ylabel('u(k)'); %Variáveis-desvio de entrada
561 xlim([1,size(U',1)]);
562 ylim([-1.1*max(max(abs(U))),1.1*max(max(abs(U)))]);
563 set(gca,'XTick',unique([1,get(gca,'XTick')]));
564 leg = cell(1,nu); %Alocação de memória para a célula leg
565 for i=1:nu
566     leg{i}=['u',num2str(i)]; %Legendas
567 end
568 legend(leg,'Orientation','vertical','Location','northeastoutside');
569 saveas(gcf,fullfile('identificação_resultados','u.fig')); %Salva a figura na
         pasta identificação_resultados com o nome u.fig
570 close; %Fecha a figura
571
572 for i=1:ny
```

```
573     figure(5)
574     plot(1:size(Ym,2),Ym(i,:),'b',1:size(Yc,2),Yc(i,:),'r');
575     xlabel('k'); %Tempo de amostragem
576     ylabel(['y_',num2str(i),'(k)']); %Variável-desvio de saída yi
577     xlim([1,size(Ym,2)]);
578     set(gca,'XTick',unique([1,get(gca,'XTick')]));
579     legend('medido','calculado','Orientation','vertical',...
580         'Location','northeast');
581     saveas(gcf,fullfile('identificação_resultados',...
582         ['y',num2str(i),'.fig'])); %Salva a figura na pasta
            identificação_resultados com o nome yi.fig
583     close; %Fecha a figura
584 end
585
586 end
587
588 %% Validação Cruzada Generalizada (Generalized Cross-Validation - GCV):
589
590 function [f,g] = GCV(omega,Na,Nb)
591
592 global ny nu PHI Y M2 M3 der_M2 der_M3;
593
594 N = size(Y,1); %Número de tempos de amostragem
595
596 M1 = (PHI/(PHI'*PHI + N*omega*eye(Na*ny+(Nb+1)*nu)))*PHI'; %Matriz M1
597 M2 = (eye(N)-M1)*Y; %Matriz M2
598 M3 = eye(N)-M1; %Matriz M3
599
600 f = N*trace(M2'*M2)/trace(M3)^2; %Critério de seleção para ômega (f deve ser
        minimizado)
601
602 if nargout>1
603
604     der_M1 = -N*PHI*inv(PHI'*PHI + N*omega*eye(Na*ny+(Nb+1)*nu))^2*PHI'; %
          Derivada primeira de M1 em relação a ômega
605     der_M2 = -der_M1*Y; %Derivada primeira de M2 em relação a ômega
606     der_M3 = -der_M1; %Derivada primeira de M3 em relação a ômega
607
608     E1 = trace(M2'*der_M2)/trace(M3)^2; %Escalar E1
609     E2 = trace(M2'*M2)*trace(der_M3)/trace(M3)^3; %Escalar E2
610
```

```
611     g = 2*N*(E1-E2); %Gradiente de f (derivada primeira de f em relação a
            ômega)
612
613 end
614
615 end
616
617 function H = GCV_hess(omega,lambda,Na,Nb) %#ok<INUSL>
618
619 global ny nu PHI Y M2 M3 der_M2 der_M3;
620
621 N = size(Y,1); %Número de tempos de amostragem
622
623 der2_M1 = 2*N^2*PHI*inv(PHI'*PHI + N*omega*eye(Na*ny+(Nb+1)*nu))^3*PHI'; %
        Derivada segunda de M1 em relação a ômega
624 der2_M2 = -der2_M1*Y; %Derivada segunda de M2 em relação a ômega
625 der2_M3 = -der2_M1; %Derivada segunda de M3 em relação a ômega
626
627 der_E1 = (trace(M3)^2*trace(M2'*der2_M2+der_M2'*der_M2) -...
628     2*trace(M2'*der_M2)*trace(M3)*trace(der_M3))/trace(M3)^4; %Derivada
            primeira de E1 em relação a ômega
629
630 der_E2 = (trace(M3)^3*(trace(M2'*M2)*trace(der2_M3)+...
631     2*trace(M2'*der_M2)*trace(der_M3)) -...
632     3*trace(M2'*M2)*trace(M3)^2*trace(der_M3)^2)/trace(M3)^6; %Derivada
            primeira de E2 em relação a ômega
633
634 H = 2*N*(der_E1-der_E2); %Hessiana de f (derivada segunda de f em relação a
        ômega)
635
636 end
637
638 %% LASSO adaptativo:
639
640 function X = lasso_adaptativo(Na,Nb)
641
642 global ny nu PHI Y X0;
643
644 X = X0; %Estimativa inicial para a matriz X (matriz de parâmetros)
645 W = abs(X0).^(-1); %Matriz de pesos
646
```

```
N = size(Y,1); %Número de tempos de amostragem
k = log(Na*ny+(Nb+1)*nu)/log(N); %Escalar k
sigma = max(0,1-1/(2*k)); %Escalar sigma

for i=1:ny %Para cada coluna da matriz Y

    y = Y(:,i); %y=yi
    PHIw = zeros(size(PHI)); %Alocação de memória para a matriz PHIw
    for j=1:Na*ny+(Nb+1)*nu
        PHIw(:,j) = PHI(:,j)/W(j,i); %Construção da matriz PHIw (LASSO
            adaptativo)
    end
    if PHIw==zeros(size(PHI)) %Caso PHIw seja a matriz nula
        continue; %Passar para a próxima iteração (i->i+1)
    end

    xs = sign(X0(:,i)); %Vetor de parâmetros selecionado
    e = y-PHIw*xs; %Vetor de erros
    S = sum(e.^2); %Soma do quadrado dos erros
    csi = nchoosek(Na*ny+(Nb+1)*nu,nnz(xs)); %Escalar csi
    EBIC_min = N*log(S/N) + (nnz(xs)+1)*log(N) + 2*sigma*log(csi); %Menor
        valor obtido para o Critério de Informação Bayesiano Estendido (EBIC
        )

    x = zeros(Na*ny+(Nb+1)*nu,1); %Alocação de memória para o vetor de
        parâmetros x
    nA = 0; %Número de elementos no conjunto A (inicialmente, nA=0)
    j_til = []; %Elemento que deve ser retirado do conjunto A (inicialmente,
         j_til=[])

    while nA<Na*ny+(Nb+1)*nu %Enquanto o conjunto A não estiver completo

        mu = PHIw*x; %Estimativa para o vetor y
        c = PHIw'*(y-mu); %Vetor de correlações. O j-ésimo componente de c é
             igual ao produto escalar da j-ésima coluna de PHIw com o vetor
            y-mu
        C = max(abs(c)); %Máxima correlação absoluta
        if isempty(j_til)==1 %Caso não seja necessário remover nenhum
            elemento de A
            nA = nA+1; %Um novo elemento é colocado no conjunto A
            [v,Id] = sort(abs(c),'descend');
```

```
680             A = Id(1:nA); %Conjunto A, formado pelos índices das colunas de
                    PHIw cuja correlação absoluta com y-mu é C
681             Ac = Id(nA+1:end); %Conjunto A complementar (Ac), formado pelos
                    índices das outras colunas de PHIw
682         else %Caso seja necessário remover algum elemento de A
683             A(A==j_til) = []; %O elemento j_til é retirado do conjunto A
684             Ac = union(Ac,j_til);%j_til é colocado no conjunto Ac
685             j_til = [];
686         end
687
688         s = sign(c(A)); %Vetor dos sinais das correlações (considera-se
                somente as colunas de PHIw cujo índice está em A)
689         PHI_A = PHIw(:,A');
690         for j=1:nA
691             PHI_A(:,j) = s(j)*PHI_A(:,j); %Construção da matriz PHI_A
692         end
693
694         GA = PHI_A'*PHI_A; %Matriz GA
695         if rcond(GA)<=eps %Caso a matriz GA esteja mal condicionada
696             break; %Sair do loop while
697         end
698         umA = ones(nA,1); %Vetor umA
699         AA = (umA'*(GA\umA))^(-1/2); %Escalar AA
700         wA = AA*(GA\umA);%Vetor wA
701         uA = PHI_A*wA; %Vetor uA = vetor unitário equiangular às colunas de
                PHI_A
702         a = PHIw'*uA; %Vetor a
703
704         if isempty(Ac)==0 %Caso esta não seja a última iteração do loop
                while
705             v1 = [(C-c(Ac))./(AA-a(Ac)); (C+c(Ac))./(AA+a(Ac))];
706             gamma = min(v1(v1>0)); %Deslocamento de gamma unidades na
                    direção e sentido de uA (passo do algoritmo)
707             if isempty(gamma)==1 %Caso gamma seja vazio
708                 break; %Sair do loop while
709             end
710         else %Caso esta seja a última iteração do loop while
711             gamma = C/AA;
712         end
713         d = zeros(Na*ny+(Nb+1)*nu,1);
714         d(A) = s.*wA; %Construção do vetor d
```

```
715         v2 = -x./d;
716         gamma_til = min(v2(v2>0)); %gamma_til = deslocamento necessário para
                zerar o componente xj_til do vetor x
717         if isempty(gamma_til)==0 && gamma_til<gamma %Caso gamma_til seja
                inferior a gamma
718             nA = nA-1;
719             j_til = find(v2==gamma_til); %O elemento j_til deve ser retirado
                    do conjunto A
720             gamma = gamma_til; %O passo do algoritmo é interrompido após um
                    deslocamento de gamma_til unidades
721         end
722
723         x = x + gamma*d; %Estimativa para x
724         e = y-PHIw*x; %Vetor de erros
725         S = sum(e.^2); %Soma do quadrado dos erros
726         csi = nchoosek(Na*ny+(Nb+1)*nu,nA); %Escalar csi
727         EBIC = N*log(S/N) + (nA+1)*log(N) + 2*sigma*log(csi); %EBIC obtido
                para o modelo cujo vetor de parâmetros é x
728
729         if EBIC < EBIC_min %Caso EBIC seja inferior a EBIC_min
730             EBIC_min = EBIC; %Atualização de EBIC_min
731             xs = x; %Atualização de xs
732         end
733
734     end
735
736     X(:,i) = xs./W(:,i); %Obtenção da i-ésima coluna da matriz X
737     for n=1:Na*ny+(Nb+1)*nu
738         if abs(X(n,i)) < max(abs(X(:,i)))/1000
739             X(n,i) = 0; %Anulação dos números muito pequenos da i-ésima
                    coluna de X (este procedimento evita a superparametrização)
740         end
741     end
742
743 end
744
745 end
746
747 %% Função objetivo:
748
749 function [f,g] = obj(X,Id,Na,Nb)
```

```
750
751 global ny nu PHI Y;
752
753 vetor = zeros((Na*ny+(Nb+1)*nu)*ny,1);
754 vetor(Id) = X;
755 X = vetor;
756 X = reshape(X,[Na*ny+(Nb+1)*nu ny]); %Reconstrução da matriz X
757
758 E = Y - PHI*X; %Matriz de erros (E)
759 f = sum(sum(E.^2)); %Função objetivo (norma de Frobenius ao quadrado da
       matriz E)
760
761 if nargout>1
762
763     g = -2*PHI'*(Y-PHI*X);
764     g = g(:);
765     g = g(Id); %Gradiente da função objetivo
766
767 end
768
769 end
770
771 %% Restrições:
772
773 function [cin,ceq,gcin,gceq] = restr(X,Id,Na,Nb,K,R)
774
775 global ny nu Nin Neq der_AT der_BT norma_PHI S A Ar adj_Ar...
776     r der_r rho der_rho a w Sa der_Sa A1 B1;
777
778 vetor = zeros((Na*ny+(Nb+1)*nu)*ny,1);
779 vetor(Id) = X;
780 X = vetor;
781 X = reshape(X,[Na*ny+(Nb+1)*nu ny]); %Reconstrução da matriz X
782
783 lin = zeros(1,Na*ny+(Nb+1)*nu); %Alocação de memória para o vetor-linha lin
784 col = cell(1,Na*ny+(Nb+1)*nu); %Alocação de memória para a célula col
785 for i=1:Na*ny+(Nb+1)*nu
786     if norm(X(i,:))~=0 %Caso a i-ésima linha de X possua elementos não nulos
787         lin(i) = i; %Armazenar o índice da linha
788         vetor = zeros(1,ny);
789         for j=1:ny
```

```
790             if X(i,j)~=0 %Caso X(i,j) seja um elemento não nulo
791                 vetor(j) = j; %Armazenar o índice da coluna
792             end
793         end
794         col{i} = vetor(vetor~=0); %Vetor-linha dos índices das colunas dos
                elementos não nulos da i-ésima linha de X
795     end
796 end
797 lin = lin(lin~=0); %Vetor-linha dos índices das linhas não nulas de X
798
799 for i=lin
800     X(i,:) = X(i,:)/norma_PHI(i); %Recuperação da matriz X original
801 end
802 XT = X'; %Transposição de X
803
804 A = zeros(ny,ny,Na);
805 for i=1:Na
806     A(:,:,i) = XT(:,(i-1)*ny+1:i*ny); %Obtenção das matrizes Ai
807 end
808
809 B = zeros(ny,nu,Nb+1);
810 for i=0:Nb
811     B(:,:,i+1) = XT(:,Na*ny+i*nu+1:Na*ny+(i+1)*nu); %Obtenção das matrizes
            Bi
812 end
813
814 cin = zeros(Nin,1); %Alocação de memória para o vetor cin (vetor das
    restrições de desigualdade)
815 ceq = zeros(Neq,1); %Alocação de memória para o vetor ceq (vetor das
    restrições de igualdade)
816
817 P = zeros(ny,(Na+1)*ny); %Alocação de memória para a matriz P
818 P(:,1:ny) = eye(ny);
819 for i=1:Na
820     P(:,i*ny+1:(i+1)*ny) = -A(:,:,i); %P=[I -A1 -A2 ... -ANa] -> Conjunto
            dos coeficientes da matriz polinomial A(z)
821 end
822 P = reshape(P,[ny^2 Na+1]); %P=[I(:) -A1(:) -A2(:) ... -ANa(:)]
823 Ndet = Na*ny; %Grau máximo do determinante de A(z)
824 Q = fft(P,Ndet+1,2); %Q=[Q1(:) Q2(:) ... QNdet+1(:)] -> Transformada de
    Fourier Discreta de P (calculada ao longo de suas linhas)
```

```
825
826 D = zeros(1,Ndet+1); %Alocação de memória para o vetor-linha D
827 for i=1:Ndet+1
828     if i<=ceil(Ndet/2)+1
829         D(i) = det(reshape(Q(:,i),[ny ny])); %Qi(:) -> Qi(ny x ny) -> D(i)=
                det(Qi)
830     else
831         D(i) = conj(D(Ndet+3-i)); %Propriedade da Transformada de Fourier
                Discreta para matrizes polinomiais com coeficientes reais
832     end
833 end
834 det_Az = ifft(D); %Transformada de Fourier inversa de D -> det_Az = vetor-
    linha dos coeficientes do determinante de A(z)
835 for i=1:Ndet+1
836     if abs(det_Az(i))<eps
837         det_Az(i) = 0; %Anulação dos componentes muito pequenos de det_Az
838     end
839 end
840 det_Az = det_Az(1:find(det_Az,1,'last')); %Eliminação dos componentes nulos
    finais de det_Az
841
842 r = roots(det_Az); %Polos da matriz de transferência do sistema
843 if isempty(r)==0 %Caso o determinante de A(z) seja um polinômio de grau
    maior ou igual a 1
844     rho = r.*conj(r); %rho=abs(r).^2
845     a = 1e6; %Escalar a
846     b = a*max(rho); %Escalar b
847     w = exp(a*rho-b-log(sum(exp(a*rho-b)))); %Vetor-linha de pesos (a e b
            são escolhidos de modo a evitar o "overflow" no cálculo de w)
848     Sa = w'*rho; %Aproximação diferenciável da função max(rho)
849     cin(1) = Sa - R^2; %Sa<=R^2 (restrição de desigualdade sobre a
            localização dos polos)
850 end
851
852 A1 = eye(ny);
853 for i=1:Na
854     A1 = A1 - A(:,:,i); %Matriz A(z) calculada em z=1
855 end
856
857 B1 = zeros(ny,nu);
858 for i=0:Nb
```

```
859     B1 = B1 + B(:,:,i+1); %Matriz B(z) calculada em z=1
860 end
861
862 Kp = A1\B1; %Matriz de ganhos estáticos do processo
863
864 in = 2; %Índice da próxima restrição de desigualdade
865 eq = 1; %Índice da próxima restrição de igualdade
866 for i=1:ny
867     for j=1:nu
868         if K.inf(i,j)<K.sup(i,j) %Há 2 restrições de desigualdade sobre Kp(i
                ,j)
869             cin(in) = K.inf(i,j)-Kp(i,j); %Kp(i,j)>=K.inf(i,j)
870             cin(in+1) = Kp(i,j)-K.sup(i,j); %Kp(i,j)<=K.sup(i,j)
871             in = in+2; %Atualização de in
872         else %K.inf(i,j)==K.sup(i,j) -> Há 1 restrição de igualdade sobre Kp
                (i,j)
873             ceq(eq) = Kp(i,j)-K.sup(i,j); %Kp(i,j)==K.sup(i,j)
874             eq = eq+1; %Atualização de eq
875         end
876     end
877 end
878
879 if nargout>2
880
881     gcin = zeros(Na*ny+(Nb+1)*nu,ny,Nin); %Alocação de memória para a matriz
            gcin (gradiente das restrições de desigualdade)
882     gceq = zeros(Na*ny+(Nb+1)*nu,ny,Neq); %Alocação de memória para a matriz
            gceq (gradiente das restrições de igualdade)
883
884     if isempty(r)==0 %Caso o determinante de A(z) seja um polinômio de grau
            maior ou igual a 1
885
886         Ar = zeros(ny,ny,length(r));
887         for n=1:length(r)
888             Ar(:,:,n) = eye(ny);
889             for k=1:Na
890                 Ar(:,:,n) = Ar(:,:,n) - A(:,:,k)*r(n)^-k; %Matriz A(z)
                        calculada em z=r(n)
891             end
892         end
893
```

```
894         adj_Ar = zeros(ny,ny,length(r));
895         for n=1:length(r)
896             adj_Ar(:,:,n) = adjunta(Ar(:,:,n)); %Adjunta de A(z) calculada
                    em z=r(n)
897         end
898
899         der_r = zeros(Na*ny+(Nb+1)*nu,ny,length(r));
900         S = zeros(ny,ny,length(r));
901         for n=1:length(r)
902             Soma = zeros(ny);
903             for k=1:Na
904                 Soma = Soma + A(:,:,k)*k*r(n)^-(k+1);
905             end
906             S(:,:,n) = Soma; %Matriz S
907             for i=lin(lin<=Na*ny)
908                 k0 = ceil(i/ny);
909                 for j=col{i}
910                     der_r(i,j,n) = adj_Ar(i-(k0-1)*ny,j,n)*r(n)^(-k0)/...
911                         trace(adj_Ar(:,:,n)*S(:,:,n)); %Derivada primeira de
                                r(n) em relação a xij
912                 end
913             end
914         end
915
916         der_rho = zeros(Na*ny+(Nb+1)*nu,ny,length(r));
917         for n=1:length(r)
918             der_rho(:,:,n) = r(n)*conj(der_r(:,:,n))+...
919                 der_r(:,:,n)*conj(r(n)); %Derivada primeira de rho(n) em
                        relação à matriz X
920         end
921         der_rho = real(der_rho); %Eliminação dos termos imaginários (erros
                numéricos)
922         der_rho = permute(der_rho,[3 1 2]); %der_rho(i,j,n) -> der_rho(n,i,j
                )
923
924         der_Sa = w.*(1+a*(rho-Sa)); %Derivada primeira de Sa em relação ao
                vetor rho
925
926         for i=lin
927             for j=col{i}
928                 gcin(i,j,1) = der_Sa'*der_rho(:,i,j); %Derivada primeira de
```

```
                    cin(1) em relação a xij -> Gradiente de cin(1)
            end
        end

    end

    der_Kp = zeros(ny,nu,Na*ny+(Nb+1)*nu,ny);
    for i=lin
        if i<=Na*ny
            k0 = ceil(i/ny);
            for j=col{i}
                der_Kp(:,:,i,j) = (A1\der_AT{k0,i,j}')*(A1\B1); %Derivada
                    primeira da matriz Kp em relação a xij
            end
        else
            k0 = ceil((i-Na*ny)/nu)-1;
            for j=col{i}
                der_Kp(:,:,i,j) = (A1\der_BT{k0+1,i,j}');
            end
        end
    end
    der_Kp = permute(der_Kp,[3 4 1 2]); %der_Kp(u,v,i,j) -> der_Kp(i,j,u,v)

    in = 2; %Índice da próxima restrição de desigualdade
    eq = 1; %Índice da próxima restrição de igualdade
    for u=1:ny
        for v=1:nu
            if K.inf(u,v)<K.sup(u,v)
                %der_Kp(:,:,u,v) = derivada primeira do elemento (u,v) de Kp
                     em relação à matriz X
                gcin(:,:,in) = -der_Kp(:,:,u,v); %Gradiente de cin(in)
                gcin(:,:,in+1) = der_Kp(:,:,u,v); %Gradiente de cin(in+1)
                in = in+2; %Atualização de in
            else
                gceq(:,:,eq) = der_Kp(:,:,u,v); %Gradiente de ceq(eq)
                eq = eq+1; %Atualização de eq
            end
        end
    end

    for i=lin
```

```
967         gcin(i,:,:) = gcin(i,:,:)/norma_PHI(i); %Obtenção de gcin em termos
                das variáveis xij da matriz X modificada
968     end
969     gcin = reshape(gcin,[(Na*ny+(Nb+1)*nu)*ny Nin]); %Reestruturação de gcin
            -> [gcin(1) gcin(2) ... gcin(Nin)]
970     if isempty(gceq)==0 %Caso existam restrições de igualdade
971         for i=lin
972             gceq(i,:,:) = gceq(i,:,:)/norma_PHI(i); %Obtenção de gceq em
                    termos das variáveis xij da matriz X modificada
973         end
974         gceq = reshape(gceq,[(Na*ny+(Nb+1)*nu)*ny Neq]); %Reestruturação de
                gceq -> [gceq(1) gceq(2) ... gceq(Neq)]
975     end
976
977     gcin = gcin(Id,:); %Gradiente das restrições de desigualdade
978     if isempty(gceq)==0 %Caso existam restrições de igualdade
979         gceq = gceq(Id,:); %Gradiente das restrições de igualdade
980     end
981
982     if nnz(gcin)/numel(gcin)<0.5 %Caso o número de elementos não nulos de
            gcin seja inferior a 50%
983         gcin = sparse(gcin); %gcin é convertida em uma matriz esparsa
984     end
985     if isempty(gceq)==0 %Caso existam restrições de igualdade
986         if nnz(gceq)/numel(gceq)<0.5 %Caso o número de elementos não nulos
                de gceq seja inferior a 50%
987             gceq = sparse(gceq); %gceq é convertida em uma matriz esparsa
988         end
989     end
990
991 end
992
993 end
994
995 %% Hessiana do operador de Lagrange:
996
997 function H = hess(X,lambda,Id,Na,Nb,K)
998
999 global ny nu Nin Neq der_AT der_BT norma_PHI PHI S A Ar adj_Ar...
1000     r der_r rho der_rho a w Sa der_Sa A1 B1;
1001
```

```
1002 vetor = zeros((Na*ny+(Nb+1)*nu)*ny,1);
1003 vetor(Id) = X;
1004 X = vetor;
1005 X = reshape(X,[Na*ny+(Nb+1)*nu ny]); %Reconstrução da matriz X
1006
1007 lin = zeros(1,Na*ny+(Nb+1)*nu); %Alocação de memória para o vetor-linha lin
1008 col = cell(1,Na*ny+(Nb+1)*nu); %Alocação de memória para a célula col
1009 for i=1:Na*ny+(Nb+1)*nu
1010     if norm(X(i,:))~=0 %Caso a i-ésima linha de X possua elementos não nulos
1011         lin(i) = i; %Armazenar o índice da linha
1012         vetor = zeros(1,ny);
1013         for j=1:ny
1014             if X(i,j)~=0 %Caso X(i,j) seja um elemento não nulo
1015                 vetor(j) = j; %Armazenar o índice da coluna
1016             end
1017         end
1018         col{i} = vetor(vetor~=0); %Vetor-linha dos índices das colunas dos
                elementos não nulos da i-ésima linha de X
1019     end
1020 end
1021 lin = lin(lin~=0); %Vetor-linha dos índices das linhas não nulas de X
1022
1023 Hobj = zeros(Na*ny+(Nb+1)*nu,ny,Na*ny+(Nb+1)*nu,ny); %Alocação de memória
    para a matriz Hobj (hessiana da função objetivo)
1024 for p=lin
1025     for q=col{p}
1026         for i=lin
1027             for j=col{i}
1028                 if q==j
1029                     Hobj(p,q,i,j) = 2*PHI(:,p)'*PHI(:,i); %Derivada segunda
                        da função objetivo em relação a xij e a xpq
1030                 end
1031             end
1032         end
1033     end
1034 end
1035 Hobj = reshape(Hobj,[(Na*ny+(Nb+1)*nu)*ny (Na*ny+(Nb+1)*nu)*ny]); %
    Reestruturação de Hobj -> matriz bidimensional
1036
1037 Hin = zeros(Na*ny+(Nb+1)*nu,ny,Na*ny+(Nb+1)*nu,ny,Nin); %Alocação de memória
     para a matriz Hin (hessiana das restrições de desigualdade)
```

```
1038 Heq = zeros(Na*ny+(Nb+1)*nu,ny,Na*ny+(Nb+1)*nu,ny,Neq); %Alocação de memória
      para a matriz Heq (hessiana das restrições de igualdade)
1039
1040 if isempty(r)==0 %Caso o determinante de A(z) seja um polinômio de grau
     maior ou igual a 1
1041
1042     der_adj_Ar = zeros(ny,ny,length(r),Na*ny+(Nb+1)*nu,ny);
1043     if ny>1 %Se ny = 1 -> adj_Ar = ones(ny,ny,length(r)) -> der_adj_Ar =
         zeros(ny,ny,length(r),Na*ny+(Nb+1)*nu,ny)
1044         for n=1:length(r)
1045             for u=1:ny
1046                 for v=1:ny
1047                     Soma = zeros(ny-1);
1048                     for k=1:Na
1049                         M1 = A(:,:,k);
1050                         M1(v,:) = [];
1051                         M1(:,u) = []; %Matriz M1
1052                         Soma = Soma + M1*k*r(n)^-(k+1);
1053                     end
1054                     M2 = Ar(:,:,n);
1055                     M2(v,:) = [];
1056                     M2(:,u) = []; %Matriz M2
1057                     for p=lin(lin<=Na*ny)
1058                         k1 = ceil(p/ny);
1059                         for q=col{p}
1060                             M3 = der_AT{k1,p,q};
1061                             M3(u,:) = [];
1062                             M3(:,v) = []; %Matriz M3
1063                             %der_adj_Ar(u,v,n,p,q) = derivada primeira do
                                 elemento (u,v) da adjunta de A(z) calculada
                                 em z=r(n) em relação a xpq
1064                             if rcond(M2)<=eps %Se M2 estiver mal
                                 condicionada -> Utilizar a adjunta de M2
                                 para calcular der_adj_Ar(u,v,n,p,q)
1065                                 adj_M2 = adjunta(M2);
1066                                 der_adj_Ar(u,v,n,p,q) = (-1)^(u+v)*...
1067                                     (trace(adj_M2*Soma)*der_r(p,q,n)-...
1068                                     (r(n)^-k1)*trace(adj_M2*M3'));
1069                             else %Se M2 estiver bem condicionada -> Não
                                 utilizar a adjunta de M2 (redução do custo
                                 computacional)
```

```
                                der_adj_Ar(u,v,n,p,q) = (-1)^(u+v)*...
                                    det(M2)*(trace(M2\Soma)*...
                                    der_r(p,q,n)-(r(n)^-k1)*trace(M2\M3'));
                            end
                        end
                    end
                end
            end
        end
    end

    der2_r = zeros(Na*ny+(Nb+1)*nu,ny,Na*ny+(Nb+1)*nu,ny,length(r));
    for n=1:length(r)
        f1 = trace(adj_Ar(:,:,n)*S(:,:,n)); %Fator f1 (escalar)
        Soma = zeros(ny);
        for k=1:Na
            Soma = Soma + A(:,:,k)*k*(k+1)*r(n)^-(k+2);
        end
        for i=lin(lin<=Na*ny)
            k0 = ceil(i/ny);
            for j=col{i}
                f3 = adj_Ar(i-(k0-1)*ny,j,n)*r(n)^-k0; %Fator f3 (escalar)
                for p=lin(lin<=Na*ny)
                    k1 = ceil(p/ny);
                    for q=col{p}
                        f2 = der_adj_Ar(i-(k0-1)*ny,j,n,p,q)*r(n)^-k0 -...
                            der_r(p,q,n)*adj_Ar(i-(k0-1)*ny,j,n)*...
                            k0*r(n)^-(k0+1); %Fator f2 (escalar)
                        f4 = adj_Ar(p-(k1-1)*ny,q,n)*k1*r(n)^-(k1+1) -...
                            der_r(p,q,n)*trace(adj_Ar(:,:,n)*Soma) +...
                            trace(der_adj_Ar(:,:,n,p,q)*S(:,:,n)); %Fator f4
                                (escalar)
                        der2_r(p,q,i,j,n) = (f1*f2-f3*f4)/f1^2; %Derivada
                            segunda de r(n) em relação a xij e a xpq
                    end
                end
            end
        end
    end

    der2_rho = zeros(Na*ny+(Nb+1)*nu,ny,Na*ny+(Nb+1)*nu,ny,length(r));
```

```
1109      for n=1:length(r)
1110          for i=lin(lin<=Na*ny)
1111              for j=col{i}
1112                  der2_rho(:,:,i,j,n) = r(n)*conj(der2_r(:,:,i,j,n)) +...
1113                      der_r(:,:,n)*conj(der_r(i,j,n)) +...
1114                      der_r(i,j,n)*conj(der_r(:,:,n)) +...
1115                      der2_r(:,:,i,j,n)*conj(r(n)); %Derivada segunda de rho(n
                             ) em relação a xij e a xpq
1116              end
1117          end
1118      end
1119      der2_rho = real(der2_rho); %Eliminação dos termos imaginários (erros
             numéricos)
1120      der2_rho = permute(der2_rho,[5 1 2 3 4]); %der2_rho(p,q,i,j,n) ->
             der2_rho(n,p,q,i,j)
1121
1122      der2_Sa = zeros(length(r));
1123      I = eye(length(r));
1124      for n=1:length(r)
1125          der2_Sa(:,n) = a*w(n)*(I(:,n)*(2+a*(rho(n)-Sa)) -...
1126              w.*(2+a*(rho+rho(n)-2*Sa))); %Derivada segunda de Sa em relação
                  a rho(n) e a rho(m)
1127      end
1128
1129      for p=lin(lin<=Na*ny)
1130          for q=col{p}
1131              for i=lin(lin<=Na*ny)
1132                  for j=col{i}
1133                      Hin(p,q,i,j,1) = der_Sa'*der2_rho(:,p,q,i,j) +...
1134                          der_rho(:,p,q)'*der2_Sa*der_rho(:,i,j); %Derivada
                                 segunda de cin(1) em relação a xij e a xpq ->
                                 Hessiana de cin(1)
1135                  end
1136              end
1137          end
1138      end
1139
1140  end
1141
1142  der2_Kp = zeros(ny,nu,Na*ny+(Nb+1)*nu,ny,Na*ny+(Nb+1)*nu,ny);
1143  for p=lin
```

```
    for q=col{p}
        for i=lin
            for j=col{i}
                if i<=Na*ny && p<=Na*ny
                    k0 = ceil(i/ny);
                    k1 = ceil(p/ny);
                    der2_Kp(:,:,p,q,i,j) = (A1\der_AT{k1,p,q}')*...
                        (A1\der_AT{k0,i,j}')*(A1\B1) +...
                        (A1\der_AT{k0,i,j}')*(A1\der_AT{k1,p,q}')*...
                        (A1\B1); %Derivada segunda da matriz Kp em relação a
                            xij e a xpq
                elseif i<=Na*ny && p>Na*ny
                    k0 = ceil(i/ny);
                    k1 = ceil((p-Na*ny)/nu)-1;
                    der2_Kp(:,:,p,q,i,j) = (A1\der_AT{k0,i,j}')*...
                        (A1\der_BT{k1+1,p,q}');
                elseif i>Na*ny && p<=Na*ny
                    k0 = ceil((i-Na*ny)/nu)-1;
                    k1 = ceil(p/ny);
                    der2_Kp(:,:,p,q,i,j) = (A1\der_AT{k1,p,q}')*...
                        (A1\der_BT{k0+1,i,j}');
                end
            end
        end
    end
end
der2_Kp = permute(der2_Kp,[3 4 5 6 1 2]); %der2_Kp(u,v,p,q,i,j) -> der2_Kp(p
    ,q,i,j,u,v)

in = 2; %Índice da próxima restrição de desigualdade
eq = 1; %Índice da próxima restrição de igualdade
for u=1:ny
    for v=1:nu
        if K.inf(u,v)<K.sup(u,v)
            %der2_Kp(:,:,:,:,u,v) = derivada segunda do elemento (u,v) de Kp
                em relação à matriz X
            Hin(:,:,:,:,in) = -der2_Kp(:,:,:,:,u,v); %Hessiana de cin(in)
            Hin(:,:,:,:,in+1) = der2_Kp(:,:,:,:,u,v); %Hessiana de cin(in+1)
            in = in+2; %Atualização de in
        else
            Heq(:,:,:,:,eq) = der2_Kp(:,:,:,:,u,v); %Hessiana de ceq(eq)
```

```
1182            eq = eq+1; %Atualização de eq
1183        end
1184    end
1185 end
1186
1187 for p=lin
1188     for i=lin
1189         Hin(p,:,i,:,:) = Hin(p,:,i,:,:)/(norma_PHI(p)*norma_PHI(i)); %
                Obtenção de Hin em termos das variáveis xij e xpq da matriz X
                modificada
1190     end
1191 end
1192 Hin = reshape(Hin,[(Na*ny+(Nb+1)*nu)*ny (Na*ny+(Nb+1)*nu)*ny Nin]); %
    Reestruturação de Hin -> Hin(:,:,k) = hessiana da k-ésima restrição de
    desigualdade
1193 if isempty(Heq)==0 %Caso existam restrições de igualdade
1194     for p=lin
1195         for i=lin
1196             Heq(p,:,i,:,:) = Heq(p,:,i,:,:)/(norma_PHI(p)*norma_PHI(i)); %
                    Obtenção de Heq em termos das variáveis xij e xpq da matriz
                    X modificada
1197         end
1198     end
1199     Heq = reshape(Heq,[(Na*ny+(Nb+1)*nu)*ny (Na*ny+(Nb+1)*nu)*ny Neq]); %
        Reestruturação de Heq -> Heq(:,:,k) = hessiana da k-ésima restrição
        de igualdade
1200 end
1201
1202 H = Hobj;
1203 for i=1:Nin
1204     H = H + lambda.ineqnonlin(i)*Hin(:,:,i); %lambda = estrutura dos
        multiplicadores de Lagrange
1205 end
1206 if isempty(Heq)==0 %Caso existam restrições de igualdade
1207     for i=1:Neq
1208         H = H + lambda.eqnonlin(i)*Heq(:,:,i);
1209     end
1210 end
1211 H = H(Id,Id); %Hessiana do operador de Lagrange
1212
1213 if nnz(H)/numel(H)<0.5 %Caso o número de elementos não nulos de H seja
```

```
    inferior a 50%
    H = sparse(H); %H é convertida em uma matriz esparsa
end

end

%% Adjunta:

function adjA = adjunta(A)

NL = size(A,1); %Número de linhas da matriz A
NC = size(A,2); %Número de colunas da matriz A

if NL~=NC %Caso A não seja uma matriz quadrada
    disp('Erro: a adjunta só está definida para matrizes quadradas');
    return;
else
    n = NL; %Ordem da matriz quadrada
end

if n==1
    adjA = 1; %Adjunta de uma matriz quadrada de ordem 1
else
    D = zeros(n); %Alocação de memória para a matriz D
    Cof = zeros(n); %Alocação de memória para a matriz Cof
    for i=1:n
        for j=1:n
            M = A;
            M(i,:) = [];
            M(:,j) = []; %A submatriz M é gerada a partir da remoção da i-
                ésima linha e da j-ésima coluna da matriz A
            D(i,j) = det(M); %Determinante da submatriz M
            Cof(i,j) = (-1)^(i+j)*D(i,j); %Cofator do elemento (i,j) de A
        end
    end
    adjA = Cof.'; %A adjunta de A é a matriz de seus cofatores transposta
end

end
```

B.4 Algoritmo 4: *teste_GBN.m*

```
function [U2,delta] = teste_GBN(tau,G,dtg,Gpi,u,y,fs,seed,g)
%
%   A função TESTE_GBN faz o planejamento de sinais para a etapa do teste
%GBN (Generalized Binary Noise).
%
%*************************************************************************
%
%SINTAXE:
%
%   [U2,delta] = teste_GBN(tau,G,dtg,Gpi,u,y,fs,seed,g)
%
%*************************************************************************
%
%ARGUMENTOS DE ENTRADA:
%
%   tau = Estrutura formada pelos parâmetros tau.min(nu x 1) e
%   tau.max(nu x 1), onde tau.min(j) e tau.max(j) são a menor e a maior
%   constante de tempo obtidas para a entrada uj, respectivamente e nu é o
%   número total de entradas. Os valores de tau são expressos em tempos de
%   amostragem (k). Esta estrutura é gerada pela função teste_degrau.
%
%   G = {G{1},G{2},...,G{ng}} é uma célula, onde G{i} é um vetor-linha que
%   contém os índices das entradas pertencentes ao i-ésimo grupo GBN e ng é
%   o número total de grupos. (Default: G={1:nu})
%
%   dtg(1 x ng) = Durações dos grupos GBN, expressas em tempos de
%   amostragem (k). Este vetor-linha é gerado pela função teste_degrau.
%
%   Gpi = Matriz de transferência do processo identificada (na variável
%   z^-1). Esta matriz é gerada pela função identificador.
%
%   u = Estrutura formada pelos parâmetros u.min(nu x 1) e u.max(nu x 1),
%   onde u.min(j) e u.max(j) são o menor e o maior valor obtidos para a
%   entrada uj, respectivamente. Os valores de u são dados em termos de
%   variáveis-desvio. Esta estrutura é gerada pela função identificador.
%
%   y = Estrutura formada pelos parâmetros y.min(ny x 1) e y.max(ny x 1),
%   onde y.min(i) e y.max(i) são o menor e o maior valor obtidos para a
%   saída yi, respectivamente. Os valores de y são dados em termos de
%   variáveis-desvio. Esta estrutura é gerada pela função identificador.
%
```

```
42 %   fs = Fator de segurança. Com base na matriz Gpi (modelo identificado),
43 %   os sinais GBN são projetados de modo a assegurar que as saídas yi
44 %   fiquem compreendidas entre fs*y.min(i) e fs*y.max(i). Se as incertezas
45 %   sobre Gpi forem desprezíveis, deve-se atribuir a fs o valor 1. Caso
46 %   contrário, um valor menor deve ser utilizado. fs deve ser especificado
47 %   como um número positivo menor ou igual a 1. (Default: fs=1)
48 %
49 %   seed = Semente para a função MATLAB rand, que gera números aleatórios
50 %   uniformemente distribuídos entre 0 e 1. seed deve ser especificado como
51 %   um número inteiro não negativo. (Default: seed=0)
52 %
53 %   g = Comando gráfico. Fazer g=1 para o algoritmo plotar o gráfico das
54 %   entradas-desvio projetadas em função do tempo. (Default: g=1)
55 %
56 %*************************************************************************
57 %
58 %ARGUMENTOS DE SAÍDA:
59 %
60 %   U2(nu x N) = [u1(1:N)';u2(1:N)';u3(1:N)';...;unu(1:N)'], onde
61 %   u1,...,unu são as entradas projetadas para a etapa do teste GBN,
62 %   expressas em termos de variáveis-desvio.
63 %
64 %   delta(nu x 1) = [delta(1);delta(2);...;delta(nu)], onde delta(j)>0 é a
65 %   amplitude do sinal GBN de uj.
66 %
67 %*************************************************************************
68 %
69 %OBSERVAÇÃO:
70 %
71 %   Todos os componentes do vetor delta devem ser não nulos. Caso algum
72 %   componente seja 0, deve-se modificar o valor de seed e executar a
73 %   função novamente.
74 %
75 %*************************************************************************
76 %
77 %AUTOR: Cristiano Salah Mussoi
78 %DATA DE CRIAÇÃO: 05/01/2019
79
80 %% Verificação dos argumentos de entrada:
81
82 if nargin<6
```

```
83      disp('Erro: o argumento de entrada y deve ser fornecido');
84      return;
85  elseif nargin==6
86      if isempty(tau)
87          disp('Erro: o argumento de entrada tau deve ser fornecido');
88          return;
89      end
90      if isempty(G)
91          G = {1:length(tau.min)};
92      elseif iscell(G)==0
93          disp('Erro: G deve ser especificado como uma célula');
94          return;
95      else
96          for n=1:length(G)
97              if isvector(G{n})==0
98                  disp('Erro: os elementos de G devem ser vetores');
99                  return;
100             end
101         end
102     end
103     if isempty(dtg)
104         disp('Erro: o argumento de entrada dtg deve ser fornecido');
105         return;
106     end
107     if isempty(Gpi)
108         disp('Erro: o argumento de entrada Gpi deve ser fornecido');
109         return;
110     end
111     if isempty(u)
112         disp('Erro: o argumento de entrada u deve ser fornecido');
113         return;
114     end
115     if isempty(y)
116         disp('Erro: o argumento de entrada y deve ser fornecido');
117         return;
118     end
119     fs = 1;
120     seed = 0;
121     g = 1;
122 elseif nargin==7
123     if isempty(tau)
```

```
124         disp('Erro: o argumento de entrada tau deve ser fornecido');
125         return;
126     end
127     if isempty(G)
128         G = {1:length(tau.min)};
129     elseif iscell(G)==0
130         disp('Erro: G deve ser especificado como uma célula');
131         return;
132     else
133         for n=1:length(G)
134             if isvector(G{n})==0
135                 disp('Erro: os elementos de G devem ser vetores');
136                 return;
137             end
138         end
139     end
140     if isempty(dtg)
141         disp('Erro: o argumento de entrada dtg deve ser fornecido');
142         return;
143     end
144     if isempty(Gpi)
145         disp('Erro: o argumento de entrada Gpi deve ser fornecido');
146         return;
147     end
148     if isempty(u)
149         disp('Erro: o argumento de entrada u deve ser fornecido');
150         return;
151     end
152     if isempty(y)
153         disp('Erro: o argumento de entrada y deve ser fornecido');
154         return;
155     end
156     if isempty(fs)
157         fs = 1;
158     elseif isscalar(fs)==0 || fs<=0 || fs>1
159         disp('Erro: valor de fs inválido');
160         return;
161     end
162     seed = 0;
163     g = 1;
164 elseif nargin==8
```

```
165     if isempty(tau)
166         disp('Erro: o argumento de entrada tau deve ser fornecido');
167         return;
168     end
169     if isempty(G)
170         G = {1:length(tau.min)};
171     elseif iscell(G)==0
172         disp('Erro: G deve ser especificado como uma célula');
173         return;
174     else
175         for n=1:length(G)
176             if isvector(G{n})==0
177                 disp('Erro: os elementos de G devem ser vetores');
178                 return;
179             end
180         end
181     end
182     if isempty(dtg)
183         disp('Erro: o argumento de entrada dtg deve ser fornecido');
184         return;
185     end
186     if isempty(Gpi)
187         disp('Erro: o argumento de entrada Gpi deve ser fornecido');
188         return;
189     end
190     if isempty(u)
191         disp('Erro: o argumento de entrada u deve ser fornecido');
192         return;
193     end
194     if isempty(y)
195         disp('Erro: o argumento de entrada y deve ser fornecido');
196         return;
197     end
198     if isempty(fs)
199         fs = 1;
200     elseif isscalar(fs)==0 || fs<=0 || fs>1
201         disp('Erro: valor de fs inválido');
202         return;
203     end
204     if isempty(seed)
205         seed = 0;
```

```
206     elseif isscalar(seed)==0 || seed<0 || seed~=round(seed)
207         disp('Erro: valor de seed inválido');
208         return;
209     end
210     g = 1;
211 elseif nargin==9
212     if isempty(tau)
213         disp('Erro: o argumento de entrada tau deve ser fornecido');
214         return;
215     end
216     if isempty(G)
217         G = {1:length(tau.min)};
218     elseif iscell(G)==0
219         disp('Erro: G deve ser especificado como uma célula');
220         return;
221     else
222         for n=1:length(G)
223             if isvector(G{n})==0
224                 disp('Erro: os elementos de G devem ser vetores');
225                 return;
226             end
227         end
228     end
229     if isempty(dtg)
230         disp('Erro: o argumento de entrada dtg deve ser fornecido');
231         return;
232     end
233     if isempty(Gpi)
234         disp('Erro: o argumento de entrada Gpi deve ser fornecido');
235         return;
236     end
237     if isempty(u)
238         disp('Erro: o argumento de entrada u deve ser fornecido');
239         return;
240     end
241     if isempty(y)
242         disp('Erro: o argumento de entrada y deve ser fornecido');
243         return;
244     end
245     if isempty(fs)
246         fs = 1;
```

```
247     elseif isscalar(fs)==0 || fs<=0 || fs>1
248         disp('Erro: valor de fs inválido');
249         return;
250     end
251     if isempty(seed)
252         seed = 0;
253     elseif isscalar(seed)==0 || seed<0 || seed~=round(seed)
254         disp('Erro: valor de seed inválido');
255         return;
256     end
257     if isempty(g)
258         g = 1;
259     end
260 else
261     disp('Erro: número de argumentos de entrada inválido');
262     return;
263 end
264 
265 %% Verificação dos argumentos de saída:
266 
267 if nargout~=2 %O número de argumentos de saída deve ser igual a 2
268     disp('Erro: devem ser fornecidos 2 argumentos de saída para a função');
269     return;
270 end
271 
272 %% Obtenção de ny, nu, ng e N:
273 
274 ny = size(Gpi,1); %Número de variáveis de saída
275 nu = size(Gpi,2); %Número de variáveis de entrada
276 ng = length(G); %Número de grupos GBN
277 
278 N = 2+sum(dtg); %Número total de tempos de amostragem
279 
280 %% Cálculo do vetor p:
281 
282 alpha = 2; %Constante alpha
283 beta = 3; %Constante beta
284 
285 ni_min = zeros(nu,1);
286 ni_max = zeros(nu,1);
287 p = zeros(nu,1);
```

```
288 for j=1:nu
289     ni_min(j) = min(1/(beta*tau.max(j)),pi);
290     ni_max(j) = min(alpha/tau.min(j),pi); %[ni_min(j) ni_max(j)] = intervalo
            de interesse do espectro de frequência da entrada uj
291     p(j) = 1/(1 + sqrt(tan(ni_min(j)/2)*tan(ni_max(j)/2))); %Probabilidade
            ótima de não mudança de nível de uj
292 end
293
294 %% Geração de nu sinais GBN de amplitude unitária:
295
296 warning('off','all');
297 rand('twister',seed);
298 warning('on','all');
299
300 U2 = zeros(nu,N);
301 k0 = 2; %Tempo de amostragem inicial
302 for n=1:ng
303     if size(G{n},2)==1 %Caso G{n} seja um vetor-coluna
304         G{n} = G{n}'; %G{n} passa a ser um vetor-linha
305     end
306     for j=G{n}
307         R = rand(1,dtg(n)); %Vetor-linha de dtg(n) componentes aleatórios
                compreendidos entre 0 e 1
308         if R(1)>0.5
309             P_M = 1; %Nível inicial de uj
310         else
311             P_M = -1;
312         end
313         GBN = zeros(1,dtg(n));
314         for k=1:dtg(n)
315             if R(k)<1-p(j) %1-p(j) = probabilidade ótima de mudança de nível
                    de uj
316                 P_M = -P_M; %Mudança de nível
317             end
318             GBN(k) = P_M; %Nível de uj no k-ésimo tempo de amostragem
319         end
320         U2(j,k0:k0+dtg(n)-1) = GBN; %Sinal GBN de uj
321     end
322     k0 = k0+dtg(n); %Atualização de k0
323 end
324
```

```
%% Maximização das amplitudes dos sinais GBN:

%delta = [delta(1);delta(2);...;delta(nu)], onde delta(j) é a amplitude do
    sinal GBN de uj

%Variáveis de decisão: componentes do vetor delta
%Função objetivo: max(-f'*delta) = min(f'*delta)
%Restrições: A*delta<=b e lb=<delta<=ub

f = -ones(nu,1); %Vetor dos coeficientes da função objetivo

A_til = zeros(ny*N,nu);
for i=1:ny
    for j=1:nu
        A_til((i-1)*N+1:i*N,j) = lsim(Gpi(i,j),U2(j,:)');
    end
end
A = [A_til; -A_til]; %Matriz A

b_sup = zeros(ny*N,1);
b_inf = zeros(ny*N,1);
for i=1:ny
    b_sup((i-1)*N+1:i*N) = fs*y.max(i)*ones(N,1); %fs*y.max(i) = limite
        superior de yi
    b_inf((i-1)*N+1:i*N) = fs*y.min(i)*ones(N,1); %fs*y.min(i) = limite
        inferior de yi
end
b = [b_sup; -b_inf]; %Vetor b

lb = zeros(nu,1); %Limite inferior de delta
ub = min(abs(u.min),abs(u.max)); %Limite superior de delta

options = optimset('LargeScale','on','Display','iter',...
    'TolFun',1e-8,'MaxIter',Inf);

delta = linprog(f,A,b,[],[],lb,ub,[],options); %Programação Linear (Linear
    Programming - LP)

for j=1:nu
    U2(j,:) = U2(j,:)*delta(j); %Atualização da matriz U2
end
```

```
362
363 %% Gráfico das entradas-desvio projetadas para o teste GBN:
364
365 if g==1
366     figure(6)
367     stairs(U2');
368     hold on;
369     plot(zeros(N,1),'k');
370     hold off;
371     title('Teste GBN');
372     xlabel('k'); %Tempo de amostragem
373     ylabel('u(k)'); %Variáveis-desvio de entrada
374     xlim([1,N]);
375     ylim([-1.1*max(max(abs(U2))),1.1*max(max(abs(U2)))]);
376     set(gca,'XTick',unique([1,get(gca,'XTick')]));
377     leg = cell(1,nu); %Alocação de memória para a célula leg
378     for j=1:nu
379         leg{j}=['u',num2str(j)]; %Legendas
380     end
381     legend(leg,'Orientation','vertical','Location','northeastoutside');
382 end
383
384 end
```