IDENTIFICAÇÃO DE SISTEMAS NÃO-LINEARES BASEADA NO ALGORITMO LAR: PROPOSTA DE UM CRITÉRIO DE PARADA GEOMÉTRICO E SUA APLICAÇÃO PARA A CONSCIÊNCIA SITUACIONAL

Catia Valdman

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Elétrica.

Orientadores: Marcello Luiz Rodrigues de Campos
José Antonio Apolinário Junior

Rio de Janeiro
Fevereiro de 2014

IDENTIFICAÇÃO DE SISTEMAS NÃO-LINEARES BASEADA NO ALGORITMO LAR: PROPOSTA DE UM CRITÉRIO DE PARADA GEOMÉTRICO E SUA APLICAÇÃO PARA A CONSCIÊNCIA SITUACIONAL

Catia Valdman

TESE 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 DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. Marcello Luiz Rodrigues de Campos, Ph.D.

Prof. José Antonio Apolinário Junior, D.Sc.

Prof. Sergio Lima Netto, Ph.D.

Prof. Vitor Heloiz Nascimento, Ph.D.

Prof. Ernesto Leite Pinto, D.Sc.

Prof. Alexandre Santos de la Vega, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2014

Valdman, Catia

Identificação de sistemas não-lineares baseada no algoritmo LAR: proposta de um critério de parada geométrico e sua aplicação para a consciência situacional/Catia Valdman. – Rio de Janeiro: UFRJ/COPPE, 2014.

XVI, 117 p.: il.; 29,7cm.

Orientadores: Marcello Luiz Rodrigues de Campos
José Antonio Apolinário Junior

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2014.

Referências Bibliográficas: p. 112 – 117.

1. Algoritmo Least Angle Regression. 2. Algoritmo Recursive Least Squares. 3. Sistemas não-lineares. 4. Filtro de Volterra. 5. Seleção de modelo. 6. Consciência situacional. I. Campos, Marcello Luiz Rodrigues de *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Filha de peixe, peixinho é!
Aos meus peixes,
que me ensinaram a nadar
só para frente.

Agradecimentos

Meu período de doutoramento foi uma jornada. Para realizar um desejo pessoal, iniciei o doutorado há cinco anos, em Março de 2009. Numa família de acadêmicos, não podia fugir à regra (pelo menos não dessa!). Só não esperava que tantas coisas memoráveis, boas e ruins, fossem acontecer neste período.

Infelizmente, minha mãe, Belkis Valdmanז"ל, não está mais aqui presente para ver que consegui concluir. Tenho certeza que estaria, e está, muito orgulhosa de mim, assim como está meu pai, Benjamin Valdman. Meu pai, invariavelmente ao meu lado, viu de perto todas dificuldades superadas neste período. Não existem palavras que bastem para agradecer aos meus pais.

Minhas irmãs, Andrea Valdman e Erika Valdman, também são muito especiais. Às vezes mais perto, às vezes mais longe, mas sempre me ajudando um pouco aqui um pouco acolá, com as suas experiências acadêmicas e de vida.

Quero agradecer em especial à minha tia, Keti Tenenblat, por me ajudar nas pedras do meu caminho e me dar autoconfiança em momentos críticos. Ao meu tio, Jo Dweck, quero agradecer por seu sábios conselhos – sempre tinha um, se precisasse ou não!

Neste período também ocorreram mudanças no meu trabalho como examinadora de patentes, no Instituto Nacional de Propriedade Industrial. Muitas responsabilidades foram adquiridas em viagens internacionais. Se não fosse por minha chefe, Telma Alcantara, e meu amigo Renato Dutra, o caminho teria sido mais sinuoso.

Conselhos e conversas com amigos também fazem parte do meu viver. Alguns amigos em particular me ajudaram a "quebrar um galho", por isso gostaria de agradecer especialmente ao Rafael Brandão e Fabiano Castoldi, que sempre responderam aos meus pedidos, minhas intermináveis dúvidas "técnicas", facilitando minha jornada. No quesito ajuda pessoal, tem uma lista grande de amigos... Em especial, gostaria de agradecer a Viviane Lajter Segal, Nira Bessler, Luciana Gerber, Miriam Berenguer, Andrei Battistel e Florian Knäble, por longas conversas que sempre trazem uma sensação de calma e segurança em minha vida.

Por fim, gostaria de agradecer ao Marcello Campos e ao José Apolinário, por terem confiado em mim e aceitado a me orientar como aluna de doutorado em tempo parcial.

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

IDENTIFICAÇÃO DE SISTEMAS NÃO-LINEARES BASEADA NO ALGORITMO LAR: PROPOSTA DE UM CRITÉRIO DE PARADA GEOMÉTRICO E SUA APLICAÇÃO PARA A CONSCIÊNCIA SITUACIONAL

Catia Valdman

Fevereiro/2014

Orientadores: Marcello Luiz Rodrigues de Campos
José Antonio Apolinário Junior

Programa: Engenharia Elétrica

Esta tese apresenta duas propostas para identificação de sistemas não-lineares.

A primeira proposta, chamada de algoritmo GLAR, é criada a partir de um novo critério de parada geométrico que atua ativamente no algoritmo *Least Angle Regression* (LAR), interrompendo-o após N iterações. Tal critério possui um fator de penalidade, influenciado pela quantidade de dados disponível e pela esparsidade do sistema. Quando comparado com outros critérios de seleção de modelo no estado da arte, o resultado do critério de parada geométrico mostra ser mais eficiente, indicando uma quantidade de coeficientes que resulta em um erro mínimo quadrático próximo ao ruído de observação do sistema.

A segunda proposta, chamada de algoritmo GLAR-RLS, é criada a partir da combinação do algoritmo GLAR e do algoritmo *Recursive Least Squares* (RLS). Com a indicação pelo algoritmo GLAR de quais coeficientes são necessários para a identificação do sistema, o algoritmo RLS estima o vetor de coeficientes recursivamente no tempo, com N coeficientes ao invés do total J. O termo consciência situacional é introduzido nesta proposta como a percepção de uma modificação no sistema; para esta tarefa, o erro de estimação é usado, identificando modificações bruscas e acionando o algoritmo GLAR. Considerando sistemas não-lineares esparsos, em que $N << J$, o algoritmo GLAR-RLS possui maior precisão e menor complexidade computacional do que o algoritmo RLS. Sistemas não-lineares com modelos esparsos são simulados para avaliação de ambos os algoritmos. Para identificar os sistemas não-lineares simulados, filtros de Volterra de terceira ordem e de quinta ordem são utilizados em conjunto com os algoritmos propostos.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

NONLINEAR SYSTEM IDENTIFICATION BASED ON THE LAR ALGORITHM: PROPOSAL OF A GEOMETRICAL STOPPING CRITERION AND ITS USE FOR SITUATIONAL AWARENESS

Catia Valdman

February/2014

Advisors: Marcello Luiz Rodrigues de Campos
José Antonio Apolinário Junior

Department: Electrical Engineering

In this thesis two new schemes for nonlinear system identification are proposed.

The first one, called the GLAR algorithm, is created using a new geometrical stopping criterion, which interrupts the *Least Angle Regression* (LAR) algorithm after N iterations. Such criterion has a penalty factor, ruled by the amount of available data and the system sparsity. When compared to other criteria in the state of the art, the results of the geometrical stopping criterion shows to be more efficient, indicating an amount of coefficients that yields a minimum quadratic error closer to the system observation noise.

The second one, called the GLAR-RLS algorithm, combines the GLAR and the *Recursive Least Squares* (RLS) algorithms. The RLS algorithm estimates the coefficient vector recursively in time, estimating N, given by the GLAR algorithm, instead of the total J coefficients. The term situational awareness is used in this thesis as the perception of a modification in the system, carried out by the estimated error which identifies abrupt modifications and triggers the GLAR algorithm. Considering sparse nonlinear systems, where $N << J$, the GLAR-RLS algorithm is more accurate and less computationally complex than the RLS algorithm. Nonlinear systems with sparse models are simulated to evaluate both algorithms. To identify the simulated nonlinear systems, third-order and fifth-order Volterra filters are combined with the proposed schemes.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Identificar sistemas, sejam lineares ou não-lineares, sempre foi um desafio. Identificação de sistemas pode ser referida como sendo uma área de conhecimento que estuda maneiras de modelar e analisar sistemas a partir de observações, ou seja, dados [1]. Apesar de ser um problema antigo, amplamente investigado e com diversas soluções propostas, o desafio ainda permanece.

Atualmente, sistemas de aquisição de dados confiáveis e capazes de monitorar variáveis de processos reais com altas taxas de amostragem, de maneira a garantir uma representação dinâmica do sistema, estão disponíveis com maior facilidade [1]. O desafio agora é a identificação de sistemas em tempo real com baixa complexidade computacional e, assim, baixo custo de energia. Sistemas *real time* são dinâmicos no tempo. A percepção de uma modificação no sistema identificado é chamada nesta tese de consciência situacional.

Consciência situacional, do termo em inglês *situational awareness*, foi criado no final da década de 80 por pesquisadores da área aeroespacial. Endsley [2] define consciência situacional como o modelo que o piloto da aeronave faz do mundo à sua volta em qualquer ponto no tempo. A capacidade individual, o treinamento, as experiências, o objetivo e as habilidades para responder a uma tarefa determinam a consciência situacional do piloto [2]. Atualmente, este termo é utilizado em outras áreas de uma forma mais abrangente. Em [3], o monitoramento de sistemas industriais foi utilizado neste contexto. Em [4], o conceito de consciência situacional foi utilizado para descrever como sensores de aquisição de dados devem processar e responder o conhecimento da situação em curso, objetivando reduzir a quantidade de pacotes de dados enviados para o sistema de monitoramento.

Dentre os algoritmos no estado da arte disponíveis para identificação de sistemas, o algoritmo *Least Angle Regression*, conhecido como LAR, é abordado nesta tese. O algoritmo LAR foi proposto por Efron et al. e publicado pela primeira vez em 2004 [5]. Nesta ocasião, o acrônimo dado ao algoritmo foi LARS, onde S indica os algoritmos LASSO e *Stagewise* [5], cujos resultados podem ser obtidos com uma

pequena modificação no pseudocódigo do algoritmo original. O acrônimo LAR, ao invés de LARS, é encontrado pela primeira vez em 2009 na publicação do livro *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* [6], dos mesmos autores do artigo original. O algoritmo ainda é, entretanto, referenciado na literatura técnica de ambas as formas.

O algoritmo LAR é um algoritmo de regressão similar ao *forward stepwise* [5], ou seja, a estimação dos coeficientes é sequencial e os coeficientes são adicionados um a um ao modelo, compondo o chamado **conjunto ativo**. A cada interação, o "melhor" coeficiente é identificado e adicionado ao modelo [6]. No algoritmo LAR, o "melhor" coeficiente é aquele que tem maior correlação com o erro residual, i.e., o coeficiente que mais influencia o erro e altera a estimação do sinal de saída. No algoritmo LAR, o primeiro coeficiente adicionado ao modelo é estimado de modo a atingir seu valor de erro quadrado mínimo; porém, ao identificar outro coeficiente tão correlato com o erro residual quanto o primeiro, este segundo coeficiente é adicionado ao conjunto ativo e ambos coeficientes são estimados de forma interligada [6]. O algoritmo LAR continua até que todos os coeficientes componham o conjunto ativo, resultando, se não for acionado um critério de parada, a mesma resposta do algoritmo *Least Squares* (LS) [5].

Vários estudos sobre o algoritmo LAR foram apresentados em livros [6, 7] e outros tipo de publicações [8–10]. O algoritmo LAR foi desenvolvido e baseado em estudos de diabetes [5], sendo utilizado em diversas aplicações desde então.

Em [11] e [12], modelos de classificação de textos (TC – *Text Classification*) foram desenvolvidos utilizando o LAR. O algoritmo LAR foi considerado o mais adequado por escolher e ordenar os termos mais relevantes. Em ambos artigos, foi concluído que o LAR é um algoritmo eficiente para problemas de classificação textuais.

Em [13], o algoritmo LAR foi usado para estimar a variabilidade de desempenho de circuitos integrados com um maior número de coeficientes, na ordem de 10^4 a 10^6. Ao comparar a resposta do algoritmo LS e do algoritmo LAR, os autores concluíram que o algoritmo LAR diminui o tempo de execução em até 25 vezes, sem comprometer a exatidão.

Em 2010, o algoritmo foi empregado na área de processamento de imagem, em representação e reconhecimento facial [14], bem como em estimativa facial e estimativa de idade [15]. Em ambos os estudos, o algoritmo LAR foi utilizado como ferramenta para avaliar os resultados dos parâmetros.

Nesta tese, o algoritmo LAR é utilizado para a identificação de sistemas não-lineares. Modelos de sistemas não-lineares são usados em diversas áreas, tais como sistemas de comunicação, amplificadores de potência, alto-falantes com distorção harmônica, entre outros [16]. Para a identificação de sistemas não-lineares, o filtro de Volterra é comumente empregado.

Filtro de Volterra é um dispositivo não-linear com memória baseado na série de Volterra [17]. A série de Volterra pode ser considerada uma generalização da série de Taylor de uma função com memória, e pode possuir infinitos coeficientes em seu núcleo, ou *kernel*. Para evitar um grande número de coeficientes, a maioria das abordagens tende a limitar a ordem do filtro em dois.

Em [18, 19], filtros adaptativos de Volterra de segunda ordem foram utilizados para modelar um sinal não-linear proveniente do eco usando o algoritmo *Normalised Least Mean Squares* (NLMS). Em [20], sinais estacionários e não estacionários que surgem a partir do modelo de Volterra foram estimados utilizando redes neurais; novamente, o modelo de Volterra de segunda ordem foi considerado suficiente para avaliação do efeito. Em [21], um estudo foi realizado com vários algoritmos combinados com filtro de Volterra para identificar sistemas não-lineares. Os algoritmos estudados foram: *Least Mean Squares* (LMS), NLMS, *Recursive Least Squares* (RLS), *affine projection* e *summation affine projection*; mais uma vez, apenas componentes não-lineares de até segunda ordem foram tratados [21].

Uma característica do filtro de Volterra é a tendência a gerar sistemas esparsos [17]. Sistemas esparsos, por definição, possuem muitos coeficientes nulos. O algoritmo LAR mostra-se útil ao ser usado em sistemas esparsos, pois estima, a cada iteração, um novo coeficiente, escolhido devido à sua influência no erro residual.

O modo de estimação de coeficientes motivou o desenvolvimento de um critério de parada para o algoritmo LAR, interrompendo-o antes de todos os coeficientes serem estimados. Ao interromper o algoritmo LAR na iteração N, N coeficientes são estimados e $J-N$ coeficientes permanecem iguais a zero, onde J é a quantidade total de coeficientes. Os N coeficientes estimados pelo algoritmo LAR são chamados de coeficientes ativos.

A interrupção do algoritmo LAR antes de todos os coeficientes serem estimados é vantajosa na identificação de sistemas esparsos, pois sabemos *a priori* que apenas $N << J$ coeficientes não-nulos precisam ser estimados. Esta análise foi o estudo inicial desta tese, apresentado em [22, 23], através da simulação de um sistema não-linear com o valor de N conhecido. Neste estudo foi concluído que, ao usar o algoritmo LAR com o filtro de Volterra, é possível identificar os coeficientes mais importantes em sistemas não-lineares, independente de suas posições no kernel, permitindo, assim, o uso de filtros de Volterra de ordens mais elevadas [23].

No entanto, na maior parte de casos reais, o valor de N é desconhecido e precisa ser determinado. Para tal, alguns critérios de seleção de modelo são conhecidos na literatura técnica. Nesta tese, são abordados os critérios *Akaike Information Criterion* (AIC) [24], *Bayesian Information Criterion* (BIC) [25] e *Mallows* C_p (C_p) [26]. Todos estes métodos são amplamente conhecidos e aplicados em diversas áreas, desde ecologia até tecnologia [27–31].

Os critérios AIC, BIC e C_p são baseados em funções custo *versus* benefício, e todas as ordens possíveis do modelo precisam ser testadas para, então, o ponto ótimo da função ser definido. Por isso, estes critérios necessitam que o algoritmo LAR iteraja até o final — somente após a estimação de todos os coeficientes é determinada sua quantidade ótima. Desta forma, estes critérios não usam a vantagem do algoritmo LAR de calcular um coeficiente por iteração.

A n-ésima iteração do algoritmo LAR ($n = 1, \cdots, J$) não é relacionada ao instante de tempo daquela amostra, mas sim relacionada à quantidade de coeficientes não-nulos selecionados até então. Na iteração n as seguintes informações são conhecidas:

1. o vetor de coeficientes estimado com n elementos, ou coeficientes, não-nulos;
2. o vetor de sinal de saída estimado;
3. a matriz de sinal de entrada; e
4. o vetor de sinal de saída de referência.

Utilizar o valor do erro médio quadrático (MSE) mínimo para determinar quando parar o algoritmo nem sempre é possível. Se o MSE mínimo não for conhecido, i.e., se o ruído do sistema não for conhecido, não é possível determinar se o MSE desejado foi atingido.

O critério de parada proposto nesta tese para o algoritmo LAR é chamado de critério de parada geométrico. O critério de parada geométrico, primeiro apresentado em [32], é baseado na matriz de sinal de entrada e no vetor de erro de estimação. O critério de parada geométrico foi desenvolvido através da observação das respostas de dois cenários de sistemas não-lineares simulados: um sistema não-linear polinomial de terceira ordem e um sistema não-linear polinomial de quinta ordem. O algoritmo LAR com o critério de parada geométrico é chamado de algoritmo GLAR, a primeira contribuição desta tese.

Para validar o algoritmo GLAR, os algoritmos *Least Squares* (LS), *Constrained Least Squares* (CLS) e *Subset Selection* (SSS) foram usados para avaliação de desempenho quanto ao erro nos coeficientes estimados e ao erro de estimação.

O algoritmo GLAR usa dados em lote, normalizados por coeficiente e não por tempo, impedindo o desenvolvimento de uma versão recursiva no tempo. Foi desenvolvido, então, um algoritmo que utiliza alternadamente o algoritmo GLAR e o algoritmo RLS, chamado de algoritmo GLAR-RLS. O algoritmo GLAR-RLS utiliza a indicação dos N coeficientes dada pelo algoritmo GLAR para estimá-los, recursivamente no tempo, através do algoritmo RLS.

A utilização do algoritmo GLAR-RLS em conjunto com o filtro de Volterra permite a identificação de sistemas não-lineares com modificações bruscas do vetor de

coeficientes, possuindo, assim, consciência situacional. Para validar o algoritmo GLAR-RLS, o algoritmo RLS foi usado para avaliação de desempenho quanto ao erro nos coeficientes estimados e ao erro de estimação.

Ao longo desta tese, letras minúsculas em negrito identificam vetores, como $\mathbf{x}$, $\mathbf{y}$ e $\mathbf{u}$, e letras maiúsculas em negrito identificam matrizes, como $\mathbf{X}$ e $\mathbf{M}$. As principais conclusões são resumidas ao final de cada capítulo.

O Capítulo 2 é uma revisão dos conceitos abordados: representação de sistemas não-lineares, incluindo filtros de Volterra; os algoritmos de estimação LAR, LS, CLS e SSS; e os critérios de seleção de modelos AIC, BIC e C_p.

O Capítulo 3 trata do algoritmo GLAR. Neste capítulo, o critério de parada geométrico é demonstrado e o algoritmo LAR é desenvolvido de modo recursivo por iteração. Na Seção 3.3, a identificação de sistemas não-lineares com o algoritmo GLAR em conjunto com o filtro de Volterra é avaliada utilizando diferentes cenários. Sistemas não-lineares do tipo polinômio de terceira ordem, polinômio de quinta ordem e tangente hiperbólica foram simulados.

O Capítulo 4 descreve o algoritmo GLAR-RLS. O algoritmo GLAR-RLS em conjunto com o filtro de Volterra é avaliado na identificação de sistemas não-lineares estacionários por partes na Seção 4.3. Os sistemas não-lineares são ditos estacionários por partes pois o vetor de coeficientes sofre alterações bruscas ao longo do tempo. Diferentes cenários são simulados para validação do algoritmo GLAR-RLS.

Finalmente, no Capítulo 5, as conclusões dos capítulos anteriores são reunidas, as contribuições desse trabalho apresentadas e trabalhos futuros são sugeridos.

Capítulo 2

Conceitos Básicos

2.1 Introdução do capítulo

O objetivo deste capítulo é apresentar conceitos, já conhecidos no estado da arte, necessários para o entendimento do trabalho desenvolvido. Para tal, três assuntos são abordados: representação de sistemas não-lineares, algoritmos para estimação dos coeficientes do sistema e critérios de seleção de modelos.

No processo de identificação de sistemas não-lineares, a primeira etapa é definir a sua representação – tema apresentado na primeira seção deste capítulo. Dentro desta abordagem, o filtro de Volterra é apresentado como a opção escolhida para a representação dos sistemas não-lineares a serem identificados.

A etapa seguinte no processo de identificação de sistemas compreende empregar um algoritmo para a estimação dos coeficientes do sistema. Para isso, algoritmos com o objetivo de minimizar o erro quadrático são apresentados na segunda seção deste capítulo. Os algoritmos aqui descritos são: o algoritmo *Least Squares* (LS), o algoritmo *Least Angle Regression* (LAR), e o algoritmo *Least Absolute Shrinkage and Selection Operator* (LASSO). O algoritmo LAR é apresentado com detalhes, uma vez que foi o objeto principal de estudo desta tese.

O último ponto abordado neste capítulo é a apresentação de critérios de seleção de modelos. Com este objetivo, os critérios *Akaike Information Criterion* (AIC), *Bayesian Information Criterion* (BIC) e *Mallows* C_p (C_p) são apresentados. Estes critérios são utilizados para comparar o desempenho do critério de parada geométrico proposto no Capítulo 3. No processo de identificação de sistemas, o critério de seleção de modelos pode ser aplicado em uma etapa posterior ou na mesma etapa da estimação dos coeficientes.

2.2 Representação de sistemas não-lineares

Sistemas dinâmicos encontrados na prática são, em última instância, não-lineares [1]. Muitas vezes estes sistemas podem ser aproximados por sistemas lineares, porém outras vezes não. A principal motivação para o uso de sistemas não-lineares é o fato de modelos não-lineares produzirem certos regimes dinâmicos que modelos lineares não conseguem representar [1]. Alto-falantes atuando próximo à saturação e supressores de eco são sistemas não-lineares típicos [16]. O objetivo desta seção é apresentar algumas representações de sistemas não-lineares, com ênfase no filtro de Volterra.

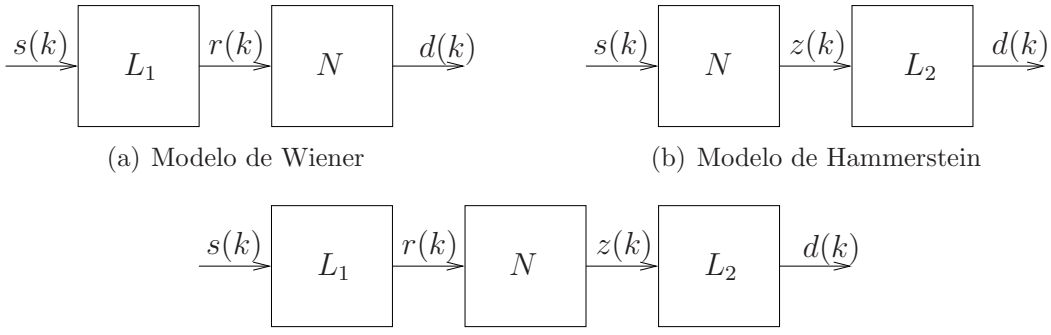
2.2.1 Modelos em cascata

Uma forma simples de representar sistemas não-lineares é usando modelos em cascata [33]. A Figura 2.1 apresenta os seguintes modelos em cascata [33]:

(a) LN – este modelo é composto por um filtro linear (L) seguido de um dispositivo não-linear (N) sem memória, conhecido como modelo de Wiener.

(b) NL – este modelo é composto por um dispositivo não-linear (N) sem memória seguido de um filtro linear (L), conhecido como modelo de Hammerstein.

(c) LNL – este modelo é composto por um filtro linear (L), seguido de um dispositivo não-linear (N) sem memória e de um segundo filtro linear (L).



(a) Modelo de Wiener

(b) Modelo de Hammerstein

(c) Modelo LNL

Figura 2.1: Representação de sistemas não-lineares em modo cascata.

Para o (primeiro) dispositivo linear, L_1, a saída de um sistema causal, discreto e invariante no tempo pode ser representada por

$$r(k) = h_{L_1}(1)s(k) + \cdots + h_{L_1}(m_{L_1})s(k - m_{L_1} + 1), \tag{2.1}$$

onde k é o índice temporal, $s(k)$ é o sinal de entrada no sistema, $r(k)$ é o sinal de saída do (primeiro) dispositivo linear, $m_{L_1} - 1$ é o tamanho de memória do

(primeiro) dispositivo linear, e $h_{L_1}(i), 1 \leq i \leq m_{L_1}$, é a resposta ao impulso do (primeiro) dispositivo linear.

Para o dispositivo linear L_2, a saída de um sistema causal, discreto e invariante no tempo pode ser representada por

$$d(k) = h_{L_2}(1)z(k) + \cdots + h_{L_2}(m_{L_2})z(k - m_{L_2} + 1), \tag{2.2}$$

onde k é o índice temporal, $z(k)$ é o sinal de entrada no (segundo) dispositivo linear, $d(k)$ é o sinal de saída de referência, $m_{L_2} - 1$ é o tamanho de memória do (segundo) dispositivo linear, e $h_{L_2}(i), 1 \leq i \leq m_{L_2}$, é a resposta ao impulso do (segundo) dispositivo linear.

Para os dispositivos não-lineares sem memória, assumidos polinomiais, a saída de cada modelo é aqui vista separadamente. A saída do modelo de Wiener pode ser representada em tempo discreto por

$$d(k) = h_N(1)r(k) + h_N(2)r^2(k) + \cdots + h_N(l)r^l(k), \tag{2.3}$$

enquanto que a saída do dispositivo não-linear do modelo de Hammerstein pode ser representada em tempo discreto por

$$z(k) = h_N(1)s(k) + h_N(2)s^2(k) + \cdots + h_N(l)s^l(k). \tag{2.4}$$

Já para o modelo LNL, a saída da não-linearidade sem memória pode ser representada em tempo discreto por

$$z(k) = h_N(1)r(k) + h_N(2)r^2(k) + \cdots + h_N(l)r^l(k). \tag{2.5}$$

Nas três últimas expressões, k é o índice temporal, $s(k)$ é o sinal de entrada no sistema, $r(k)$ é o sinal de saída do (primeiro) dispositivo linear, $z(k)$ é o sinal de entrada no (segundo) dispositivo linear, $d(k)$ é o sinal de saída de referência, l é a ordem de não-linearidade, e $h_N(i), 1 \leq i \leq l$, são os coeficientes do polinômio do dispositivo não-linear, i.e., $h_N(1)$ é o coeficiente linear, $h_N(2)$ é o coeficiente quadrático ou de segunda ordem, $h_N(3)$ é o coeficiente cúbico ou de terceira ordem, e assim sucessivamente.

Assim, a saída do modelo de Wiener, usando (2.1) e (2.3), é dada por

$$\begin{aligned} d(k) =& h_N(1)\left(h_{L_1}(1)s(k) + \cdots + h_{L_1}(m_{L_1})s(k - m_{L_1} + 1)\right) + \cdots + \\ & h_N(l)\left(h_{L_1}(1)s(k) + \cdots + h_{L_1}(m_{L_1})s(k - m_{L_1} + 1)\right)^l, \end{aligned} \tag{2.6}$$

enquanto que a saída do modelo de Hammerstein, usando (2.2) e (2.4), é dada por

$$\begin{aligned} d(k) =& h_{L_2}(1)\left(h_N(1)s(k) + \cdots + h_N(l)s^l(k)\right) + \cdots + \\ & h_{L_2}(m_{L_2})\left(h_N(1)s(k-m_{L_2}+1) + \cdots + h_N(l)s^l(k-m_{L_2}+1)\right); \end{aligned} \tag{2.7}$$

finalmente, a saída do modelo LNL, usando (2.1), (2.2) e (2.5), é dada por

$$\begin{aligned} d(k) =& h_{L_2}(1)\left(h_N(1)r(k) + \cdots + h_N(l)r^l(k)\right) + \cdots + \\ & h_{L_2}(m_{L_2})\left(h_N(1)r(k-m_{L_2}+1) + \cdots + h_N(l)r^l(k-m_{L_2}+1)\right), \\ =& h_{L_2}(1)h_N(1)\left(h_{L_1}(1)s(k) + \cdots + h_{L_1}(m_{L_1})s(k-m_{L_1}+1)\right) + \cdots + \\ & h_{L_2}(1)h_N(l)\left(h_{L_1}(1)s(k) + \cdots + h_{L_1}(m_{L_1})s(k-m_{L_1}+1)\right)^l + \cdots + \\ & h_{L_2}(m_{L_2})h_N(1)\left(h_{L_1}(1)s(k-m_{L_2}+1) + \cdots + h_{L_1}(m_{L_1})s(k-m_{L_1}-m_{L_2}+2)\right) + \cdots + \\ & h_{L_2}(m_{L_2})h_N(l)\left(h_{L_1}(1)s(k-m_{L_2}+1) + \cdots + h_{L_1}(m_{L_1})s(k-m_{L_1}-m_{L_2}+2)\right)^l. \end{aligned} \tag{2.8}$$

2.2.2 Filtros de Volterra

Outra forma de representar sistemas não-lineares é através do filtro de Volterra – quando as memórias dos dispositivos lineares de um modelo LNL são pequenas, o filtro de Volterra pode ser mais vantajoso devido à sua modularidade [33]. Os modelos de Wiener e Hammerstein são casos particulares do modelo LNL, que podem, por sua vez, ser representados através de um filtro de Volterra, como apresentado na Figura 2.2. Uma vantagem da modelagem através do sistema LNL é o número reduzido de coeficientes quando comparado ao filtro de Volterra. Entretanto, computar os coeficientes do modelo LNL pode ser bem mais complicado do que calcular os coeficientes usando um filtro de Volterra [17].

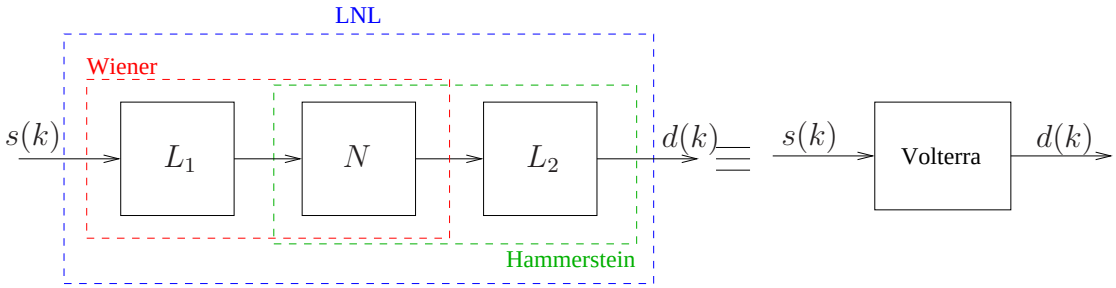


Figura 2.2: Nos modelos de Wiener, Hammerstein e LNL são necessários dois ou três dispositivos, enquanto que no filtro de Volterra apenas um dispositivo modela o sistema não-linear.

O filtro de Volterra descreve a relação de entrada e saída em um dispositivo não-linear com memória [17]. Um sistema não-linear causal, discreto e invariante no tempo é representado pela seguinte expansão da série Volterra [21]:

$$d(k) \;=\; h_0 + \sum_{m_1=0}^{\infty} h_1(m_1)s(k-m_1)$$

$$
\begin{aligned}
&+ \sum_{m_1=0}^{\infty} \sum_{m_2=0}^{\infty} h_2(m_1,m_2)s(k-m_1)s(k-m_2) \\
&+ \sum_{m_1=0}^{\infty} \sum_{m_2=0}^{\infty} \sum_{m_3=0}^{\infty} h_3(m_1,m_2,m_3)s(k-m_1)s(k-m_2)s(k-m_3) \\
&+ \ \cdots + \sum_{m_1=0}^{\infty} \sum_{m_2=0}^{\infty} \cdots \sum_{m_l=0}^{\infty} h_l(m_1,\cdots,m_l) \\
&s(k-m_1)s(k-m_2)\cdots s(k-m_l) + \ \cdots ,
\end{aligned} \tag{2.9}
$$

onde k é o índice temporal, $d(k)$ é o sinal de saída de referência, $s(k)$ é o sinal de entrada, l é a ordem do filtro, $m_{1,2,\cdots,l}$ são as memórias do filtro, e $h_l(m_1,\cdots,m_l)$ são os l-ésimos coeficientes do *kernel* Volterra, i.e., h_0 é o termo independente (*bias*, ou componente DC), $h_1(m_1)$ são os coeficientes lineares ou de primeira ordem, $h_2(m_1,m_2)$ são os coeficientes quadráticos ou de segunda ordem, $h_3(m_1,m_2,m_3)$ são os coeficientes cúbicos ou de terceira ordem, e assim por diante [20].

Observamos em (2.9) que o número de coeficientes $h_l(m_1,\cdots,m_l)$ no *kernel* Volterra pode ser infinito, bem como a memória $m_{1,2,\cdots,l}$ pode ser infinita. Na prática, o filtro de Volterra é truncado, i.e., com a ordem e o número de coeficientes finitos. O número de coeficientes no *kernel*, em uma representação truncada da série Volterra, para uma ordem l e memória finita m ($m_{1,2,\cdots,l} = 0,1,\cdots,m$), é calculado por [34]

$$
J = \left(\begin{array}{c} l+m+1 \\ l \end{array} \right) = \frac{(l+m+1)!}{l!(m+1)!}, \tag{2.10}
$$

onde J é o total de coeficientes, l é a ordem do filtro de Volterra, dado pelo fator de não-linearidade, e m é o tamanho de memória do filtro de Volterra, também conhecido como ordem dinâmica.

Um exemplo da série Volterra truncada para um filtro de segunda ordem ($l = 2$) com tamanho de memória unitário ($m = 1$) é dado por

$$
\begin{aligned}
d(k) &= h_0 + \sum_{m_1=0}^{1} h_1(m_1)s(k-m_1) + \\
&\quad \sum_{m_1=0}^{1} \sum_{m_2=0}^{1} h_2(m_1,m_2)s(k-m_1)s(k-m_2) \\
&= h_0 + h_1(0)s(k) + h_1(1)s(k-1) + h_2(0,0)s(k)s(k) + \\
&\quad h_2(0,1)s(k)s(k-1) + h_2(1,0)s(k-1)s(k) + \\
&\quad h_2(1,1)s(k-1)s(k-1) \\
&= h_0 + h_1(0)s(k) + h_1(1)s(k-1) + h_2(0,0)s^2(k) + \\
&\quad 2h_2(0,1)s(k)s(k-1) + h_2(1,1)s^2(k-1),
\end{aligned} \tag{2.11}
$$

considerando $h_2(0,1) = h_2(1,0)$, uma vez que ambos os termos multiplicam $s(k)s(k-1)$.

Podemos utilizar notação matricial para representar o filtro de Volterra: desconsiderando o termo constante h_0 (não é de interesse na área em pesquisa), (2.11) pode ser expressa por

$$d(k) = \mathbf{h}^{\mathrm{T}}\mathbf{s}(k), \tag{2.12}$$

onde

$$\mathbf{s}(k) = [s(k) \;\; s(k-1) \;\; s^2(k) \;\; s(k)s(k-1) \;\; s^2(k-1)]^{\mathrm{T}} \tag{2.13}$$

é o vetor do sinal de entrada, e

$$\mathbf{h} = [h_1(0) \;\; h_1(1) \;\; h_2(0,0) \;\; 2h_2(0,1) \;\; h_2(1,1)]^{\mathrm{T}} \tag{2.14}$$

é o vetor de coeficientes. Este exemplo é ilustrado na Figura 2.3.

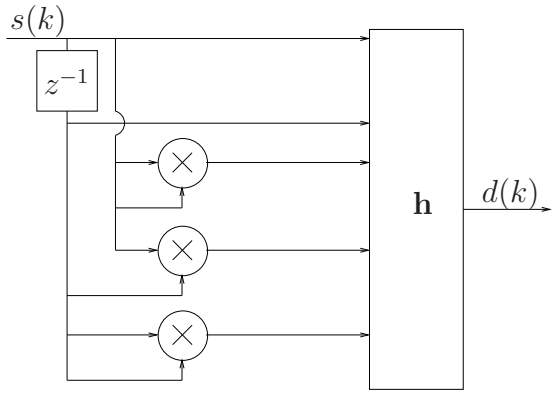


Figura 2.3: Diagrama ilustrativo de um sinal de entrada em um filtro de segunda ordem com memória unitária.

Refazendo este exemplo, agora numericamente, seja

$$\mathbf{h} = [0 \;\; 0 \;\; 1 \;\; -1 \;\; 0{,}25]^{\mathrm{T}}.$$

O sinal de saída, dado por (2.12), é igual a

$$d(k) = s^2(k) - s(k)s(k-1) + 0{,}25s^2(k-1).$$

O sistema resultante é, de fato, um sistema de Wiener (LN), como apresentado na Figura 2.4.

O número total de coeficientes no *kernel* Volterra é dado por (2.10)

$$J = \frac{(2+1+1)!}{2!(1+1)!} = \frac{24}{4} = 6.$$

Desprezando a componente DC, dos 5 (cinco) coeficientes possíveis no *kernel*

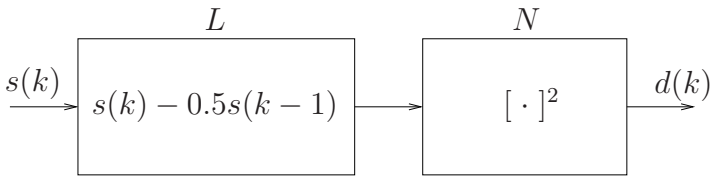


Figura 2.4: Exemplo de um sistema de Wiener criado a partir das equações de uma série Volterra truncada de segunda ordem com tamanho de memória unitário.

Volterra, apenas três são diferentes de zero. Este exemplo já indica que o uso de filtros de Volterra resulta, muitas vezes, em sistemas com modelos esparsos.

Nesta trabalho, como será visto no Capítulo 3 e Capítulo 4, dois cenários de sistemas não-lineares são simulados. No primeiro, um filtro de Volterra de terceira ordem ($l = 3$) com tamanho de memória igual a quatro ($m = 4$) é utilizado para identificação dos sistemas. No segundo, um filtro de Volterra de quinta ordem ($l = 5$) com tamanho de memória igual a seis ($m = 6$) é utilizado para identificação de sistemas não-lineares. Consequentemente, o número total de coeficientes no *kernel*, calculado por (2.10), é dado por

$$\text{Cenário 1: } J = \frac{(3+4+1)!}{3!(4+1)!} = 56, \tag{2.15}$$

$$\text{Cenário 2: } J = \frac{(5+6+1)!}{5!(6+1)!} = 792. \tag{2.16}$$

Como a componente DC não foi levada em consideração, o total de coeficientes estimado foi 55 e 791 para o primeiro e segundo cenários, respectivamente. Outros valores de l e m são possíveis; os valores aqui utilizados resultam em uma grande quantidade de coeficientes – o suficiente para avaliar os algoritmos propostos, dentro da espectativa da quantidade de coeficientes necessária para modelar sistemas reais.

O principal problema ao usar um filtro adaptativo baseado no modelo de Volterra é que, geralmente, não é prático, seja porque leva muito tempo para convergir, quando usado com algoritmos do tipo *Least Mean Squares* (LMS), seja porque possui muitos coeficientes, e, portanto, muito complexo, quando usado com algoritmos do tipo *Least Squares* (LS) [17]. Entretanto, modelar um sistema não-linear LNL utilizando um filtro de Volterra gera um grande número de coeficientes nulos, como visto no exemplo acima. Devido a esta característica de esparsidade, é vantajoso combinar filtros de Volterra com o algoritmo LAR, como descrito no Capítulo 3 e no Capítulo 4.

2.3 Algoritmos de Estimação

Na seção anterior foram apresentadas algumas modelagens usadas para sistemas não-lineares, com ênfase no modelo de Volterra. A etapa seguinte para a identificação de um sistema não-linear é a determinação dos coeficientes do modelo adotado. Dentre os diversos algoritmos para estimação de coeficientes, o desenvolvimento desta tese foi baseado no algoritmo *Least Angle Regression* (LAR). Para fazer uma comparação de desempenho do algoritmo LAR em combinação com o filtro de Volterra para identificação de sistemas não-lineares, outros quatro algoritmos foram estudados: *Least Squares* (LS), *Subset Selection* (SSS), *Constrained Least Squares* (CLS), e *Least Absolute Shrinkage and Selection Operator* (LASSO). O objetivo desta seção é apresentar todos os algoritmos analisados.

2.3.1 Algoritmos LS – *Least Squares*

Da família dos algoritmos de mínimos quadrados (LS), três algoritmos foram estudados: LS, SSS e CLS.

Algoritmo LS

O algoritmo LS foi primeiramente desenvolvido por Carl Friedrich Gauss em 1795, porém apresentado por Legendre somente em 1805 [35]. O objetivo do algoritmo LS é encontrar uma solução que minimiza a soma dos erros ao quadrado [6, 36]. A origem da ideia pode ser encontrada nos trabalhos de Gauss sobre estudos astronômicos [1].

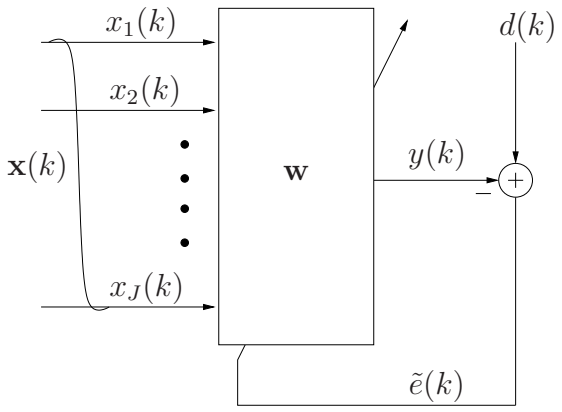


Figura 2.5: Configuração básica para identificação de sistemas. Note que $y(k) = \mathbf{w}^{\mathrm{T}}\mathbf{x}(k)$, de modo que o erro de estimação seja $\tilde{e}(k) = d(k) - \mathbf{w}^{\mathrm{T}}\mathbf{x}(k)$.

A Figura 2.5 apresenta a configuração básica (genérica) para identificação de um sistema por um filtro linear. Seja

$$\mathbf{w} = [w_1 \ \cdots \ w_j \ \cdots \ w_J]^{\mathrm{T}} \tag{2.17}$$

o vetor de coeficientes, onde $1 \leq j \leq J$ é o índice do coeficiente, e

$$\mathbf{X} = [\mathbf{x}(1) \ \cdots \ \mathbf{x}(k) \ \cdots \ \mathbf{x}(K)] \tag{2.18}$$

a matriz de sinal de entrada $(J \times K)$, onde

$$\mathbf{x}(k) = [x_1(k) \ \cdots \ x_j(k) \ \cdots \ x_J(k)]^{\mathrm{T}} \tag{2.19}$$

é o vetor de sinal de entrada do tempo k. O sinal de saída estimado do tempo 1 ao tempo K, colocado num vetor $\mathbf{y}$, é dado por

$$\mathbf{y} = \mathbf{X}^{\mathrm{T}}\mathbf{w}. \tag{2.20}$$

Considerando ainda $\mathbf{d}$ como o vetor de sinal de saída desejado (do tempo 1 ao tempo K), a solução LS é dada pela otimização da função objetivo [7]

$$\min_{\mathbf{w}} \|\mathbf{d} - \mathbf{y}\|^2. \tag{2.21}$$

Substituindo (2.20) em (2.21), podemos escrever a função custo por

$$\varepsilon = \|\mathbf{d} - \mathbf{X}^{\mathrm{T}}\mathbf{w}\|^2 = \|\mathbf{e}\|^2. \tag{2.22}$$

A solução ótima é obtida quando fazemos $\nabla_{\mathbf{w}}\varepsilon = \mathbf{0}$. Calculando o gradiente da função custo encontramos

$$\begin{aligned}
\nabla_{\mathbf{w}}\varepsilon &= \nabla_{\mathbf{w}}\|\mathbf{e}\|^2 = \nabla_{\mathbf{w}}\{\mathbf{e}^{\mathrm{T}}\mathbf{e}\} \\
&= \nabla_{\mathbf{w}}\{[\mathbf{d}^{\mathrm{T}} - \mathbf{w}^{\mathrm{T}}\mathbf{X}][\mathbf{d} - \mathbf{X}^{\mathrm{T}}\mathbf{w}]\} \\
&= -\nabla_{\mathbf{w}}\{(\mathbf{X}\mathbf{d})^{\mathrm{T}}\mathbf{w}\} - \nabla_{\mathbf{w}}\{\mathbf{w}^{\mathrm{T}}(\mathbf{X}\mathbf{d})\} + \nabla_{\mathbf{w}}\{\mathbf{w}^{\mathrm{T}}\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{w}\}.
\end{aligned}$$

Sabendo que $\nabla_{\mathbf{w}}\mathbf{w}^{\mathrm{T}}\mathbf{a} = \nabla_{\mathbf{w}}\mathbf{a}^{\mathrm{T}}\mathbf{w} = \mathbf{a}$ e que $\nabla_{\mathbf{w}}\{\mathbf{w}^{\mathrm{T}}\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{w}\} = 2\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{w}$, uma vez que $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ é simétrica, chegamos a

$$\nabla_{\mathbf{w}}\varepsilon = -2\mathbf{X}\mathbf{d} + 2\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{w}, \tag{2.23}$$

de onde obtemos, com $\nabla_{\mathbf{w}}\varepsilon = \mathbf{0}$, a resposta LS:

$$\mathbf{w}_{LS} = (\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{X}\mathbf{d}, \tag{2.24}$$

assumindo que seja possível calcular $(\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}$.

Ao utilizar o algoritmo LS para modelar sistemas esparsos, devido a estimação

dos coeficientes nulos *a priori*, o resultado do vetor de coeficientes possui baixo *bias*, porém alta variância [6]. Uma forma de melhorar a acurácia é anular (ou diminuir) alguns coeficientes; desta forma, o *bias* é sacrificado de modo a reduzir a variância dos valores estimados e, assim, melhorar a estimação em geral [6]. Por este motivo, dois algoritmos que selecionam subgrupos da resposta LS foram analisados: o *Subset Selection* (SSS) [6] e o *Constrained Least Squares* (CLS) [17]. No algoritmos SSS, coeficientes de menor magnitude assumem valores iguais a zero. No algoritmo CLS, é utilizada uma matriz de restrição para forçar que determinados coeficientes sejam nulos.

Algoritmo SSS

O algoritmo SSS considera a quantidade $\bar{N}$ de coeficientes nulos conhecida. Este algoritmo utiliza a resposta LS, inserindo $\bar{N}$ zeros nos coeficientes de menor valor absoluto [6]. Isto pode ser descrito por

$$[\mathbf{w}_{SSS}]_j = \begin{cases} 0, & \text{para os } \bar{N} \text{ menores valores absolutos de } \mathbf{w}_{LS} \\ [\mathbf{w}_{LS}]_j, & \text{caso contrário.} \end{cases} \tag{2.25}$$

Algoritmo CLS

O algoritmo CLS considera conhecida, além da quantidade $\bar{N}$ de coeficientes nulos, as suas posições dentro do modelo. O algoritmo utiliza a chamada matriz de restrição $\mathbf{C}$ (*constraint matrix*, daí o nome do algoritmo) para colocar zeros nas posições correspondentes daqueles coeficientes dados.

A matriz $\mathbf{C}$ é composta por J (quantidade total de coeficientes) linhas e $\bar{N}$ (quantidade total de coeficientes nulos) colunas. Em cada coluna apenas uma posição possui o valor 1: naquela em que é desejado anular o coeficiente; o restante possui valor 0. Por exemplo, seja um sistema com cinco coeficientes, onde sabemos que os coeficientes w_2 e w_5 são nulos. A matriz de restrição é

$$\mathbf{C} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}. \tag{2.26}$$

A função objetivo do algoritmo CLS é dada por [17])

$$\min_{\mathbf{w}} \|\mathbf{d} - \mathbf{y}\|^2 \text{ s.t. } \mathbf{C}^{\mathrm{T}}\mathbf{w} = \mathbf{0}. \tag{2.27}$$

Podemos observar que, usando a matriz $\mathbf{C}$ do nosso exemplo e $\mathbf{w} = [w_1 \cdots w_5]^{\mathrm{T}}$, ao fazermos $\mathbf{C}^{\mathrm{T}}\mathbf{w} = 0$, iremos forçar $w_2 = w_5 = 0$.

A nova função custo, usando o vetor de multiplicador de Lagrange $\boldsymbol{\lambda}$, é dada por

$$\varepsilon = \|\mathbf{d} - \mathbf{X}^{\mathrm{T}}\mathbf{w}\|^2 + \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{C}^{\mathrm{T}}\mathbf{w}. \tag{2.28}$$

Para encontrar a solução da função objetivo, devemos calcular o gradiente da função custo,

$$\nabla_{\mathbf{w}}\varepsilon = \nabla_{\mathbf{w}}\{[\mathbf{d}^{\mathrm{T}} - \mathbf{w}^{\mathrm{T}}\mathbf{X}][\mathbf{d} - \mathbf{X}^{\mathrm{T}}\mathbf{w}]\} + \nabla_{\mathbf{w}}\{\boldsymbol{\lambda}^{\mathrm{T}}\mathbf{C}^{\mathrm{T}}\mathbf{w}\}. \tag{2.29}$$

Usando a resposta encontrada anteriormente (2.23) para a primeira parte da equação (2.29), encontramos

$$\nabla_{\mathbf{w}}\varepsilon = -2\mathbf{X}\mathbf{d} + 2\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{w} + \mathbf{C}\boldsymbol{\lambda}. \tag{2.30}$$

Fazendo $\nabla_{\mathbf{w}}\varepsilon = 0$, encontramos

$$\mathbf{w} = (\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\left(\mathbf{X}\mathbf{d} - \frac{\mathbf{C}\boldsymbol{\lambda}}{2}\right). \tag{2.31}$$

Usando a restrição $\mathbf{C}^{\mathrm{T}}\mathbf{w} = \mathbf{0}$,

$$\begin{aligned}
\mathbf{C}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\left(\mathbf{X}\mathbf{d} - \frac{\mathbf{C}\boldsymbol{\lambda}}{2}\right) &= \mathbf{0} \\
\frac{1}{2}\mathbf{C}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{C}\boldsymbol{\lambda} &= \mathbf{C}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{X}\mathbf{d} \\
\boldsymbol{\lambda} &= 2[\mathbf{C}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{C}]^{-1}\mathbf{C}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{X}\mathbf{d}.
\end{aligned} \tag{2.32}$$

Substituindo (2.32) em (2.31), obtemos

$$\begin{aligned}
\mathbf{w} &= (\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\left(\mathbf{X}\mathbf{d} - \frac{2\mathbf{C}[\mathbf{C}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{C}]^{-1}\mathbf{C}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{X}\mathbf{d}}{2}\right) \\
&= (\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{X}\mathbf{d} - (\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{C}[\mathbf{C}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{C}]^{-1}\mathbf{C}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{X}\mathbf{d},
\end{aligned} \tag{2.33}$$

de onde, usando a resposta LS (2.24), encontramos a resposta CLS:

$$\mathbf{w}_{CLS} = \mathbf{w}_{LS} - (\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{C}[\mathbf{C}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{C}]^{-1}\mathbf{C}^{\mathrm{T}}\mathbf{w}_{LS}. \tag{2.34}$$

Exemplo

Para ficar clara a diferença entre os algoritmos LS, SSS e CLS, seja o exemplo a seguir. Em um sistema com o total de coeficientes a serem estimados $J = 5$ possui, no tempo $K = 6$, a matriz de sinal de entrada e o sinal desejado dados por

$$\mathbf{X} = \begin{bmatrix} 4.00 & 2.15 & 3.00 & 3.30 & 3.40 & 2.90 \\ 5.00 & 2.65 & 5.10 & 2.80 & 3.50 & 3.40 \\ 5.50 & 2.90 & 5.50 & 3.30 & 3.80 & 3.70 \\ 6.00 & 3.15 & 5.50 & 4.00 & 3.90 & 4.30 \\ 7.60 & 3.95 & 7.30 & 4.00 & 3.70 & 3.90 \end{bmatrix} \quad \text{e} \quad \mathbf{d} = \begin{bmatrix} 6.70 \\ 7.80 \\ 5.60 \\ 8.80 \\ 6.80 \\ 7.30 \end{bmatrix}.$$

Seja também conhecido que os coeficientes w_2 e w_5 são nulos.

A resposta da estimação dos coeficientes do sistema, quando aplicado cada algoritmo, é dada por

$$\mathbf{w}_{LS} = (\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{X}\mathbf{d} = \begin{bmatrix} 1.81 \\ -2.08 \\ -0.18 \\ 2.69 \\ -0.44 \end{bmatrix},$$

$$[\mathbf{w}_{SSS}]_n = \begin{cases} 0, & \text{para os } \bar{N} \text{ menores valores absolutos de } [\mathbf{w}_{LS}]_n \\ [\mathbf{w}_{LS}]_n, & \text{caso contrário} \end{cases} = \begin{bmatrix} 1.81 \\ -2.08 \\ 0.00 \\ 2.69 \\ 0.00 \end{bmatrix},$$

$$\mathbf{w}_{CLS} = \mathbf{w}_{LS} - (\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{C}[\mathbf{C}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{C}]^{-1}\mathbf{C}^{\mathrm{T}}\mathbf{w}_{LS} = \begin{bmatrix} 2.33 \\ 0.00 \\ -2.92 \\ 2.60 \\ 0.00 \end{bmatrix}.$$

Note que o algoritmo SSS corretamente atribuiu zero à w_5, porém w_3 também foi anulado, erradamente. O algoritmo CLS, por sua vez, atribuiu zero aos coeficientes w_2 e w_5 (tínhamos este conhecimento *a priori*).

2.3.2 Algoritmo LAR – *Least Angle Regression*

O algoritmo LAR foi proposto em 2004 por Efron et al. [5]. A compreensão completa do algoritmo não é trivial; na primeira apresentação feita pelos autores, em [5], a justificativa de todas as suas equações e o comportamento gráfico dos vetores não

são apresentados.

Nos anos seguintes vários estudos sobre o algoritmo LAR foram realizados [6–10]. Dentre estes, dois se destacam: Berghen [8] apresentou graficamente o comportamento do algoritmo LAR, bem como equações simples e diretas para a normalização dos dados; e Khan et al. [9] descreveu detalhadamente como as equações do algoritmo são obtidas.

A nomenclatura utilizada para descrever o algoritmo LAR deve ser claramente definida, caso contrário o pleno entendimento pode ser prejudicado. As equações principais serão aqui apresentadas, buscando uma proximidade da notação utilizada na área de processamento de sinais, com base na Figura 2.6, onde $\tilde{\mathbf{w}}$ $(J \times 1)$ é o vetor de coeficientes. Devido à normalização necessária do algoritmo LAR, explicada mais adiante nesta seção, vetores e matrizes com dados não normalizados recebem o símbolo "˜" no topo de cada variável, por exemplo em $\tilde{\mathbf{w}}$ e $\tilde{\mathbf{x}}$.

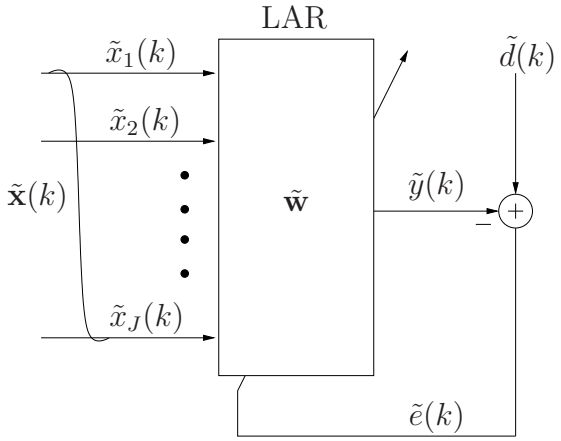


Figura 2.6: Configuração básica de um modelo linear usado em identificação de sistemas. Note que $\tilde{y}(k) = \tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}(k)$ e $\tilde{d}(k)$ é o sinal desejado, de modo que o erro de estimação seja $\tilde{e}(k) = \tilde{d}(k) - \tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}(k)$.

O vetor de sinal de entrada $(J \times 1)$ do tempo k é definido por

$$\tilde{\mathbf{x}}(k) = [\tilde{x}_1(k) \ \cdots \ \tilde{x}_j(k) \ \cdots \ \tilde{x}_J(k)]^{\mathrm{T}}, \tag{2.35}$$

onde $k = 1,2,\cdots,K$, é o índice temporal, e $j = 1,2,\cdots,J$ é o índice do canal. Agrupando todas as amostras de entrada de 1 a K, a matriz de sinal de entrada $(J \times K)$ é definida por

$$\tilde{\mathbf{X}} \ = \ [\tilde{\mathbf{x}}(1) \ \cdots \ \tilde{\mathbf{x}}(k) \ \cdots \ \tilde{\mathbf{x}}(K)]$$

$$= \begin{bmatrix} \tilde{x}_1(1) & \cdots & \tilde{x}_1(k) & \cdots & \tilde{x}_1(K) \\ \vdots & & \vdots & & \vdots \\ \tilde{x}_j(1) & \cdots & \tilde{x}_j(k) & \cdots & \tilde{x}_j(K) \\ \vdots & & \vdots & & \vdots \\ \tilde{x}_J(1) & \cdots & \tilde{x}_J(k) & \cdots & \tilde{x}_J(K) \end{bmatrix}, \tag{2.36}$$

i.e., cada coluna corresponde a um vetor de sinal de entrada para um dado tempo k.

O algoritmo LAR utiliza essa matriz de outro ponto de vista. Seja o canal j, de agora em diante chamado de coeficiente j, com todas as suas amostras; podemos rescrever a matriz do sinal de entrada como

$$\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1 \ \cdots \ \tilde{\mathbf{x}}_j \ \cdots \ \tilde{\mathbf{x}}_J]^{\mathrm{T}}, \tag{2.37}$$

onde

$$\tilde{\mathbf{x}}_j = [\tilde{x}_j(1) \ \cdots \ \tilde{x}_j(k) \ \cdots \ \tilde{x}_j(K)]^{\mathrm{T}} \tag{2.38}$$

é o vetor $(K \times 1)$ de dados do coeficiente j. $\tilde{x}_j(k)$ é o sinal de entrada do coeficiente j no instante k.

Todos os K instantes são também agrupados no vetor de sinal de saída desejado definido por

$$\tilde{\mathbf{d}} = \left[\tilde{d}(1) \ \cdots \ \tilde{d}(k) \ \cdots \tilde{d}(K)\right]^{\mathrm{T}}. \tag{2.39}$$

Os únicos dados de entrada do algoritmo LAR são estes definidos acima, a matriz do sinal de entrada $\tilde{\mathbf{X}}$ e o vetor de sinal de saída desejado $\tilde{\mathbf{d}}$. Estes dados devem ser normalizados [5] antes do início do algoritmo propriamente dito, como apresentado a seguir. Consequentemente, o resultado do vetor de coeficientes estimado é normalizado, e deve então ser transformado para a sua condição original, como explicado em seguida. O algoritmo LAR é detalhado na última parte desta seção.

Normalização dos dados de entrada

A normalização dos dados imposta pelo algoritmo LAR busca garantir ao algoritmo LAR que seus dados de entrada satisfaçam as seguintes condições, sendo $j = 1, \cdots, J$ o índice de coeficientes e $k = 1, \cdots, K$ o índice temporal:

$$\sum_{k=1}^{K} \tilde{x}_j(k) = 0; \tag{2.40}$$

$$\sum_{k=1}^{K} \tilde{x}_j^2(k) = 1; \tag{2.41}$$

$$\sum_{k=1}^{K} \tilde{d}(k) = 0; \text{ e} \tag{2.42}$$

$$\sum_{k=1}^{K} \tilde{d}^2(k) = 1. \tag{2.43}$$

Isto é, o vetor de dados de cada coeficiente j ($\tilde{\mathbf{x}}_j, j = 1, \cdots, J$) deve ter média zero (2.40) e norma quadrada unitária (2.41), bem como o vetor de sinal de saída deve ter média zero (2.42). A última condição (2.43) não é necessária originalmente, i.e., não é apresentada no artigo original do algoritmo LAR [5]; no entanto, além de aumentar a estabilidade numérica, como sugerido por Berghen [8], auxilia no desenvolvimento do critério de parada do algoritmo proposto, como será visto mais adiante.

Sendo assim, sempre que os dados originais ($\tilde{\mathbf{x}}_j$ e $\tilde{\mathbf{d}}$) não satisfizerem (2.40)–(2.43), eles devem ser normalizados para $\mathbf{x}_j$ e $\mathbf{d}$, de modo a obter tais condições.

Para tal, o vetor de dados normalizado do coeficiente j ($K \times 1$) é dado por

$$\mathbf{x}_j = \frac{\tilde{\mathbf{x}}_j - m_{\tilde{\mathbf{x}}_j}}{n_{\mathbf{x}_j}}, j = 1, \cdots, J \tag{2.44}$$

onde

$$m_{\tilde{\mathbf{x}}_j} = \frac{\sum_{k=1}^{K} \tilde{x}_j(k)}{K} \tag{2.45}$$

é a média dos elementos do vetor de dados do coeficiente j e

$$n_{\mathbf{x}_j} = ||\tilde{\mathbf{x}}_j - m_{\tilde{\mathbf{x}}_j}|| \tag{2.46}$$

é o comprimento do vetor de dados já com média nula do coeficiente j. Definindo o vetor de média de todos os coeficientes ($J \times 1$) por

$$\mathbf{m}_{\tilde{\mathbf{x}}} = \begin{bmatrix} m_{\tilde{\mathbf{x}}_1} & \cdots & m_{\tilde{\mathbf{x}}_j} & \cdots & m_{\tilde{\mathbf{x}}_J} \end{bmatrix}^{\mathrm{T}}, \tag{2.47}$$

podemos agrupar as médias de todas as amostras de entrada, de 1 a K, na matriz $\mathbf{M}_{\tilde{\mathbf{x}}}$ ($K \times J$) definida por

$$\begin{aligned} \mathbf{M}_{\tilde{\mathbf{x}}} &= \begin{bmatrix} \mathbf{m}_{\tilde{\mathbf{x}}} & \cdots \mathbf{m}_{\tilde{\mathbf{x}}} \end{bmatrix}^{\mathrm{T}} \\ &= \begin{bmatrix} m_{\tilde{\mathbf{x}}_1} & \cdots & m_{\tilde{\mathbf{x}}_J} \\ \vdots & \ddots & \vdots \\ m_{\tilde{\mathbf{x}}_1} & \cdots & m_{\tilde{\mathbf{x}}_J} \end{bmatrix}. \end{aligned} \tag{2.48}$$

Definindo também a matriz diagonal $\mathbf{N_x}$ $(J \times J)$ por

$$\mathbf{N_x} = \begin{bmatrix} n_{\mathbf{x}_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & n_{\mathbf{x}_J} \end{bmatrix}, \tag{2.49}$$

e considerando a definição em (2.37), a matriz de sinal de entrada normalizada é dada por

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{x}_J^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \dfrac{\tilde{\mathbf{x}}_1^{\mathrm{T}} - m_{\tilde{\mathbf{x}}_1}}{n_{\mathbf{x}_1}} \\ \vdots \\ \dfrac{\tilde{\mathbf{x}}_J^{\mathrm{T}} - m_{\tilde{\mathbf{x}}_J}}{n_{\mathbf{x}_J}} \end{bmatrix} = \mathbf{N}_{\mathbf{x}}^{-1} \left(\tilde{\mathbf{X}} - \mathbf{M}_{\tilde{\mathbf{x}}} \right). \tag{2.50}$$

De modo similar, o vetor de sinal de saída desejado normalizado é dado por

$$\mathbf{d} = \frac{\tilde{\mathbf{d}} - m_{\tilde{\mathbf{d}}}}{n_{\mathbf{d}}} \tag{2.51}$$

onde

$$m_{\tilde{\mathbf{d}}} = \frac{\sum_{k=1}^{K} \tilde{d}(k)}{K} \tag{2.52}$$

é a média dos elementos do vetor de sinal de saída desejado e

$$n_{\mathbf{d}} = ||\tilde{\mathbf{d}} - m_{\tilde{\mathbf{d}}}|| \tag{2.53}$$

é o comprimento do vetor de sinal de saída já com média nula.

O processo de normalização está representado na Figura 2.7.

O novo sistema com o sinal normalizado irá resultar em um vetor de coeficientes relativo aos dados normalizados, gerando $\mathbf{w}$ ao invés de $\tilde{\mathbf{w}}$. A seguir, a transformação necessária para retornar o valor de $\mathbf{w}$ para $\tilde{\mathbf{w}}$ é explicada.

Desnormalização do vetor de coeficientes

Como visto no tópico anterior, os dados de entrada do algoritmo devem, sempre que não obedecerem (2.40)–(2.43), ser normalizados antes de iniciar o algoritmo. Assim, também temos que encontrar a transformação necessária para o vetor de coeficientes, uma vez que encontramos, com dados de entrada normalizados, o vetor de coeficientes normalizado ($\mathbf{w}$), enquanto queremos conhecê-lo em sua condição original ($\tilde{\mathbf{w}}$).

O modelo do sistema desconhecido a ser estimado é indicado na Figura 2.8, onde

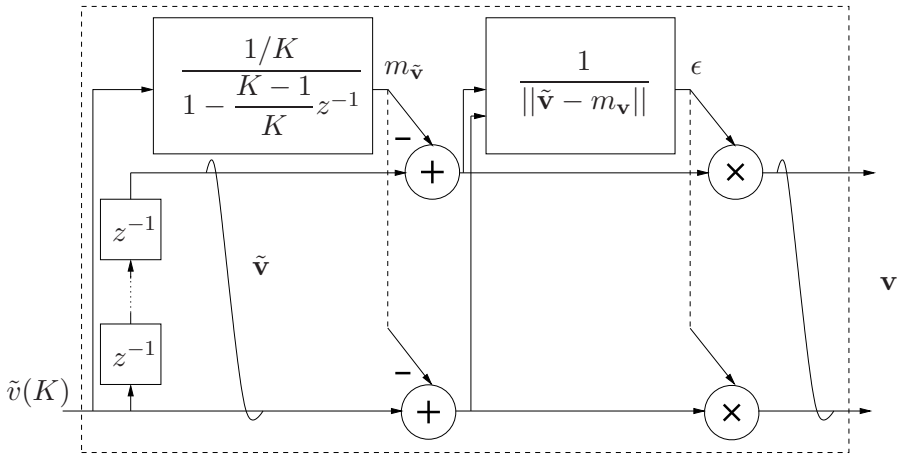


Figura 2.7: Bloco de normalização necessário pelo algoritmo LAR. O sinal de entrada, aqui representado por $\tilde{v}(k)$, pode ser $\tilde{x}_j(k), 1 \leq j \leq J$, ou $\tilde{d}(k)$; consequentemente, o vetor de sinal de saída, aqui representado por $\mathbf{v}$, é $\mathbf{x}_j, 1 \leq j \leq J$, ou $\mathbf{d}$.

uma componente DC (constante D) é incluída para possibilitar sinais de entrada e de saída com médias diferentes de zero e comprimentos não unitários. Na Figura 2.8, o sinal a ser estimado é dado por

$$\tilde{y}_o(k) = \tilde{\mathbf{w}}_o^{\mathrm{T}} \tilde{\mathbf{x}}(k) + D. \tag{2.54}$$

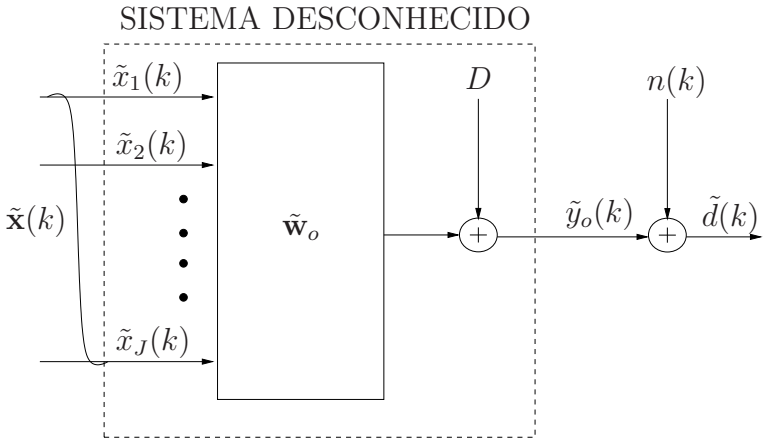


Figura 2.8: Modelo do sistema desconhecido quando assumimos dados com médias diferentes de zero e comprimentos não unitários. Note que $n(k)$ é o ruído (aditivo) de observação.

O modelo linear do vetor de coeficientes $\tilde{\mathbf{w}}$, estimativa de $\tilde{\mathbf{w}}_o$, e o valor constante D devem ser encontrados de modo a termos uma boa predição de $\tilde{y}_o(k)$. A Figura 2.9 indica como obtermos a predição de $\tilde{y}_o(k)$ a partir dos coeficientes não-normalizados.

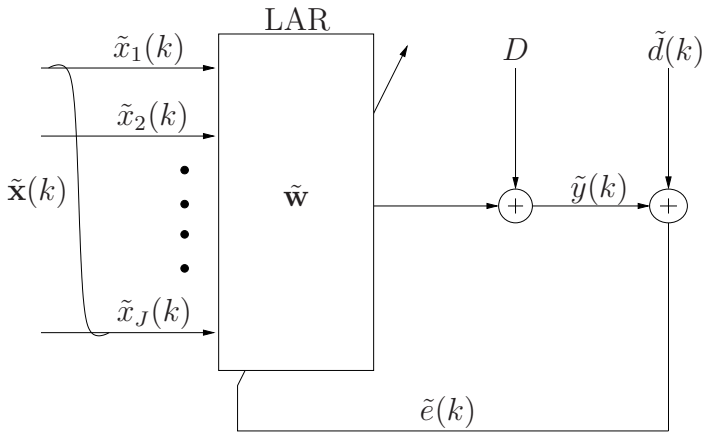


Figura 2.9: Estimativa da saída do sistema desconhecido pelo algoritmo LAR. Observe que $\tilde{y}(k)$ é uma predição de $\tilde{y}_o(k)$ da Figura 2.8.

O vetor de coeficientes $\tilde{\mathbf{w}}$, do ponto de vista determinístico, é obtido quando a média do erro é igual a zero e sua variância é minimizada. Para tal, fazemos

$$\begin{aligned} m_{\tilde{\mathbf{e}}} &= \frac{\sum_{k=1}^{K}(\tilde{d}(k)-\tilde{y}(k))}{K} \\ &= \frac{\sum_{k=1}^{K}(\tilde{d}(k)-\tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}(k)-D)}{K} \\ &= \frac{\sum_{k=1}^{K}\tilde{d}(k)}{K} - \frac{\tilde{\mathbf{w}}^{\mathrm{T}}\sum_{k=1}^{K}\tilde{\mathbf{x}}(k)}{K} - D \\ &= m_{\tilde{\mathbf{d}}} - \tilde{\mathbf{w}}^{\mathrm{T}}\mathbf{m}_{\tilde{\mathbf{x}}} - D \\ &= 0, \end{aligned} \tag{2.55}$$

donde

$$D = m_{\tilde{\mathbf{d}}} - \tilde{\mathbf{w}}^{\mathrm{T}}\mathbf{m}_{\tilde{\mathbf{x}}}. \tag{2.56}$$

A fim de minimizar a variância do erro, temos o seguinte problema de otimização

$$\min_{\tilde{\mathbf{w}}} \frac{\sum_{k=1}^{K}(\tilde{d}(k)-\tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}(k)-D)^2}{K-1},$$

o qual, usando o valor de D em (2.56), resulta em

$$\min_{\tilde{\mathbf{w}}} \frac{\sum_{k=1}^{K}(\tilde{d}(k)-\tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}(k)-m_{\tilde{\mathbf{d}}}+\tilde{\mathbf{w}}^{\mathrm{T}}\mathbf{m}_{\tilde{\mathbf{x}}})^2}{K-1},$$

i.e.,

$$\min_{\tilde{\mathbf{w}}} \frac{\sum_{k=1}^{K}(\tilde{d}(k)-m_{\tilde{\mathbf{d}}}-\tilde{\mathbf{w}}^{\mathrm{T}}(\tilde{\mathbf{x}}(k)-\mathbf{m}_{\tilde{\mathbf{x}}}))^2}{K-1}. \tag{2.57}$$

Usando as equações de normalização encontradas, dadas por

$$\mathbf{x}(k) = \mathbf{N}_{\mathbf{x}}^{-1}(\tilde{\mathbf{x}}(k) - \mathbf{m}_{\tilde{\mathbf{x}}}) \iff \tilde{\mathbf{x}}(k) - \mathbf{m}_{\tilde{\mathbf{x}}} = \mathbf{N}_{\mathbf{x}}\mathbf{x}(k), \text{ e}$$
$$d(k) = \frac{\tilde{d}(k) - m_{\tilde{\mathbf{d}}}}{n_{\mathbf{d}}} \iff \tilde{d}(k) - m_{\tilde{\mathbf{d}}} = n_{\mathbf{d}}d(k),$$

obtemos

$$\min_{\tilde{\mathbf{w}}} \frac{\sum_{k=1}^{K}(n_{\mathbf{d}}d(k) - \tilde{\mathbf{w}}^{\mathrm{T}}\mathbf{N}_{\mathbf{x}}\mathbf{x}(k))^2}{K-1},$$

equivalente a

$$\min_{\tilde{\mathbf{w}}} \frac{\sum_{k=1}^{K}\left(d(k) - \tilde{\mathbf{w}}^{\mathrm{T}}\dfrac{\mathbf{N}_{\mathbf{x}}}{n_{\mathbf{d}}}\mathbf{x}(k)\right)^2}{K-1}, \tag{2.58}$$

De (2.58), a equação para o vetor de coeficientes normalizado é diretamente obtida por

$$\mathbf{w}^{\mathrm{T}} = \tilde{\mathbf{w}}^{\mathrm{T}}\frac{\mathbf{N}_{\mathbf{x}}}{n_{\mathbf{d}}}.$$

Uma vez que $\mathbf{N}_{\mathbf{x}}$ é simétrica, $\mathbf{N}_{\mathbf{x}}^{\mathrm{T}} = \mathbf{N}_{\mathbf{x}}$, encontramos a relação

$$\mathbf{w} = \frac{\mathbf{N}_{\mathbf{x}}}{n_{\mathbf{d}}}\tilde{\mathbf{w}} \iff \tilde{\mathbf{w}} = n_{\mathbf{d}}\mathbf{N}_{\mathbf{x}}^{-1}\mathbf{w}. \tag{2.59}$$

Com isso, todas as normalizações necessárias para o algoritmo LAR foram descritas. Após estimar o vetor de coeficientes, este deve ser desnormalizado conforme descrito nesta seção, sendo a estimativa de saída do sistema desconhecido dada por

$$\begin{aligned}\tilde{y}(k) &= \tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}(k) + D \\ &= \tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}(k) + m_{\tilde{\mathbf{d}}} - \tilde{\mathbf{w}}^{\mathrm{T}}\mathbf{m}_{\tilde{\mathbf{x}}} \\ &= \tilde{\mathbf{w}}^{\mathrm{T}}(\tilde{\mathbf{x}}(k) - \mathbf{m}_{\tilde{\mathbf{x}}}) + m_{\tilde{\mathbf{d}}} \\ &= n_{\mathbf{d}}\mathbf{w}^{\mathrm{T}}\mathbf{x}(k) + m_{\tilde{\mathbf{d}}}.\end{aligned} \tag{2.60}$$

O algoritmo LAR é descrito a seguir com detalhes. A partir deste momento, assumimos que todas as devidas normalizações foram realizadas, i.e., os dados obedecem a (2.40)-(2.43).

Revisão detalhada do algoritmo LAR

A cada iteração do algoritmo LAR um coeficiente é adicionado ao conjunto de elementos ativos, totalizando no máximo J iterações [5]. Para que o coeficiente ativo seja escolhido e o algoritmo venha a convergir, todo vetor ou matriz é dependente da iteração que está sendo computada naquele momento, com exceção apenas de $\mathbf{X}$ e $\mathbf{d}$ – dados necessários para aplicar o algoritmo LAR. Neste trabalho, a iteração

do algoritmo LAR é representada pela letra $n = 1, \cdots, J$, sempre apresentada no modo subscrito em cada vetor ou matriz. O desenvolvimento feito neste trabalho está considerando apenas sistemas sobredeterminados.

Primeiro, definimos a equação base do algoritmo LAR, dada pelo vetor $(J \times 1)$ de correlação [5],

$$\mathbf{c}_n = \mathbf{X}\mathbf{e}_n, \tag{2.61}$$

onde $\mathbf{X}$ é a matriz de sinal de entrada normalizada $(J \times K)$, definida em (2.50), e $\mathbf{e}_n$ é o vetor de erro de predição $(K \times 1)$, definido por

$$\mathbf{e}_n = \mathbf{d} - \mathbf{y}_{n-1}, \tag{2.62}$$

onde $\mathbf{y}_{n-1}$ é o vetor de sinal de saída estimado na iteração anterior ($\mathbf{y}_0 = \mathbf{0}$), e $\mathbf{d}$ é o vetor de sinal de saída desejado, definido em (2.51).

O vetor de correlação pode, então, ser reescrito por

$$\begin{aligned}
\mathbf{c}_n &= \begin{bmatrix} c_{1,n} & \cdots & c_{j,n} & \cdots & c_{J,n} \end{bmatrix}^{\mathrm{T}} \\
&= \begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_j & \cdots & \mathbf{x}_J \end{bmatrix}^{\mathrm{T}} \mathbf{e}_n \\
&= \begin{bmatrix} \mathbf{x}(1) \cdots \mathbf{x}(K) \end{bmatrix} \begin{bmatrix} e_n(1) \\ \vdots \\ e_n(K) \end{bmatrix} \\
&= \begin{bmatrix} \sum_{k=1}^{K} x_1(k) e_n(k) \\ \vdots \\ \sum_{k=1}^{K} x_J(k) e_n(k) \end{bmatrix}.
\end{aligned} \tag{2.63}$$

Como pode ser observado, o j-ésimo elemento do vetor de correlação na n-ésima iteração do algoritmo, $c_{j,n}$, corresponde a correlação estimada entre o erro de estimação e o vetor de dados do coeficiente j.

Chamamos de conjunto ativo, $\boldsymbol{\mathcal{A}}_n$, o conjunto de coeficientes estimados no modelo na iteração n. A cada iteração, um coeficiente é adicionado a este grupo. O vetor de predição do algoritmo LAR, $\mathbf{y}_n$, é redirecionado a cada iteração, como o nome *Least* ***Angle*** *Regression* sugere, para a direção equiangular dentre os vetores do modelo. A direção a ser seguida, definida pelo vetor $\mathbf{u}_n$, tem o mesmo ângulo para todo $\mathbf{x}_j, j \in \boldsymbol{\mathcal{A}}_n$. Para isso, três regras devem ser obedecidas [9]:

1. ter comprimento unitário:

$$||\mathbf{u}_n||^2 = \mathbf{u}_n^{\mathrm{T}} \mathbf{u}_n = 1; \tag{2.64}$$

2. ser de modo que o ângulo e, consequentemente, o cosseno deste ângulo, α_n,

entre cada vetor de dados do coeficiente no conjunto ativo e $\mathbf{u}_n$ seja o mesmo:

$$s_j \mathbf{x}_j^{\mathrm{T}} \mathbf{u}_n = \alpha_n, \forall j \in \boldsymbol{\mathcal{A}_n}, \tag{2.65}$$

onde $s_j = \pm 1$, dado pelo sinal de $c_{j,n}$. Este sinal deve ser utilizado para que o ângulo fique dentro do primeiro quadrante, e o cosseno do ângulo α_n seja sempre positivo. Sendo $||\mathbf{x}_j|| = ||\mathbf{u}_n|| = 1$, a Equação (2.65) pode ser escrita como

$$\mathbf{X}_n \mathbf{u}_n = \alpha_n \mathbf{1}_n, \tag{2.66}$$

onde $\mathbf{1}_n$ é um vetor com o mesmo número de elementos de $\boldsymbol{\mathcal{A}}_n$ composto por 1's, e $\mathbf{X}_n$ é a matriz composta pelos vetores de dados dos coeficientes do conjunto ativo definida por

$$\mathbf{X}_n = \left[\begin{array}{ccc} s_{j_1}\mathbf{x}_{j_1} & \cdots & s_{j_n}\mathbf{x}_{j_n} \end{array} \right]^{\mathrm{T}}; \tag{2.67}$$

3. ser uma combinação linear dos vetores de dados dos coeficientes que estão no conjunto ativo:

$$\mathbf{u}_n = \mathbf{X}_n^{\mathrm{T}} \mathbf{p}_n, \tag{2.68}$$

onde $\mathbf{p}_n$ é um vetor de pesos $(n \times 1)$, determinado a seguir.

Para determinar $\mathbf{p}_n$, primeiro substituímos (2.68) em (2.64), de modo que

$$\mathbf{p}_n^{\mathrm{T}} \mathbf{X}_n \mathbf{X}_n^{\mathrm{T}} \mathbf{p}_n = \mathbf{p}_n^{\mathrm{T}} \mathbf{R}_n \mathbf{p}_n = 1, \tag{2.69}$$

onde $\mathbf{R}_n = \mathbf{X}_n \mathbf{X}_n^{\mathrm{T}}$ é a matriz de correlação dos vetores de dados dos coeficientes do conjunto ativo. Em seguida, substituindo (2.68) em (2.66), teremos

$$\mathbf{X}_n \mathbf{X}_n^{\mathrm{T}} \mathbf{p}_n = \mathbf{R}_n \mathbf{p}_n = \alpha_n \mathbf{1}_n, \tag{2.70}$$

onde

$$\mathbf{p}_n = \alpha_n \mathbf{R}_n^{-1} \mathbf{1}_n. \tag{2.71}$$

Considerando que o sistema é sobredeterminado, $\mathbf{R}_n^{-1}$ sempre será possível de ser calculado.

Finalmente, α_n é calculado substituindo (2.71) em (2.69),

$$\mathbf{p}_n^{\mathrm{T}} \mathbf{R}_n \mathbf{p}_n = (\alpha_n \mathbf{R}_n^{-1} \mathbf{1}_n)^{\mathrm{T}} \mathbf{R}_n (\alpha_n \mathbf{R}_n^{-1} \mathbf{1}_n) = \alpha_n^2 \mathbf{1}_n^{\mathrm{T}} \mathbf{R}_n^{-1} \mathbf{1}_n = 1,$$

usando $\mathbf{S}_n = \mathbf{R}_n^{-1}$,

$$\alpha_n = (\mathbf{1}_n^{\mathrm{T}} \mathbf{S}_n \mathbf{1}_n)^{-1/2}. \tag{2.72}$$

Substituindo (2.71) em (2.68), $\mathbf{u}_n$ é obtido por

$$\mathbf{u}_n = \alpha_n \mathbf{X}_n^{\mathrm{T}} \mathbf{S}_n \mathbf{1}_n. \tag{2.73}$$

O algoritmo LAR começa com o vetor de saída estimado igual a zero, i.e., $\mathbf{y}_0 = \mathbf{0}$. O vetor de saída estimado $\mathbf{y}_n$ é atualizado de acordo com a seguinte equação [5]:

$$\mathbf{y}_n = \mathbf{y}_{n-1} + \gamma_n \mathbf{u}_n, \tag{2.74}$$

onde $\mathbf{u}_n$ é dado por (2.73), e γ_n é o tamanho do passo dado na direção $\mathbf{u}_n$. O tamanho do passo depende dos coeficientes que estão fora do modelo, i.e., não estão no conjunto ativo. Os coeficientes que estão fora do modelo compõem o chamado conjunto inativo, $\bar{\boldsymbol{\mathcal{A}}}_n$.

O último ponto do algoritmo é determinar qual coeficiente do conjunto inativo deve ser escolhido para ser transferido ao conjunto ativo e, assim, calcular o passo γ_n. O coeficiente escolhido, dentre os coeficientes de $\bar{\boldsymbol{\mathcal{A}}}_n$, é aquele que está mais correlacionado com o erro residual, calculado como se segue.

O vetor de correlação, substituindo (2.62) em (2.61), é dado por

$$\mathbf{c}_n = \mathbf{X}(\mathbf{d} - \mathbf{y}_{n-1}), \tag{2.75}$$

de modo que o valor de correlação relativo ao j-ésimo coeficiente é dado por

$$c_{j,n} = \mathbf{x}_j^{\mathrm{T}}(\mathbf{d} - \mathbf{y}_{n-1}). \tag{2.76}$$

Substituindo (2.74) em (2.76) temos

$$\begin{aligned} c_{j,n} &= \mathbf{x}_j^{\mathrm{T}}(\mathbf{d} - \mathbf{y}_{n-2} - \gamma_{n-1}\mathbf{u}_{n-1}) \\ &= \mathbf{x}_j^{\mathrm{T}}(\mathbf{d} - \mathbf{y}_{n-2}) - \gamma_{n-1}\mathbf{x}_j^{\mathrm{T}}\mathbf{u}_{n-1} \\ &= c_{j,n-1} - \gamma_{n-1}\mathbf{x}_j^{\mathrm{T}}\mathbf{u}_{n-1}. \end{aligned} \tag{2.77}$$

Por definição, todos os coeficientes no conjunto ativo têm o mesmo valor de correlação absoluto, $|c_{j,n}| = C_{max,n}, j \in \boldsymbol{\mathcal{A}}_n$, e o mesmo ângulo cujo cosseno é α_n, dado em (2.65). Logo, para os coeficientes ativos teremos

$$C_{max,n} = C_{max,n-1} - \gamma_{n-1}\alpha_{n-1}, \;\; j \in \boldsymbol{\mathcal{A}}_n,$$

equivalente a

$$C_{max,n+1} = C_{max,n} - \gamma_n \alpha_n, \;\; j \in \boldsymbol{\mathcal{A}}_{n+1}, \tag{2.78}$$

sendo este o valor máximo de correlação da iteração seguinte.

Definindo um vetor auxiliar por

$$\mathbf{a}_n = \mathbf{X}\mathbf{u}_n, \tag{2.79}$$

um coeficiente de índice j que está no conjunto inativo terá este mesmo valor de correlação se, para $c_{j,n} > 0$,

$$\begin{aligned} c_{j,n} - \gamma_n \mathbf{x}_j^{\mathrm{T}} \mathbf{u}_n &= C_{max,n} - \gamma_j \alpha_n, \;\; j \in \bar{\boldsymbol{\mathcal{A}}}_n \\ \gamma_j &= \frac{C_{max,n} - c_{j,n}}{\alpha_n - a_{j,n}} \end{aligned} \tag{2.80}$$

ou, para $c_{j,n} < 0$,

$$\begin{aligned} -c_{j,n} + \gamma_n \mathbf{x}_j^{\mathrm{T}} \mathbf{u}_s &= C_{max,n} - \gamma_j \alpha_n, \;\; j \in \bar{\boldsymbol{\mathcal{A}}}_n \\ \gamma_j &= \frac{C_{max,n} + c_{j,n}}{\alpha_n + a_{j,n}}. \end{aligned} \tag{2.81}$$

O próximo coeficiente mais correlacionado, o escolhido j_n, é aquele que resulta no menor valor positivo de γ_j,

$$\gamma_n = \min_{\gamma_j > 0} \gamma_j, \tag{2.82}$$

onde γ_j são valores encontrados em (2.80) e (2.81) para $j \in \bar{\boldsymbol{\mathcal{A}}}_n$. O dito coeficiente, identificado pelo índice j_n, é removido do conjunto inativo e inserido no conjunto ativo, e o valor γ_n é considerado na iteração atual n. Na última iteração, quando $n = J$, há uma exceção, uma vez que $\bar{\boldsymbol{\mathcal{A}}}_J = [\ ]$ e nenhuma das equações acima pode ser calculada; neste momento o algoritmo utiliza $\gamma_J = C_{max,J}/\alpha_J$, de forma a resultar na solução LS [5].

A seguir, com base na Figura 2.10, é apresentado um exemplo gráfico simples de como é incrementado o algoritmo LAR. Seja um sistema com quatro coeficientes, j = 1,2,3,4; logo, os vetores de dados dos coeficientes são representados por $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, e $\mathbf{x}_4$.

O primeiro coeficiente a entrar no conjunto ativo, em $n = 1$, é encontrado a partir da equação $\mathbf{c}_1 = \mathbf{X}^{\mathrm{T}}\mathbf{d}$. Seja o maior valor absoluto de $\mathbf{c}_1$ calculado para o coeficiente $j = 1$, ou seja, $j = 1$ é o mais correlato: $|c_{1,1}| > |c_{2,1}|$, $|c_{1,1}| > |c_{3,1}|$ e $|c_{1,1}| > |c_{4,1}|$. $\mathbf{y}_1$ aumenta o valor γ_1 na direção $\mathbf{u}_1$ ($\mathbf{y}_1 = \mathbf{y}_0 + \gamma_1\mathbf{u}_1$, $\mathbf{y}_0 = \mathbf{0}$), sendo γ_1 o menor valor encontrado para que um novo coeficiente seja tão correlato quanto o coeficiente $j = 1$.

Seja o novo coeficiente mais correlato $j = 2$, ou seja, calculando $\mathbf{c}_2 = \mathbf{X}^{\mathrm{T}}(\mathbf{d} - \mathbf{y}_1)$, obtém-se $|c_{1,2}| = |c_{2,2}| > |c_{3,2}|$ e $|c_{1,2}| = |c_{2,2}| > |c_{4,2}|$. Neste momento, $\mathbf{y}_2$ aumenta o valor γ_2 na direção $\mathbf{u}_2$, direção esta equiangular entre $\mathbf{x}_1$ e $\mathbf{x}_2$. Novamente, o valor γ_2 foi calculado tal que um novo coeficiente seja tão correlato quanto $j = 1$ e $j = 2$,

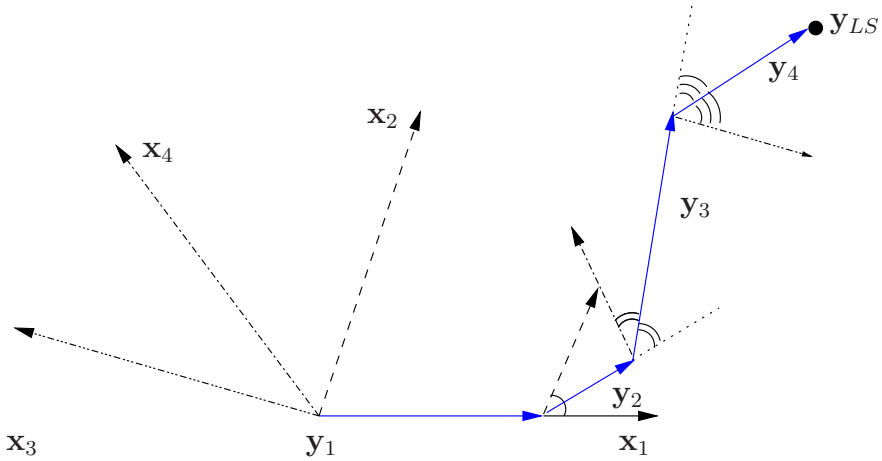


Figura 2.10: Exemplo da implementação do algoritmo LAR em um sistema com quatro coeficientes. O vetor de saída estimado $\mathbf{y}_n$ é incrementado conforme os vetores de cor azul. Os vetores com o mesmo padrão de traço são do mesmo coeficiente. Note que o vetor de dados do coeficiente $j = 3, \mathbf{x}_3$, selecionado em $n = 4$, teve sua direção invertida ($j_4 = 3, s_{j_4} = -1$).

por exemplo $j = 4$ ($|c_{1,3}| = |c_{2,3}| = |c_{4,3}| > |c_{3,3}|$). $\mathbf{y}_3$ aumenta, então, o valor γ_3 na direção $\mathbf{u}_3$, direção esta equiangular entre $\mathbf{x}_4$, $\mathbf{x}_2$ e $\mathbf{x}_1$.

Finalmente, o último coeficiente mais correlato só pode ser $j = 3$. Neste momento, não há nenhum outro coeficiente restante, logo o valor de γ_4 é calculado tal que a resposta do LAR equivalha à resposta LS. A direção seguida é equiangular a todos os vetores de dados de coeficientes do sistema.

Estimação do Vetor de Coeficientes

No algoritmo LAR, o vetor de coeficientes estimados pode ser calculado apenas após a última iteração realizada. Dependendo da aplicação, estimar o vetor de coeficientes apenas no final não é prejudicial. No entanto, para a aplicação de identificação de sistemas não-lineares aqui abordada, é desejado estimar o vetor de coeficientes a cada iteração. Este adicional, não evidenciado em [5], é apresentado a seguir.

Seja o vetor de coeficientes ativos estimado ($n \times 1$) pelo algoritmo LAR na iteração n dado por

$$\mathbf{w}_n = [w_{j_1,n} \quad \cdots \quad w_{j_n,n}]^{\mathrm{T}}, \tag{2.83}$$

e o vetor de coeficientes completo estimado ($J \times 1$) pelo algoritmo LAR na iteração n dado por

$$\hat{\mathbf{w}}_n = [w_{1,n} \quad \cdots \quad w_{j,n} \quad \cdots \quad w_{J,n}]^{\mathrm{T}}. \tag{2.84}$$

O vetor de coeficientes ativos é idealmente estimado por

$$\mathbf{w}_n = (\mathbf{X}_n \mathbf{X}_n^{\mathrm{T}})^{-1} \mathbf{X}_n \mathbf{y}_n = \mathbf{S}_n \mathbf{X}_n \mathbf{y}_n. \tag{2.85}$$

Note que $\mathbf{w}_n$ tem tamanho $n \times 1$, ordenado como em $\mathcal{A}$.

Usando o vetor de sinal de saída estimado, dado por (2.74), encontramos

$$\begin{aligned} \mathbf{w}_n &= \mathbf{S}_n\mathbf{X}_n(\mathbf{y}_{n-1} + \gamma_n\mathbf{u}_n) \\ &= \mathbf{S}_n\mathbf{X}_n\mathbf{y}_{n-1} + \gamma_n\mathbf{S}_n\mathbf{X}_n\mathbf{u}_n. \end{aligned} \tag{2.86}$$

Para resolver a primeira parte da equação, conhecemos que a saída estimada é calculada por

$$\mathbf{y}_{n-1} = \mathbf{X}^{\mathrm{T}}\hat{\mathbf{w}}_{n-1}, \tag{2.87}$$

de onde também encontramos $\hat{\mathbf{w}}_n$,

$$\hat{\mathbf{w}}_n = (\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{X}\mathbf{y}_n. \tag{2.88}$$

Podemos escrever a relação entre $\mathbf{X}$ e $\mathbf{X}_n$, baseado em (2.67), por

$$\mathbf{X}_n = \mathbf{D}_n\mathbf{P}_n\mathbf{X} \tag{2.89}$$

onde

$$\mathbf{P}_n = \begin{bmatrix} \mathbf{p}_{j_1}^{\mathrm{T}} \\ \vdots \\ \mathbf{p}_{j_n}^{\mathrm{T}} \end{bmatrix}, \tag{2.90}$$

é a matriz seleção como em $\boldsymbol{\mathcal{A}}_n$ $(n \times J)$, onde $\mathbf{p}_{j_n}$ é um vetor com a posição j_n igual a 1 e as demais iguais a zero, e

$$\mathbf{D}_n = \mathrm{diag}(s_{j_1} \cdots s_{j_n}) = \mathrm{diag}(\mathbf{P}_n\ \mathrm{sign}(\mathbf{c}_n)) \tag{2.91}$$

é a matriz diagonal composta por ± 1 $(n \times n)$, dado pelo sinal de $s_{j_n} = \mathrm{sign}(c_{j,n})$,$j \in \boldsymbol{\mathcal{A}}_n$. Ambas as matrizes são compostas de acordo com a ordem em que os coeficientes entram no conjunto ativo. Por exemplo, seja um sistema com cinco coeficientes $(J = 5)$, em que o algoritmo LAR encontra-se na terceira iteração $(n = 3)$ com

$$\boldsymbol{\mathcal{A}}_3 = \begin{bmatrix} 3 & 2 & 5 \end{bmatrix}^{\mathrm{T}} \text{ e } \mathbf{c}_3 = \begin{bmatrix} 0.2 & -0.5 & 0.5 & -0.4 & 0.5 \end{bmatrix}^{\mathrm{T}}.$$

Neste caso, teremos

$$\mathbf{P}_3 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \text{ e } \mathbf{D}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Assim, temos que $\mathbf{D}_n^{-1} = \mathbf{D}_n$ e $\mathbf{D}_n^2 = \mathbf{P}_n\mathbf{P}_n^{\mathrm{T}} = \mathbf{I}_n$.

Por definição, temos

$$\hat{\mathbf{w}}_n = \mathbf{P}_n^{\mathrm{T}}\mathbf{D}_n\mathbf{w}_n. \tag{2.92}$$

Substituindo (2.92) em (2.87) e usamdo (2.89), mostramos que

$$\mathbf{y}_{n-1} = \mathbf{X}^{\mathrm{T}}\mathbf{P}_{n-1}^{\mathrm{T}}\mathbf{D}_{n-1}\mathbf{w}_{n-1} = \mathbf{X}_{n-1}^{\mathrm{T}}\mathbf{w}_{n-1}. \tag{2.93}$$

Logo, (2.92) é solução de (2.87). Note que $\hat{\mathbf{w}}_n$ tem tamanho $J \times 1$, enquanto que $\mathbf{w}_n$ tem tamanho $n \times 1$. A multiplicação de $\mathbf{w}_n$ pela transposta da matriz de seleção $\mathbf{P}_n$, $n \times J$, retorna ao tamanho original o vetor de coeficientes.

Usando (2.67) e (2.93) para solucionar a primeira parte da equação (2.86), teremos

$$\begin{aligned}
\mathbf{S}_n\mathbf{X}_n\mathbf{y}_{n-1} &= \mathbf{S}_n\mathbf{X}_n\mathbf{X}_{n-1}^{\mathrm{T}}\mathbf{w}_{n-1} \\
&= \mathbf{S}_n\mathbf{X}_n[\mathbf{X}_{n-1}^{\mathrm{T}}\ s_{j_n}\mathbf{x}_{j_n}]\begin{bmatrix}\mathbf{w}_{n-1}\\ 0\end{bmatrix} \\
&= \mathbf{I}_n\begin{bmatrix}\mathbf{w}_{n-1}\\ 0\end{bmatrix} \\
&= \begin{bmatrix}\mathbf{w}_{n-1}\\ 0\end{bmatrix}.
\end{aligned} \tag{2.94}$$

Para a segunda parte da equação (2.86), sabemos, de (2.73), que

$$\mathbf{u}_n = \alpha_n\mathbf{X}_n^{\mathrm{T}}\mathbf{S}_n\mathbf{1}_n.$$

Substituindo (2.73) e (2.94) em (2.86), temos

$$\begin{aligned}
\mathbf{w}_n &= \mathbf{S}_n\mathbf{X}_n\mathbf{y}_{n-1} + \gamma_n\mathbf{S}_n\mathbf{X}_n\mathbf{u}_n \\
&= \begin{bmatrix}\mathbf{w}_{n-1}\\ 0\end{bmatrix} + \gamma_n\alpha_n\mathbf{S}_n\mathbf{X}_n\mathbf{X}_n^{\mathrm{T}}\mathbf{S}_n\mathbf{1}_n \\
&= \begin{bmatrix}\mathbf{w}_{n-1}\\ 0\end{bmatrix} + \gamma_n\alpha_n\mathbf{S}_n\mathbf{1}_n,
\end{aligned}$$

de onde encontramos

$$\mathbf{w}_n = \begin{bmatrix}\mathbf{w}_{n-1}\\ 0\end{bmatrix} + \gamma_n\boldsymbol{\omega}_n. \tag{2.95}$$

onde

$$\boldsymbol{\omega}_n = \alpha_n\mathbf{S}_n\mathbf{1}_n. \tag{2.96}$$

Assim, temos o algoritmo completo, com todas equações utilizadas apresentadas

no Algoritmo 1. Deve-se lembrar que é necessário desnormalizar o vetor de coeficientes, dado em (2.59). Note ainda que um vetor $\mathbf{w}_n$ esparso permanece esparso após a desnormalização.

Algoritmo 1 – LAR

entrada: $\tilde{\mathbf{X}}$, $\tilde{\mathbf{d}}$
normalizar para $\mathbf{X}, \mathbf{d}$
$\boldsymbol{\mathcal{A}}_0 \leftarrow [\ ]$
$\mathbf{y}_0 \leftarrow \mathbf{0}$
$\mathbf{P}_n \leftarrow [\ ]^{\mathrm{T}}$
for $n = 1 : J$ **do**
  $\mathbf{e}_n \leftarrow \mathbf{d} - \mathbf{y}_{n-1}$
  $\mathbf{c}_n \leftarrow \mathbf{X}\mathbf{e}_n$
  $[C_{max}, pos_{max}] \leftarrow max(|\mathbf{c}_n|)$
  if $n = 1$ **then**
    $j_n \leftarrow pos_{max}$
  end if
  $\boldsymbol{\mathcal{A}}_n \leftarrow \left[\ \boldsymbol{\mathcal{A}}_{n-1} \quad j_n\ \right]^{\mathrm{T}}$
  $\bar{\boldsymbol{\mathcal{A}}}_n \leftarrow \{1, 2, \cdots, J\} - \boldsymbol{\mathcal{A}}_n$
  $\mathbf{p}_{j_n} \leftarrow \mathrm{zeros}(J,1)$
  $p_{j_n}(j_n) \leftarrow 1$
  $\mathbf{P}_n \leftarrow [\mathbf{P}_{n-1}^{\mathrm{T}} \ \mathbf{p}_{j_n}]^{\mathrm{T}}$
  $\mathbf{D}_n \leftarrow \mathrm{diag}(\mathrm{sign}(\mathbf{P}_n\mathbf{c}_n))$
  $\mathbf{X}_n \leftarrow \mathbf{D}_n\mathbf{P}_n\mathbf{X}$
  $\mathbf{S}_n \leftarrow (\mathbf{X}_n\mathbf{X}_n^{\mathrm{T}})^{-1}$
  $\mathbf{1}_n \leftarrow \mathrm{ones}(n,1)$
  $\alpha_n \leftarrow (\mathbf{1}_n^{\mathrm{T}}\mathbf{S}_n\mathbf{1}_n)^{-1/2}$
  $\boldsymbol{\omega}_n \leftarrow \alpha_n\mathbf{S}_n\mathbf{1}_n$
  $\mathbf{u}_n \leftarrow \mathbf{X}_n\boldsymbol{\omega}_n$
  $\mathbf{a}_n \leftarrow \mathbf{X}\mathbf{u}_n$
  if $n = J$ **then**
    $\gamma_n \leftarrow C_{max}/\alpha_n$
  else
    for $j = 1$ to $J - n$ **do**
      $\beta_1 \leftarrow \dfrac{C_{max} - c_{\bar{\mathcal{A}}(j),n}}{\alpha_n - a_{\bar{\mathcal{A}}(j),n}}$
      $\beta_2 \leftarrow \dfrac{C_{max} + c_{\bar{\mathcal{A}}(j),n}}{\alpha_n + a_{\bar{\mathcal{A}}(j),n}}$
      $\boldsymbol{\gamma}_j \leftarrow \min(\beta_1 > 0, \beta_2 > 0)$
    end for
    $[\gamma_n, pos_n] \leftarrow \min(\boldsymbol{\gamma}_j)$
    $j_n \leftarrow \bar{\mathcal{A}}_n(pos_n)$
  end if
  $\mathbf{y}_n \leftarrow \mathbf{y}_{n-1} + \gamma_n\mathbf{u}_n$
  $\mathbf{w}_n \leftarrow [\mathbf{w}_{n-1}^{\mathrm{T}} \ 0]^{\mathrm{T}} + \gamma_n\boldsymbol{\omega}_n$
end for
$\hat{\mathbf{w}} \leftarrow \mathbf{P}_J^{\mathrm{T}}\mathbf{D}_J\mathbf{w}_J$
saída: desnormalizar $\hat{\mathbf{w}}$

2.3.3 Algoritmo LASSO – *Least Absolute Shrinkage and Selection Operator*

Robert Tibshirani apresentou, em 1996, o algoritmo LASSO como um novo método para estimação de modelos lineares [37]. Tibshirani também é coautor do algoritmo LAR, apresentado em 2004. Uma modificação no algoritmo LAR permite que todas as respostas do algoritmo LASSO sejam obtidas [5]. O conteúdo aqui apresentado é esta modificação necessária no algoritmo LAR, de modo que o algoritmo LAR modificado satisfaça a função objetivo do algoritmo LASSO.

A função objetivo do algoritmo LASSO é minimizar o erro residual sujeito a um valor absoluto máximo dos coeficientes estimados [37]. Em outras palavras, o algoritmo LASSO resolve o problema LS sujeito a uma restrição de norma-1 dos seus coeficientes, cuja função objetivo é dada por [38]

$$\min_{\mathbf{w}} \|\mathbf{d} - \mathbf{y}\|^2 \text{ s.t. } \|\mathbf{w}\|_1 < t, \tag{2.97}$$

onde $\|\mathbf{w}\|_1 = \sum_j |w_j|$, e t é uma constante definida *a priori*.

Em 1999, Osborne et al. apresentaram em [39, 40] melhorias no processamento do algoritmo LASSO, que futuramente foi chamado de *Homotopy*, ou Homotopia. A resposta do algoritmo Homotopia é a mesma resposta do algoritmo LASSO. De maneira análoga, Tibshirani apresentou em [5] uma modificação no algoritmo LAR para que obtivesse também a mesma resposta do algoritmo LASSO. O próprio autor disse que o algoritmo LAR modificado para o algoritmo LASSO tem um procedimento semelhante ao algoritmo de Homotopia, porém de uma forma mais direta [5].

Donoho e Tsaig [38] desenvolveram um critério de parada, chamado de propriedade da solução do passo-k, usado para o algoritmo *Homotopy*, algoritmo com resultado semelhante ao algoritmo LASSO. Este critério foi desenvolvido para o caso de sistemas subdeterminados e, segundo os autores, pode também ser utilizado no algoritmo LAR. Após atingir a quantidade ideal segundo a propriedade apresentada, o algoritmo é interrompido antes de todos os coeficientes serem estimados.

O enfoque aqui apresentado é a modificação necessária no algoritmo LAR para que tenha a resposta do algoritmo LASSO. Maiores detalhes do algoritmo LASSO não são apresentados, uma vez que o algoritmo LASSO não foi objeto de estudo aprofundado para o desenvolvimento deste trabalho.

Vamos rescrever a função objetivo do algoritmo LASSO com a nomenclatura utilizada na Seção 2.3.2:

$$\min_{\hat{\mathbf{w}}_n} \|\mathbf{d} - \mathbf{y}_n\|^2 \text{ s.t. } \|\hat{\mathbf{w}}_n\|_1 < t, \tag{2.98}$$

onde $\|\hat{\mathbf{w}}_n\|_1 = \sum_{j=1}^{J} |\hat{w}_{j,n}|$, e t é uma constante definida *a priori*.

Para que a resposta do algoritmo LAR seja a resposta do algoritmo LASSO é necessário que o sinal de cada coeficiente estimado seja o mesmo sinal do seu valor de correlação, ou seja [5]

$$\text{sign}(w_{j,n}) = \text{sign}(c_{j,n}) = s_j. \tag{2.99}$$

O algoritmo LAR não impõe esta restrição, porém pode ser modificado para que isto ocorra.

Para encontrar a solução da função objetivo do LASSO (2.97), devemos calcular o gradiente da função custo, definido por

$$\nabla_{\hat{\mathbf{w}}_n}\varepsilon = \nabla_{\hat{\mathbf{w}}_n}\left\{[\mathbf{d}^{\mathrm{T}} - \hat{\mathbf{w}}_n^{\mathrm{T}}\mathbf{X}][\mathbf{d} - \mathbf{X}^{\mathrm{T}}\hat{\mathbf{w}}_n]\right\} + \nabla_{\hat{\mathbf{w}}_n}\left\{\lambda \sum_j |\hat{w}_{j,n}|\right\}. \tag{2.100}$$

Relembrando a relação entre $\hat{\mathbf{w}}_n$ e $\mathbf{w}_n$ (2.92)

$$\hat{\mathbf{w}}_n = \mathbf{P}_n^{\mathrm{T}}\mathbf{D}_n\mathbf{w}_n,$$

onde $\mathbf{w}_n$ é definido em (2.83) e $\hat{\mathbf{w}}_n$ é definido em (2.84), a norma L_1 usada na restrição do algoritmo pode ser reduzida a

$$\|\hat{\mathbf{w}}_n\|_1 = \sum_{j=1}^{J} |\hat{w}_{j,n}| = \sum_{i=1}^{n} s_{j_i} w_{j_i,n} = \mathbf{w}_n^{\mathrm{T}}\mathbf{D}_n\mathbf{1}_n. \tag{2.101}$$

Desta forma, substituindo (2.92) e (2.101) em (2.100), bem como usando o multiplicador de Lagrange λ, encontramos

$$\begin{aligned}
\nabla_{\mathbf{w}_n}\varepsilon &= \nabla_{\mathbf{w}_n}\{[\mathbf{d}^{\mathrm{T}} - (\mathbf{P}_n^{\mathrm{T}}\mathbf{D}_n\mathbf{w}_n)^{\mathrm{T}}\mathbf{X}][\mathbf{d} - \mathbf{X}^{\mathrm{T}}\mathbf{P}_n^{\mathrm{T}}\mathbf{D}_n\mathbf{w}_n]\} + \nabla_{\mathbf{w}_n}\{\lambda\mathbf{w}_n^{\mathrm{T}}\mathbf{D}_n\mathbf{1}_n\} \\
&= -\nabla_{\mathbf{w}_n}\{\mathbf{d}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{P}_n^{\mathrm{T}}\mathbf{D}_n\mathbf{w}_n\} - \nabla_{\mathbf{w}_n}\{\mathbf{w}_n^{\mathrm{T}}\mathbf{D}_n\mathbf{P}_n\mathbf{X}\mathbf{d}\} \quad \cdots \\
&\quad +\nabla_{\mathbf{w}_n}\{\mathbf{w}_n^{\mathrm{T}}\mathbf{D}_n\mathbf{P}_n\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{P}_n^{\mathrm{T}}\mathbf{D}_n\mathbf{w}\} + \nabla_{\mathbf{w}_n}\{\lambda\mathbf{w}_n^{\mathrm{T}}\mathbf{D}_n\mathbf{1}_n\} \\
&= -2\mathbf{D}_n\mathbf{P}_n\mathbf{X}\mathbf{d} + 2\mathbf{D}_n\mathbf{P}_n\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{P}_n^{\mathrm{T}}\mathbf{D}_n\mathbf{w}_n + \lambda\mathbf{D}_n\mathbf{1}_n \\
&= -2\mathbf{X}_n\mathbf{d} + 2\mathbf{X}_n\mathbf{X}_n^{\mathrm{T}}\mathbf{w}_n + \lambda\mathbf{D}_n\mathbf{1}_n,
\end{aligned} \tag{2.102}$$

onde $\mathbf{X}_n = \mathbf{D}_n\mathbf{P}_n\mathbf{X}$ (2.89). Fazendo $\nabla_{\mathbf{w}_n}\varepsilon = 0$, encontramos

$$2\mathbf{X}_n(\mathbf{d} - \mathbf{X}_n^{\mathrm{T}}\mathbf{w}_n) = \lambda\mathbf{D}_n\mathbf{1}_n. \tag{2.103}$$

Substituindo $\mathbf{y}_n = \mathbf{X}_n^{\mathrm{T}}\mathbf{w}_n$ (2.93) em (2.103) e voltando com $\mathbf{X}_n = \mathbf{D}_n\mathbf{P}_n\mathbf{X}$,

$$2\mathbf{D}_n\mathbf{P}_n\mathbf{c}_n = \lambda\mathbf{D}_n\mathbf{1}_n. \tag{2.104}$$

Nota-se que $\mathbf{P}_n\mathbf{c}_n$ é o valor de $c_{j,n}, j \in \boldsymbol{\mathcal{A}}_n$. Logo, $C_{max,n} = \dfrac{\lambda}{2}$. Por outro lado, também temos, igualando (2.103) e (2.104),

$$\mathbf{P}_n\mathbf{X}(\mathbf{d} - \mathbf{X}_n^{\mathrm{T}}\mathbf{w}_n) = \mathbf{P}_n\mathbf{c}_n. \tag{2.105}$$

Em ambos os lados, a multiplicação por $\mathbf{P}_n$ indica que a igualdade deve ser mantida apenas para os coeficientes ativos. Neste caso, temos então que $\mathrm{sign}(c_{j,n}) = \mathrm{sign}(w_{j,n})$, como queríamos demonstrar em (2.99).

Para que (2.99) seja satisfeita, suponha que o algoritmo esteja na iteração n. Sabemos que a relação entre $\hat{\mathbf{w}}_n$ e $\mathbf{w}_n$ (2.92) é dada por

$$\hat{\mathbf{w}}_n = \mathbf{P}_n^{\mathrm{T}}\mathbf{D}_n\mathbf{w}_n,$$

bem como (2.95)

$$\mathbf{w}_n = \begin{bmatrix} \mathbf{w}_{n-1} \\ 0 \end{bmatrix} + \gamma_n\boldsymbol{\omega}_n$$

onde $\boldsymbol{\omega}_n = [\ \omega_{j_1,n} \quad \cdots \quad \omega_{j_n,n}\ ]^{\mathrm{T}} = \alpha_n\mathbf{S}_n\mathbf{1}_n$ (2.96).

Logo,

$$\begin{aligned}\hat{\mathbf{w}}_n &= \mathbf{P}_n^{\mathrm{T}}\mathbf{D}_n\begin{bmatrix} \mathbf{w}_{n-1} \\ 0 \end{bmatrix} + \gamma_n\mathbf{P}_n^{\mathrm{T}}\mathbf{D}_n\boldsymbol{\omega}_n \\ &= \hat{\mathbf{w}}_{n-1} + \gamma_n\mathbf{P}_n^{\mathrm{T}}\mathbf{D}_n\boldsymbol{\omega}_n. \end{aligned} \tag{2.106}$$

Os sinais de $\hat{\mathbf{w}}_n$ e $\hat{\mathbf{w}}_{n-1}$ serão diferentes a partir de

$$\gamma_n\mathbf{P}_n^{\mathrm{T}}\mathbf{D}_n\boldsymbol{\omega}_n = -\hat{\mathbf{w}}_{n-1}. \tag{2.107}$$

Uma vez que $w_{j,n-1} = 0, j \in \bar{\mathcal{A}}_n$ (o mesmo resultado obtemos para o lado esquerdo da equação), apenas os coeficientes ativos devem ser verificados. Para os coeficientes ativos, isto vai ocorrer quando

$$\begin{aligned}\gamma_{j_n}s_{j_n}\omega_{j_n,n} &= -w_{j_n,n-1} \\ \gamma_{j_n} &= -\frac{w_{j_n,n-1}}{s_{j_n}\omega_{j_n,n}}, \end{aligned} \tag{2.108}$$

o primeiro ocorrendo em

$$\gamma_{j_{LASSO}} = \min_{\gamma_{j_n}>0}\{\gamma_{j_n}\}. \tag{2.109}$$

Quando não há valor positivo para γ_{j_n}, $\gamma_{j_{LASSO}}$ é considerado infinito.

Por fim, quando $\gamma_{j_{LASSO}}$, dado por (2.109), for menor que γ_n, dado por (2.82), o

coeficiente j_{LASSO} é retirado do conjunto ativo

$$\boldsymbol{\mathcal{A}}_n = \boldsymbol{\mathcal{A}}_n - j_{LASSO}, \tag{2.110}$$

e o tamanho do passo de iteração é igual $\gamma_{j_{LASSO}}$, ao invés de γ_n. Ou seja, o valor máximo de correlação é dado por (2.78)

$$C_{max,n+1} = C_{max,n} - \gamma_{j_{LASSO}}\alpha_n, \;\; j \in \boldsymbol{\mathcal{A}}_n, \tag{2.111}$$

bem como o vetor de saída estimado $\mathbf{y}_n$ é dado por (2.74)

$$\mathbf{y}_n = \mathbf{y}_{n-1} + \gamma_{j_{LASSO}}\mathbf{u}_n, \tag{2.112}$$

e o vetor de coeficientes $\mathbf{w}_n$ dado por (2.95)

$$\mathbf{w}_n = \begin{bmatrix} \mathbf{w}_{n-1} \\ 0 \end{bmatrix} + \gamma_{LASSO}\boldsymbol{\omega}_n. \tag{2.113}$$

O pseudocódigo do algoritmo LAR modificado para obter as respostas do algoritmo LASSO é apresentado no Algoritmo 2.

2.4 Critérios para seleção de modelo

Seleção de modelo é a escolha de parâmetros, aqui chamados de coeficientes, na tentativa de criar um modelo com a menor complexidade possível para uma quantidade de dados finita [6, 41]. Se a quantidade de coeficientes usada for muito menor do que a quantidade efetiva do sistema real, o modelo não possuirá a complexidade estrutural necessária para reproduzir a dinâmica do sistema. Por outro lado, se a quantidade for muito maior do que a necessária, a estimação dos coeficientes será mal dimensionada [1]. Neste trabalho, o critério para seleção de modelo irá definir a quantidade e quem serão os coeficientes não-nulos do sistema, também chamados de coeficientes ativos na Seção 2.3.

Três critérios para seleção de modelo foram estudados: *Akaike Information Criterion* (AIC), *Bayesian Information Criterion* (BIC) e *Mallows* C_p. Os dois primeiros critérios, propostos na década de 70, são amplamente utilizados em seleção de modelos nas mais diversas áreas, como em modelos urbanos [27], modelos sociológicos [30] e modelos aplicados na área de ecologia e evolução [28, 31]. O terceiro critério, também conhecido e proposto na década de 70, foi estudado em particular por ter sido sugerido por Efron et al. [5], como critério de seleção especificamente para ser usado com o algoritmo LAR. Todos estes critérios são medidas de qualidade de

Algoritmo 2 – LAR modificado para LASSO

entrada: $\tilde{\mathbf{X}}$, $\tilde{\mathbf{d}}$
normalizar para $\mathbf{X}, \mathbf{d}$
$\boldsymbol{\mathcal{A}}_0 \leftarrow [\ ]$
$\mathbf{y}_0 \leftarrow \mathbf{0}$
$\mathbf{P}_n \leftarrow [\ ]^{\mathrm{T}}$
for $n = 1 : J$ **do**
  $\mathbf{e}_n \leftarrow \mathbf{d} - \mathbf{y}_{n-1}$
  $\mathbf{c}_n \leftarrow \mathbf{X}\mathbf{e}_n$
  $[C_{max}, pos_{max}] \leftarrow max(|\mathbf{c}_n|)$
  if $n = 1$ **then**
    $j_n \leftarrow pos_{max}$
  end if
  $\boldsymbol{\mathcal{A}}_n \leftarrow \begin{bmatrix} \boldsymbol{\mathcal{A}}_{n-1} & j_n \end{bmatrix}^{\mathrm{T}}$
  $\bar{\boldsymbol{\mathcal{A}}}_n \leftarrow \{1, 2, \cdots, J\} - \boldsymbol{\mathcal{A}}_n$
  $\mathbf{p}_{j_n} \leftarrow \text{zeros}(J,1)$
  $p_{j_n}(j_n) \leftarrow 1$
  $\mathbf{P}_n \leftarrow [\mathbf{P}_{n-1}^{\mathrm{T}} \ \mathbf{p}_{j_n}]^{\mathrm{T}}$
  $\mathbf{D}_n \leftarrow \text{diag}(\text{sign}(\mathbf{P}_n\mathbf{c}_n))$
  $\mathbf{X}_n \leftarrow \mathbf{D}_n\mathbf{P}_n\mathbf{X}$
  $\mathbf{S}_n \leftarrow (\mathbf{X}_n\mathbf{X}_n^{\mathrm{T}})^{-1}$
  $\mathbf{1}_n \leftarrow \text{ones}(n,1)$
  $\alpha_n \leftarrow (\mathbf{1}_n^{\mathrm{T}}\mathbf{S}_n\mathbf{1}_n)^{-1/2}$
  $\boldsymbol{\omega}_n \leftarrow \alpha_n\mathbf{S}_n\mathbf{1}_n$
  $\mathbf{u}_n \leftarrow \mathbf{X}_n\boldsymbol{\omega}_n$
  $\mathbf{a}_n \leftarrow \mathbf{X}\mathbf{u}_n$
  if $n = J$ **then**
    $\gamma_n \leftarrow C_{max}/\alpha_n$
  else
    for $j = 1$ to $J - n$ **do**
      $\beta_1 \leftarrow \dfrac{C_{max} - c_{\bar{\mathcal{A}}(j),n}}{\alpha_n - a_{\bar{\mathcal{A}}(j),n}}$
      $\beta_2 \leftarrow \dfrac{C_{max} + c_{\bar{\mathcal{A}}(j),n}}{\alpha_n + a_{\bar{\mathcal{A}}(j),n}}$
      $\boldsymbol{\gamma}_j \leftarrow \min(\beta_1 > 0, \beta_2 > 0)$
    end for
    $[\gamma_n, pos_n] \leftarrow \min(\boldsymbol{\gamma}_j)$
    $j_n \leftarrow \bar{\mathcal{A}}_n(pos_n)$
  end if
  for $j = 1$ to n **do**
    $\tilde{\gamma}_{LASSO}(j) = -w_{j,n-1}/(s_j\omega_{j,n})$
  end for
  $[\gamma_{LASSO}, pos_{LASSO}] \leftarrow \min(\tilde{\boldsymbol{\gamma}}_{LASSO} > 0)$
  $j_{LASSO} \leftarrow \mathcal{A}_n(pos_{LASSO})$
  if $\gamma_{LASSO} < \gamma_n$ **then**
    $\mathbf{y}_n \leftarrow \mathbf{y}_{n-1} + \gamma_{LASSO}\mathbf{u}_n$
    $\mathbf{w}_n \leftarrow [\mathbf{w}_{n-1}^{\mathrm{T}} \ 0]^{\mathrm{T}} + \gamma_{LASSO}\boldsymbol{\omega}_n$
    $\boldsymbol{\mathcal{A}}_n \leftarrow \boldsymbol{\mathcal{A}}_n - j_{LASSO}$
    $\mathbf{P}_n \leftarrow \mathbf{P}_n - \mathbf{p}_{j_{LASSO}}$
  else
    $\mathbf{y}_n \leftarrow \mathbf{y}_{n-1} + \gamma_n\mathbf{u}_n$
    $\mathbf{w}_n \leftarrow [\mathbf{w}_{n-1}^{\mathrm{T}} \ 0]^{\mathrm{T}} + \gamma_n\boldsymbol{\omega}_n$
  end if
end for
$\hat{\mathbf{w}} \leftarrow \mathbf{P}_J^{\mathrm{T}}\mathbf{D}_J\mathbf{w}_J$
saída: desnormalizar $\hat{\mathbf{w}}$

ajuste ou de incerteza para uma dada quantidade de amostras [42].

A quantidade de coeficientes indicada por cada critério é função do número de amostras utilizadas, do número de coeficientes estimados e do erro de estimação. A quantidade ótima de coeficientes é escolhida pelo menor resultado da equação de

cada critério [24, 25]. Para isso, todos os modelos possíveis devem ser calculados, ou seja, deve ser estimado o modelo com $1, 2, \cdots$, até J coeficientes. O menor resultado apontado pelos métodos AIC/BIC/C_p proporciona o modelo com a quantidade de coeficientes ideal para aquele critério em particular.

Isto já demonstra uma desvantagem destes critérios para serem usados em conjunto com o algoritmo LAR: o algoritmo deve ser processado iterativamente até que todos os coeficientes sejam estimados; somente após a última iteração pode-se então definir, de todos os coeficientes estimados, quantos devem ser efetivamente utilizados. Em outras palavras, nenhum destes critérios é um critério de parada de um algoritmo, apenas critérios para seleção do número de coeficientes do modelo ótimo.

A seguir, é apresentado cada critério separadamente.

2.4.1 *Akaike Information Criterion*

Hirotsugu Akaike introduziu, em 1973, o conceito de critério de informação como uma ferramenta para a seleção ótima de modelo. Akaike sugeriu a seguinte equação para avaliação de seleção de modelo [24]:

$$AIC = K \log\left(\frac{\|\mathbf{d} - \mathbf{y}\|^2}{K}\right) + 2N, \tag{2.114}$$

onde K é a quantidade de amostras usada no cálculo do modelo, N é a quantidade de coeficientes estimados, $\mathbf{d}$ é o vetor de referência, e $\mathbf{y}$ é o vetor de saída estimado, de modo que $\|\mathbf{d} - \mathbf{y}\|$ é o erro de estimação.

O primeiro termo mede a falta de ajuste do modelo, enquanto que o segundo termo é uma penalidade para coeficientes adicionados ao modelo [42]. Assim, quando o número de coeficientes estimados N aumenta, o erro estimado diminui, porém a penalidade do modelo aumenta. Outra interpretação, dada por Aguirre [1], é a de que à medida que termos são incluídos no modelo, o número de graus de liberdade aumenta permitindo um ajuste aos dados mais exato.

Para um número pequeno de amostras, quando $K/N < 40$, deve ser usado o chamado *Akaike Information Criterion* de segunda ordem [24], dado por

$$AIC = K \log\left(\frac{\|\mathbf{d} - \mathbf{y}\|^2}{K}\right) + 2N + \frac{2N(N+1)}{K - N + 1}. \tag{2.115}$$

Para grandes quantidade de amostras (maiores valores de K), o último termo tende a zero, de modo que (2.115) resulta em (2.114).

Para a identificação dos sistemas não-lineares proposta, diversas quantidades de amostras foram utilizadas, desde uma pequena quantidade de amostras ($K/N < 40$) até uma grande quantidade delas ($K/N > 40$). Desta forma, (2.115) foi a equação utilizada para avaliação do critério AIC.

2.4.2 *Bayesian Information Criterion*

Gideon Schwarz apresentou, em 1978, uma modificação no critério AIC, chamado de *Bayesian Information Criterion*, dado por [25]

$$BIC = K \log\left(\frac{\|\mathbf{d}-\mathbf{y}\|^2}{K}\right) + N \log K, \tag{2.116}$$

onde K é a quantidade de amostras usada no cálculo do modelo, N é a quantidade de coeficientes estimados, $\mathbf{d}$ é o vetor de referência, e $\mathbf{y}$ é o vetor de saída estimado, de modo que $\|\mathbf{d}-\mathbf{y}\|$ é o erro de estimação.

É clara a sua similaridade ao critério AIC: o primeiro termo mede a falta de ajuste do modelo, enquanto que o segundo termo é uma penalidade para coeficientes adicionados ao modelo. A diferença é no termo de penalidade, maior para o BIC do que para o AIC ($\log(K) > 2$ para $K > 8$) [25].

2.4.3 *Mallows C_p*

Colin Mallows apresentou, também em 1973, assim como o Akaike, um critério desenvolvido particularmente para modelos estimados através do algoritmo *Ordinary Least Squares* (OLS) [26]. Por este motivo, foi sugerido por Efron et al. como critério para indicar a quantidade de coeficientes ótima na primeira proposta apresentada em [5]. O critério, conhecido por C_p, onde p é o número de coeficientes dentro do modelo, é dado por [26]

$$C_p = \frac{\|\mathbf{d}-\mathbf{y}\|^2}{\sigma_{\mathbf{d}}^2} - K + 2N, \tag{2.117}$$

onde K é a quantidade de amostras usada no cálculo do modelo, N é a quantidade de coeficientes estimados, $\mathbf{d}$ é o vetor de saída de referência, $\mathbf{y}$ é o vetor de saída estimado, e $\sigma_{\mathbf{d}}^2$ é a variância do vetor de referência. Note que o índice p faz referência ao coeficiente, que neste trabalho é chamado de N ($p = N$). Porém, para não descaracterizar o nome do critério, foi mantido o índice p em C_p.

2.5 Conclusão do capítulo

Neste capítulo foi realizada uma apresentação dos conceitos básicos dos tópicos relacionados a esta tese. Os tópicos foram apresentados na ordem das etapas para identificação de sistemas não-lineares.

Primeiro, na Seção 2.2, diferentes maneiras para a representação de sistemas não-lineares foram apresentadas. Os três modelos em cascata LN (modelo de Wiener), NL (modelo de Hammerstein), e LNL foram abordados. Ainda nesta seção, o filtro de Volterra foi delineado.

A seguir, na Seção 2.3, algoritmos de estimação foram detalhados; dentre aqueles apresentados estão três algoritmos da família *Least Squares*: LS, SSS, e CLS. O algoritmo LAR foi, então, minuciosamente descrito, com todas as equações determinadas na maneira convencionalmente utilizada na área de processamento de sinais. O algoritmo LAR foi abordado com diversos detalhes não explícitos em trabalhos anteriores. Ao final desta seção, o algoritmo LASSO foi apresentado, com o enfoque na modificação necessária ao algoritmo LAR para que a resposta seja a mesma do algoritmo LASSO.

Finalizando, na Seção 2.4, três critérios para seleção de modelo, amplamente conhecidos, foram apresentados: AIC, BIC e Mallows C_p. Os critérios para seleção de modelo não são critérios de parada de um algoritmo, pois necessitam que todos os J modelos possíveis, sendo J a quantidade total de coeficientes, sejam criados para selecionar o melhor deles.

O levantamento do estado da arte mostrou que o algoritmo LAR, recentemente desenvolvido, é um algoritmo com características que favorecem a sua utilização na identificação de sistemas esparsos. Sua principal desvantagem é a alta complexidade computacional e a impossibilidade de ser recursivo no tempo, devido ao uso de dados em bateladas. O bom desempenho do algoritmo LAR em diversas aplicações gerou um interesse de desenvolver um critério de parada, a fim de torná-lo mais eficiente na identificação de sistemas esparsos. Este é o tema do próximo capítulo.

Capítulo 3

Algoritmo GLAR

3.1 Introdução do capítulo

Neste capítulo é apresentada a primeira contribuição desta tese: o algoritmo aqui designado por GLAR, i.e., o algoritmo LAR incluindo um critério de parada geométrico. A motivação inicial ao desenvolver o critério de parada geométrico para o algoritmo LAR era gerar um algoritmo capaz de trabalhar de modo eficiente com sistemas esparsos. Sistemas esparsos são, por definição, sistemas que geram modelos com uma grande quantidade de coeficientes nulos.

O algoritmo LAR estima um novo coeficiente a cada iteração: após n iterações, n coeficientes são estimados, permanecendo os coeficientes restantes iguais a zero. Será mostrado que o algoritmo LAR identifica corretamente os N coeficientes não-nulos do sistema esparso próximo a $n = N$, de modo que interrompê-lo antes da estimação de todos os coeficientes não prejudica a identificação do sistema. Com este objetivo, foi desenvolvido o algoritmo GLAR, nome dado ao algoritmo LAR com o critério de parada geométrico, apresentado na Seção 3.2.

Para validar o esquema proposto, na Seção 3.3, sistemas não-lineares são identificados com o algoritmo GLAR em conjunto com o filtro de Volterra. Primeiro, a quantidade de coeficientes determinada pelo critério de parada geométrico é comparada à quantidade de coeficientes determinada pelos critérios de seleção AIC, BIC e C_p. Em seguida, os vetores de coeficientes estimados resultantes dos algoritmos GLAR, LS, CLS e SSS são comparados.

3.2 Desenvolvimento do algoritmo GLAR

Conforme discutido no Capítulo 1, o primeiro objetivo ao estudar o algoritmo LAR foi desenvolver um critério de parada para interromper o algoritmo utilizando apenas a informação disponível na iteração n. Com o critério de parada geométrico é

possível interromper o algoritmo LAR antes da estimação de todos os coeficientes.

São duas as vantagens do algoritmo LAR ser interrompido antes de calcular todos os coeficientes: diminuir a complexidade computacional e, no caso de sistemas esparsos, determinar coeficientes com magnitude zero (quantos e quais são os coeficientes). Esta motivação levou ao desenvolvimento do critério de parada para o algoritmo LAR, chamado de critério de parada geométrico.

O critério de parada geométrico é assim chamado pois utiliza os ângulos, $\theta_{j,n}$, para avaliar se o algoritmo deve ser interrompido na iteração corrente n. Na Seção 3.2.1, estes ângulos são determinados. O critério de parada geométrico em si é apresentado na Seção 3.2.2.

Em seguida, o algoritmo LAR é desenvolvido de modo recursivo em n. O algoritmo LAR apresentado em detalhes no Capítulo 2, Seção 2.3.2, não apresenta nenhuma tentativa em relação à otimização – a complexidade computacional não estava em questão. A recursividade em n, como demonstrado na Seção 3.2.3, tem o objetivo de desenvolver um algoritmo mais eficiente.

Por fim, o pseudocódigo do algoritmo GLAR e sua complexidade computacional são apresentados na Seção 3.2.4, onde as equações recursivas pertinentes foram inseridas.

3.2.1 Determinação dos ângulos $\theta_{j,n}$

O vetor de correlação (2.61), definido por

$$\mathbf{c}_n = \mathbf{X}\mathbf{e}_n,$$
$$\begin{bmatrix} c_{1,n} \\ \vdots \\ c_{j,n} \\ \vdots \\ c_{J,n} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^{\mathrm{T}}\mathbf{e}_n \\ \vdots \\ \mathbf{x}_j^{\mathrm{T}}\mathbf{e}_n \\ \vdots \\ \mathbf{x}_J^{\mathrm{T}}\mathbf{e}_n \end{bmatrix}, \tag{3.1}$$

é o vetor base calculado a cada iteração do algoritmo LAR.

Na última iteração do algoritmo LAR ($n = J$), a resposta obtida é a resposta LS. Neste caso, os vetores de dados dos coeficientes são ortogonais ao vetor de erro de predição,

$$\mathbf{x}_j \perp \mathbf{e}_J, \ \ \forall j, \tag{3.2}$$

ou seja,

$$c_{j,J} = \mathbf{x}_j^{\mathrm{T}}\mathbf{e}_J = 0, \ \ \forall j, \tag{3.3}$$

conhecido como *princípio da ortogonalidade* [17]. Antes da última iteração, o ângulo

entre cada vetor de dados de coeficiente e o vetor de erro de predição é diferente de 90^o.

O critério de parada geométrico para o algoritmo LAR proposto nesta tese é baseado no conceito do princípio da ortogonalidade e no vetor conhecido $\mathbf{c}_n$. O critério de parada geométrico utiliza os ângulos entre os vetores de dados de coeficientes e o vetor de erro de predição, calculado pelo produto interno $< \mathbf{x}_j, \mathbf{e}_n >$,

$$\theta_{j,n} = \arccos \frac{\mathbf{x}_j^{\mathrm{T}} \mathbf{e}_n}{\|\mathbf{x}_j^{\mathrm{T}}\| \|\mathbf{e}_n\|} = \arccos \frac{\mathbf{x}_j^{\mathrm{T}} \mathbf{e}_n}{\|\mathbf{e}_n\|}, \tag{3.4}$$

onde $\|\mathbf{x}_j\| = 1$ devido à normalização. Os ângulos $\theta_{j,n}, \forall j$, são agrupados no vetor de ângulos $(J \times 1)$ definido por

$$\boldsymbol{\theta}_n = [\theta_{1,n} \cdots \theta_{j,n} \cdots \theta_{J,n}]^{\mathrm{T}}. \tag{3.5}$$

Comparando (3.1) e (3.4), constata-se que os ângulos $\theta_{j,n}$ e o vetor de correlação $\mathbf{c}_n$ estão diretamente relacionados. O comportamento de $\theta_{j,n}$, à medida que o valor de n aumenta, é alterado junto com o valor máximo de correlação, $C_{max,n}$.

O valor máximo de correlação é monotonicamente decrescente, como demonstrado em (2.78), dado por

$$C_{max,n} = C_{max,n-1} - \gamma_n \alpha_n. \tag{3.6}$$

Pelo algoritmo LAR, $C_{max,n}$ é o valor de correlação absoluto dos coeficientes ativos, enquanto que o valor de correlação absoluto dos coeficientes inativos é sempre menor do que aquele, ou seja,

$$|c_{j,n}| \begin{cases} = C_{max,n}, & j \in \mathcal{A}_n \\ < C_{max,n}, & j \in \bar{\mathcal{A}}_n. \end{cases} \tag{3.7}$$

Para $j \in \bar{\mathcal{A}}_n$, (3.7) pode ser simplificado para $c_{j,n} < C_{max,n}$.

Para os coeficientes ativos, $j \in \mathcal{A}_n$, o ângulo é dado por

$$\theta_{j,n} = \begin{cases} \arccos \dfrac{-C_{max,n}}{\|\mathbf{e}_n\|}, & c_{j,n} < 0 \\ \arccos \dfrac{C_{max,n}}{\|\mathbf{e}_n\|}, & c_{j,n} > 0. \end{cases} \tag{3.8}$$

Baseado na Figura 3.1, pode-se perceber que

$$\left| 90^o - \arccos \frac{C_{max,n}}{\|\mathbf{e}_n\|} \right| = \left| 90^o - \arccos \frac{-C_{max,n}}{\|\mathbf{e}_n\|} \right|. \tag{3.9}$$

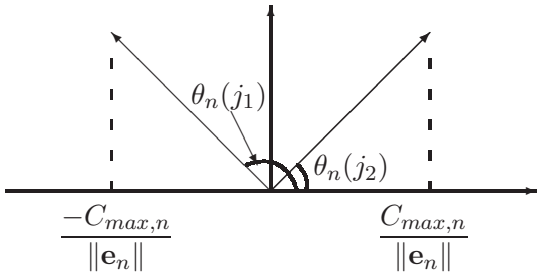


Figura 3.1: Exemplo gráfico dos ângulos de dois coeficientes, j_1 e $j_2 \in \mathcal{A}$, onde $c_{j_1,n} = -C_{max,n}$ e $c_{j_2,n} = C_{max,n}$.

Sabendo que

$$a < b \quad \Rightarrow \quad \arccos a > \arccos b, \tag{3.10}$$

uma vez que $c_{j,n} < C_{max,n}, j \in \bar{\mathcal{A}}_n$,

$$\arccos \frac{c_{j,n}}{\|\mathbf{e}_n\|} > \arccos \frac{C_{max,n}}{\|\mathbf{e}_n\|}, \tag{3.11}$$

assim como

$$\left| 90^o - \arccos \frac{c_{j,n}}{\|\mathbf{e}_n\|} \right| < 90^o - \arccos \frac{C_{max,n}}{\|\mathbf{e}_n\|}. \tag{3.12}$$

Este resultado, ilustrado na Figura 3.2, indica que os ângulos de coeficientes inativos sempre estão mais próximos a 90^o do que os ângulos de coeficientes ativos. Os coeficientes ativos, escolhidos até a iteração atual n, são aqueles com maior valor absoluto de correlação $|c_{j,n}|$, que mais aumentam o erro residual e mais longe estão de 90^o. Por este motivo são escolhidos para serem estimados, e, assim, diminuir o erro residual.

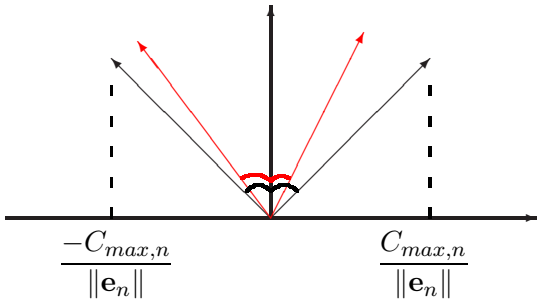


Figura 3.2: Os ângulos dos coeficientes inativos, representados em vermelho, estão sempre mais próximos a 90^o do que os ângulos dos coeficientes ativos, representados em preto.

3.2.2 O critério de parada geométrico

A partir de sistemas esparsos conhecidos, foi observado o comportamento dos ângulos $\theta_{j,n}, \forall j$. A iteração ótima, $n = N$, é definida pelo momento em que todos

os coeficientes não-nulos foram estimados e, consequentemente, o algoritmo LAR já é passível de ser interrompido.

Com as observações realizadas, o primeiro critério de parada geométrico, inicialmente sugerido em [32], foi o de interromper o algoritmo LAR quando

$$\Delta\theta_n \leq \sigma_{\boldsymbol{\theta}_1}, \tag{3.13}$$

onde

$$\Delta\theta_n = \max(\boldsymbol{\theta}_n) - \min(\boldsymbol{\theta}_n); \tag{3.14}$$

ou seja, quando a extensão dos ângulos ($\Delta\theta_n$) é igual ou menor do que o desvio padrão inicial dos ângulos ($\sigma_{\boldsymbol{\theta}_1}$), obtido na primeira iteração, assume-se que todos os coeficientes necessários foram estimados e o algoritmo é interrompido. Todos os ângulos estavam contidos dentro de um cone limitado pelo desvio padrão inicial dos ângulos, como sugerido pela Figura 3.2 para um modelo com duas dimensões. Com o decorrer desta investigação, o critério foi melhorado, conforme descrito a seguir.

Seguindo as conclusões da Seção 3.2.1, ao calcular $c_{j,n}, j \in \mathcal{A}_n$, a cada iteração, uma das três situações a seguir ocorre:

1. só existe $c_{j,n} = +C_{max,n}, j \in \mathcal{A}_n$: o ângulo mínimo é gerado por coeficiente(s) ativo(s);

2. só existe $c_{j,n} = -C_{max,n}, j \in \mathcal{A}_n$: o ângulo máximo é gerado por coeficiente(s) ativo(s); ou

3. existem $c_{j,n} = \pm C_{max,n}, j \in \mathcal{A}_n$: ambos os ângulos, mínimo e máximo, são gerados por coeficientes ativos.

Na primeira iteração, o caso (1) ou (2) acima ocorre, dependendo se $c_{j_1,1} > 0$ ou $c_{j_1,1} < 0, j_1 \in \mathcal{A}_1$. A partir da segunda iteração, uma das três opções apontadas a seguir ocorre:

- novamente o caso (1), se $c_{j_1,2} = c_{j_2,2} > 0, j_{1,2} \in \mathcal{A}_2$;
- novamente o caso (2), se $c_{j_1,2} = c_{j_2,2} < 0, j_{1,2} \in \mathcal{A}_2$; ou
- o caso (3), quando $|c_{j_1,2}| = |c_{j_2,2}|, j_{1,2} \in \mathcal{A}_2$.

Mesmo que o caso (3) não aconteça na segunda iteração, é esperado que ocorra logo. Uma vez que o caso (3) aconteça, ele se repete até o final do algoritmo, uma vez que o coeficiente ativo permanece ativo com seu valor de correlação diminuindo em magnitude. A terceira situação é apresentada na Figura 3.3.

Quando a situação (3) ocorre, $\Delta\theta_n$ calculado por (3.14)

$$\Delta\theta_n = \max(\boldsymbol{\theta}_n) - \min(\boldsymbol{\theta}_n) = \theta_{max,n} - \theta_{min,n},$$

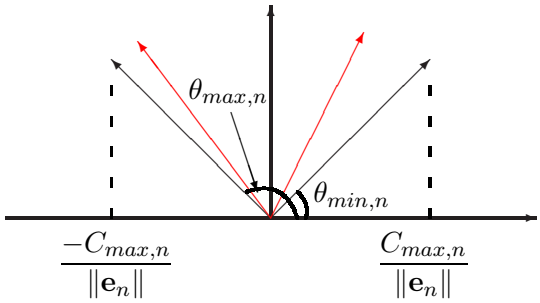


Figura 3.3: Os ângulos máximo, $\theta_{max,n}$, e mínimo, $\theta_{min,n}$, são de coeficientes ativos, $j \in \mathcal{A}_n$. Os outros ângulos, oriundos dos vetores em vermelho, são de coeficientes inativos, $\theta_{max,n} > \theta_{j,n} > \theta_{min,n}, j \in \bar{\mathcal{A}}_n$.

baseado na Figura 3.3, tem como resultado duas vezes a distância do maior (ou menor) ângulo até 90^o. Assim, é desnecessário calcular todos os ângulos para saber qual o valor de $\Delta\theta_n$, podendo ser reduzido a

$$\Delta\theta_n = 2\left(90^o - \arccos\frac{C_{max,n}}{\|\mathbf{e}_n\|}\right). \tag{3.15}$$

Isto diminui muito a complexidade computacional do critério proposto em [32].

Voltando às possíveis situações que podem ocorrer, uma situação que ocorre com menor probabilidade é quando apenas o caso (1) ou apenas o caso (2) acontece em todas as iterações, fato ilustrado na Figura 3.4. Caso os ângulos oriundos dos coeficientes inativos já estejam muito próximos a 90^o, calcular $\Delta\theta_n = \theta_{max,n} - \theta_{min,n}$ pode satisfazer de forma prematura $\Delta\theta_n < \sigma_{\boldsymbol{\theta}_1}$, interrompendo o algoritmo de estimação com vetores de dados de coeficientes ativos ainda longe de 90^o.

Em outras palavras, calcular $\Delta\theta_n = \theta_{max,n} - \theta_{min,n}$, como em (3.14), pode resultar em um valor menor do que $\Delta\theta_n = 2\left(90^o - \arccos\frac{C_{max,n}}{\|\mathbf{e}_n\|}\right)$, como em (3.15), levando o algoritmo a ser interrompido antes dos vetores de dados de coeficientes ativos estarem próximos de 90^o. Como consequência, o erro de estimação seria maior. Assim, é mais eficiente utilizar o cálculo de $\Delta\theta_n$ como em (3.15), pois é necessário que os coeficientes ativos estejam próximos a 90^o para uma correta estimação do vetor de coeficientes.

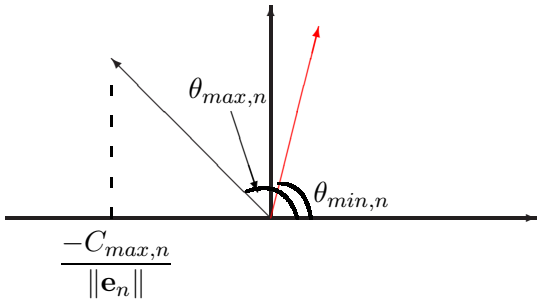


Figura 3.4: Ilustração em que ocorre apenas a situação (2) ao longo das iterações do algoritmo LAR.

Um último ponto é adicionado ao critério de parada geométrico proposto em [32]: o fator de penalidade, $0 \leq \mu \leq 1$. Todos os critérios de seleção aqui avaliados, conforme apresentado na Seção 2.4, possuem um fator de penalidade que determina o modelo com a menor complexidade possível para a quantidade de dados disponível. No critério de parada geométrico aqui apresentado este fator de penalidade é dado pela constante μ. Quanto menor o valor de μ, maior é a penalidade, porém melhor é o ajuste.

O objetivo de μ é aprimorar o resultado do critério de parada geométrico para uma quantidade de dados limitada. Quanto menor o valor de μ, mais próximos estão os vetores de dados de coeficientes a 90^o; consequentemente, o critério de parada geométrico fica mais restrito. Quanto menor o valor de μ, maior a quantidade de coeficientes estimada. Quando $\mu = 0$, todos os coeficientes são estimados e a resposta LS é obtida. O valor de μ poderia ser maior que um; no entanto, durante a investigação do critério de parada geométrico, foi observado que para $\mu > 1$, os vetores de dados de coeficientes estavam ainda longe de 90^o, e os coeficientes pertinentes não haviam sido escolhidos.

O critério de parada geométrico final é aqui proposto por

$$\Delta\theta_n \leq \mu\sigma_{\boldsymbol{\theta}_1}, \tag{3.16}$$

onde

$$0 \leq \mu \leq 1, \text{ e} \tag{3.17}$$

$$\Delta\theta_n = 2\left(90^o - \arccos\frac{C_{max,n}}{\|\mathbf{e}_n\|}\right). \tag{3.18}$$

Na primeira iteração em que (3.16) é satisfeito, o algoritmo é interrompido em $n = N$, determinando N como a quantidade de coeficientes escolhida.

3.2.3 Recursividade em n

Sejam as seguintes equações do algoritmo LAR, conforme definido na Seção 2.3.2, repetidas aqui por conveniência:

$$\mathbf{x}_j = [x_j(1) \ \cdots \ x_j(K)]^{\mathrm{T}}, \tag{3.19}$$

$$\mathbf{X} = [\mathbf{x}_1 \ \cdots \ \mathbf{x}_J]^{\mathrm{T}}, \tag{3.20}$$

$$\mathbf{1}_n = [\mathbf{1}_{n-1}^{\mathrm{T}} \ \ 1]^{\mathrm{T}} = [1 \cdots 1]^{\mathrm{T}}, \text{ com tamanho } n, \tag{3.21}$$

$$\mathcal{A}_n = [j_1 \ \cdots \ j_n]^{\mathrm{T}} = [\mathcal{A}_{n-1}^{\mathrm{T}} \ \ j_n]^{\mathrm{T}}, \tag{3.22}$$

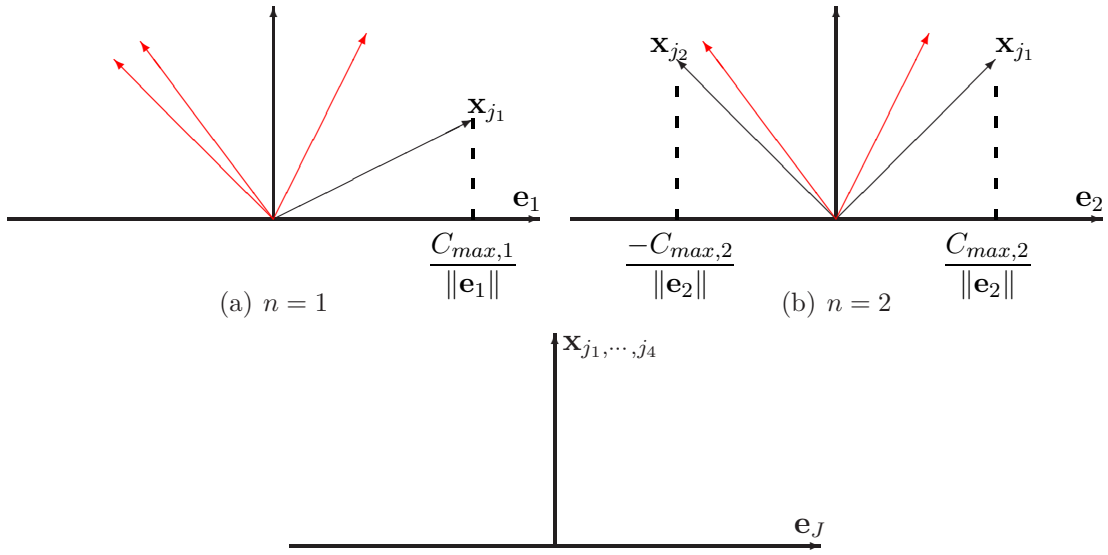


Figura 3.5: Os ângulos de todos os coeficientes, ativos (ilustrados em preto) e inativos (ilustrados em vermelho), se aproximam de 90^o com n. Neste exemplo, quatro coeficientes são identificados. Em (a), o primeiro coeficiente x_{j_1} resulta em $c_{j_1,1} = C_{max,1} > 0$. Em (b), $|c_{j_1,2}| = |c_{j_2,2}| = C_{max,2}$, porém $c_{j_1,2} > 0$ e $c_{j_2,2} < 0$. Em (c), todos os coeficientes são estimados e o ângulo com o vetor de erro residual é de 90^o.

$$\mathbf{P}_n = \begin{bmatrix} \mathbf{p}_{j_1}^{\mathrm{T}} \\ \vdots \\ \mathbf{p}_{j_n}^{\mathrm{T}} \end{bmatrix} = [\mathbf{P}_{n-1}^{\mathrm{T}} \;\; \mathbf{p}_{j_n}]^{\mathrm{T}}, \tag{3.23}$$

$$\mathbf{D}_n = \mathrm{diag}(s_{j_1} \cdots s_{j_n}) = \mathrm{diag}(\mathbf{P}_n \; \mathrm{sign}(\mathbf{c}_n)) \tag{3.24}$$

$$\mathbf{X}_n = \mathbf{D}_n \mathbf{P}_n \mathbf{X}, \tag{3.25}$$

$$\mathbf{S}_n = (\mathbf{X}_n \mathbf{X}_n^{\mathrm{T}})^{-1}, \tag{3.26}$$

$$\alpha_n = (\mathbf{1}_n^{\mathrm{T}} \mathbf{S}_n \mathbf{1}_n)^{-1/2}, \tag{3.27}$$

$$\boldsymbol{\omega}_n = \alpha_n \mathbf{S}_n \mathbf{1}_n, \tag{3.28}$$

$$\mathbf{u}_n = \mathbf{X}_n^{\mathrm{T}} \boldsymbol{\omega}_n, \text{ e} \tag{3.29}$$

$$\mathbf{a}_n = \mathbf{X} \mathbf{u}_n. \tag{3.30}$$

Tendo em vista que (3.19) e (3.20) não variam em n, bem como (3.21) a (3.23) já estão recursivas em n, um estudo sobre a viabilidade de calcular as equações (3.24) a (3.30) de modo recursivo é apresentado a seguir.

(3.24) $\mathbf{D}_n$ recursivo

A primeira equação a ser definida no modo recursivo é a matriz diagonal de sinais $\mathbf{D}_n$. Uma vez dentro do conjunto ativo, o sinal de $c_{j_n,n}$ não muda [5]. Assim, a matriz sinal tem a sua recursividade descrita por

$$\mathbf{D}_n = \begin{bmatrix} \mathbf{D}_{n-1} & 0 \\ 0 & \text{sign}(\mathbf{p}_{j_n}^{\mathrm{T}}\mathbf{c}_n) \end{bmatrix}. \tag{3.31}$$

(3.25) $\mathbf{X}_n$ recursivo

A matriz de dados normalizada, $\mathbf{X}_n$, é calculada de modo recursivo substituindo (3.23) e (3.31) em (3.25),

$$\begin{aligned} \mathbf{X}_n &= \mathbf{D}_n\mathbf{P}_n\mathbf{X} \\ &= \begin{bmatrix} \mathbf{D}_{n-1} & 0 \\ 0 & \text{sign}(\mathbf{p}_{j_n}^{\mathrm{T}}\mathbf{c}_n) \end{bmatrix} \begin{bmatrix} \mathbf{P}_{n-1} \\ \mathbf{p}_{j_n}^{\mathrm{T}} \end{bmatrix} \mathbf{X} \\ &= \begin{bmatrix} \mathbf{D}_{n-1}\mathbf{P}_{n-1}\mathbf{X} \\ \text{sign}(\mathbf{p}_{j_n}^{\mathrm{T}}\mathbf{c}_n)\mathbf{p}_{j_n}^{\mathrm{T}}\mathbf{X} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{X}_{n-1} \\ \mathbf{x}_{j_n}^{\mathrm{T}} \end{bmatrix}, \end{aligned} \tag{3.32}$$

onde

$$\mathbf{x}_{j_n}^{\mathrm{T}} = \text{sign}(\mathbf{p}_{j_n}^{\mathrm{T}}\mathbf{c}_n)\mathbf{p}_{j_n}^{\mathrm{T}}\mathbf{X}. \tag{3.33}$$

(3.26) $\mathbf{S}_n$ recursivo

A matriz de correlação inversa, $\mathbf{S}_n$, pode ser calculada usando o Lema de Inversão de Matrizes em Bloco [43]. Para $\mathbf{C}$ $(n \times n)$, $\mathbf{f}$ $(n \times 1)$, e h (1×1), o Lema de Inversão de Matrizes em Bloco [43] estabelece

$$\mathbf{B} = \begin{bmatrix} \mathbf{C} & \mathbf{f} \\ \mathbf{f}^{\mathrm{T}} & h \end{bmatrix} \;\Rightarrow\; \mathbf{B}^{-1} = \begin{bmatrix} \mathbf{C}^{-1} + \dfrac{1}{g}\mathbf{C}^{-1}\mathbf{f}\mathbf{f}^{\mathrm{T}}\mathbf{C}^{-1} & -\dfrac{1}{g}\mathbf{C}^{-1}\mathbf{f} \\ -\dfrac{1}{g}\mathbf{f}^{\mathrm{T}}\mathbf{C}^{-1} & \dfrac{1}{g} \end{bmatrix}, \tag{3.34}$$

onde $g = h - \mathbf{f}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{f}$.

Desenvolvendo $\mathbf{R}_n$, lembrando que $\mathbf{x}_{j_n}^{\mathrm{T}}\mathbf{x}_{j_n} = \|\mathbf{x}_{j_n}\|^2 = 1$,

$$\begin{aligned} \mathbf{R}_n &= \mathbf{X}_n\mathbf{X}_n^{\mathrm{T}} \\ &= \begin{bmatrix} \mathbf{X}_{n-1} \\ \mathbf{x}_{j_n}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{n-1}^{\mathrm{T}} & \mathbf{x}_{j_n} \end{bmatrix} \end{aligned}$$

$$
\begin{aligned}
&= \begin{bmatrix} \mathbf{X}_{n-1}\mathbf{X}_{n-1}^{\mathrm{T}} & \mathbf{X}_{n-1}\mathbf{x}_{j_n} \\ \mathbf{x}_{j_n}^{\mathrm{T}}\mathbf{X}_{n-1}^{\mathrm{T}} & \mathbf{x}_{j_n}^{\mathrm{T}}\mathbf{x}_{j_n} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{R}_{n-1} & \mathbf{X}_{n-1}\mathbf{x}_{j_n} \\ \mathbf{x}_{j_n}^{\mathrm{T}}\mathbf{X}_{n-1}^{\mathrm{T}} & 1 \end{bmatrix}. \qquad (3.35)
\end{aligned}
$$

Aplicando o Lema de Inversão de Matrizes em Bloco, usando $\mathbf{S} = \mathbf{R}^{-1}$,

$$
\mathbf{S}_n = \begin{bmatrix} \mathbf{S}_{n-1} + \frac{1}{g_n}\mathbf{S}_{n-1}\mathbf{f}_n\mathbf{f}_n^{\mathrm{T}}\mathbf{S}_{n-1} & -\frac{1}{g_n}\mathbf{S}_{n-1}\mathbf{f}_n \\ -\frac{1}{g_n}\mathbf{f}_n^{\mathrm{T}}\mathbf{S}_{n-1} & \frac{1}{g_n} \end{bmatrix}, \qquad (3.36)
$$

onde

$$
\begin{aligned}
\mathbf{f}_n &= \mathbf{X}_{n-1}\mathbf{x}_{j_n}, \text{ e} \qquad (3.37) \\
g_n &= 1 - \mathbf{f}_n^{\mathrm{T}}\mathbf{S}_{n-1}\mathbf{f}_n. \qquad (3.38)
\end{aligned}
$$

Chamando de

$$
\mathbf{v}_n = \mathbf{S}_{n-1}\mathbf{f}_n, \qquad (3.39)
$$

(3.36) é reduzida a

$$
\mathbf{S}_n = \begin{bmatrix} \mathbf{S}_{n-1} + \frac{1}{g_n}\mathbf{v}_n\mathbf{v}_n^{\mathrm{T}} & -\frac{1}{g_n}\mathbf{v}_n \\ -\frac{1}{g_n}\mathbf{v}_n^{\mathrm{T}} & \frac{1}{g_n} \end{bmatrix}, \qquad (3.40)
$$

assim como (3.38) é reduzida a

$$
g_n = 1 - \mathbf{f}_n^{\mathrm{T}}\mathbf{v}_n. \qquad (3.41)
$$

As equações acima apresentadas compõem a forma mais simples para calcular $\mathbf{S}_n$, pois $\mathbf{f}_n$, $\mathbf{v}_n$ e g_n não podem ser reduzidas, como demonstrado a seguir:

- $\mathbf{f}_n$ – Substituindo (3.32) em (3.37),

$$
\begin{aligned}
\mathbf{f}_n &= \mathbf{X}_{n-1}\mathbf{x}_{j_n} \\
&= \begin{bmatrix} \mathbf{X}_{n-2} \\ \mathbf{x}_{j_{n-1}}^{\mathrm{T}} \end{bmatrix}\mathbf{x}_{j_n} \\
&= \begin{bmatrix} \mathbf{X}_{n-2}\mathbf{x}_{j_n} \\ \mathbf{x}_{j_{n-1}}^{\mathrm{T}}\mathbf{x}_{j_n} \end{bmatrix}. \qquad (3.42)
\end{aligned}
$$

  Por exemplo, para n = 1,2,3,4:

  1. $\mathbf{f}_1 = [\,]$;
  2. $\mathbf{X}_{n-1} = \mathbf{X}_1 = \mathbf{x}_{j_1}^{\mathrm{T}}$ e $\mathbf{x}_{j_n} = \mathbf{x}_{j_2}$, logo $\mathbf{f}_2 = \mathbf{X}_1\mathbf{x}_{j_2} = \mathbf{x}_{j_1}^{\mathrm{T}}\mathbf{x}_{j_2}$;

3. $\mathbf{X}_{n-1} = \mathbf{X}_2 = \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{x}_{j_2}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{j_1}^{\mathrm{T}} \\ \mathbf{x}_{j_2}^{\mathrm{T}} \end{bmatrix}$ e $\mathbf{x}_{j_n} = \mathbf{x}_{j_3}$, logo $\mathbf{f}_3 = \mathbf{X}_2\mathbf{x}_{j_3} = \begin{bmatrix} \mathbf{x}_{j_1}^{\mathrm{T}}\mathbf{x}_{j_3} \\ \mathbf{x}_{j_2}^{\mathrm{T}}\mathbf{x}_{j_3} \end{bmatrix}$;

4. $\mathbf{X}_{n-1} = \mathbf{X}_3 = \begin{bmatrix} \mathbf{X}_2 \\ \mathbf{x}_{j_3}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{j_1}^{\mathrm{T}} \\ \mathbf{x}_{j_2}^{\mathrm{T}} \\ \mathbf{x}_{j_3}^{\mathrm{T}} \end{bmatrix}$ e $\mathbf{x}_{j_n} = \mathbf{x}_{j_4}$, logo $\mathbf{f}_4 = \mathbf{X}_3\mathbf{x}_{j_4} = \begin{bmatrix} \mathbf{x}_{j_1}^{\mathrm{T}}\mathbf{x}_{j_4} \\ \mathbf{x}_{j_2}^{\mathrm{T}}\mathbf{x}_{j_4} \\ \mathbf{x}_{j_3}^{\mathrm{T}}\mathbf{x}_{j_4} \end{bmatrix}$.

Assim, não há recursividade entre $\mathbf{f}_n$ e $\mathbf{f}_{n-1}$.

- $\mathbf{v}_n$ – Substituindo (3.40) e (3.42) em (3.39),

$$\begin{aligned} \mathbf{v}_n &= \mathbf{S}_{n-1}\mathbf{f}_n \\ &= \begin{bmatrix} \mathbf{S}_{n-2} + \frac{1}{g_{n-1}}\mathbf{v}_{n-1}\mathbf{v}_{n-1}^{\mathrm{T}} & -\frac{1}{g_{n-1}}\mathbf{v}_{n-1} \\ -\frac{1}{g_{n-1}}\mathbf{v}_{n-1}^{\mathrm{T}} & \frac{1}{g_{n-1}} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{n-2}\mathbf{x}_{j_n} \\ \mathbf{x}_{j_{n-1}}\mathbf{x}_{j_n}^{\mathrm{T}} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{S}_{n-2}\mathbf{X}_{n-2}\mathbf{x}_{j_n} + \frac{1}{g_{n-1}}\mathbf{v}_{n-1}\mathbf{v}_{n-1}^{\mathrm{T}}\mathbf{X}_{n-2}\mathbf{x}_{j_n} - \frac{1}{g_{n-1}}\mathbf{v}_{n-1}\mathbf{x}_{j_{n-1}}^{\mathrm{T}}\mathbf{x}_{j_n} \\ -\frac{1}{g_{n-1}}\mathbf{v}_{n-1}^{\mathrm{T}}\mathbf{X}_{n-2}\mathbf{x}_{j_n} + \frac{1}{g_{n-1}}\mathbf{x}_{j_{n-1}}^{\mathrm{T}}\mathbf{x}_{j_n} \end{bmatrix}. \end{aligned} \quad (3.43)$$

O primeiro termo, de modo geral, é o termo recursivo, o que não ocorre aqui, pois $\mathbf{X}_{n-2}\mathbf{x}_{j_n} \neq \mathbf{f}_{n-1}$. Apesar de $\mathbf{v}_{n-1}$ estar presente em outros termos, a complexidade computacional de (3.39) é menor do que (3.43), pois além da maior quantidade de matrizes, existe a presença de $\mathbf{X}_{n-2}$, de tamanho $(n-2 \times K)$.

- g_n – Substituindo (3.42) e (3.43) em (3.41),

$$\begin{aligned} g_n &= 1 - \mathbf{f}_n^{\mathrm{T}}\mathbf{v}_n \\ &= 1 - \begin{bmatrix} \mathbf{x}_{j_n}^{\mathrm{T}}\mathbf{X}_{n-2}^{\mathrm{T}} & \mathbf{x}_{j_n}^{\mathrm{T}}\mathbf{x}_{j_{n-1}} \end{bmatrix} \begin{bmatrix} \mathbf{S}_{n-2}\mathbf{X}_{n-2}\mathbf{x}_{j_n} + \frac{1}{g_{n-1}}\mathbf{v}_{n-1}\mathbf{v}_{n-1}^{\mathrm{T}}\mathbf{X}_{n-2}\mathbf{x}_{j_n} - \frac{1}{g_{n-1}}\mathbf{v}_{n-1}\mathbf{x}_{j_{n-1}}^{\mathrm{T}}\mathbf{x}_{j_n} \\ -\frac{1}{g_{n-1}}\mathbf{v}_{n-1}^{\mathrm{T}}\mathbf{X}_{n-2}\mathbf{x}_{j_n} + \frac{1}{g_{n-1}}\mathbf{x}_{j_{n-1}}^{\mathrm{T}}\mathbf{x}_{j_n} \end{bmatrix} \\ &= 1 - \mathbf{x}_{j_n}^{\mathrm{T}}\mathbf{X}_{n-2}^{\mathrm{T}}\mathbf{S}_{n-2}\mathbf{X}_{n-2}\mathbf{x}_{j_n} + \frac{1}{g_{n-1}}\mathbf{x}_{j_n}^{\mathrm{T}}\mathbf{X}_{n-2}^{\mathrm{T}}\mathbf{v}_{n-1}\mathbf{v}_{n-1}^{\mathrm{T}}\mathbf{X}_{n-2}\mathbf{x}_{j_n} \\ &\quad + \frac{1}{g_{n-1}}\mathbf{x}_{j_n}^{\mathrm{T}}\mathbf{X}_{n-2}^{\mathrm{T}}\mathbf{v}_{n-1}\mathbf{x}_{j_{n-1}}^{\mathrm{T}}\mathbf{x}_{j_n} + \frac{1}{g_{n-1}}\mathbf{x}_{j_n}^{\mathrm{T}}\mathbf{x}_{j_{n-1}}\mathbf{v}_{n-1}^{\mathrm{T}}\mathbf{X}_{n-2}\mathbf{x}_{j_n} - \frac{1}{g_{n-1}}\mathbf{x}_{j_n}^{\mathrm{T}}\mathbf{x}_{j_{n-1}}\mathbf{x}_{j_{n-1}}^{\mathrm{T}}\mathbf{x}_{j_n} \\ &= 1 - \mathbf{x}_{j_n}^{\mathrm{T}}\mathbf{X}_{n-2}^{\mathrm{T}}\mathbf{S}_{n-2}\mathbf{X}_{n-2}\mathbf{x}_{j_n} + \frac{1}{g_{n-1}}\mathbf{x}_{j_n}^{\mathrm{T}}\mathbf{X}_{n-2}^{\mathrm{T}}\mathbf{v}_{n-1}\left(\mathbf{v}_{n-1}^{\mathrm{T}}\mathbf{X}_{n-2} + 2\mathbf{x}_{j_{n-1}}^{\mathrm{T}}\right)\mathbf{x}_{j_n} \\ &\quad - \frac{1}{g_{n-1}}(\mathbf{x}_{j_n}^{\mathrm{T}}\mathbf{x}_{j_{n-1}})^2. \end{aligned} \quad (3.44)$$

Como $\mathbf{x}_{j_n}^{\mathrm{T}}\mathbf{X}_{n-2}^{\mathrm{T}} \neq \mathbf{f}_{n-1}^{\mathrm{T}}$, não há recursividade em $1 - \mathbf{x}_{j_n}^{\mathrm{T}}\mathbf{X}_{n-2}^{\mathrm{T}}\mathbf{S}_{n-2}\mathbf{X}_{n-2}\mathbf{x}_{j_n}$. Com este resultado, a complexidade computacional de (3.41) é menor do que (3.44), pelos mesmos motivos apresentados para $\mathbf{v}_n$.

(3.27) α_n recursivo

O cosseno do ângulo entre cada vetor de dados de coeficiente no conjunto ativo e a direção equiangular a ser seguida, α_n, isto é, o cosseno no ângulo entre $\mathbf{x}_j, j \in \mathcal{A}$, e $\mathbf{u}_n$, é obtido de forma recursiva substituindo (3.21) e (3.40) em (3.27),

$$\begin{aligned}
\alpha_n &= (\mathbf{1}_n^{\mathrm{T}}\mathbf{S}_n\mathbf{1}_n)^{-1/2} \\
&= \left(\begin{bmatrix} \mathbf{1}_{n-1}^{\mathrm{T}} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{S}_{n-1} + \frac{1}{g_n}\mathbf{v}_n\mathbf{v}_n^{\mathrm{T}} & -\frac{1}{g_n}\mathbf{v}_n \\ -\frac{1}{g_n}\mathbf{v}_n^{\mathrm{T}} & \frac{1}{g_n} \end{bmatrix} \begin{bmatrix} \mathbf{1}_{n-1} \\ 1 \end{bmatrix} \right)^{-1/2} \\
&= \left(\begin{bmatrix} \mathbf{1}_{n-1}^{\mathrm{T}}\mathbf{S}_{n-1} + \frac{1}{g_n}\mathbf{1}_{n-1}^{\mathrm{T}}\mathbf{v}_n\mathbf{v}_n^{\mathrm{T}} - \frac{1}{g_n}\mathbf{v}_n^{\mathrm{T}} & -\frac{1}{g_n}\mathbf{1}_{n-1}^{\mathrm{T}}\mathbf{v}_n + \frac{1}{g_n} \end{bmatrix} \begin{bmatrix} \mathbf{1}_{n-1} \\ 1 \end{bmatrix} \right)^{-1/2} \\
&= \left(\mathbf{1}_{n-1}^{\mathrm{T}}\mathbf{S}_{n-1}\mathbf{1}_{n-1} + \frac{1}{g_n}\mathbf{1}_{n-1}^{\mathrm{T}}\mathbf{v}_n\mathbf{v}_n^{\mathrm{T}}\mathbf{1}_{n-1} - \frac{1}{g_n}\mathbf{v}_n^{\mathrm{T}}\mathbf{1}_{n-1} - \frac{1}{g_n}\mathbf{1}_{n-1}^{\mathrm{T}}\mathbf{v}_n + \frac{1}{g_n} \right)^{-1/2}. \quad (3.45)
\end{aligned}$$

Chamando de

$$t_n = \mathbf{1}_{n-1}^{\mathrm{T}}\mathbf{v}_n = \mathbf{v}_n^{\mathrm{T}}\mathbf{1}_{n-1}, \quad (3.46)$$

e percebendo que $\mathbf{1}_{n-1}^{\mathrm{T}}\mathbf{S}_{n-1}\mathbf{1}_{n-1} = (\alpha_{n-1})^{-2}$, (3.45) é reduzida a

$$\begin{aligned}
\alpha_n &= \left(\alpha_{n-1}^{-2} + \frac{(t_n^2 - 2t_n + 1)}{g_n} \right)^{-1/2} \\
&= \left(\alpha_{n-1}^{-2} + \frac{(t_n - 1)^2}{g_n} \right)^{-1/2}. \quad (3.47)
\end{aligned}$$

(3.28) $\boldsymbol{\omega}_n$ recursivo

O vetor auxiliar para cálculo, $\boldsymbol{\omega}_n$, é calculado de modo recursivo substituindo (3.40), (3.21) e (3.46) em (3.28),

$$\begin{aligned}
\boldsymbol{\omega}_n &= \alpha_n\mathbf{S}_n\mathbf{1}_n \\
&= \alpha_n \begin{bmatrix} \mathbf{S}_{n-1} + \frac{1}{g_n}\mathbf{v}_n\mathbf{v}_n^{\mathrm{T}} & -\frac{1}{g_n}\mathbf{v}_n \\ -\frac{1}{g_n}\mathbf{v}_n^{\mathrm{T}} & \frac{1}{g_n} \end{bmatrix} \begin{bmatrix} \mathbf{1}_{n-1} \\ 1 \end{bmatrix} \\
&= \alpha_n \begin{bmatrix} \mathbf{S}_{n-1}\mathbf{1}_{n-1} + \frac{1}{g_n}\mathbf{v}_n\mathbf{v}_n^{\mathrm{T}}\mathbf{1}_{n-1} - \frac{1}{g_n}\mathbf{v}_n \\ -\frac{1}{g_n}\mathbf{v}_n^{\mathrm{T}}\mathbf{1}_{n-1} + \frac{1}{g_n} \end{bmatrix} \\
&= \begin{bmatrix} \alpha_n\mathbf{S}_{n-1}\mathbf{1}_{n-1} + \frac{\alpha_n t_n}{g_n}\mathbf{v}_n - \frac{\alpha_n}{g_n}\mathbf{v}_n \\ -\frac{\alpha_n t_n}{g_n} + \frac{\alpha_n}{g_n} \end{bmatrix} \\
&= \begin{bmatrix} \alpha_n\frac{\alpha_{n-1}}{\alpha_{n-1}}\mathbf{S}_{n-1}\mathbf{1}_{n-1} + \alpha_n\frac{t_n - 1}{g_n}\mathbf{v}_n \\ -\alpha_n\frac{t_n - 1}{g_n} \end{bmatrix}
\end{aligned}$$

$$= \begin{bmatrix} \dfrac{\alpha_n}{\alpha_{n-1}}\boldsymbol{\omega}_{n-1} + \alpha_n \dfrac{t_n - 1}{g_n}\mathbf{v}_n \\ -\alpha_n \dfrac{t_n - 1}{g_n} \end{bmatrix}. \tag{3.48}$$

Em (3.28) são N multiplicações e $N(N-1)$ somas. Em (3.48) são $2N-1$ multiplicações, 2 divisões, e $N-1$ somas. Ou seja, a recursividade diminui o número de somas, porém aumenta o número de multiplicações. Como multiplicação tem maior complexidade computacional que soma, é mais econômico utilizar a expressão dada por (3.28).

(3.29) $\mathbf{u}_n$ recursivo

A direção equiangular, $\mathbf{u}_n$, a ser seguida é calculada de modo recursivo substituindo (3.32) e (3.48) em (3.29)

$$\begin{aligned} \mathbf{u}_n &= \mathbf{X}_n^{\mathrm{T}}\boldsymbol{\omega}_n \\ &= \begin{bmatrix} \mathbf{X}_{n-1}^{\mathrm{T}} & \mathbf{x}_{j_n} \end{bmatrix} \begin{bmatrix} \dfrac{\alpha_n}{\alpha_{n-1}}\boldsymbol{\omega}_{n-1} + \alpha_n \dfrac{t_n - 1}{g_n}\mathbf{v}_n \\ -\alpha_n \dfrac{t_n - 1}{g_n} \end{bmatrix} \\ &= \frac{\alpha_n}{\alpha_{n-1}}\mathbf{X}_{n-1}^{\mathrm{T}}\boldsymbol{\omega}_{n-1} + \alpha_n \frac{t_n - 1}{g_n}\mathbf{X}_{n-1}^{\mathrm{T}}\mathbf{v}_n - \alpha_n \frac{t_n - 1}{g_n}\mathbf{x}_{j_n} \\ &= \frac{\alpha_n}{\alpha_{n-1}}\mathbf{u}_{n-1} + \alpha_n \frac{t_n - 1}{g_n}(\mathbf{X}_{n-1}^{\mathrm{T}}\mathbf{v}_n - \mathbf{x}_{j_n}). \end{aligned} \tag{3.49}$$

Foi provado anteriormente, em (3.43), que não há recursividade em $\mathbf{v}_n$. Portanto, o termo $\mathbf{X}_{n-1}^{\mathrm{T}}\mathbf{v}_n$ não pode ser reduzido.

A complexidade computacional de (3.29) é de KN multiplicações, enquanto que a complexidade computacional de (3.49) é de $KN + K$ multiplicações. Logo, apesar de existir $\mathbf{u}_n$ recursivo, é mais econômica a expressão dada por (3.29).

(3.30) $\mathbf{a}_n$ recursivo

O vetor $\mathbf{a}_n$ é utilizado para identificar o próximo coeficiente que entrará no conjunto ativo, logo não precisa ser calculado com todos os J coeficientes. Apenas os $J - n$ coeficientes pertencentes ao conjunto inativo $\bar{\boldsymbol{\mathcal{A}}}$ são necessários.

Definindo a matriz de dados dos coeficientes inativos por

$$\bar{\mathbf{X}}_n = [\cdots \mathbf{x}_j \cdots]^{\mathrm{T}}, j \in \bar{\boldsymbol{\mathcal{A}}}, \tag{3.50}$$

(3.30) é escrita por

$$\mathbf{a}_n = \bar{\mathbf{X}}_n \mathbf{u}_n. \tag{3.51}$$

Como foi concluído que é ineficiente calcular $\mathbf{u}_n$ de modo recursivo, tampouco $\mathbf{a}_n$ é reduzido mais do que em (3.51).

3.2.4 Pseudocódigo e complexidade computacional

O critério de parada geométrico proposto é adicionado ao algoritmo LAR, gerando o algoritmo GLAR. O pseudocódigo do algoritmo GLAR é apresentado no Algoritmo 3, no final deste capítulo. A quantidade de multiplicações ($\times$), divisões ($\div$), somas ($+$), subtrações ($-$) e potências ($\wedge$) de cada linha do algoritmo estão em vermelho, no lado direito do Algoritmo 3. Esta apresentação facilita o cálculo de complexidade computacional resultante do algoritmo.

As equações recursivas pertinentes, i.e., que diminuem a complexidade computacional, foram também inseridas neste novo algoritmo. Estas equações são referentes ao cômputo de $\mathbf{D}_n$, $\mathbf{X}_n$, $\mathbf{S}_n$, α_n e $\bar{\mathbf{X}}_n$.

Além de multiplicação, divisão, soma, subtração e potência, as seguintes considerações foram feitas para o cálculo da complexidade computacional:

- calcular o arco-cosseno de um valor equivale a quatro potências, uma soma e uma subtração ($\arccos(\theta) = -i \ln\left[\theta + \sqrt{\theta^2 - 1}\right]$ [44]);
- calcular a norma de um vetor de tamanho $(K \times 1)$ equivale a $K + 1$ potências e $K - 1$ somas ($\|\mathbf{x}\| = \sqrt{\sum x(k)^2}$ [44]);
- encontrar o valor máximo ou mínimo de um vetor de tamanho $J - N$ e a sua posição foi considerado $J - N$ somas;
- excluir ou inserir um elemento em um conjunto foi considerado sem complexidade;
- multiplicação por ± 1 foi considerado uma soma.

A Tabela 3.1 apresenta a complexidade computacional do algoritmo GLAR e do algoritmo LAR, caso fosse conhecido o valor de N. A ordem do algoritmo GLAR é $N^3/3 + 3KN$ menor do que a ordem do algoritmo LAR, devido à recursividade aqui apresentada.

3.3 Avaliação do algoritmo GLAR

Para avaliar o critério de parada geométrico, primeiro é analisada sua atuação nos dados de Diabetes. Estes dados foram usados para validar o algoritmo LAR [5]. Mesmo não sendo um sistema esparso, a quantidade de coeficientes dada pelos critérios de

[1] caso a inversa seja obtida pela solução de um sistema do tipo $\mathbf{Ax} = \mathbf{b}$

Tabela 3.1: Complexidade computacional do algoritmo GLAR e do algoritmo LAR

	Algoritmo GLAR	Algoritmo LAR
Inversa ($[\,]^{-1}$)	0	N^3 (ou $N^3/3$)[1]
Potência ($\wedge$)	$2K+8$	$2K+6$
Multiplicação ($\times$)	$3KJ+KN+K$ $+2N^2+3N+3$	$3KJ+4KN+K$ $+N^2+3N+3$
Divisão ($\div$)	$2J-N+4$	$2J-2N+3$
Soma ($+$)	$3KJ+KN+2K$ $-J+2N^2+3N$	$3KJ+4KN+2K$ $-J+2N^2-N+1$
Subtração ($-$)	$K+4J-4N+6$	$K+4J-4N+5$
Ordem	$3KJ+KN$	$N^3+3KJ+4KN$

seleção AIC, BIC e C_p e pelo critério de parada geométrico são comparados na Seção 3.3.1.

Nas seções seguintes, o algoritmo GLAR é avaliado tanto em sua capacidade de indicar corretamente a quantidade de coeficientes não-nulos, quanto em sua capacidade de estimação do vetor de coeficientes na identificação de um sistema não-linear. Para identificar os sistemas não-lineares, é utilizado o algoritmo GLAR em conjunto com um filtro de Volterra, conforme apresentado na Figura 3.6.

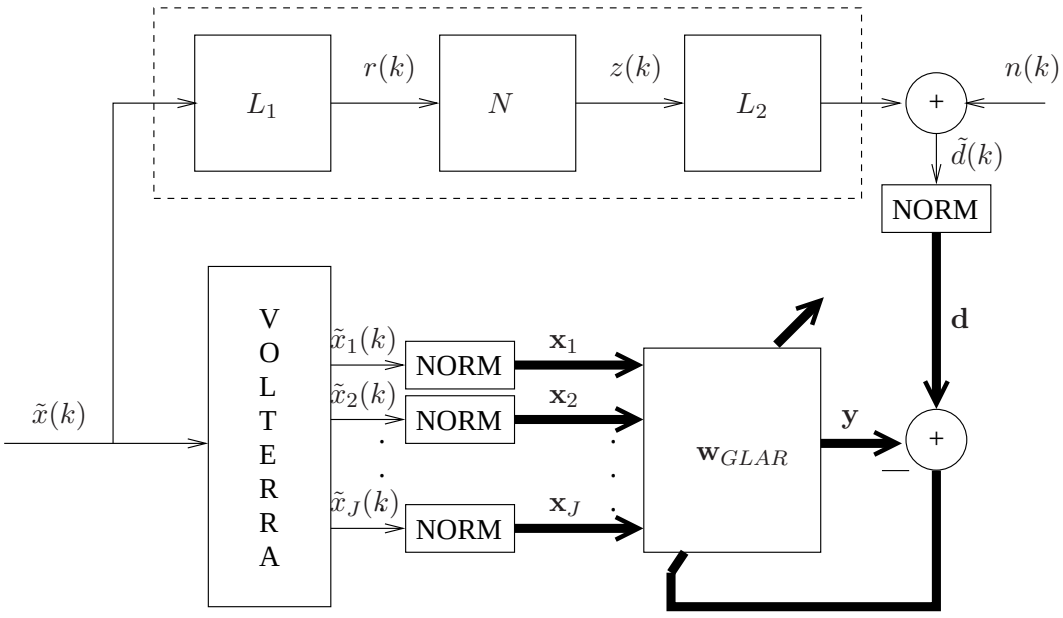


Figura 3.6: Identificação de sistemas não-lineares com filtro de Volterra e o algoritmo GLAR. O bloco "NORM", ilustrado na Figura 3.7, é devido à normalização necessária para o algoritmo GLAR que, por se tratar de um algoritmo de processamento em lote, precisa ter K amostras coletadas.

O sistema não-linear é representado por um modelo em cascata LNL, composto

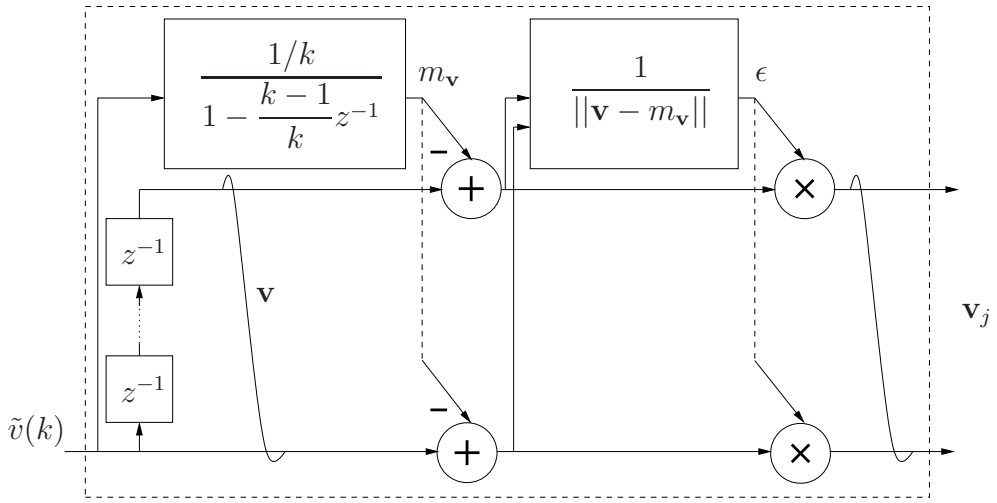


Figura 3.7: Bloco de normalização necessário para o algoritmo GLAR. O sinal de entrada, $\tilde{v}(k)$, representa $\tilde{x}_j(k), 1 \leq j \leq J$, ou $\tilde{d}(k)$. O vetor de saída, $\mathbf{v}_j$ $(K \times 1)$, representa $\mathbf{x}_j, 1 \leq j \leq J$, ou $\mathbf{d}$.

por dois filtros lineares com memória (L_1 e L_2) e um dispositivo não-linear sem memória (N). Na Seção 3.3.2, a não-linearidade é dada por equações polinomiais de terceira ordem. Na Seção 3.3.3, a não-linearidade é dada por equações polinomiais de quinta ordem. Na Seção 3.3.4, a não-linearidade é dada pela função tangente hiperbólica.

Nos sistemas cuja não-linearidade é dada por equações polinomiais, a quantidade de coeficientes não-nulos é conhecida pelo modelo LNL. Para mostrar que o critério de parada geométrico consegue, nestes cenários, indicar a quantidade de coeficientes não-nulos do sistema em análise, os resultados são comparados aos critérios de seleção AIC (2.115), BIC (2.116) e C_p (2.117).

Nos diversos cenários, o resultado da quantidade de coeficientes determinada pelo critério de parada geométrico para diferentes valores de μ é avaliado. Para evitar rodar o algoritmo GLAR diversas vezes, cada vez com um valor diferente de μ, foram realizadas todas as iterações do algoritmo GLAR, até $n = J$, e, a cada iteração, o valor de μ foi calculado a partir da definição do critério de parada geométrico (3.16), levando a

$$\mu = \frac{\Delta\boldsymbol{\theta}_n}{\sigma_{\boldsymbol{\theta}_1}}. \tag{3.52}$$

Para avaliar o algoritmo GLAR quanto à estimação do vetor de coeficientes, o algoritmo GLAR em conjunto com o filtro de Volterra é utilizado para identificação dos sistemas não-lineares. O resultado do algoritmo GLAR é comparado com os resultados dos algoritmos LS (2.24), SSS (2.25) e CLS (2.34). Para os algoritmos SSS e CLS, o algoritmo GLAR é o responsável por determinar a quantidade e a posição dos coeficientes nulos.

3.3.1 Dados de Diabetes

Nesta seção, o critério de parada geométrico é aplicado nos dados de diabetes usados no primeiro artigo de Efron et al. [5]. O conjunto de dados da diabetes foi amplamente utilizado para validação do algoritmo LAR, encontrado em [8].

Em [5], Efron et al. usou o critério de seleção C_p para identificar quantos coeficientes, do total de dez, eram suficientes como marcadores de diabetes. O critério C_p indicou 7 (sete) como a quantidade de marcadores (coeficientes) necessária, resultado aferido por médicos experientes.

O algoritmo GLAR é aplicado nos dados de Diabetes e a resposta de μ avaliada a cada iteração do algoritmo. A quantidade de coeficientes calculada é diretamente proporcional ao μ, como apreciado na Figura 3.8.

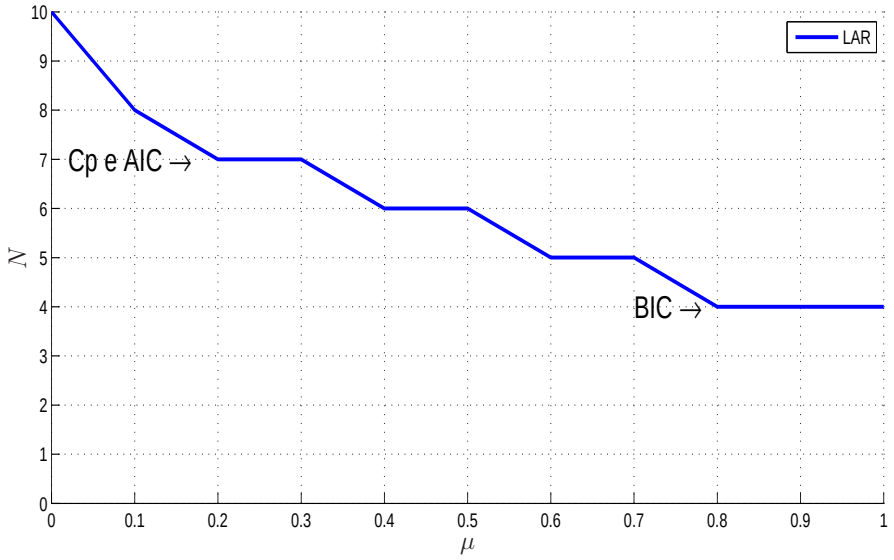


Figura 3.8: A quantidade de coeficientes dada pelos critérios de seleção AIC e C_p é de sete coeficientes, enquanto que para o critério de seleção BIC é de apenas quatro coeficientes.

Para Efron et al. [5] o resultado de sete coeficientes estimados é satisfatório. Este resultado é obtido pelo critério de parada geométrico com $0{,}2 \leq \mu \leq 0{,}3$. O critério de parada geométrico tem a flexibilidade de prover um resultado com mais coeficientes, quando $\mu \to 0$, ou com menos coeficientes, quando $\mu \to 1$. Mesmo neste cenário, que não é esparso, o critério de parada geométrico pode resultar na quantidade de coeficientes esperada para um técnico no assunto.

3.3.2 Identificação de sistemas não-lineares do tipo polinômio de 3^a ordem

Neste cenário, o modelo LNL, conforme apresentado na Figura 3.6, é construído por

$$\begin{aligned} L_1: \quad r(k) &= \mathbf{w}_1^{\mathrm{T}}\tilde{\mathbf{x}}(k) \\ N: \quad z(k) &= ar(k) - br^3(k) \\ L_2: \quad d(k) &= \mathbf{w}_2^{\mathrm{T}}\mathbf{z}(k) + n(k) \end{aligned}$$

onde

$$\begin{aligned} \mathbf{w}_1 &= [0{,}5 \quad 1 \quad 0{,}5]^{\mathrm{T}} \\ \tilde{\mathbf{x}}(k) &= [\tilde{x}(k) \quad \tilde{x}(k-1) \quad \tilde{x}(k-2)]^{\mathrm{T}} \\ \mathbf{w}_2 &= [0{,}1 \quad -0{,}5 \quad 0{,}1]^{\mathrm{T}} \\ \mathbf{z}(k) &= [z(k) \quad z(k-1) \quad z(k-2)]^{\mathrm{T}} \end{aligned}$$

em que $\mathbf{w}_1$ e $\mathbf{w}_2$ são os vetores do primeiro e segundo filtro, respectivamente, $\tilde{\mathbf{x}}(k)$, $\mathbf{z}(k)$ e $r(k)$ são sinais de entrada como apresentado na Figura 3.6. Dois sistemas não-lineares são simulados. O primeiro, chamado de NL1, possui $a = 0{,}1$ e $b = 0{,}01$; o segundo, chamado de NL2, possui $a = b = 1$. A diferença entre ambos é apenas na magnitude final dos coeficientes, enquanto que as posições dos coeficientes no *kernel* são mantidas. Para o sinal de entrada $\tilde{x}(k)$ é simulado um ruído branco com média nula e variância unitária. Para o ruído de observação $n(k)$ é simulado um ruído branco com média nula e variância de 10^{-6} e, em outra simulação, com média nula e variância de 10^{-2}.

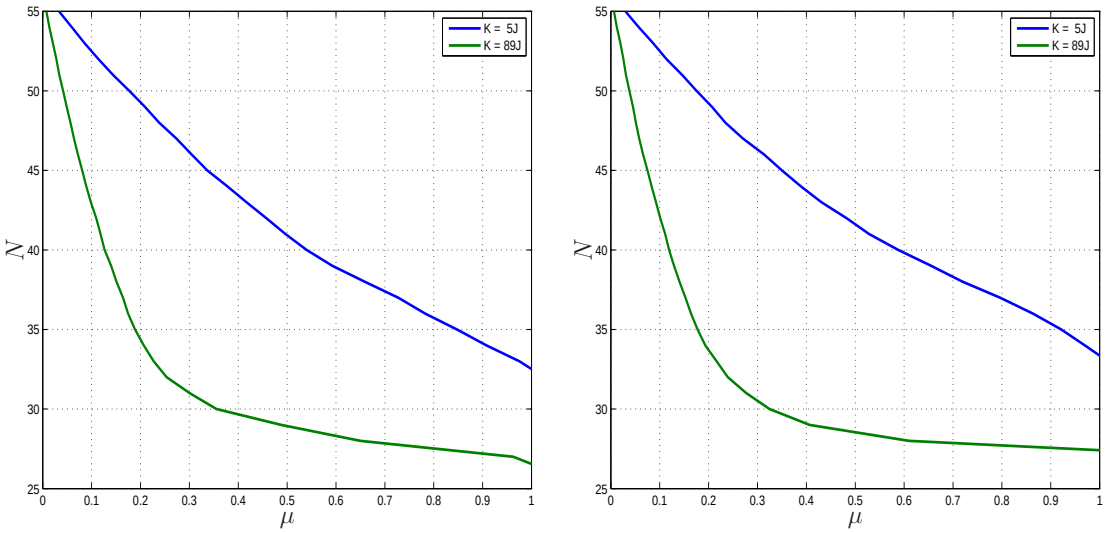
O filtro de Volterra correspondente possui 55 coeficientes ($J = 55$), resultado de (2.10) com $l = 3$ e $m = 4$ menos a componente DC. Pelas equações do sistema LNL é possível concluir que apenas 27 destes coeficientes são não-nulos, portanto 28 coeficientes são nulos.

Em cada simulação, para gerar os diversos resultados, são tiradas as médias de 100 experimentos completos do algoritmo GLAR para trinta valores de K, $K = 2J{,}5J, 8J, \cdots, 89J$. Uma amostra extra é utilizada em cada intervalo para calcular o erro *a priori* pertinente.

NL1 e NL2 com ruído de $-60dB$

Para primeiro avaliar o critério de parada geométrico, o resultado da quantidade de coeficientes determinada pelo critério de parada geométrico para vários valores de μ com $K = 5J$ e $K = 89J$ é apresentado na Figura 3.9. Quanto mais dados estão disponíveis (maior valor de K), melhor é a estimação dos coeficientes e a

determinação da quantidade de coeficientes não-nulos. Por estas razões, quanto maior o valor de K menor o valor de μ para resultar na mesma quantidade de coeficientes. Por exemplo, no sistema NL1, $\mu = 0{,}52$ para $K = 5J$ e $\mu = 0{,}12$ para $K = 89J$ resultam em 40 coeficientes estimados. A amplitude dos coeficientes não altera significativamente as curvas apresentas na Figura 3.9.



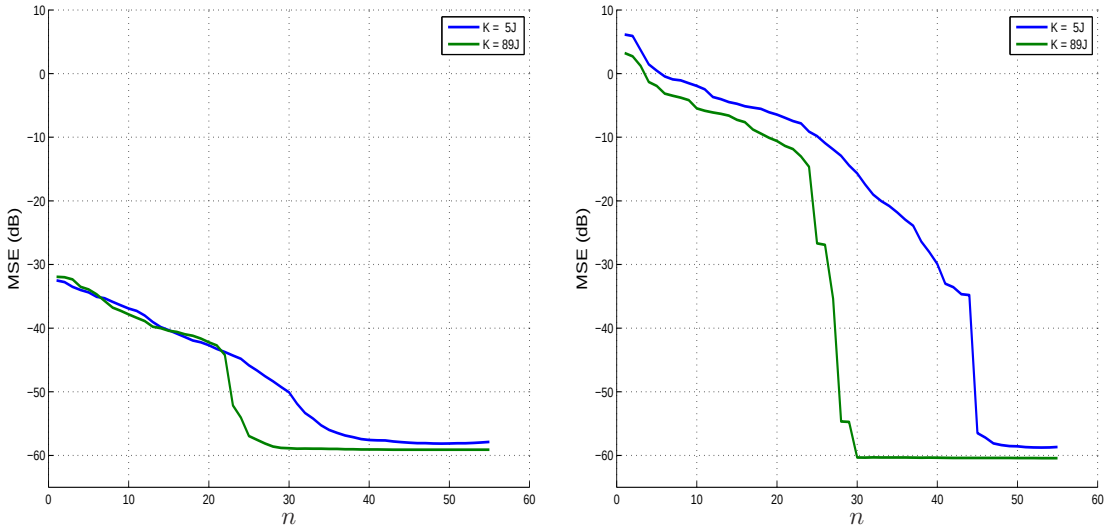
(a) NL1 (b) NL2

Figura 3.9: O valor de N varia com a quantidade de dados disponíveis. A escolha de μ define quantos coeficientes são estimados pelo algoritmo GLAR.

Lembrando que a quantidade de coeficientes estimada é o mesmo valor da iteração n, com a Figura 3.10, a quantidade de coeficientes estimada necessária para que o erro de estimação seja mínimo é determinada. O maior valor de K resulta no MSE mínimo mais próximo do valor de coeficientes não-nulos conhecido (27). Para $K = 5J$, são necessários aproximadamente 40 e 43 coeficientes para que o MSE seja mínimo em NL1 e NL2, respectivamente.

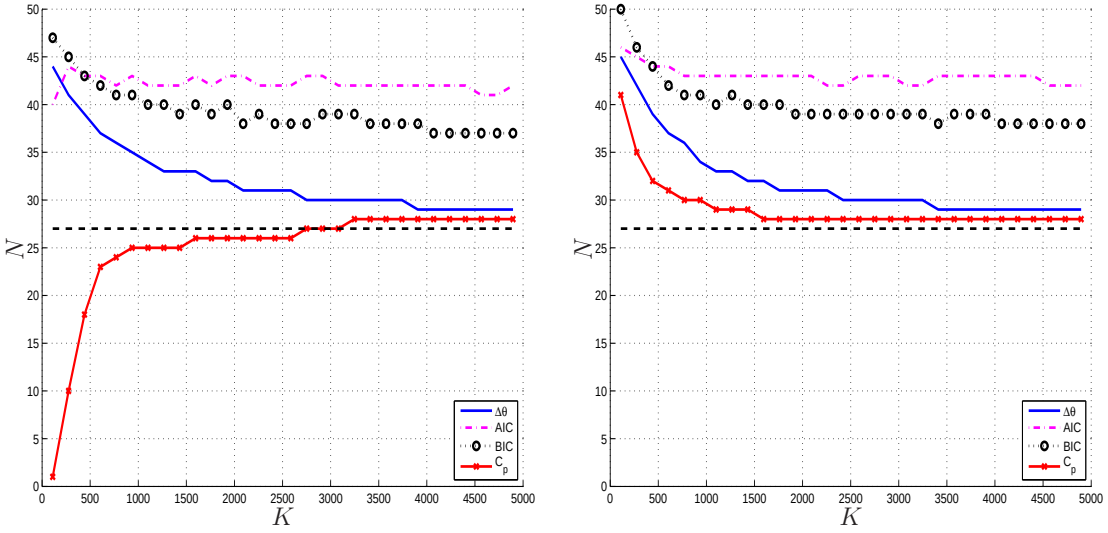
A quantidade de coeficientes estimada dada pelo critério de parada geométrico com $\mu = 0{,}5$, bem como pelos critérios de seleção AIC, BIC e C_p são comparadas na Figura 3.11. Neste momento, deve-se relembrar que o algoritmo LAR precisa ser executado até o final, i.e., até $n = J$, para que então os critérios de seleção AIC, BIC e C_p sejam computados e o resultado ótimo obtido. Esta é uma clara vantagem do critério de parada geométrico: uma vez que o critério é atingido, o algoritmo é interrompido (algoritmo GLAR).

Com o valor de μ constante, a quantidade de coeficientes estimada para maiores valores de K diminui até uma quantidade em que a curva é estabilizada, observado na Figura 3.11. Este resultado pode também ser obtido pela Figura 3.9: manter a quantidade de coeficientes fixa significa diminuir o valor de μ para maiores valores de K; enquanto que para o valor de μ fixo, a quantidade de coeficientes é maior para



(a) NL1 (b) NL2

Figura 3.10: MSE, usando o erro *a priori*, calculado a cada iteração do algoritmo GLAR. Os coeficientes de NL2 tem maior magnitude do que os coeficientes de NL1, gerando uma maior variação de MSE e uma queda mais brusca quando os coeficientes são estimados corretamente.



(a) NL1 (b) NL2

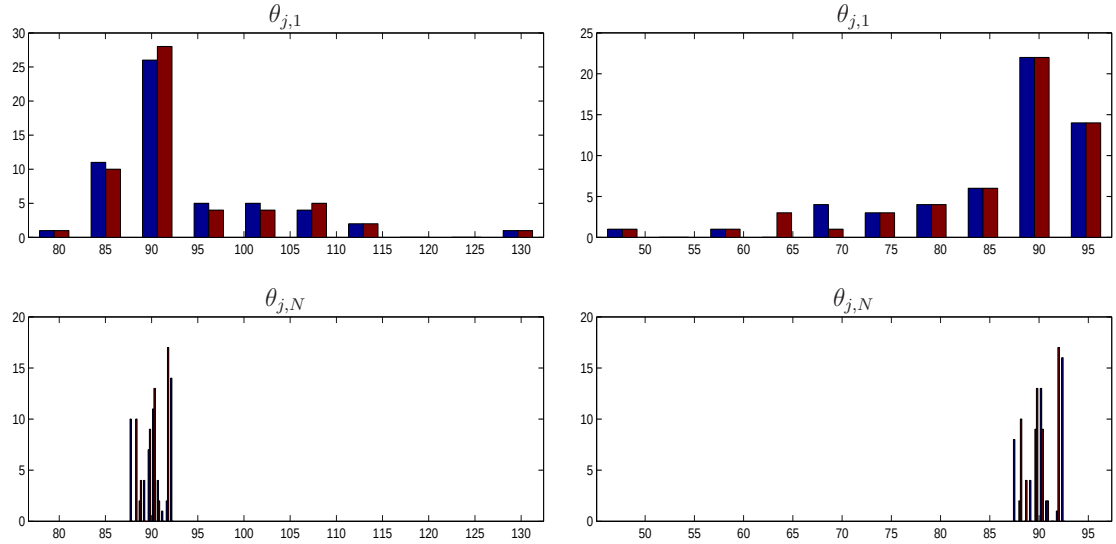
Figura 3.11: Quantidade de coeficientes estimada para cada um dos critérios avaliados. Para o critério de parada geométrico é usado $\mu = 0{,}5$. A reta em pontilhado representa a quantidade total de coeficientes não-nulos conhecida, dada pelo modelo LNL por 27 coeficientes.

K menor.

Os critérios que mais se aproximam do valor conhecido de coeficientes não-nulos são o critério de parada geométrico e o critério de seleção C_p. No entanto, em NL1 o critério de seleção C_p resulta em menos coeficientes do que a quantidade conhecida para $K < 2.700$, aproximadamente, insatisfatório para a identificação

de sistemas. Ao usar o algoritmo LAR, estimar menos coeficientes do que aquele conhecido pelo modelo LNL resulta em um erro de estimação alto. A quantidade de coeficientes conhecida pelo modelo LNL é o limite inferior de coeficientes que devem ser estimados.

O histograma dos ângulos entre cada vetor de dados dos coeficientes $j = 1, \cdots, J$ e o erro residual é apresentado na Figura 3.12, para a primeira iteração ($n = 1$) e para a iteração quando o critério de parada geométrico com $\mu = 0{,}5$ é atingido, em $n = N$ diferente para cada valor de K (Figura 3.11). Isto confirma a análise feita na Seção 3.2.1: todos os coeficientes se aproximam de 90^o, sejam coeficientes ativos ou inativos.



(a) NL1 (b) NL2

Figura 3.12: Histograma na primeira iteração $n = 1$ e $n = N$, quando $\Delta\boldsymbol{\theta}_N \leq 0{,}5\sigma_{\boldsymbol{\theta}_1}$, para $K = 5J$, em azul, e $K = 89J$, em marrom.

Em $n = 1$, em ambos os experimentos os ângulos variam em faixas próximas a 50^o, deslocadas quando comparadas: de 80^o até 130^o para NL1 (histograma superior na Figura 3.12(a)) e de 45^o até 95^o para NL2 (histograma superior na Figura 3.12(b)). Na primeira iteração, os ângulos estão mais espalhados, enquanto que na iteração N apresentam-se mais homogêneos em torno de 90^o. Na iteração $n = N$, o vetor de erro de predição não é ortogonal a cada vetor de dados de coeficiente, mas o erro resultante é tolerado, como será verificado com os resultados de MSE apresentados posteriormente. Na última iteração possível, $n = J$, o histograma é uma única linha em 90^o para ambos os casos, correspondente à solução LS.

O desvio padrão dos ângulos iniciais é apresentado na Tabela 3.2. Tanto em NL1 quanto em NL2, o valor de σ_{θ_1} não varia significativamente com a quantidade de dados K. Comparando σ_{θ_1} para o mesmo valor de K em NL1 e NL2 observa-se

uma maior variação, i.e. a amplitude dos coeficientes influencia no desvio padrão dos ângulos iniciais.

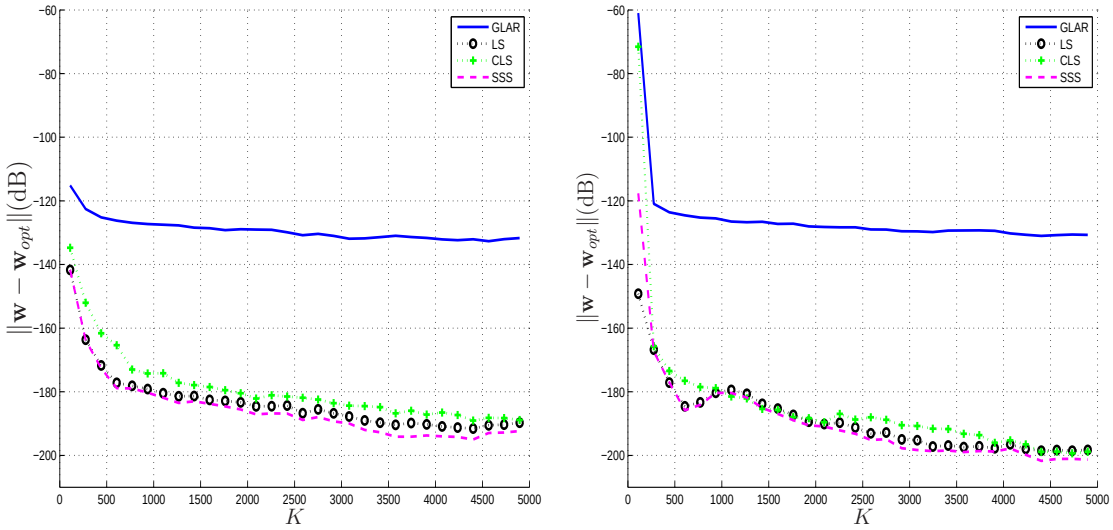
Tabela 3.2: Desvio padrão dos ângulos iniciais (σ_{θ_1})

	$K = 5J$	$K = 89J$
NL1	9,4	9,7
NL2	10,5	10,6

Uma vez comprovada a eficiência do critério de parada geométrico, o conjunto do algoritmo GLAR em combinação com filtro de Volterra é avaliado na identificação dos sistemas não-lineares. O resultado do algoritmo GLAR com $\mu = 0{,}5$ é comparado com os algoritmos LS, SSS e CLS. O algoritmo SSS utiliza a resposta do algoritmo GLAR para determinar a quantidade de coeficientes nulos. O algoritmo CLS utiliza a resposta do algoritmo GLAR para a determinar as posições dos coeficientes nulos no *kernel*, necessário para a criação da matriz de restrição.

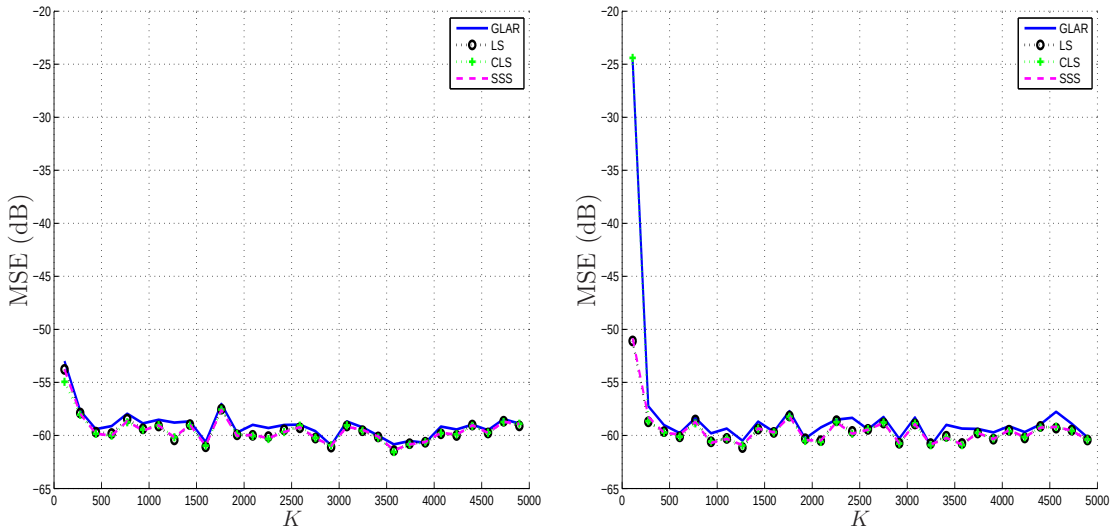
A norma da diferença entre o vetor de coeficientes estimado por cada algoritmo e o vetor de coeficientes ideal, conhecido pelo sistema LNL, é apresentada na Figura 3.13. Para o sistema NL2, com $K = 2J$, o algoritmo CLS apresentou uma resposta mais degradada quando comparada ao algoritmo LS. Isto mostra que esta quantidade de dados é insuficiente para indicar a posição dos coeficientes nulos corretamente. Para os outros valores de K, em ambos os sistemas, o resultado do algoritmo CLS teve sua curva ligeiramente acima da curva do algoritmo LS, sugerindo que a resposta do algoritmo GLAR é incorreta em relação à posição dos coeficientes nulos com todas as quantidades de dados avaliadas. Por outro lado, o resultado do algoritmo SSS, com sua curva ligeiramente abaixo da curva do algoritmo LS, sugere que a quantidade de coeficientes nulos é correta. A diferença dos resultados dos algoritmo LS, CLS e SSS é muito pequena, todos os resultados estão abaixo de $-160dB$, e a resposta é inconclusiva baseada apenas na Figura 3.13.

O resultado degradado do algoritmo GLAR, apresentado na Figura 3.13, indica que a sua estimação não é tão precisa quanto a do algoritmo LS. Entretanto, seu resultado é abaixo de $-120dB$ para $K \geq 5J$; resultados da norma da diferença entre o vetor de coeficientes estimado e o vetor de coeficientes ideal abaixo de $-120dB$ tornam imperceptíveis as diferenças das curvas de resposta de MSE dos algoritmos avaliados, apresentado na Figura 3.14. Com exceção do algoritmo GLAR, todos os outros algoritmos precisam estimar os 55 coeficientes (resultado LS) antes de forçar alguns deles a zero, enquanto o algoritmo GLAR estima apenas o número N de coeficientes.



(a) NL1 (b) NL2

Figura 3.13: Diferença entre o vetor de coeficientes estimado por cada algoritmo avaliado e o vetor de coeficiente ideal conhecido pelo sistema LNL. A precisão dos vetores de coeficientes estimados é alta: para $K \geq 5J$, em todos os casos, a resposta ficou abaixo de $-120dB$. Para o critério de parada geométrico é usado $\mu = 0{,}5$.



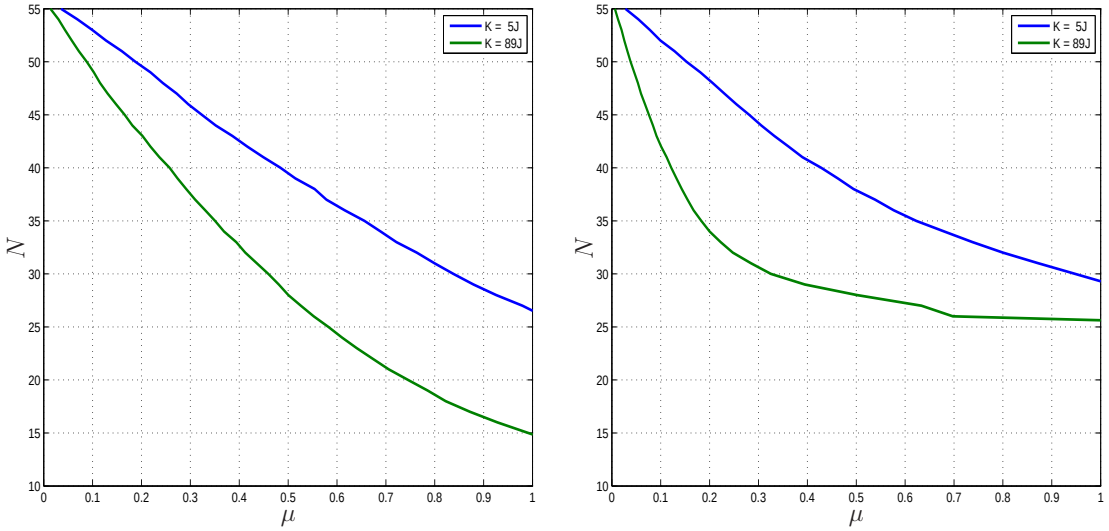
(a) NL1 (b) NL2

Figura 3.14: MSE, calculado com o erro *a priori*, para os diversos algoritmos avaliados. Para obter as respostas dos algoritmos SSS e CLS, quantos (e quais) coeficientes são nulos é definido pelo critério de parada geométrico, diferente para cada valor de K como apresentado na Figura 3.11. Para o critério de parada geométrico é usado $\mu = 0{,}5$.

NL1 e NL2 com ruído de $-20dB$

Neste cenário, o ruído de observação é mais alto, elevando o grau de dificuldade na estimação do vetor de coeficientes. Entretanto, por mais precisa que seja a estimação do vetor de coeficientes, o MSE resultante será em torno de $-20dB$.

O resultado de N para diferentes valores de μ com $K = 5J$ e $K = 89J$ é apresentado na Figura 3.15. Quanto maior o valor de K melhor é a resposta N. Enquanto que para NL2 as curvas sugerem em torno da quantidade correta de coeficientes não-nulos conhecida (27), a curva para $K = 89J$ em NL1 chega a indicar apenas 15 coeficientes.



(a) NL1 (b) NL2

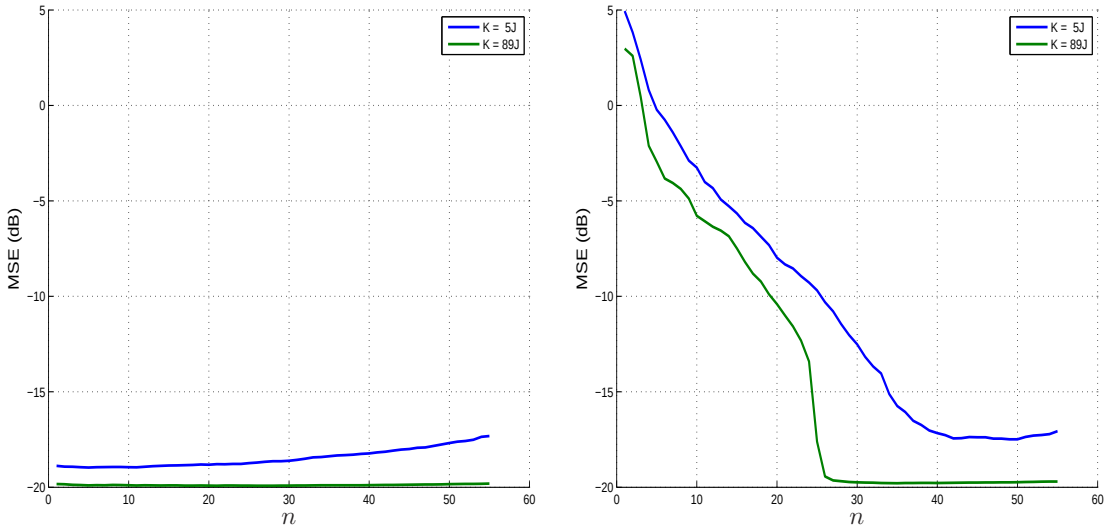
Figura 3.15: Em NL1, o resultado de N chega a 15 coeficientes apenas, enquanto que em NL2, N chega a 26 coeficientes. Isto mostra a influência do ruído no sistema, quanto maior o ruído mais difícil é identificar corretamente a quantidade de coeficientes.

O resultado de MSE, calculado com o erro *a priori*, a cada iteração do algoritmo LAR é apresentado na Figura 3.16. Em NL2, o resultado de MSE para $K = 89J$, Figura 3.16(b), converge em torno de 30 coeficientes. Com os resultados apresentados de MSE, é escolhido o valor de $\mu = 0{,}5$, baseado na Figura 3.15.

O alto valor do ruído não impediu o critério de parada geométrico de indicar N próximo ao ideal, dado pelo modelo LNL, como apresentado na Figura 3.17. O mesmo não aconteceu para os outros critérios de seleção avaliados, que tiveram o resultado muito degradado devido ao ruído de observação do sistema. O critério de parada geométrico mostrou ser o melhor critério para todos os valores de K.

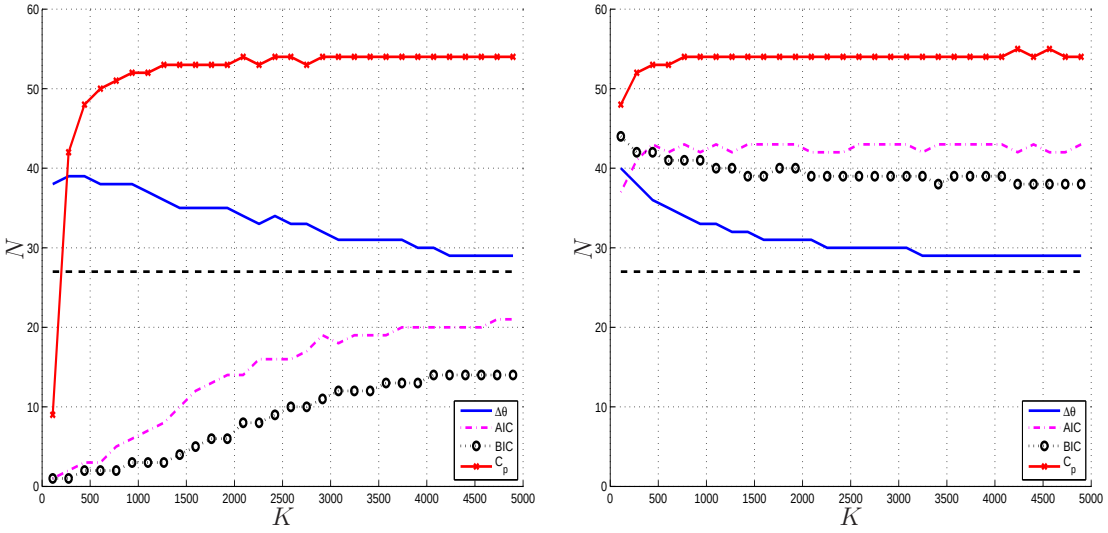
Uma vez que o ruído não atrapalhou na curva de N para o critério de parada geométrico, apresentado na Figura 3.17, já era esperado que tampouco o histograma dos ângulos entre cada vetor de dados dos coeficientes $j = 1, \cdots, J$ e o erro residual seria influenciado pelo ruído, como pode ser observado na Figura 3.18.

O algoritmo GLAR, com $\mu = 0{,}5$, em combinação com filtro de Volterra é utilizado para identificar os sistemas em questão. A norma da diferença entre o vetor de coeficientes estimado por cada algoritmo e o vetor de coeficientes ideal, conhecido pelo sistema LNL, é apresentada na Figura 3.19. Com o ruído de observação



(a) NL1 (b) NL2

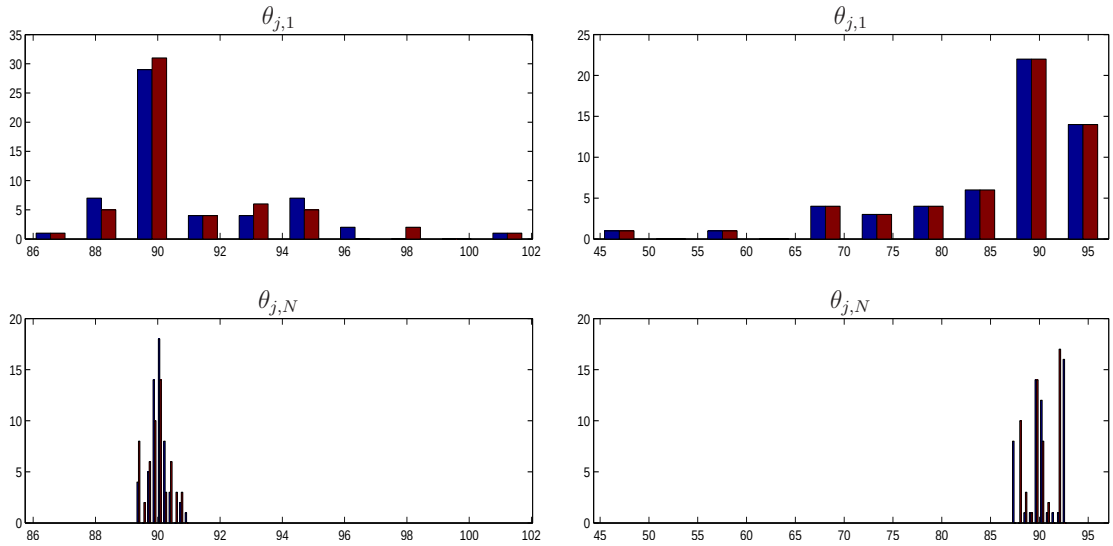
Figura 3.16: MSE, usando o erro *a priori*, calculado a cada iteração do algoritmo GLAR. Como o ruído é alto e os coeficientes em NL1 são de baixa magnitude, o MSE resultante é próximo ao MSE mínimo desde a primeira iteração do algoritmo GLAR.



(a) NL1 (b) NL2

Figura 3.17: Quantidade de coeficientes estimada para cada um dos critérios avaliados. Para o critério de parada geométrico é usado $\mu = 0{,}5$. A reta em pontilhado representa a quantidade total de coeficientes não-nulos conhecida, dada pelo modelo LNL por 27 coeficientes.

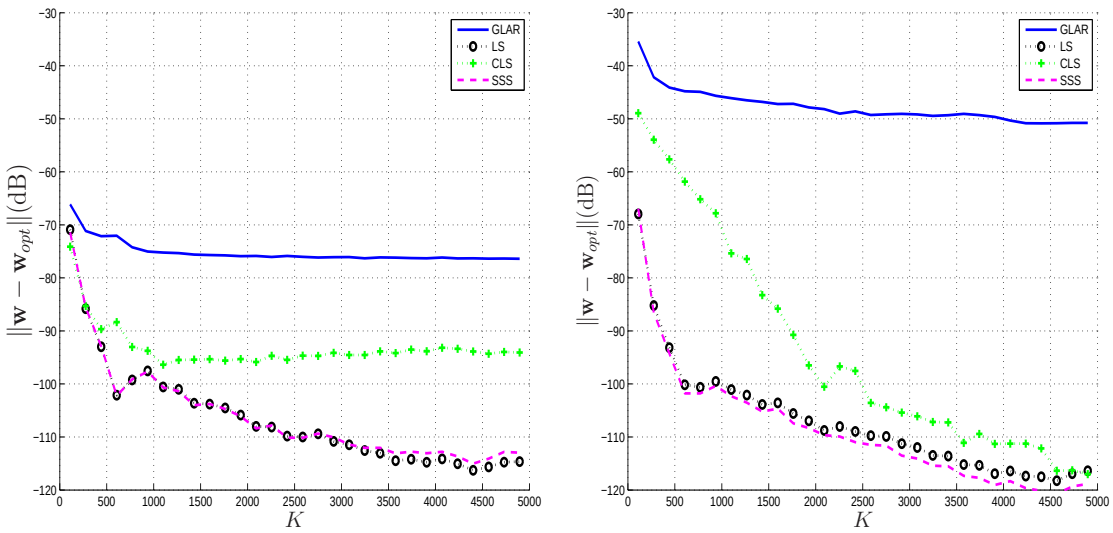
mais alto, o erro da norma da diferença entre o vetor de coeficientes estimado e o vetor de coeficientes ideal neste cenário chegou a $-120dB$, maior do que no cenário anterior quando chegou a $-200dB$. Novamente, o resultado do algoritmo CLS indica que algum (ou alguns) coeficiente está sendo considerado nulo erradamente, degradando o resultado quando comparado ao algoritmo LS. Todavia esta diferença



(a) NL1 (b) NL2

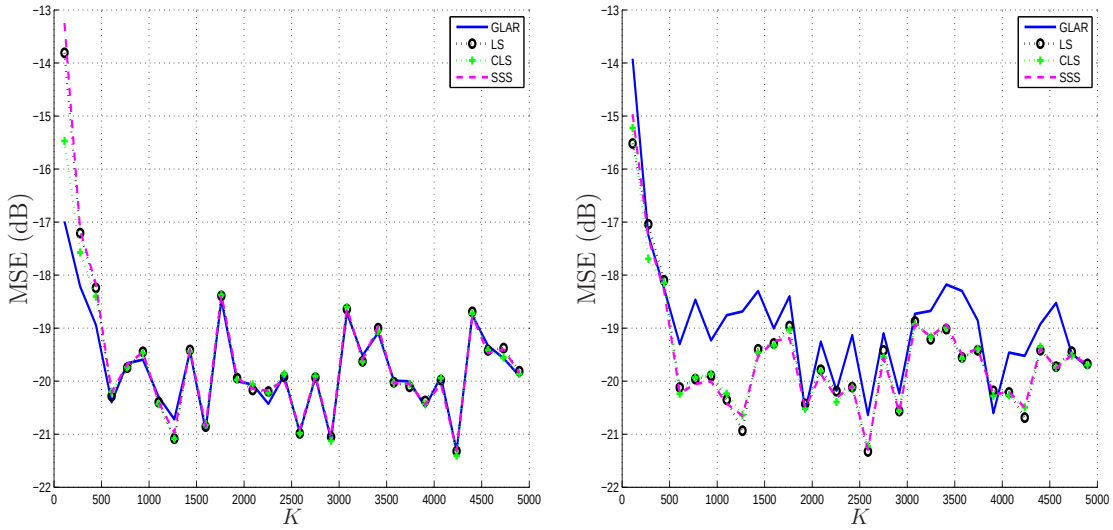
Figura 3.18: Histograma na primeira iteração $n = 1$ e $n = N$, quando $\Delta\boldsymbol{\theta}_N \leq 0{,}5\sigma_{\boldsymbol{\theta}_1}$, para $K = 5J$, em azul, e $K = 89J$, em marrom.

não é percebida na curva de MSE, ilustrada na Figura 3.20.



(a) NL1 (b) NL2

Figura 3.19: Diferença entre o vetor de coeficientes estimado por cada algoritmo avaliado e o vetor de coeficiente ideal conhecido pelo sistema LNL. Para o critério de parada geométrico é usado $\mu = 0{,}5$. Em NL2, apesar da quantidade de coeficientes estimada estar satisfatória, a estimação do algoritmo GLAR está sendo mascarada pelo ruído gerando um pior resultado.



(a) NL1 (b) NL2

Figura 3.20: MSE, calculado com o erro *a priori*, para os diversos algoritmos avaliados. Para o critério de parada geométrico é usado $\mu = 0{,}5$. Em NL2, a estimação degradada do vetor de coeficientes pelo algoritmo GLAR, observado na Figura 3.19, é identificado aqui, na curva de MSE resultante.

3.3.3 Identificação de sistemas não-lineares do tipo polinômio de 5^a ordem

Para este cenário, o modelo LNL, conforme apresentado na Figura 3.6, é construído por

$$\begin{aligned} L_1: \quad r(k) &= \mathbf{w}_1^{\mathrm{T}}\tilde{\mathbf{x}}(k) \\ N: \quad z(k) &= ar(k) - br^3(k) + cr^5(k) \\ L_2: \quad d(k) &= \mathbf{w}_2^{\mathrm{T}}\mathbf{z}(k) + n(k) \end{aligned}$$

onde

$$\begin{aligned} \mathbf{w}_1 &= [0{,}5 \quad 0{,}5 \quad 1 \quad 0{,}5]^{\mathrm{T}} \\ \tilde{\mathbf{x}}(k) &= [\tilde{x}(k) \quad \tilde{x}(k-1) \quad \tilde{x}(k-2) \quad \tilde{x}(k-3)]^{\mathrm{T}} \\ \mathbf{w}_2 &= [0{,}1 \quad -0{,}5 \quad 1 \quad -0{,}5]^{\mathrm{T}} \\ \mathbf{z}(k) &= [z(k) \quad z(k-1) \quad z(k-2) \quad z(k-3)]^{\mathrm{T}} \end{aligned}$$

em que $\mathbf{w}_1$ e $\mathbf{w}_2$ são os vetores do primeiro e segundo filtro, respectivamente, $\tilde{\mathbf{x}}(k)$, $\mathbf{z}(k)$ e $r(k)$ são sinais de entrada como apresentado na Figura 3.6. Também nas simulações desta seção, a nomenclatura NL1 e NL2 é utilizada para referenciar dois sistemas cujas magnitudes dos coeficientes são distintas: NL1 possui $a = 0{,}1$ e $b = c = 0{,}01$, e NL2 possui $a = b = c = 1$. Para o sinal de entrada $\tilde{x}(k)$ é simulado

um ruído branco com média nula e variância unitária. Para o ruído de observação é simulado um ruído branco com média zero e variância de 10^{-6}.

O filtro de Volterra correspondente possui 791 coeficientes ($J = 791$), resultado de (2.10) com $l = 5$ e $m = 6$ menos a componente DC. Pelas equações do sistema LNL é possível concluir que 211 destes coeficientes são não-nulos e, portanto, 580 coeficientes são nulos.

Em cada simulação, para gerar os diversos resultados, foram tiradas as médias de 50 experimentos completos do algoritmo GLAR para onze valores de K, $K = 2J$,$3J, 4J, \cdots, 12J$. Uma amostra extra é utilizada em cada intervalo para calcular o erro *a priori* pertinente.

Uma vez que estimar todos os coeficientes consome muito tempo e que o número de coeficientes não-nulos é de apenas 211, para calcular os resultados deste cenário o algoritmo GLAR é iterado até $n = 500$.

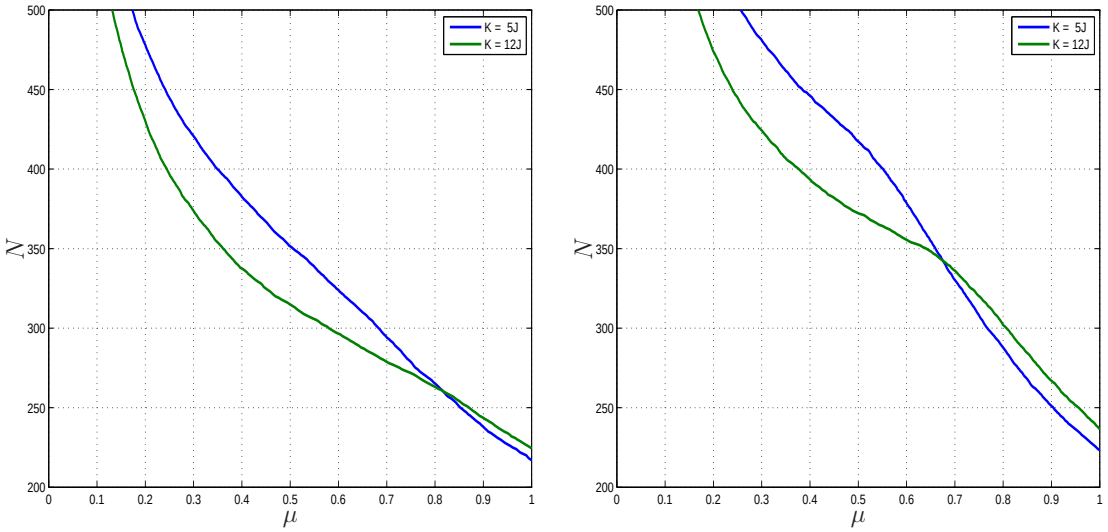
NL1 e NL2 com ruído de $-60dB$

Os resultados de N para os diversos valores de μ, usando $K = 5J$ e $K = 12J$, são apresentados na Figura 3.21. Para valores baixos de μ é impossível saber a respectiva quantidade de coeficientes (N), pois o algoritmo GLAR é interrompido em $n = 500$. Para ambos os valores de K avaliados, a curva ainda está em estado decrescente em $\mu = 1$. A diferença nas curvas de $K = 5J$ e $K = 12J$ não é tão claramente observada quanto no cenário anterior, pois os valores de K estão mais próximos (a diferença entre $K = 5J$ e $K = 12J$ é pouco mais que duas vezes, enquanto que no cenário anterior a diferença entre $K = 5J$ e $K = 89J$ é em torno de dezoito vezes). Também por este motivo as curvas $K = 5J$ e $K = 12J$ se cruzam. O valor $K = 89J = 351.995$ é muito alto, inviabilizando a geração de dados com esta quantidade de amostras.

Apesar das curvas na Figura 3.21 não deixarem clara qual a quantidade de coeficientes necessária para atingir o erro de estimação mínimo imposto pelo ruído de observação, a Figura 3.22 indica que após 400 coeficientes, aproximadamente, o resultado desejado é atingido.

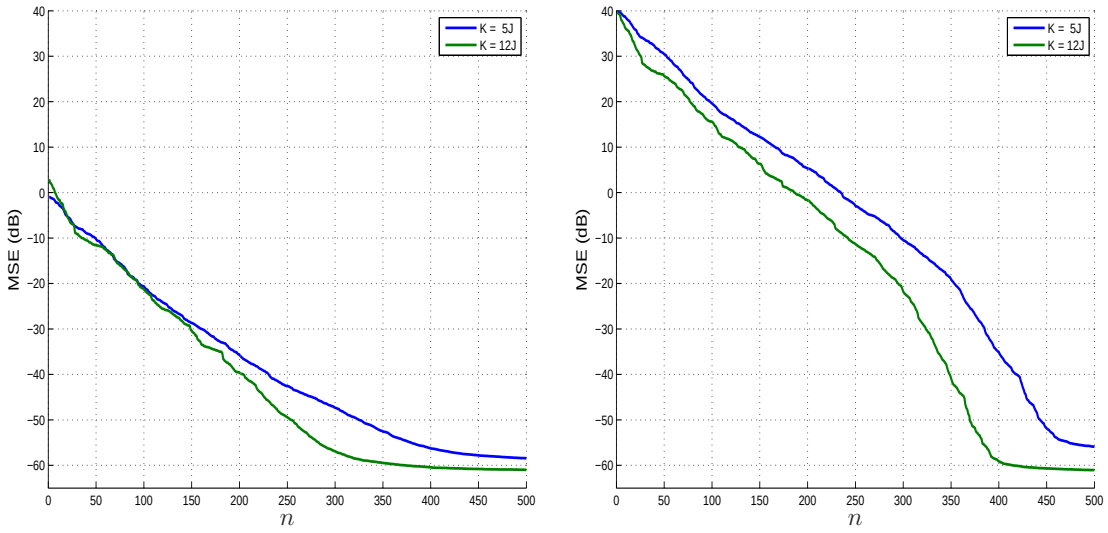
Com os resultados apresentados na Figura 3.21 e na Figura 3.22, o valor de MSE mínimo é atingido com $\mu = 0{,}3$. O algoritmo GLAR com $\mu = 0{,}3$, em conjunto com o filtro de Volterra, é, então, utilizado na identificação destes sistemas não-lineares. O primeiro resultado apresentado é a quantidade de coeficientes estimada por cada critério avaliado, apresentado na Figura 3.23.

Todos os critérios resultaram em uma quantidade de coeficientes acima daquela determinada pelo modelo LNL. Isto mostra que quanto maior a ordem do sistema, mais difícil é encontrar a quantidade de coeficientes correta do sistema em análise. Devido à interrupção do algoritmo em $n = 500$, os resultados dos critérios de seleção



(a) NL1 (b) NL2

Figura 3.21: Para as quantidades de dados avaliadas, as curvas se mantiveram decrescentes; porém, mesmo para $\mu = 1$, o valor de N é acima daquele conhecido pelo modelo LNL.

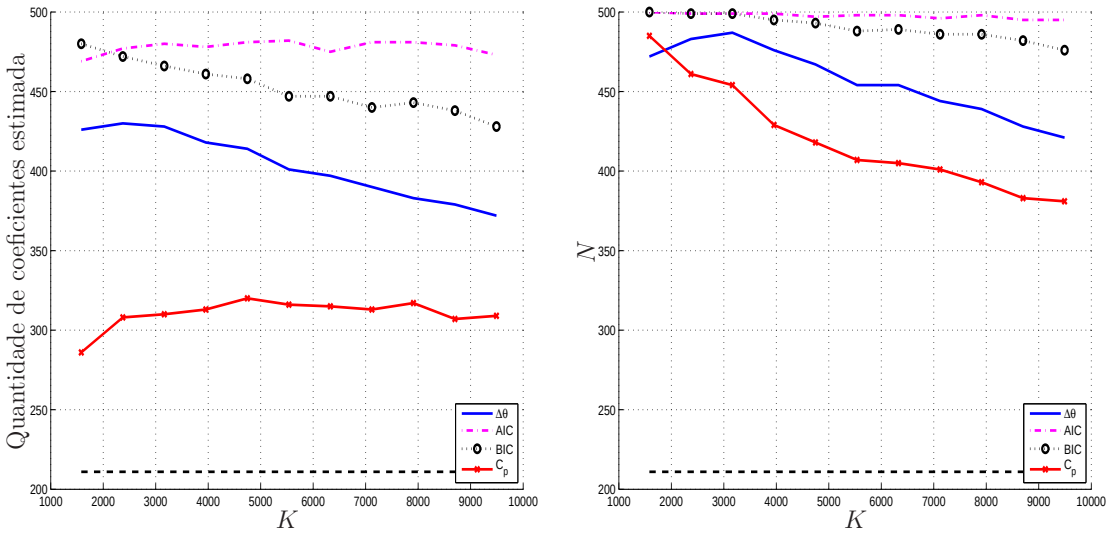


(a) NL1 (b) NL2

Figura 3.22: MSE, usando o erro *a priori*, calculado a cada iteração do algoritmo GLAR. NL1, com coeficientes de menor magnitude, possui menor variação de MSE.

AIC, BIC e C_p poderiam ter resultados diferentes destes apresentados caso o algoritmo GLAR fosse interrompido apenas em $n = J = 791$.

Para $\mu > 0{,}3$, o critério de parada geométrico resultaria em uma quantidade de coeficientes mais próxima ao valor ideal conhecido pelo modelo LNL, porém comprometeria o resultado de MSE. Em NL2, a quantidade de coeficientes indicada pelos critérios de seleção AIC e BIC certamente não é a quantidade correta final, uma vez que resultaram $N = 500$ para diversos valores de K. Ou seja, as curvas da

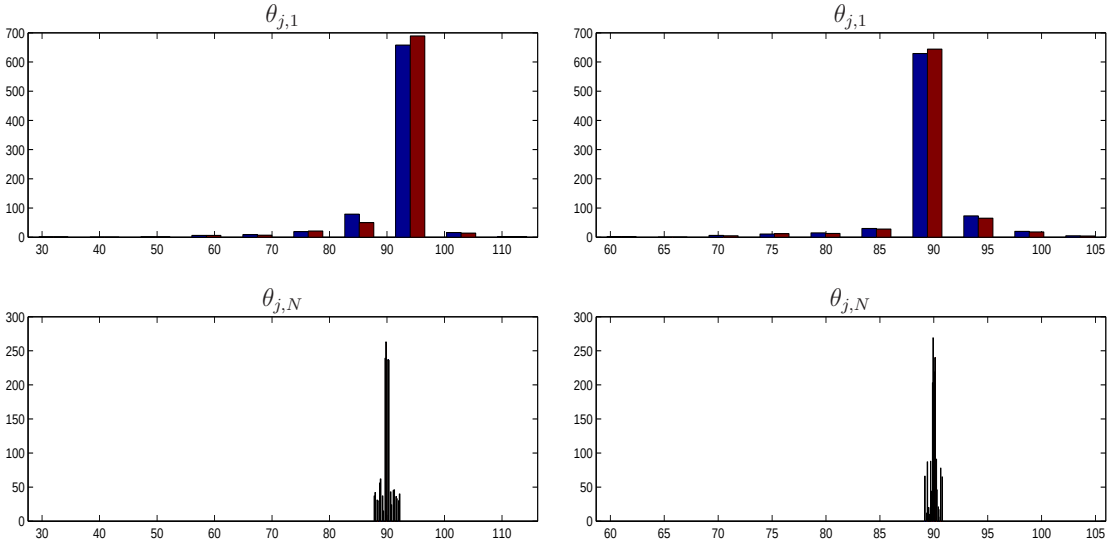


(a) NL1 (b) NL2

Figura 3.23: Quantidade de coeficientes estimada para cada um dos critérios avaliados. Para o algoritmo GLAR é usado $\mu = 0{,}3$. A reta em pontilhado representa a quantidade total de coeficientes não-nulos conhecida, dada pelo modelo LNL por 211 coeficientes.

funções AIC e BIC ainda estão na descendente, e o ponto de mínimo de cada função não é atingido.

O histograma dos ângulos é apresentado na Figura 3.24.

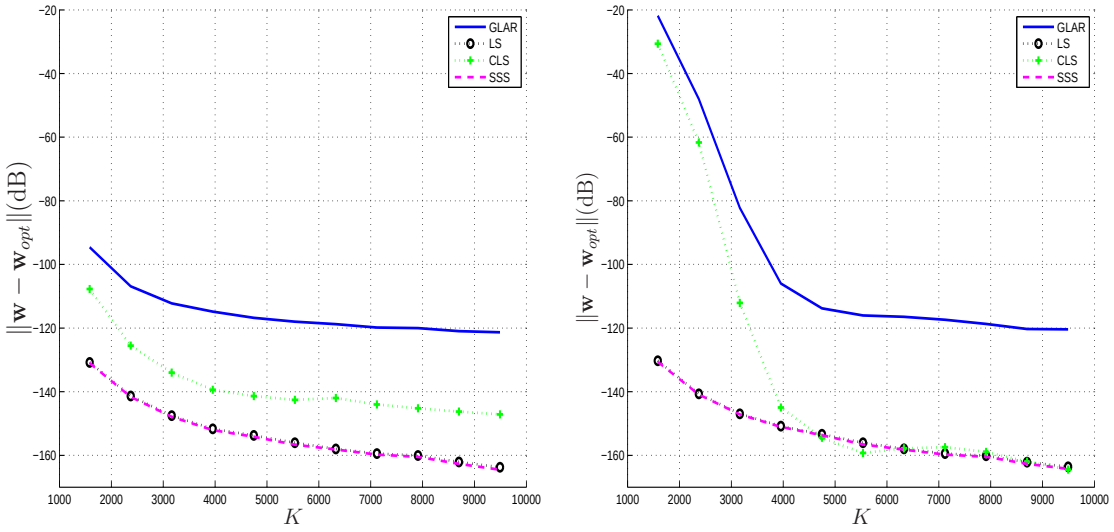


(a) NL1 (b) NL2

Figura 3.24: Histograma na primeira iteração $n = 1$ e $n = N$, quando $\Delta\boldsymbol{\theta}_N \leq 0{,}3\sigma_{\boldsymbol{\theta}_1}$, para $K = 5J$, em azul, e $K = 89J$, em marrom.

Pela curva da norma da diferença entre o vetor de coeficientes estimado pelo algoritmo GLAR e o vetor de coeficientes ideal, apresentada na Figura 3.25, a quantidade mínima para um resultado estável do algoritmo GLAR é de, aproximada-

mente, $k \geq 5J = 3.955$. A partir deste valor de K, todos os algoritmos possuem erro de estimação do vetor de coeficientes inferior a $-100dB$ nos sistemas NL1 e NL2, indicando uma boa precisão do vetor de coeficientes estimado.



(a) NL1 (b) NL2

Figura 3.25: Diferença entre o vetor de coeficientes estimado por cada algoritmo avaliado e o vetor de coeficiente ideal conhecido pelo sistema LNL. Para o algoritmo GLAR é usado $\mu = 0{,}3$.

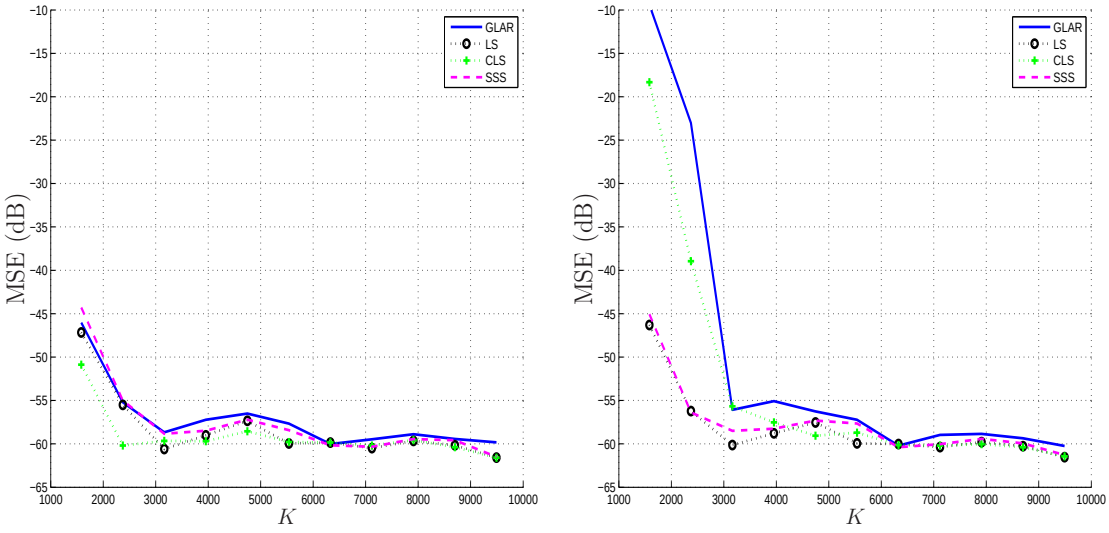
Em NL2, Figura 3.25(b), para $k \leq 3J$, a quantidade de dados não é suficiente para a correta estimação do vetor de coeficientes pelo algoritmo GLAR. Em NL1, Figura 3.25(a), o resultado do algoritmo CLS indica que o algoritmo GLAR não consegue indicar corretamente quais são todos os coeficientes nulos do sistema. No entanto, a julgar pelo resultado satisfatório da estimação do vetor de coeficientes, tanto pelo algoritmo GLAR quanto pelo algoritmo CLS, apenas um ou dois coeficientes, certamente de baixa magnitude, indicados como nulo são indevidos.

Todos os algoritmos possuem um *tempo de convergência* similar para a curva de MSE, Figura 3.26, em torno de $k \geq 4J = 3.164$; a partir deste ponto, a diferença entre os algoritmos avaliados é desprezível. O valor mínimo imposto pelo ruído do sistema é atingido por todos os algoritmos.

3.3.4 Identificação de sistemas não-lineares do tipo tangente hiperbólica

No caso da função tangente hiperbólica, não há quantidade de coeficientes ideal. A função tangente hiperbólica pode ser aproximada pela Série de Taylor por [45]

$$\text{tangh}(r(k)) = r(k) - \frac{r^3(k)}{3} + \frac{2r^5(k)}{15} - \frac{17r^7(k)}{315} + \dots . \tag{3.53}$$



(a) NL1 (b) NL2

Figura 3.26: MSE, calculado com o erro *a priori*, para os diversos algoritmos avaliados. Todos os algoritmo necessitam de $k \geq 4J$ para atingir a resposta mínima imposta pelo ruído de observação. Para o algoritmo GLAR é usado $\mu = 0{,}3$.

Ao usar um filtro de Volterra, a quantidade total de coeficientes é determinada pelo tamanho do filtro de Volterra. Dois sistemas não-lineares do tipo tangente hiperbólica foram simulados e identificados por um filtro de Volterra ora de terceira ordem ora de quinta ordem. Nos dois sistemas, para o ruído de observação $n(k)$ é simulado um ruído branco com média zero e variância de 10^{-6}.

Identificação de um sistema usando de um filtro de Volterra de 3^a Ordem

Para este cenário, o modelo LNL, conforme apresentado na Figura 3.6, é construído por

$$\begin{aligned} L_1: \quad r(k) &= \mathbf{w}_1^{\mathrm{T}} \tilde{\mathbf{x}}(k) \\ N: \quad z(k) &= \tanh(r(k)) \\ L_2: \quad d(k) &= \mathbf{w}_2^{\mathrm{T}} \mathbf{z}(k) + n(k) \end{aligned}$$

onde

$$\begin{aligned} \mathbf{w}_1 &= [0{,}5 \quad 1 \quad 0{,}5]^{\mathrm{T}} \\ \tilde{\mathbf{x}}(k) &= [\tilde{x}(k) \quad \tilde{x}(k-1) \quad \tilde{x}(k-2)]^{\mathrm{T}} \\ \mathbf{w}_2 &= [0{,}1 \quad -0{,}5 \quad 0{,}1]^{\mathrm{T}} \\ \mathbf{z}(k) &= [z(k) \quad z(k-1) \quad z(k-2)]^{\mathrm{T}} \end{aligned}$$

em que $\mathbf{w}_1$ e $\mathbf{w}_2$ são os vetores do primeiro e segundo filtro, respectivamente, $\tilde{\mathbf{x}}(k)$, $\mathbf{z}(k)$ e $r(k)$ são sinais de entrada como apresentado na Figura 3.6. Para o sinal de entrada $\tilde{x}(k)$ é simulado um ruído branco com média nula e variância 10^{-2}.

Ao usar o filtro de Volterra com $l = 3$ e $m = 4$, a função tangente hiperbólica é aproximada a partir destes coeficientes do *kernel* Volterra de terceira ordem, totalizando no máximo 55 coeficientes.

Utilizando uma aproximação da função tangente hiperbólica com dois termos da Série de Taylor em (3.53), o resultado de $z(k)$ é aproximado por

$$\begin{aligned} z(k) &= \text{tangh}(r(k)) \\ &= \text{tangh}(\mathbf{w}_1^{\text{T}}\tilde{\mathbf{x}}(k)) \\ &= \text{tangh}(0{,}5\tilde{x}(k) + \tilde{x}(k-1) + 0{,}5\tilde{x}(k-2)) \\ &\approx 0{,}5\tilde{x}(k) + \tilde{x}(k-1) + 0{,}5\tilde{x}(k-2) - \frac{(0{,}5\tilde{x}(k) + \tilde{x}(k-1) + 0{,}5\tilde{x}(k-2))^3}{3}, \end{aligned}$$

que, após expandida, resulta em uma equação polinomial de terceira ordem com 13 coeficientes. Ao utilizar esta expressão aproximada para o sinal $z(k)$, após passar pelo segundo filtro, o sinal de saída desejado $\mathbf{d}(k)$ é aproximado em uma equação polinomial de terceira ordem com 27 coeficientes.

Para gerar os resultados a seguir, são tiradas as médias de 50 experimentos completos do algoritmo GLAR para trinta valores de K, $K = 2J{,}5J, 8J, \cdots, 89J$. Uma amostra extra é utilizada em cada intervalo para calcular o erro *a priori* pertinente.

A Figura 3.27 sugere que o sistema é identificado corretamente com, aproximadamente, 12 coeficientes. A escolha de 12 coeficientes ao invés de 27 coeficientes como sugerido pela aproximação da Série de Taylor ocorre por dois motivos. Primeiro, os coeficientes escolhidos no *kernel* Volterra pelo algoritmo GLAR podem ser diferentes dos coeficientes utilizados pela aproximação com a Série de Taylor. Segundo, alguns dos 27 coeficientes sugeridos pela Série de Taylor possuem magnitude muito baixa quando comparados a outros coeficientes mais significativos, e podem ser desprezados.

Pela Figura 3.28, é identificado que o MSE atingiu o valor mínimo determinado pelo ruído do sistema ainda antes de 12 coeficientes, tendo pouca influência a quantidade de dados disponíveis (K). A identificação deste sistema não-linear do tipo tangente hiperbólica usando um filtro de Volterra de terceira ordem reflete bem a característica de esparsidade do sistema.

O MSE mínimo, pela Figura 3.28, já foi atingido em $n = 15$. Pela Figura 3.27, para $N = 15$ é necessário ter $\mu = 0{,}8$, sendo este o valor usado para avaliação do algoritmo GLAR. A quantidade de coeficientes a ser estimada por cada critério avaliado é apresentada na Figura 3.29. Diferente dos outros critérios de seleção, a

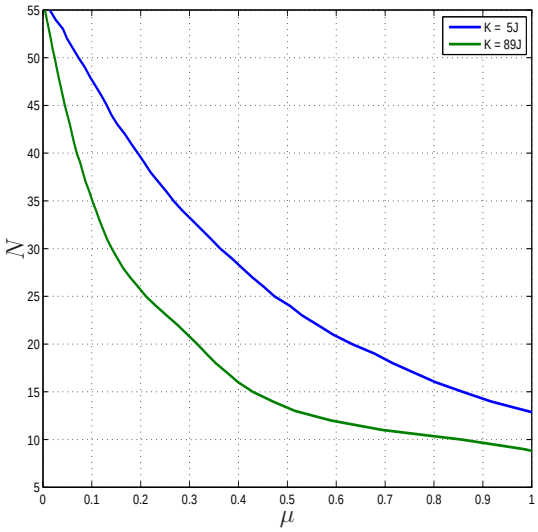


Figura 3.27: O critério de parada geométrico sugere aproximadamente 12 coeficientes não-nulos de um filtro de Volterra de terceira ordem para modelar o sistema não-linear do tipo tangente hiperbólica simulado.

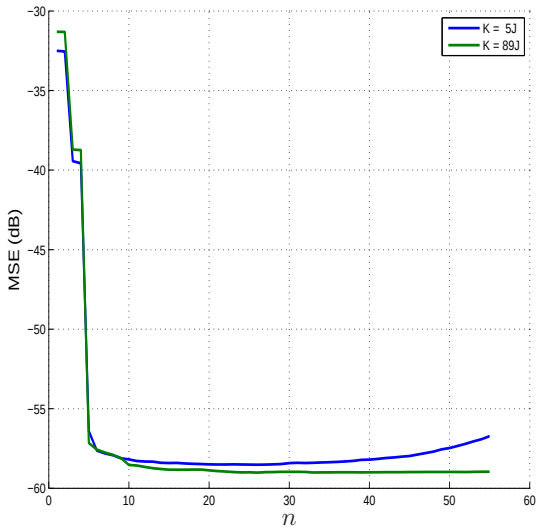


Figura 3.28: MSE, usando o erro *a priori*, calculado a cada iteração do algoritmo GLAR para o sistema não-linear tangente hiperbólica.

curva de $\Delta\theta$ é decrescente em K, pois, sendo o sistema esparso, mais dados levam a uma estimação mais precisa do vetor de coeficientes – fato que melhora o resultado do critério de parada geométrico.

O critério de seleção C_p indica que, para $K = 5J$, cinco coeficientes são necessários para a estimação do vetor de coeficientes. Pela Figura 3.28, em $n = 5$ o resultado de MSE ainda não é o mínimo imposto pelo sistema. Os outros critérios de seleção, AIC e BIC, sugerem mais de 20 coeficientes, quantidade suficiente porém desnecessariamente alta. Desta forma, o critério de parada geométrico obteve o melhor resultado de N.

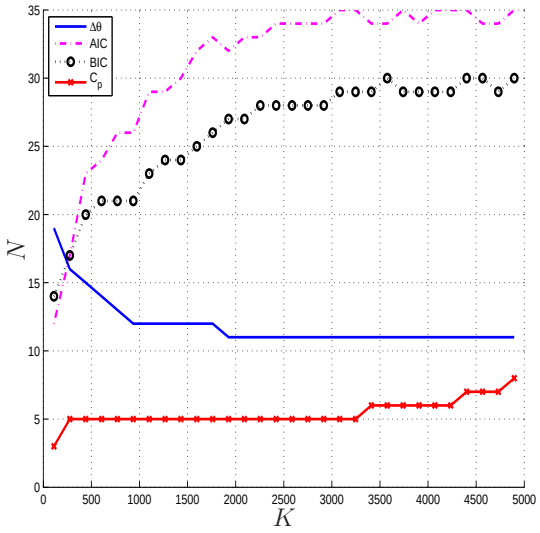


Figura 3.29: Quantidade de coeficientes estimada para cada um dos critérios avaliados. Para o algoritmo GLAR é usado $\mu = 0{,}8$. Neste caso não há quantidade de coeficientes ideal.

O histograma dos ângulos é apresentado na Figura 3.30.

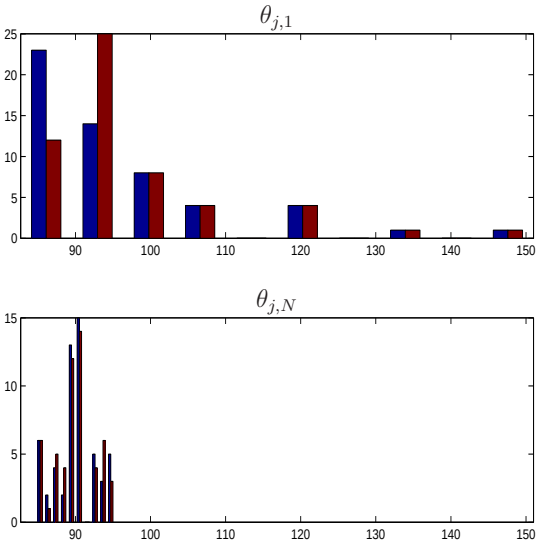


Figura 3.30: Histograma na primeira iteração $n = 1$ e $n = N$, quando $\Delta\boldsymbol{\theta}_N \leq 0{,}8\sigma_{\boldsymbol{\theta}_1}$, para $K = 5J$, em azul, e $K = 89J$, em marrom.

O resultado de MSE ao identificar o sistema com o algoritmo GLAR, com $\mu = 0{,}8$, em combinação com filtro de Volterra é apresentado na Figura 3.31. O erro de estimação resultante do algoritmo CLS, próximo ao mínimo imposto pelo ruído de observação, sugere que os coeficientes nulos indicados pelo algoritmo GLAR estão nas posições corretas. O resultado degradado do algoritmo SSS mostra que os coeficientes estimados, pelo algoritmo LS, de menores magnitudes não devem ser igualados a zero. Aumentar o valor de K não melhora a resposta do algoritmo SSS, indicando que a estimação por mínimos quadrados, feita pelo algoritmo LS, não é

tão precisa.

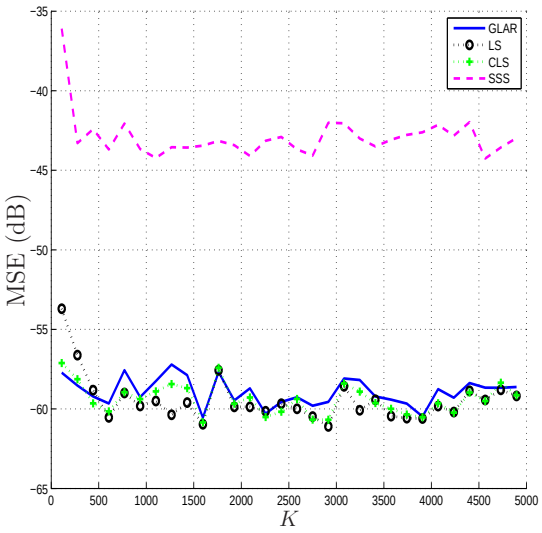


Figura 3.31: MSE, calculado com o erro *a priori*, para os diversos algoritmos avaliados. Para o algoritmo GLAR é usado $\mu = 0{,}8$.

A curva de MSE resultante para o algoritmo GLAR confirma que com apenas 11 coeficientes é possível alcançar o erro de estimação imposto pelo ruído de observação. O algoritmo GLAR obteve um resultado satisfatório aproximando o vetor de coeficientes com apenas 11 coeficientes não-nulos, enquanto que a reposta do algoritmo LS obteve um resultado satisfatório aproximando o vetor de coeficientes com 55 coeficientes não-nulos. É clara a vantagem de utilizar o algoritmo GLAR neste caso.

Identificação de um sistema usando de um filtro de Volterra de 5^a Ordem

Para este cenário, o modelo LNL, conforme apresentado na Figura 3.6, é construído por

$$\begin{aligned} L_1: \quad r(k) &= \mathbf{w}_1^{\mathrm{T}} \tilde{\mathbf{x}}(k) \\ N: \quad z(k) &= \mathrm{tangh}(r(k)) \\ L_2: \quad d(k) &= \mathbf{w}_2^{\mathrm{T}} \mathbf{z}(k) + n(k) \end{aligned}$$

onde

$$\begin{aligned} \mathbf{w}_1 &= [0{,}5 \quad 0{,}5 \quad 1 \quad 0{,}5]^{\mathrm{T}} \\ \tilde{\mathbf{x}}(k) &= [\tilde{x}(k) \quad \tilde{x}(k-1) \quad \tilde{x}(k-2) \quad \tilde{x}(k-3)]^{\mathrm{T}} \\ \mathbf{w}_2 &= [0{,}1 \quad -0{,}5 \quad 1 \quad -0{,}5]^{\mathrm{T}} \\ \mathbf{z}(k) &= [z(k) \quad z(k-1) \quad z(k-2) \quad z(k-3)]^{\mathrm{T}} \end{aligned}$$

em que $\mathbf{w}_1$ e $\mathbf{w}_2$ são os vetores do primeiro e segundo filtro, respectivamente, $\tilde{\mathbf{x}}(k)$, $\mathbf{z}(k)$ e $r(k)$ são sinais de entrada como apresentado na Figura 3.6. Para o sinal de entrada $\tilde{x}(k)$ é simulado um ruído branco com média nula e variância 5×10^{-2}.

Para identificar este sistema, é utilizado um filtro de Volterra com $l = 5$ e $m = 6$. Neste caso, a função tangente hiperbólica é aproximada a partir destes coeficientes do *kernel* Volterra de quinta ordem, totalizando no máximo 791 coeficientes. Utilizando uma aproximação da função tangente hiperbólica com três termos da Série de Taylor em (3.53), o sinal de saída desejado $\mathbf{d}(k)$ é aproximado por uma equação polinomial de quinta ordem com 211 coeficientes.

Um filtro de Volterra de terceira ordem com memória igual a seis ($l = 3$ e $m = 6$) poderia também ser utilizado para identificar este sistema. Foi escolhido um filtro de Volterra de quinta ordem com memória igual a seis ($l = 5$ e $m = 6$) devido à construção prévia deste filtro para identificação do cenário anterior, na Seção 3.3.3.

Para gerar os resultados a seguir, foram tiradas as médias de 50 experimentos completos do algoritmo GLAR para onze valores de K, $K = 2J$,$3J, 4J, \cdots, 12J$. Uma amostra extra é utilizada em cada intervalo para calcular o erro *a priori* pertinente.

A curva de N por μ, apresentada na Figura 3.32, não é significativamente influenciada por K, chegando a, aproximadamente, 100 coeficientes em $\mu = 1$. As curvas sugerem ser satisfatório e suficiente identificar o sistema simulado, cuja não-linearidade é dada por uma função tangente hiperbólica, usando de um filtro de Volterra de quinta ordem.

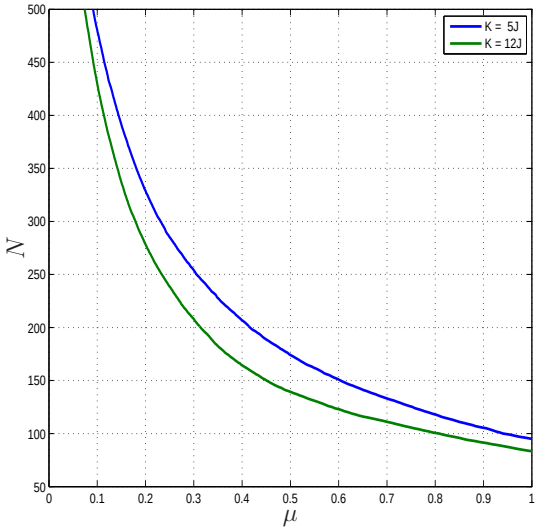


Figura 3.32: O critério de parada geométrico sugere em torno de 100 coeficientes para identificação do sistema simulado.

A curva de MSE, apresentada na Figura 3.33, mostra que o erro mínimo de $-60dB$ é atingido em $n \geq 110$, i.e., após a estimação de pouco mais de 110 co-

eficientes o mínimo imposto pelo ruído de observação é atingido. Este resultado é compatível ao encontrado na Figura 3.32; $\mu = 0{,}8$ é um valor satisfatório para identificação do sistema em análise.

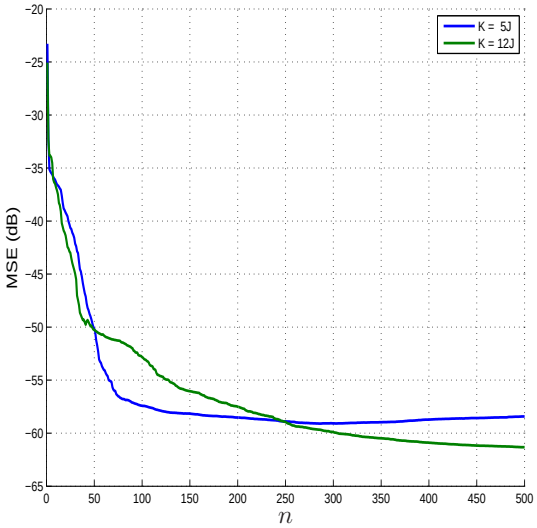


Figura 3.33: MSE, usando o erro *a priori*, calculado a cada iteração do algoritmo GLAR.

Avaliando o algoritmo GLAR com $\mu = 0{,}8$, a quantidade de coeficientes N para cada critério é apresentada na Figura 3.34. Para todos os critérios, a quantidade de coeficientes sugerida ($40 < N < 300$) é bem menor do que a quantidade total de coeficientes ($J = 791$), enfatizando a característica de esparsidade do sistema.

O resultado em $K = 5J$ para o critério de seleção C_p é de pouco mais de 50 coeficientes. Pela Figura 3.33, em $n = 50$, o MSE ainda não é o mínimo imposto pelo ruído de observação. Os outros critérios de seleção sugerem uma quantidade de coeficientes a ser estimada além da necessária. Novamente o critério de parada geométrico apresenta o melhor resultado.

O histograma dos ângulos é apresentado na Figura 3.35.

O resultado de MSE ao identificar o sistema em questão com o algoritmo GLAR, com $\mu = 0{,}8$, em combinação com filtro de Volterra é apresentado na Figura 3.36. A curva de MSE do algoritmo CLS indica que com qualquer quantidade de dados (valor de K), o algoritmo GLAR escolhe corretamente quais coeficientes são nulos.

A resposta de MSE dada pelo algoritmo SSS mostrou que nem sempre os coeficientes estimados com menor magnitude pelo algoritmo LS são aqueles que devem ser iguais a zero. Quanto maior o valor de K, melhor a estimação do vetor de coeficientes pelo algoritmo LS. Quanto melhor a estimação do algoritmo LS, mais próximo a zero são os coeficientes de fato nulos. Por isso, quanto maior a quantidade de dados, melhor a resposta do algoritmo SSS.

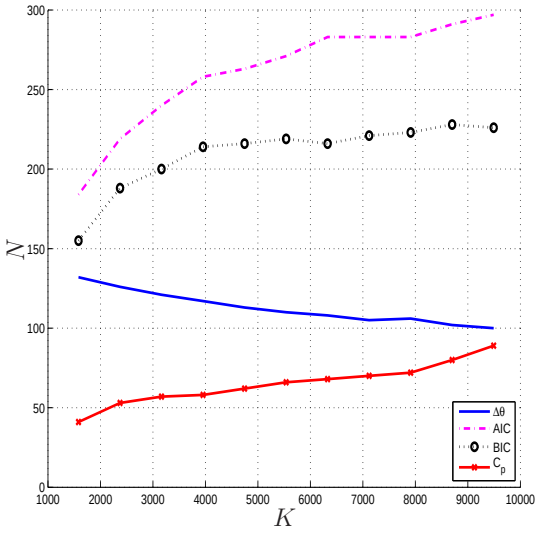


Figura 3.34: Quantidade de coeficientes estimada para cada um dos critérios avaliados. Para o critério de parada geométrico é usado $\mu = 0{,}8$. Neste caso não há quantidade de coeficientes ideal.

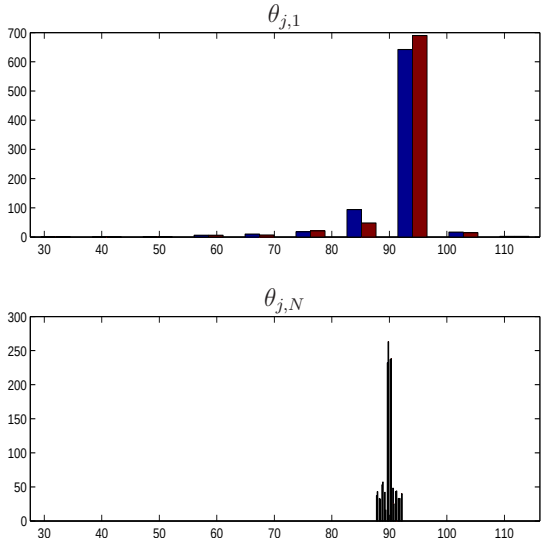


Figura 3.35: Histograma na primeira iteração $n = 1$ e $n = N$, quando $\Delta\boldsymbol{\theta}_N \leq 0{,}8\sigma_{\boldsymbol{\theta}_1}$, para $K = 5J$, em azul, e $K = 12J$, em marrom.

3.4 Conclusão do capítulo

Neste capítulo foi apresentada a primeira contribuição deste trabalho, o algoritmo GLAR. Desenvolvido na Seção 3.2, o algoritmo GLAR utiliza o critério de parada geométrico para interromper o algoritmo LAR. O critério de parada geométrico é calculado com o auxílio dos ângulos entre os vetores de dados de coeficientes e o vetor de erro de estimação. Foi identificado que a cada iteração os ângulos estão mais próximos a 90^0. Quando a condição (3.16) é satisfeita, o algoritmo GLAR é interrompido. Devido à interrupção do algoritmo, N coeficientes são estimados, ao invés do total J (podendo ser $N << J$).

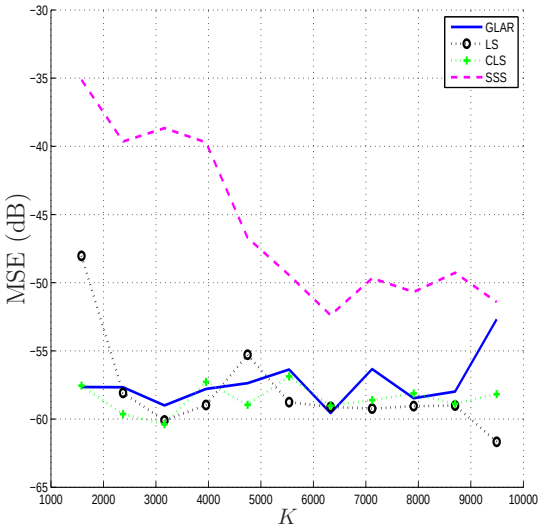


Figura 3.36: MSE, calculado com o erro *a priori*, para os diversos algoritmos avaliados. Para o algoritmo GLAR é usado $\mu = 0{,}8$.

O critério de parada geométrico mostrou-se eficiente em todos os sistemas avaliados. Primeiro, o algoritmo GLAR foi analisado com os dados da Diabetes, utilizado na validação do algoritmo LAR por Efron et al. [5]. O critério de parada geométrico com $\mu = 0{,}2$ resulta na mesma quantidade de coeficientes (marcadores) sugerida pelo critério de seleção C_p.

Para avaliar a identificação de sistemas não-lineares utilizando o conjunto GLAR e filtro de Volterra, modelos LNL simularam sistemas com três tipos de não-linearidades. As funções não-lineares foram dadas por polinômios de terceira ordem, polinômios de quinta ordem e a função tangente hiperbólica. Filtros de Volterra de terceira ordem e quinta ordem foram utilizados para a identificação dos sistemas.

Nos sistemas em que a não-linearidade foi dada por equações polinomiais, a quantidade de coeficientes era conhecida pelo modelo LNL. Com as duas ordens do filtro de Volterra utilizadas, a quantidade de coeficientes estimada foi próxima à quantidade ideal. O ruído de observação imposto pelo sistema não prejudicou a determinação correta da quantidade de coeficientes estimada. O resultado inferior da norma da diferença entre o vetor de coeficientes estimado pelo algoritmo GLAR e o vetor de coeficientes ideal mostrou-se desprezível ao ser comparado o MSE resultante. Com $K \geq 5J$ o vetor de coeficientes é estimado satisfatoriamente, resultando no MSE mínimo imposto pelo ruído de observação.

Nos sistemas em que a não-linearidade foi dada pela função tangente hiperbólica, a quantidade de coeficientes ideal não era conhecida. O algoritmo GLAR estimou corretamente o vetor de coeficientes do sistema com uma quantidade de coeficientes mais baixa do que se a função tangente hiperbólica fosse aproximada pela Série de Taylor. Em torno de 80% dos coeficientes puderam ser nulos, acentuando a

característica de esparsidade do sistema.

Na identificação dos sistemas não-lineares aqui apresentados, o valor de μ foi definido pela quantidade de iterações necessárias para atingir o MSE mínimo imposto pelo ruído de observação. Uma vez identificado $n = N$, o valor correspondente de μ, na curva $N \times \mu$, foi determinado. De modo geral, o valor de μ é determinado pela quantidade de dados disponíveis e pela razão entre coeficientes não-nulos e nulos. Quanto menor o resultado de $N/(J-N)$, i.e. mais esparso o sistema, mais próximo a 1 pode ser o valor de μ. Uma análise matemática do comportamento de μ pode ser tema de um trabalho futuro.

Ajustando o valor de μ adequadamente, a identificação de sistemas não-lineares pelo algoritmo GLAR em conjunto com o filtro de Volterra foi satisfatória em todos os cenários avaliados.

Os resultados aqui apresentados geraram três publicações, em [22, 23, 32]. Um quarto artigo foi submetido em [46].

Algoritmo 3 – Algoritmo GLAR

in: $\tilde{\mathbf{X}}$, $\tilde{\mathbf{d}}$,μ

normalizar para $\mathbf{X}, \mathbf{d}$
$n \leftarrow 1$
$\boldsymbol{\mathcal{A}}_n \leftarrow [\ ]$
$\mathbf{y}_n \leftarrow \mathbf{0}$
$S_n, u_n, a_n, w_n \leftarrow 0$
$\mathbf{c}_n \leftarrow \mathbf{X}\mathbf{d}$ ▷ $\times : KJ;\ +: J(K-1)$
$[C_{max,n}, j_n] \leftarrow \max(|\mathbf{c}_n|)$ ▷ $+ : J$
$\Delta\theta_n \leftarrow 2(90 - \arccos(C_{max,n}/\|\mathbf{d}\|))$ ▷ $\wedge : K+5;\ \times : 1;\ \div : 1;\ + : K;\ - : 2$
while $\Delta\theta_n \geq \mu\sigma_{\boldsymbol{\theta}_1}$ **do**
  $\boldsymbol{\mathcal{A}}_n \leftarrow \begin{bmatrix} \boldsymbol{\mathcal{A}}_{n-1} & j_n \end{bmatrix}^{\mathrm{T}}$
  $\bar{\boldsymbol{\mathcal{A}}}_n \leftarrow \{1, 2, \cdots, J\} - \boldsymbol{\mathcal{A}}_n$
  $\mathbf{P}(n, \mathcal{A}_n(j_n)) \leftarrow 1$
  $\mathbf{p}_{j_n} \leftarrow \mathbf{P}(n, :)^{\mathrm{T}}$
  $d_{j_n} \leftarrow \mathrm{sign}(\mathbf{p}_{j_n}^{\mathrm{T}} \mathbf{c}_n)$
  $\mathbf{x}_{j_n} \leftarrow (d_{j_n} \mathbf{p}_{j_n}^{\mathrm{T}} \mathbf{X})^{\mathrm{T}}$ ▷ $+ : N$
  $\mathbf{X}_n \leftarrow \begin{bmatrix} \mathbf{X}_{n-1}^{\mathrm{T}} & \mathbf{x}_{j_n} \end{bmatrix}^{\mathrm{T}}$
  $\mathbf{f}_n \leftarrow \mathbf{X}_{n-1}\mathbf{x}_{j_n}$ ▷ $\times : KN;\ +: N(K-1)$
  $\mathbf{v}_n \leftarrow \mathbf{S}_{n-1}\mathbf{f}_n$ ▷ $\times : N^2;\ +: N(N-1)$
  $g_n \leftarrow 1 - \mathbf{f}_n^{\mathrm{T}}\mathbf{v}_{n-1}$ ▷ $\times : N;\ +: N-1;\ - : 1$
  $\mathbf{S}_n \leftarrow \begin{bmatrix} \mathbf{S}_{n-1} + \frac{1}{g_n}\mathbf{v}_{n-1}\mathbf{v}_{n-1}^{\mathrm{T}} & -\frac{1}{g_n}\mathbf{v}_{n-1} \\ -\frac{1}{g_n}\mathbf{v}_{n-1}^{\mathrm{T}} & \frac{1}{g_n} \end{bmatrix}$ ▷ $\times : N^2;\ \div : N+1;\ + : N^2$
  $\mathbf{1}_n = \begin{bmatrix} \mathbf{1}_{n-1}^{\mathrm{T}} & 1 \end{bmatrix}^{\mathrm{T}}$
  $t_n \leftarrow \mathbf{1}_{n-1}^{\mathrm{T}}\mathbf{v}_{n-1}$ ▷ $+ : N$
  $\alpha_n \leftarrow \left(\alpha_{n-1}^{-2} + (t_n - 1)^2/g_n\right)^{-1/2}$ ▷ $\wedge : 3; \div : 1;\ + : 1;\ - : 1$
  $\boldsymbol{\omega}_n \leftarrow \alpha_n \mathbf{S}_n \mathbf{1}_n$ ▷ $\times : N;\ +: N-1$
  $\mathbf{u}_n \leftarrow \mathbf{X}_n^{\mathrm{T}}\boldsymbol{\omega}_n$ ▷ $\times : KN;\ +: K(N-1)$
  $\bar{\mathbf{X}}_n \leftarrow [\cdots \mathbf{x}_j \cdots]^{\mathrm{T}}, j \in \bar{\boldsymbol{\mathcal{A}}}_n$
  $\mathbf{a}_n \leftarrow \bar{\mathbf{X}}_n \mathbf{u}_n$ ▷ $\times : K(J-N);\ +: (K-1)(J-N)$
  if $n = J$ **then**
    $\gamma_n \leftarrow C_{max}/\alpha_n$ ▷ $\div : 1$
  else
    for $j = 1$ to $J - n$ **do**
      $\beta_1 \leftarrow \dfrac{C_{max} - c_{\bar{\mathcal{A}}(j),n}}{\alpha_n - a_{j,n}}$ ▷ $\div : J-N;\ - : 2(J-N)$
      $\beta_2 \leftarrow \dfrac{C_{max} + c_{\bar{\mathcal{A}}(j),n}}{\alpha_n + a_{j,n}}$ ▷ $\div : J-N;\ - : 2(J-N)$
      $\boldsymbol{\gamma}_j \leftarrow \min(\beta_1 > 0, \beta_2 > 0)$ ▷ $+ : 2$
    end for
    $[\gamma_n, pos_n] \leftarrow \min(\boldsymbol{\gamma}_j)$ ▷ $+ : J-N$
    $j_n \leftarrow \bar{\mathcal{A}}_n(pos_n)$
  end if
  $\mathbf{y}_n \leftarrow \mathbf{y}_{n-1} + \gamma_n \mathbf{u}_n$ ▷ $\times : K;\ +: K$
  $\mathbf{w}_n \leftarrow [\mathbf{w}_{n-1}^{\mathrm{T}}\ \ 0]^{\mathrm{T}} + \gamma_n \boldsymbol{\omega}_n$ ▷ $\times : N;\ +: N-1$
  $\mathbf{e}_n \leftarrow \mathbf{d} - \mathbf{y}_n$ ▷ $- : K$
  $n \leftarrow n + 1$ ▷ $+ : 1$
  $\mathbf{c}_n \leftarrow \mathbf{X}\mathbf{e}_{n-1}$ ▷ $\times : KJ;\ +: J(K-1)$
  $C_{max,n} = C_{max,n-1} - \gamma_{n-1}\alpha_{n-1}$ ▷ $\times : 1;\ - : 1$
  $\Delta\theta_n = 90 - \arccos(C_{max,n}/\|\mathbf{e}_{n-1}\|)$ ▷ $\wedge : K+5;\ \times : 1;\ \div : 1;\ + : K;\ - : 2$
end while
$N \leftarrow n - 1$
$\mathbf{D}_N \leftarrow \mathrm{diag}(\mathrm{sign}(\mathbf{P}\mathbf{c}_N))$
$\hat{\mathbf{w}}_N \leftarrow \mathbf{P}^{\mathrm{T}}\mathbf{D}_N\mathbf{w}_N$

Capítulo 4

Algoritmo GLAR-RLS

4.1 Introdução do capítulo

Neste capítulo é apresentada a segunda contribuição desta tese: a junção do algoritmo GLAR com o algoritmo RLS, resultando no novo algoritmo GLAR-RLS.

O algoritmo LAR, por utilizar dados em lote normalizados por coeficiente (ou canal) e não por tempo, impede sua implementação recursiva no tempo – os dados utilizados para o desenvolvimento do algoritmo GLAR foram previamente coletados. Entretanto, o algoritmo GLAR identifica satisfatoriamente os coeficientes ativos em diferentes sistemas, conforme concluído no Capítulo 3. A motivação do algoritmo GLAR-RLS foi utilizar a informação dos coeficientes ativos fornecida pelo algoritmo GLAR para implementar o algoritmo RLS, recursivo no tempo, com menor ordem. O algoritmo GLAR-RLS, apresentado na Seção 4.2, diminui a complexidade computacional e melhora a estimação do vetor de coeficientes quando comparado ao algoritmo RLS.

A Seção 4.3 apresenta a avaliação do algoritmo GLAR-RLS. Para tal, sistemas não-lineares são simulados, tendo o vetor de coeficientes modificado em períodos temporais conhecidos, e identificados usando o algoritmo GLAR-RLS em conjunto com o filtro de Volterra de terceira ordem e quinta ordem. Os resultados do algoritmo GLAR-RLS são comparados com os resultados do algoritmo RLS.

4.2 Desenvolvimento do algoritmo GLAR-RLS

Um algoritmo que alterna o emprego do algoritmo GLAR e do algoritmo RLS foi designado nesta tese como algoritmo GLAR-RLS. A motivação para o seu desenvolvimento foi obter um algoritmo com menor complexidade computacional quando comparado ao algoritmo RLS, com recursividade temporal e com consciência situacional.

Para abaixar a complexidade computacional, o algoritmo GLAR é utilizado na identificação dos $N << J$ coeficientes ativos. O algoritmo RLS é, então, aplicado recursivo no tempo, estimando apenas os N coeficientes.

Algoritmos recursivos no tempo possuem o índice temporal crescente, $k = 1,2,\ldots$ Como o algoritmo GLAR é aplicado em cima de um conjunto de dados de tamanho determinado ($K \times J$), o conceito de janela temporal é utilizado para o desenvolvimento deste algoritmo. O algoritmo GLAR é aplicado, quando necessário, às últimas L amostras.

Para definir o valor de L, o algoritmo GLAR foi avaliado quanto à quantidade de amostras necessária para uma estimação eficiente dos coeficientes do sistema em identificação. Tendo em vista os resultados do Capítulo 3, foi identificado que a quantidade mínima de amostras depende da quantidade de coeficientes do sistema em identificação. Assim, ao longo do trabalho é usado

$$L = 5J, \tag{4.1}$$

onde J é a quantidade total de coeficientes.

A ideia do algoritmo GLAR-RLS é usar o algoritmo GLAR para diminuir a complexidade do algoritmo RLS, aplicado recursivamente no tempo. Enquanto o sistema está em estado estacionário, após a aplicação do algoritmo GLAR com L amostras, o algoritmo RLS é o responsável por uma adaptação suave dos N coeficientes. Caso o sistema mude ao longo do tempo, é necessário aplicar o algoritmo GLAR novamente. Para definir quando aplicar novamente o algoritmo GLAR, a magnitude do erro *a priori* estimado pelo algoritmo GLAR-RLS é utilizado.

Para reconhecer uma modificação no sistema, devemos saber se o erro de estimação *a priori* é devido a uma modificação no sistema ou se é um erro *aceitável*. Considerando uma curva de distribuição normal, aproximadamente 70% dos erros *a priori* possuem valor absoluto abaixo do desvio padrão.

Seja $\tilde{e}(k)$ o erro *a priori* no instante k e $\sigma_{\tilde{\mathbf{e}}}$ o seu desvio padrão. Quando

$$|\tilde{e}(k)| > \sigma_{\tilde{\mathbf{e}}}, \tag{4.2}$$

o erro *a priori* pode ser um *outlier* ou um erro devido a uma modificação nos coeficientes — um novo sistema deve ser identificado. Um novo sistema poderia ter, por exemplo, a quantidade de coeficientes nulos modificada. Reaplicar o algoritmo GLAR possibilita a modificação no valor de N.

Aplicar novamente o algoritmo GLAR após a primeira ocorrência de (4.2) é ineficiente, pois este erro pode ser um *outlier* do sistema em análise (pertencer aos 30% de erros não contabilizados). Ainda que este erro seja devido a uma modificação no sistema (um novo vetor de coeficientes deve ser identificado), uma única amostra

deste novo sistema na janela L não será suficiente para estimar o novo vetor de coeficientes corretamente. Por outro lado, esperar que muitos erros ocorram diminui o tempo de convergência do algoritmo.

A quantidade de erros contabilizada para aplicar novamente o algoritmo GLAR é diretamente proporcional ao tamanho da janela L. Toda a janela temporal (ou a maior parte dela) deve conter amostras do novo sistema a ser identificado, para estimar corretamente o vetor de coeficientes. Para isso, é utilizado um contador de erros, designado aqui por τ.

O contador τ é o responsável por reaplicar o algoritmo GLAR, atuando como a consciência situacional do algoritmo. τ é incrementado em duas condições: quando $\tau < J$ e (4.2) ocorre; ou quando $\tau \geq J$, sem verificar (4.2). Ao atingir $\tau = L = 5J$, a janela de dados está completa com amostras de um novo sistema, e o algoritmo GLAR é aplicado novamente.

A razão de incrementar o contador quando $\tau \geq J$ sem verificar se $|\tilde{e}(k)| > \sigma_{\tilde{\mathbf{e}}}$ é aqui abordada. À medida que o algoritmo RLS tenta se adaptar ao novo sistema, a estimação do vetor de coeficientes melhora, e pode ocorrer do erro *a priori* deixar de ser superior a $\sigma_{\tilde{\mathbf{e}}}$. Ou seja, após algumas amostras com $|\tilde{e}(k)| > \sigma_{\tilde{\mathbf{e}}}$, a condição pode não ser mais satisfeita mesmo que o sistema tenha sido modificado. Por outro lado, uma vez que J erros aconteceram, é improvável que a totalidade de erros seja de *outliers*. Quando $\tau > J$, há grande probabilidade do sistema ter sofrido uma uma modificação, e o algoritmo espera mais $4J$ amostras para então disparar o algoritmo GLAR. Ao ser disparado, o algoritmo GLAR deve estar com a janela de amostras L completa com dados do novo sistema.

O algoritmo GLAR-RLS, apresentado em sua forma mais genérica no Algoritmo 4, é resumido pelas seguintes etapas:

1. Aplicar o algoritmo RLS com J coeficientes enquanto $k < L$;
2. Aplicar o algoritmo GLAR com as últimas L amostras temporais;
3. Aplicar novamente o algoritmo RLS com N coeficientes determinados pelo algoritmo GLAR (supostamente $N << J$);
4. Caso $\tau < J$ e $|\tilde{e}(k)| > \sigma_{\tilde{\mathbf{e}}}$, ou $\tau \geq J$, incrementar $\tau = \tau + 1$ e seguir ao passo 5. Caso contrário, retornar ao passo 3;
5. Caso $\tau = L$, retornar ao passo 2. Caso contrário, retornar ao passo 3.

O algoritmo GLAR-RLS aqui apresentado tem o objetivo de detectar mudanças bruscas no sistema. Por isso, o algoritmo RLS, quando aplicado, não possui fator de esquecimento (i.e. $\lambda = 1$), para não prejudicar na consciência situacional de mudanças bruscas. Caso o fator de esquecimento fosse utilizado não seria possível

Algoritmo 4 – Algoritmo GLAR-RLS – visão geral

1: $L \leftarrow 5J$
2: **for** $k = 1,2,3,\cdots$ **do**
3: **if** $k < L$ **then**
4: aplicar o algoritmo RLS com J coeficientes
5: **else**
6: **if** $k = L$ or $\tau = L$ **then**
7: aplicar o algoritmo GLAR e definir $N \leq J$
8: $\tau \leftarrow 0$
9: **else**
10: aplicar o algoritmo RLS com N coeficientes
11: **if** $(\tau < J$ **and** $|\tilde{e}(k)| > \sigma_{\tilde{\mathbf{e}}})$ **or** $(\tau \geq J)$ **then**
12: $\tau \leftarrow \tau + 1$
13: **end if**
14: **end if**
15: **end if**
16: **end for**

identificar a mudança brusca no vetor de coeficientes, pois o desvio padrão do erro, calculado recursivamente, ao se adaptar ao novo sistema, não atinge a condição $|e(k)| > \sigma_{\mathbf{e}}$, e impede o acionamento do algoritmo GLAR. Como consequência, o algoritmo GLAR-RLS é ruim no rastreamento e sensível à perturbações no sinal de entrada ou desejado. Contudo, a presença de perturbações não faz com que o algoritmo GLAR seja desnecessariamente acionado. Para que modificações lentas no vetor de coeficientes sejam notadas pelo algoritmo GLAR-RLS, outra forma para a consciência situacional deve ser desenvolvida, tema de trabalhos futuros.

Para que o algoritmo GLAR-RLS tenha complexidade computacional tão baixa quanto possível, dois pontos ainda são levados em consideração no seu desenvolvimento.

O primeiro ponto é em relação ao desvio padrão dos erros *a priori*. Para diminuir a complexidade computacional do algoritmo GLAR-RLS, o desvio padrão dos erros é calculado de modo recursivo, como apresentado na Seção 4.2.1.

O segundo ponto é em relação à matriz de correlação inversa. Ao ir do passo 2 ao passo 3, é desejável utilizar todas as respostas calculadas pelo algoritmo GLAR, dentre elas a matriz de correlação inversa. No entanto, devido à alteração no número de coeficientes estimados e da construção do conjunto ativo, esta tarefa não é trivial. Este ponto é abordado na Seção 4.2.2.

Por fim, o pseudocódigo completo do algoritmo GLAR-RLS, bem como sua complexidade computacional são apresentados na Seção 4.2.3.

4.2.1 Desvio padrão recursivo

A partir do desvio padrão dos erros *a priori* é determinado se o algoritmo GLAR é aplicado novamente. Para minimizar a complexidade computacional, o desvio padrão é calculado de modo recursivo no tempo dentro do algoritmo GLAR-RLS.

O desvio padrão dos erros *a priori* estimado no instante k é definido por

$$\sigma_{\tilde{\mathbf{e}}}(k) = \left[\frac{1}{k-1}\sum_{i=1}^{k}(\tilde{e}(i) - m_{\tilde{\mathbf{e}}}(k))^2\right]^{1/2} \tag{4.3}$$

onde $m_{\tilde{\mathbf{e}}}(k)$ é a média aritmética dos erros *a priori* até o instante k, dada por

$$m_{\tilde{\mathbf{e}}}(k) = \frac{\sum_{i=1}^{k}\tilde{e}(i)}{k}. \tag{4.4}$$

A média aritmética dos erros *a priori* até o instante $k+1$, usando (4.4), é dada por

$$\begin{aligned} m_{\tilde{\mathbf{e}}}(k+1) &= \frac{\sum_{i=1}^{k+1}\tilde{e}(i)}{k+1} \\ &= \frac{\tilde{e}(k+1) + \sum_{i=1}^{k}\tilde{e}(i)}{k+1} \\ &= \frac{\tilde{e}(k+1) + km_{\tilde{\mathbf{e}}}(k)}{k+1}. \end{aligned} \tag{4.5}$$

Desenvolvendo (4.3), o desvio padrão dos erros *a priori* até o instante k é calculado por

$$\begin{aligned} \sigma_{\tilde{\mathbf{e}}}(k) &= \left[\frac{1}{k-1}\sum_{i=1}^{k}(\tilde{e}(i) - m_{\tilde{\mathbf{e}}}(k))^2\right]^{1/2} \\ &= \left\{\frac{1}{k-1}\left[\sum_{i=1}^{k}\tilde{e}^2(i) - 2\sum_{i=1}^{k}\tilde{e}(i)m_{\tilde{\mathbf{e}}}(k) + \sum_{i=1}^{k}m_{\tilde{\mathbf{e}}}^2(k)\right]\right\}^{1/2}, \\ &= \left\{\frac{1}{k-1}\left[km_{\tilde{\mathbf{e}}}^2(k) - 2m_{\tilde{\mathbf{e}}}(k)\sum_{i=1}^{k}\tilde{e}(i) + \sum_{i=1}^{k}\tilde{e}^2(i)\right]\right\}^{1/2}, \end{aligned}$$

usando (4.4),

$$\begin{aligned} \sigma_{\tilde{\mathbf{e}}}(k) &= \left\{\frac{1}{k-1}\left[km_{\tilde{\mathbf{e}}}^2(k) - 2km_{\tilde{\mathbf{e}}}^2(k) + \sum_{i=1}^{k}\tilde{e}^2(i)\right]\right\}^{1/2} \\ &= \left\{\frac{1}{k-1}\left[-km_{\tilde{\mathbf{e}}}^2(k) + \sum_{i=1}^{k}\tilde{e}^2(i)\right]\right\}^{1/2}. \end{aligned} \tag{4.6}$$

A partir de (4.6) é obtida a relação

$$\sum_{i=1}^{k} \tilde{e}^2(i) = (k-1)\sigma^2_{\tilde{\mathbf{e}}}(k) + km^2_{\tilde{\mathbf{e}}}(k). \tag{4.7}$$

O desvio padrão dos erros *a priori* até o instante $k+1$, usando (4.6) e (4.7), é dado por

$$\begin{aligned}
\sigma_{\tilde{\mathbf{e}}}(k+1) &= \left\{ \frac{1}{k} \left[-(k+1)m^2_{\tilde{\mathbf{e}}}(k+1) + \sum_{i=1}^{k+1} \tilde{e}^2(i) \right] \right\}^{1/2} \\
&= \left\{ \frac{1}{k} \left[\tilde{e}^2(k+1) - (k+1)m^2_{\tilde{\mathbf{e}}}(k+1) + \sum_{i=1}^{k} \tilde{e}^2(i) \right] \right\}^{1/2} \\
&= \left[\frac{\tilde{e}^2(k+1) - (k+1)m^2_{\tilde{\mathbf{e}}}(k+1) + (k-1)\sigma^2_{\tilde{\mathbf{e}}}(k) + km^2_{\tilde{\mathbf{e}}}(k)}{k} \right]^{1/2}
\end{aligned} \tag{4.8}$$

A complexidade para o cálculo do desvio padrão de modo recursivo e não recursivo é apresentada na Tabela 4.1.

Tabela 4.1: Complexidade computacional do cálculo de desvio padrão

	Sem Recursividade	Com Recursividade
Elevado ao quadrado	k	4
Raiz quadrada	1	1
Multiplicação	0	3
Divisão	1	1
Soma	$k-1$	3
Subtração	k	2
Ordem	k	4

4.2.2 Relação entre $\tilde{\mathbf{S}}$, $\mathbf{S}_N$ e $\tilde{\mathbf{S}}_N$

O algoritmo GLAR-RLS primeiro aplica o algoritmo RLS para estimar o vetor de coeficientes de tamanho J. Nesta primeira etapa, a matriz de correlação inversa, $\tilde{\mathbf{S}}$, possui dados não normalizados de todos os coeficientes. Na etapa seguinte, o algoritmo GLAR-RLS aplica o algoritmo GLAR para estimar o vetor de coeficientes de tamanho N. Nesta etapa intermediária, a matriz de correlação inversa, $\mathbf{S}_N$, possui dados normalizados dos coeficientes ativos. Os $J-N$ coeficientes não estimados são iguais a zero. Na etapa seguinte, o algoritmo RLS é aplicado novamente para estimar apenas os N coeficientes ativos indicados pelo algoritmo GLAR. Nesta etapa, a matriz de correlação inversa, $\tilde{\mathbf{S}}_N$, possui dados não normalizados dos coeficientes

ativos.

Para melhor compreensão da relação entre $\tilde{\mathbf{S}}$, $\mathbf{S}_N$ e $\tilde{\mathbf{S}}_N$, seja a mesma quantidade de amostras temporais, de 1 a k, para calcular cada matriz de correlação $\tilde{\mathbf{S}}$, $\mathbf{S}_N$ e $\tilde{\mathbf{S}}_N$. Por esse motivo, o índice k será, sempre que possível, suprimido, uma vez que é desnecessário nesta análise.

A matriz de correlação inversa utilizada durante a primeira aplicação do algoritmo RLS, $\tilde{\mathbf{S}}$ $(J \times J)$ é dada por

$$\tilde{\mathbf{S}} = (\tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\mathrm{T}})^{-1}, \tag{4.9}$$

onde $\tilde{\mathbf{X}}$ $(J \times K)$ é a matriz de sinal de entrada.

A matriz de correlação inversa utilizada no algoritmo GLAR, $\mathbf{S}_N$ $(N \times N)$ é dada por

$$\mathbf{S}_N = \left(\mathbf{X}_N \mathbf{X}_N^{\mathrm{T}}\right)^{-1}, \tag{4.10}$$

onde $\mathbf{X}_N$ $(N \times K)$ é definida em (2.89),

$$\mathbf{X}_N = \mathbf{D}_N \mathbf{P}_N \mathbf{X},$$

onde $\mathbf{D}_N$ $(N \times N)$ é a matriz diagonal de sinais, $\mathbf{P}_N$ $(N \times J)$ é a matriz de seleção como em $\boldsymbol{\mathcal{A}}_N$, e $\mathbf{X}$ $(J \times K)$ é a matriz de sinal de entrada normalizada, definida em (2.50),

$$\mathbf{X} = \mathbf{N}_{\mathbf{x}}^{-1}\left(\tilde{\mathbf{X}} - \mathbf{M}_{\tilde{\mathbf{x}}}\right),$$

onde $\mathbf{N}_{\mathbf{x}}$ $(J \times J)$ é a matriz diagonal de comprimento de todos os coeficientes (2.49), $\mathbf{M}_{\tilde{\mathbf{x}}}$ $(J \times K)$ é a matriz de médias de todos os coeficientes (2.48), e $\tilde{\mathbf{X}}$ $(J \times K)$ é a matriz de sinal de entrada. Desta forma, $\mathbf{S}_N$ tem três particularidades: os dados para calcular a matriz estão normalizados; a matriz está ordenada como os coeficientes que entram no conjunto ativo; o sinal da matriz está alterado.

Quando o algoritmo RLS é novamente utilizado, a matriz de correlação inversa, $\tilde{\mathbf{S}}_N$ $(N \times N)$ é dada por

$$\tilde{\mathbf{S}}_N = (\tilde{\mathbf{X}}_N \tilde{\mathbf{X}}_N^{\mathrm{T}})^{-1}, \tag{4.11}$$

onde

$$\tilde{\mathbf{X}}_N = \mathbf{P}_N \tilde{\mathbf{X}}, \tag{4.12}$$

onde $\mathbf{P}_N$ $(N \times J)$ é a matriz de seleção como em $\boldsymbol{\mathcal{A}}_N$, e $\tilde{\mathbf{X}}$ $(J \times K)$ é a matriz de sinal de entrada. Ou seja, a matriz $\tilde{\mathbf{S}}_N$ continua com a ordem em que os coeficientes entram no conjunto ativo, porém sem os dados normalizados nem com sinal alterado.

A relação entre $\tilde{\mathbf{S}}$, $\mathbf{S}_N$ e $\tilde{\mathbf{S}}_N$ é vislumbrada quando estas matrizes são descritas

em função de $\tilde{\mathbf{R}} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\mathrm{T}}$. Primeiro, $\tilde{\mathbf{S}}$ é reescrita por

$$\tilde{\mathbf{S}} = (\tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\mathrm{T}})^{-1} = \tilde{\mathbf{R}}^{-1}. \tag{4.13}$$

Substituindo (4.12) em (4.11), $\tilde{\mathbf{S}}_N$ é reescrita por

$$\begin{aligned}
\tilde{\mathbf{S}}_N &= (\mathbf{P}_N\tilde{\mathbf{X}}(\mathbf{P}_N\tilde{\mathbf{X}})^{\mathrm{T}})^{-1} \\
&= (\mathbf{P}_N\tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\mathrm{T}}\mathbf{P}_N^{\mathrm{T}})^{-1} \\
&= (\mathbf{P}_N\tilde{\mathbf{R}}\mathbf{P}_N^{\mathrm{T}})^{-1}.
\end{aligned} \tag{4.14}$$

Uma vez que $\mathbf{P}_N$ não é uma matriz quadrada, esta equação não pode ser reduzida. A última matriz de correlação, $\mathbf{S}_N$, substituindo (2.89) e (2.50) em (4.10), é reescrita por

$$\begin{aligned}
\mathbf{S}_N &= (\mathbf{X}_N\mathbf{X}_N^{\mathrm{T}})^{-1} \\
&= (\mathbf{D}_N\mathbf{P}_N\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{P}_N^{\mathrm{T}}\mathbf{D}_N)^{-1} \\
&= \mathbf{D}_N(\mathbf{P}_N\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{P}_N^{\mathrm{T}})^{-1}\mathbf{D}_N \\
&= \mathbf{D}_N\left[\mathbf{P}_N\mathbf{N}_{\tilde{\mathbf{x}}}^{-1}\left(\tilde{\mathbf{X}} - \mathbf{M}_{\tilde{\mathbf{x}}}\right)\left(\tilde{\mathbf{X}} - \mathbf{M}_{\tilde{\mathbf{x}}}\right)^{\mathrm{T}}\mathbf{N}_{\tilde{\mathbf{x}}}^{-1}\mathbf{P}_N^{\mathrm{T}}\right]^{-1}\mathbf{D}_N \\
&= \mathbf{D}_N\left[\mathbf{P}_N\mathbf{N}_{\tilde{\mathbf{x}}}^{-1}\left(\tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\mathrm{T}} - \mathbf{M}_{\tilde{\mathbf{x}}}\tilde{\mathbf{X}}^{\mathrm{T}} - \tilde{\mathbf{X}}\mathbf{M}_{\tilde{\mathbf{x}}}^{\mathrm{T}} + \mathbf{M}_{\tilde{\mathbf{x}}}\mathbf{M}_{\tilde{\mathbf{x}}}^{\mathrm{T}}\right)\mathbf{N}_{\tilde{\mathbf{x}}}^{-1}\mathbf{P}_N^{\mathrm{T}}\right]^{-1}\mathbf{D}_N.
\end{aligned} \tag{4.15}$$

Para simplificar (4.15), é necessário encontrar a relação entre $\mathbf{M}_{\tilde{\mathbf{x}}}\tilde{\mathbf{X}}^{\mathrm{T}}$, $\tilde{\mathbf{X}}\mathbf{M}_{\tilde{\mathbf{x}}}^{\mathrm{T}}$ e $\mathbf{M}_{\tilde{\mathbf{x}}}\mathbf{M}_{\tilde{\mathbf{x}}}^{\mathrm{T}}$.

Pela definição em (2.48) e (2.36), $\mathbf{M}_{\tilde{\mathbf{x}}}\tilde{\mathbf{X}}^{\mathrm{T}}$ pode ser reduzida a

$$\begin{aligned}
\mathbf{M}_{\tilde{\mathbf{x}}}\tilde{\mathbf{X}}^{\mathrm{T}} &= [\mathbf{m}_{\mathbf{x}} \cdots \mathbf{m}_{\mathbf{x}}]\begin{bmatrix} \tilde{\mathbf{x}}^{\mathrm{T}}(1) \\ \vdots \\ \tilde{\mathbf{x}}^{\mathrm{T}}(k) \end{bmatrix} \\
&= \begin{bmatrix} m_{\tilde{\mathbf{x}}_1} & \cdots & m_{\tilde{\mathbf{x}}_1} \\ \vdots & \ddots & \vdots \\ m_{\tilde{\mathbf{x}}_J} & \cdots & m_{\tilde{\mathbf{x}}_J} \end{bmatrix}\begin{bmatrix} \tilde{x}_1(1) & \cdots & \tilde{x}_J(1) \\ \vdots & \ddots & \vdots \\ \tilde{x}_1(k) & \cdots & \tilde{x}_J(k) \end{bmatrix} \\
&= \begin{bmatrix} m_{\tilde{\mathbf{x}}_1}\sum_1^k \tilde{x}_1(k) & \cdots & m_{\tilde{\mathbf{x}}_1}\sum_1^k \tilde{x}_J(k) \\ \vdots & \ddots & \vdots \\ m_{\tilde{\mathbf{x}}_J}\sum_1^k \tilde{x}_1(k) & \cdots & m_{\tilde{\mathbf{x}}_J}\sum_1^k \tilde{x}_J(k) \end{bmatrix} \\
&= \begin{bmatrix} k\, m_{\tilde{\mathbf{x}}_1}^2 & \cdots & k\, m_{\tilde{\mathbf{x}}_1}\, m_{\tilde{\mathbf{x}}_J} \\ \vdots & \ddots & \vdots \\ k\, m_{\tilde{\mathbf{x}}_J}\, m_{\tilde{\mathbf{x}}_1} & \cdots & k\, m_{\tilde{\mathbf{x}}_J}^2 \end{bmatrix}.
\end{aligned} \tag{4.16}$$

De forma análoga, $\tilde{\mathbf{X}}\mathbf{M}_{\tilde{\mathbf{x}}}^{\mathrm{T}}$ pode ser reduzida a

$$\begin{aligned}
\tilde{\mathbf{X}}\mathbf{M}_{\tilde{\mathbf{x}}}^{\mathrm{T}} &= [\tilde{\mathbf{x}}(1)\ \cdots \tilde{\mathbf{x}}] \begin{bmatrix} \mathbf{m}_{\mathbf{x}}^{\mathrm{T}} \\ \vdots \\ \mathbf{m}_{\mathbf{x}}^{\mathrm{T}} \end{bmatrix} \\
&= \begin{bmatrix} \tilde{x}_1(1) & \cdots & \tilde{x}_1(k) \\ \vdots & \ddots & \vdots \\ \tilde{x}_J(1) & \cdots & \tilde{x}_J(k) \end{bmatrix} \begin{bmatrix} m_{\tilde{\mathbf{x}}_1} & \cdots & m_{\tilde{\mathbf{x}}_J} \\ \vdots & \ddots & \vdots \\ m_{\tilde{\mathbf{x}}_1} & \cdots & m_{\tilde{\mathbf{x}}_J} \end{bmatrix} \\
&= \begin{bmatrix} k\, m_{\tilde{\mathbf{x}}_1}^2 & \cdots & k\, m_{\tilde{\mathbf{x}}_1}\, m_{\tilde{\mathbf{x}}_J} \\ \vdots & \ddots & \vdots \\ k\, m_{\tilde{\mathbf{x}}_J}\, m_{\tilde{\mathbf{x}}_1} & \cdots & k\, m_{\tilde{\mathbf{x}}_J}^2 \end{bmatrix}. \qquad (4.17)
\end{aligned}$$

Por último, $\mathbf{M}_{\tilde{\mathbf{x}}}\mathbf{M}_{\tilde{\mathbf{x}}}^{\mathrm{T}}$ é reescrita por

$$\begin{aligned}
\mathbf{M}_{\tilde{\mathbf{x}}}\mathbf{M}_{\tilde{\mathbf{x}}}^{\mathrm{T}} &= [\mathbf{m}_{\mathbf{x}}\ \cdots \mathbf{m}_{\mathbf{x}}] \begin{bmatrix} \mathbf{m}_{\mathbf{x}}^{\mathrm{T}} \\ \vdots \\ \mathbf{m}_{\mathbf{x}}^{\mathrm{T}} \end{bmatrix} \\
&= \begin{bmatrix} m_{\tilde{\mathbf{x}}_1} & \cdots & m_{\tilde{\mathbf{x}}_1} \\ \vdots & \ddots & \vdots \\ m_{\tilde{\mathbf{x}}_J} & \cdots & m_{\tilde{\mathbf{x}}_J} \end{bmatrix} \begin{bmatrix} m_{\tilde{\mathbf{x}}_1} & \cdots & m_{\tilde{\mathbf{x}}_J} \\ \vdots & \ddots & \vdots \\ m_{\tilde{\mathbf{x}}_1} & \cdots & m_{\tilde{\mathbf{x}}_J} \end{bmatrix} \\
&= \begin{bmatrix} km_{\tilde{\mathbf{x}}_1}^2 & \cdots & km_{\tilde{\mathbf{x}}_1}\, m_{\tilde{\mathbf{x}}_J} \\ \vdots & \ddots & \vdots \\ km_{\tilde{\mathbf{x}}_J}\, m_{\tilde{\mathbf{x}}_1} & \cdots & km_{\tilde{\mathbf{x}}_J}^2 \end{bmatrix}. \qquad (4.18)
\end{aligned}$$

Comparando (4.16), (4.17) e (4.18),

$$\mathbf{M}_{\tilde{\mathbf{x}}}\tilde{\mathbf{X}}^{\mathrm{T}} = \tilde{\mathbf{X}}\mathbf{M}_{\tilde{\mathbf{x}}}^{\mathrm{T}} = \mathbf{M}_{\tilde{\mathbf{x}}}\mathbf{M}_{\tilde{\mathbf{x}}}^{\mathrm{T}}. \qquad (4.19)$$

A matriz de correlação inversa $\mathbf{S}_N$ (4.15) é reduzida, então, a

$$\mathbf{S}_N = \mathbf{D}_N \left[\mathbf{P}_N\mathbf{N}_{\tilde{\mathbf{x}}}^{-1}\left(\tilde{\mathbf{R}} - \mathbf{M}_{\tilde{\mathbf{x}}}\mathbf{M}_{\tilde{\mathbf{x}}}^{\mathrm{T}}\right)\mathbf{N}_{\tilde{\mathbf{x}}}^{-1}\mathbf{P}_N^{\mathrm{T}}\right]^{-1}\mathbf{D}_N. \qquad (4.20)$$

Em resumo, as equações das matrizes de correlação inversa $\tilde{\mathbf{S}}$, $\mathbf{S}_N$ e $\tilde{\mathbf{S}}_N$ são dadas por

$$\begin{aligned}
&\tilde{\mathbf{S}} = \tilde{\mathbf{R}}^{-1}, \\
&\mathbf{S}_N = \mathbf{D}_N \left[\mathbf{P}_N\mathbf{N}_{\tilde{\mathbf{x}}}^{-1}\left(\tilde{\mathbf{R}} - \mathbf{M}_{\tilde{\mathbf{x}}}\mathbf{M}_{\tilde{\mathbf{x}}}^{\mathrm{T}}\right)\mathbf{N}_{\tilde{\mathbf{x}}}^{-1}\mathbf{P}_N^{\mathrm{T}}\right]^{-1}\mathbf{D}_N, \text{ e} \\
&\tilde{\mathbf{S}}_N = (\mathbf{P}_N\tilde{\mathbf{R}}\mathbf{P}_N^{\mathrm{T}})^{-1}.
\end{aligned}$$

A relação entre $\tilde{\mathbf{S}}$, $\mathbf{S}_N$ e $\tilde{\mathbf{S}}_N$ não é trivial. Por isso, as três matrizes de correlação são calculadas de modo independente. Para minimizar a complexidade computacional, é utilizado o Lema de Inversão de Matrizes nos três casos.

A matriz de correlação inversa $\tilde{\mathbf{S}}$ é calculada por

$$\begin{aligned}
\boldsymbol{\Psi}(k) &= \tilde{\mathbf{S}}(k-1)\tilde{\mathbf{x}}(k) \\
\mathbf{g}(k) &= \frac{\boldsymbol{\Psi}(k)}{1+\tilde{\mathbf{x}}(k)^{\mathrm{T}}\boldsymbol{\Psi}(k)} \\
\tilde{\mathbf{S}}(k) &= \tilde{\mathbf{S}}(k-1) - \mathbf{g}(k)\boldsymbol{\Psi}^{\mathrm{T}}(k).
\end{aligned}$$

A matriz de correlação inversa $\mathbf{S}_N$ é calculada como apresentado na Seção 3.2.

A matriz de correlação inversa $\tilde{\mathbf{S}}_N$ é primeiro calculada paralelamente à matriz $\mathbf{S}_N$, dentro do algoritmo GLAR. Similar ao apresentado na Seção 3.2, a matriz $\tilde{\mathbf{S}}_N$ é calculada no algoritmo GLAR por

$$\tilde{\mathbf{x}}_{j_n} = (\mathbf{p}_{j_n}^{\mathrm{T}}\tilde{\mathbf{X}})^{\mathrm{T}} \tag{4.21}$$

$$\tilde{\mathbf{X}}_n = \begin{bmatrix} \tilde{\mathbf{X}}_{n-1}^{\mathrm{T}} & \tilde{\mathbf{x}}_{j_n} \end{bmatrix}^{\mathrm{T}} \tag{4.22}$$

$$\tilde{\mathbf{f}}_n = \tilde{\mathbf{X}}_{n-1}\tilde{\mathbf{x}}_{j_n} \tag{4.23}$$

$$\tilde{\mathbf{v}}_n = \tilde{\mathbf{S}}_{n-1}\tilde{\mathbf{f}}_n \tag{4.24}$$

$$\tilde{g}_n = \tilde{\mathbf{x}}_{j_n}^{\mathrm{T}}\tilde{\mathbf{x}}_{j_n} - \tilde{\mathbf{f}}_n^{\mathrm{T}}\tilde{\mathbf{v}}_{n-1} \tag{4.25}$$

$$\tilde{\mathbf{S}}_n = \begin{bmatrix} \tilde{\mathbf{S}}_{n-1} + \frac{1}{\tilde{g}_n}\tilde{\mathbf{v}}_{n-1}\tilde{\mathbf{v}}_{n-1}^{\mathrm{T}} & -\frac{1}{\tilde{g}_n}\tilde{\mathbf{v}}_{n-1} \\ -\frac{1}{\tilde{g}_n}\tilde{\mathbf{v}}_{n-1}^{\mathrm{T}} & \frac{1}{\tilde{g}_n} \end{bmatrix}. \tag{4.26}$$

Este cálculo deve ser acrescentado ao Algoritmo 3 para o funcionamento correto do algoritmo GLAR-RLS. Ao sair do algoritmo GLAR, a matriz $\tilde{\mathbf{S}}_N$ está definida e pode, a partir de então, ser calculada recursivamente por

$$\begin{aligned}
\tilde{\mathbf{x}}_N(k) &= \mathbf{P}_N\tilde{\mathbf{x}}(k) \\
\boldsymbol{\Psi}(k) &= \tilde{\mathbf{S}}_N(k-1)\tilde{\mathbf{x}}_N(k) \\
\mathbf{g}(k) &= \frac{\boldsymbol{\Psi}(k)}{1+\tilde{\mathbf{x}}_N(k)^{\mathrm{T}}\boldsymbol{\Psi}(k)} \\
\tilde{\mathbf{S}}_N(k) &= \tilde{\mathbf{S}}_N(k-1) - \mathbf{g}(k)\boldsymbol{\Psi}^{\mathrm{T}}(k).
\end{aligned}$$

4.2.3 Pseudocódigo e complexidade computacional

O pseudocódigo do algoritmo GLAR-RLS é apresentado no Algoritmo 5. O Algoritmo 5 faz referência ao Algoritmo 3 e ao Algoritmo 6. No Algoritmo 3 deve ser inserido o cálculo de $\tilde{\mathbf{S}}_N$, como em (4.21)–(4.26).

A complexidade computacional depende do instante k. Quando $k < L$, a complexidade computacional depende do algoritmo RLS com tamanho J, como apresentado na Tabela 4.2. Quando o algoritmo GLAR é aplicado, em $k \geq L$, a complexidade computacional do algoritmo GLAR-RLS é apresentada na Tabela 4.3. A complexidade computacional do algoritmo GLAR aqui neste caso é ligeiramente diferente da complexidade computacional apresentada na Seção 3.2.4, devido ao cômputo de $\tilde{\mathbf{S}}_N$. Quando $k \geq L$, sem que o algoritmo GLAR esteja sendo aplicado, a complexidade computacional depende do algoritmo RLS com uma ordem reduzida, $N << J$, como apresentado na Tabela 4.4.

Algoritmo 5 – Algoritmo GLAR-RLS

in: $\mu, J, \tilde{\mathbf{x}}(k), \tilde{d}(k)$

$L \leftarrow 5J$
$k \leftarrow 0$
$m_{\tilde{e}}(0) \leftarrow 0$
$\tilde{\mathbf{S}}(0) \leftarrow \text{diag}(\mathbf{1})$
$\sigma_{\tilde{e}}(0) \leftarrow 0$
$\tilde{\mathbf{w}}(0) \leftarrow \mathbf{0}$
for $k = 1, 2, \cdots$ **do**
  if $k < L$ **then**
    aplicar o algoritmo RLS com $N = J$ coeficientes, **Algoritmo 6**
  else
    if $k = L$ ou $\tau = L$ **then**
      $\tilde{e}(k) \leftarrow \tilde{d}(k) - \tilde{\mathbf{w}}^{\mathrm{T}}(k-1)\tilde{\mathbf{x}}(k)$ ▷ $\times : J;\ +: N-1;\ -: 1$
      aplicar o algoritmo GLAR e definir $N \leq J$, **Algoritmo 3**
      $\tau \leftarrow 0$
    else
      $\tilde{\mathbf{x}}_N(k) \leftarrow \mathbf{P}\tilde{\mathbf{x}}(k)$
      aplicar o algoritmo RLS com N coeficientes, **Algoritmo 6**
      if $\tau < J$ **then**
        if $e(k) > \sigma_{\mathbf{e}}$ **then** $\tau \leftarrow \tau + 1$
        end if
      else
        $\tau \leftarrow \tau + 1$
      end if
    end if
  end if
  $m_{\mathbf{e}}(k) \leftarrow (\tilde{e}(k) + (k-1)m_{\tilde{\mathbf{e}}}(k-1))/k$ ▷ $\times : 1; \div : 1; + : 1\ -: 1$
  $\sigma_{\mathbf{e}}(k) \leftarrow \sqrt{[\tilde{e}^2(k) - km_{\tilde{\mathbf{e}}}^2(k) + (k-1)(\sigma_{\tilde{\mathbf{e}}}^2(k-1) + m_{\tilde{\mathbf{e}}}^2(k-1))]/k}$
  ▷ $\wedge : 4;\ \times : 2;\ \div : 1;\ + : 2;\ -: 1$
end for

Algoritmo 6 – Algoritmo RLS

in: $\tilde{d}(k), \tilde{\mathbf{x}}(k)$ ou $\tilde{\mathbf{x}}_N(k), \tilde{\mathbf{w}}(k-1)$ ou $\tilde{\mathbf{w}}_N(k-1), \tilde{\mathbf{S}}(k-1)$ ou $\tilde{\mathbf{S}}_N(k-1)$

1: $\tilde{e}(k) \leftarrow \tilde{d}(k) - \tilde{\mathbf{w}}_N^{\mathrm{T}}(k-1)\tilde{\mathbf{x}}_N(k)$ ▷ $\times : N;\ +: N-1;\ -: 1$

2: $\boldsymbol{\Psi}(k) \leftarrow \tilde{\mathbf{S}}_N(k-1)\tilde{\mathbf{x}}_N(k)$ ▷ $\times : N^2;\ +: N(N-1)$

3: $\mathbf{g}(k) \leftarrow \dfrac{\boldsymbol{\Psi}(k)}{1 + \tilde{\mathbf{x}}_N(k)^{\mathrm{T}}\boldsymbol{\Psi}(k)}$ ▷ $\times : N; \div : N;\ +: N$

4: $\tilde{\mathbf{S}}_N(k) \leftarrow \tilde{\mathbf{S}}_N(k-1) - \mathbf{g}(k)\boldsymbol{\Psi}^{\mathrm{T}}(k)$ ▷ $\times : N^2;\ -: N^2$

5: $\tilde{\mathbf{w}}_N(k) \leftarrow \tilde{\mathbf{w}}_N(k-1) + \tilde{e}(k)\tilde{\mathbf{S}}(k)\tilde{\mathbf{x}}_N(k)$ ▷ $\times : N^2 + N;\ +: N^2$

Tabela 4.2: Complexidade computacional do algoritmo GLAR-RLS quando $k < L$

	Algoritmo 5 Programa principal	Algoritmo 6 RLS
Potência ($\wedge$)	4	0
Multiplicação ($\times$)	3	$3J^2 + 2J$
Divisão ($\div$)	2	J
Soma ($+$)	3	$2J^2$
Subtração ($-$)	1	J^2
Ordem	4	J^2

Tabela 4.3: Complexidade computacional do algoritmo GLAR-RLS quando $k = L$ ou $\tau = L$

	Algoritmo 5 Programa principal	Algoritmo 3 GLAR
Potência ($\wedge$)	4	$2L + 3$
Multiplicação ($\times$)	$J + 3$	$3LJ + 2LN + 2L + 3N^2 + 4N + 5$
Divisão ($\div$)	2	$2J + 5$
Soma ($+$)	$N + 2$	$3LJ + 2LN + 3L - 3J + 4N^2 + 2N - 5$
Subtração ($-$)	2	$L + 4J - 4N + 6$
Ordem	J	LJ

4.3 Avaliação do Algoritmo GLAR-RLS

Nesta seção o algoritmo GLAR-RLS é avaliado em dois pontos: estimação do vetor de coeficientes e consciência situacional. Para tal, sistemas não-lineares estacionários por partes são simulados. Os sistemas são ditos estacionários por partes por possuírem modificações repentinas em dois momentos, seja na quantidade de coeficientes nulos, seja na não-linearidade do sistema.

Para avaliar o algoritmo GLAR-RLS em conjunto com o filtro de Volterra na identificação de sistemas estacionários por partes, a configuração apresentada na Figura 4.1 é utilizada. A combinação do algoritmo RLS, com o fator de esquecimento

Tabela 4.4: Complexidade computacional do algoritmo GLAR-RLS quando $k > L$ e $\tau < L$

	Algoritmo 5 Programa principal	Algoritmo 6 RLS
Potência ($\wedge$)	4	0
Multiplicação ($\times$)	3	$3N^2 + 2N$
Divisão ($\div$)	2	N
Soma ($+$)	3	$2N^2$
Subtração ($-$)	1	N^2
Ordem	4	N^2

$\lambda = 0{,}99$, em conjunto com o filtro de Volterra é também analisada, para comparação de resultados.

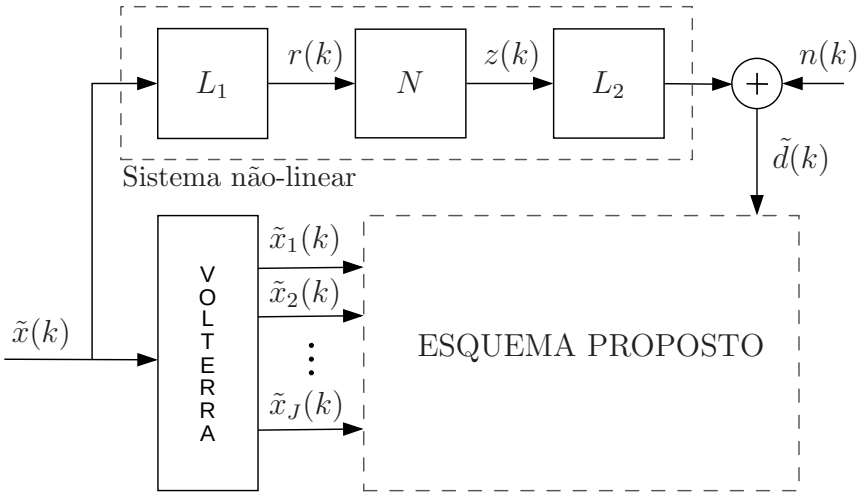


Figura 4.1: Identificação de sistemas não-lineares estacionários por partes com filtro de Volterra e o algoritmo GLAR-RLS. O bloco "ESQUEMA PROPOSTO" pode ser o algoritmo RLS ou o algoritmo GLAR, de acordo com o Algoritmo 5.

Primeiro, um filtro de Volterra de terceira ordem é utilizado na identificação dos sistemas simulados. Para avaliação, sistemas com a não-linearidade representada por uma equação polinomial de terceira ordem, com uma modificação brusca na quantidade de coeficientes nulos, são simulados. Sistemas com a não-linearidade caracterizada ora por uma equação polinomial de terceira ordem ora por uma função tangente hiperbólica são também avaliados.

Em seguida, um filtro de Volterra de quinta ordem é utilizado na identificação do sistema simulado. Neste caso, um sistema com a não-linearidade caracterizada ora por uma equação polinomial de quinta ordem ora por uma função tangente hiperbólica é simulado.

Em todos os cenários, os resultados são variantes com k. Um ruído branco com

média zero e variância de 10^{-6} é usado como ruído de observação, $n(k)$.

4.3.1 Identificação de um sistema não-linear estacionário por partes usando de um filtro de Volterra de 3^a ordem

Diferentes cenários são avaliados, com modificações na quantidade de coeficientes nulos e na função característica da não-linearidade do sistema.

Modificação na quantidade de coeficientes nulos

Nos sistemas deste cenário, a não-linearidade do sistema é caracterizada por uma equação polinomial de terceira ordem. A quantidade de coeficientes nulos no vetor de coeficientes é modificado repentinamente em $k = 4.000$ e $k = 7.000$. O modelo LNL, conforme apresentado na Figura 4.1, é construído por

$$\begin{aligned} L_1: \quad r(k) &= \mathbf{w}_1^{\mathrm{T}} \tilde{\mathbf{x}}(k) \\ N: \quad z(k) &= ar(k) - br^3(k) \\ L_2: \quad d(k) &= \mathbf{w}_2^{\mathrm{T}} \mathbf{z}(k) + n(k) \end{aligned}$$

onde

$$\mathbf{w}_1 = \begin{cases} [0{,}5 \quad 1 \quad 0{,}5]^{\mathrm{T}}, & k < 4.000 \text{ e } k \geq 7.000 \\ [0{,}5 \quad 0 \quad 0{,}5]^{\mathrm{T}}, & 4.000 \leq k < 7.000 \end{cases}$$

$$\tilde{\mathbf{x}}(k) = [\tilde{x}(k) \quad \tilde{x}(k-1) \quad \tilde{x}(k-2)]^{\mathrm{T}}$$

$$\mathbf{w}_2 = \begin{cases} [0{,}1 \quad -0{,}5 \quad 0{,}1]^{\mathrm{T}}, & k < 4.000 \text{ e } k \geq 7.000 \\ [0{,}1 \quad -0{,}5 \quad 0]^{\mathrm{T}}, & 4.000 \leq k < 7.000 \end{cases}$$

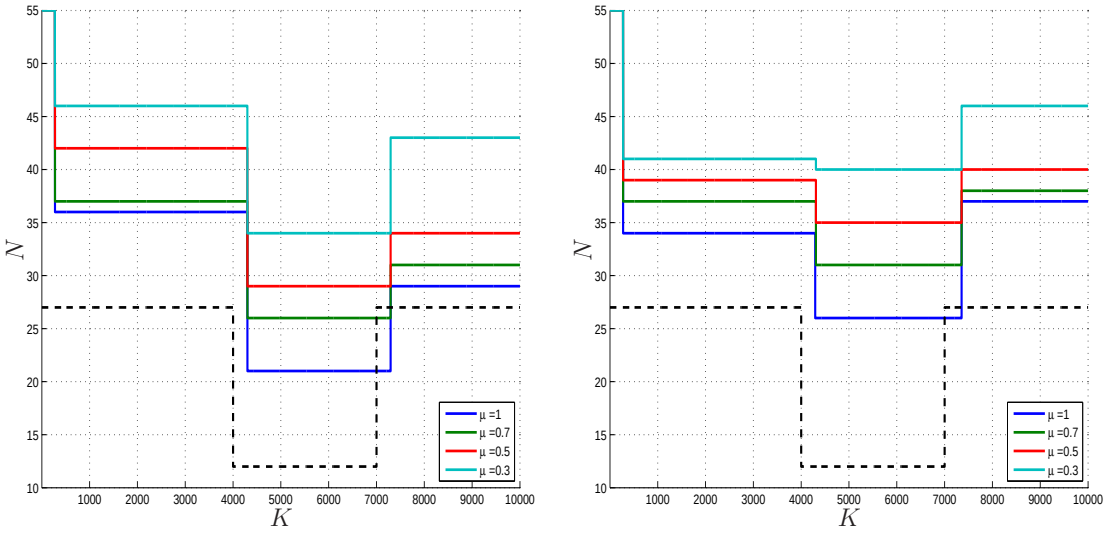
$$\mathbf{z}(k) = [z(k) \quad z(k-1) \quad z(k-2)]^{\mathrm{T}}$$

onde $\mathbf{w}_1$ e $\mathbf{w}_2$ são os vetores do primeiro e segundo filtro, respectivamente, $\tilde{\mathbf{x}}(k)$, $\mathbf{z}(k)$ e $r(k)$ são sinais de entrada como apresentado na Figura 4.1. Novamente, dois sistemas não-lineares são simulados: NL1 com $a = 0{,}1$ e $b = 0{,}01$; e NL2 com $a = b = 1$. Para o sinal de entrada $\tilde{x}(k)$ é simulado um ruído branco com média nula e variância unitária.

O filtro de Volterra utilizado, com $l = 3$ e $m = 4$, resulta em 55 coeficientes ($J = 55$) no seu *kernel*. Pelas equações do sistema LNL, para $k < 4.000$ e $k \geq 7.000$ 27 destes coeficientes são pertinentes, resultando em 28 coeficientes nulos no *kernel* Volterra. Para $4.000 \leq k < 7.000$ a quantidade de coeficientes no modelo LNL diminui para 12 coeficientes, totalizando 43 coeficientes nulos no *kernel* Volterra.

Para facilitar a compreensão da atuação do algoritmo GLAR no algoritmo GLAR-RLS, os resultados apresentados a seguir representam um experimento completo de $k = 1$ a $k = 10.000$.

Com a Figura 4.2 é possível identificar quando o algoritmo GLAR é acionado, no momento em que ocorre uma modificação na quantidade de coeficientes calculada. A primeira vez que o algoritmo GLAR é acionado ocorre em $k = L$. Nos dois momentos seguintes o algoritmo GLAR é acionado devido a $\tau = L$, em $k = 4.320$ e $k = 7.315$ para NL1 e em $k = 4.355$ e $k = 7.460$ para NL2. Isto mostra que o algoritmo GLAR-RLS avaliou corretamente a modificação do sistema em análise em $k = 4.000$ e $k = 7.000$, e, consequentemente, após $\tau = L$ amostras, acionou o algoritmo GLAR.



(a) NL1 (b) NL2

Figura 4.2: Quantidade de coeficientes estimados ao longo do tempo. A reta em pontilhado representa a quantidade total de coeficientes não-nulos conhecida, dada pelo modelo LNL por 27 coeficientes, em $k < 4.000$ e $k \geq 7.000$, e 12 coeficientes, em $4.000 \leq k < 7.000$. Para $k < L$, J coeficientes são estimados. Para $k > L$, N coeficientes são estimados, determinado pelo algoritmo GLAR.

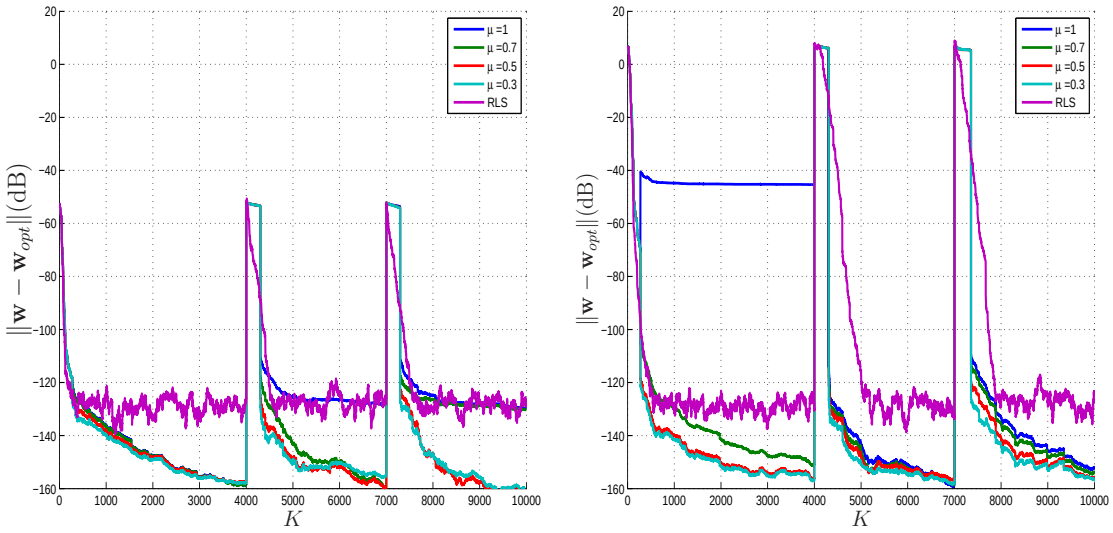
Em um resultado ideal, sendo $L = 5J = 275$, o algoritmo GLAR seria acionado em $k = 4.275$ e $k = 7.275$. Dois motivos justificam o atraso no acionamento do algoritmo GLAR. Primeiro, o algoritmo RLS adapta o vetor de coeficientes ao novo sistema, alterando a variância do erro. Uma vez que o algoritmo RLS tenta adaptar o vetor de coeficientes estimado a uma nova situação, o erro diminui, podendo ser inferior a $\sigma_{\mathbf{e}}$. O segundo motivo é que, devido ao sinal ser aleatório, erros de baixa magnitude ocorrem mesmo com o sistema alterado. Nos dois casos, o contador τ não é incrementando. O atraso para o sistema NL2 é maior porque os coeficientes tem maior magnitude, resultando em uma variância do erro ($\sigma^2_{\mathbf{e}}$) maior.

Uma última análise feita com a Figura 4.2 refere-se aos valores de μ. Diminuir

o valor de μ aumenta a quantidade de coeficientes estimada, resultado esperado e observado no Capítulo 3. Poderíamos esperar que, para o valor de μ constante, a quantidade de coeficientes escolhida em $k < 4.000$ e $k \geq 7.000$ seria igual, uma vez que o vetor de coeficientes é o mesmo. Analisando a Figura 4.2 em $k < 4.000$ e $k \geq 7.000$, constata-se que isso não ocorre. Esta diferença é devido ao conjunto de dados utilizado para aplicar o algoritmo GLAR.

No caso de sistemas esparsos, uma vez que os coeficientes não-nulos foram estimados, estimar mais coeficientes não melhora a acurácia do vetor de coeficientes, como observado na Figura 4.3 para os resultados de $\mu = 0{,}3$ e $\mu = 0{,}5$ em ambos os cenários. Por outro lado, se os coeficientes não-nulos não foram estimados corretamente, ou, numa situação extrema, não foram escolhidos pelo algoritmo GLAR, o resultado será degradado. Esta situação é clarificada no caso apresentado de NL2 para $k < 4.000$: pela Figura 4.2(b), a diferença da quantidade de coeficientes calculada para $\mu = 0{,}7$ e $\mu = 1$ é de três coeficientes; estimar três coeficientes a menos gerou um resultado até $100dB$ pior, como pode ser observado na Figura 4.3(b).

Ainda na Figura 4.3(b), o resultado de $\mu = 1$ em $k < 4.000$ está degradado quando comparado com seu resultado em $k \geq 7.000$, períodos em que o vetor de coeficientes ideal é idêntico. O conjunto de dados utilizado pelo algoritmo GLAR na primeira vez que é acionado, em $k = 275$ (dados nos instantes $1 \geq k \geq 275$), e o conjunto de dados utilizado pelo algoritmo GLAR na terceira vez que é acionado, em $k = 7.460$ (dados nos instantes $7.185 \geq k \geq 7.460$), resultaram em respostas diferentes para o mesmo valor de μ e o mesmo vetor de coeficientes.



(a) NL1 (b) NL2

Figura 4.3: Diferença entre o vetor de coeficientes estimado pelo algoritmo GLAR-RLS, com diferentes valores de μ, e o vetor de coeficiente ideal, bem como a diferença entre o vetor de coeficientes estimado pelo algoritmo RLS com $\lambda = 0{,}99$ e o vetor de coeficientes ideal.

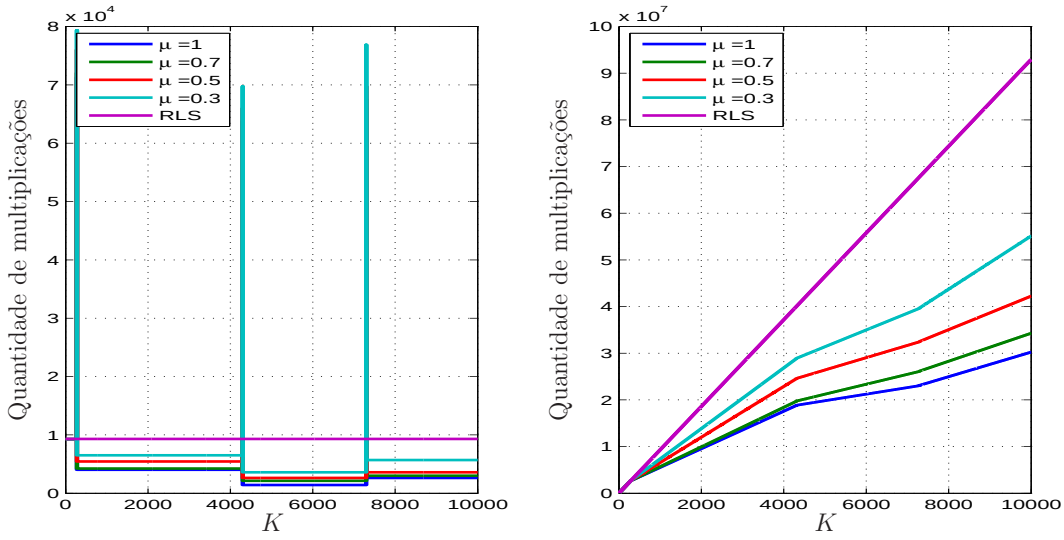
Para o cenário de NL1, Figura 4.3(a), o algoritmo GLAR-RLS, para todos os valores de μ, obteve uma resposta melhor ou equiparada ao algoritmo RLS. No entanto, vemos que no intervalo $4.000 \leq k < 7.000$ para $\mu = 1$ o resultado é pior quando comparado com os resultados do algoritmo GLAR-RLS com outros valores de μ. A resposta de $\mu = 1$, neste intervalo, indica estar no limite para encontrar a quantidade de coeficientes correta. Em $k \geq 7.000$, tanto com $\mu = 1$ quanto com $\mu = 0{,}7$, a resposta do algoritmo GLAR-RLS está degradada quando comparada com os resultados do algoritmo GLAR-RLS com $\mu = 0{,}5$ e $\mu = 0{,}3$. Voltando à Figura 4.2(a), a quantidade de coeficientes estimada para os valores de $\mu = 1$ e $\mu = 0{,}7$ diminuiu muito em $k \geq 7.000$, levando a $N = 29$ e $N = 31$, respectivamente; estes valores de N foram mais baixos do que o menor resultado do algoritmo GLAR-RLS em $k < 4.000$ (com o mesmo vetor de coeficientes), de $N = 36$ para $\mu = 1$.

No resultado apresentado na Figura 4.3 também pode ser observado quando o algoritmo GLAR entrou em ação, momento em que ocorreram mudanças bruscas nos resultados (em $k = 4.320$ e $k = 7.315$ para NL1 e em $k = 4.355$ e $k = 7.460$ para NL2). A primeira vez, entretanto, que o algoritmo GLAR é acionado ($k = 275$) não é tão perceptível.

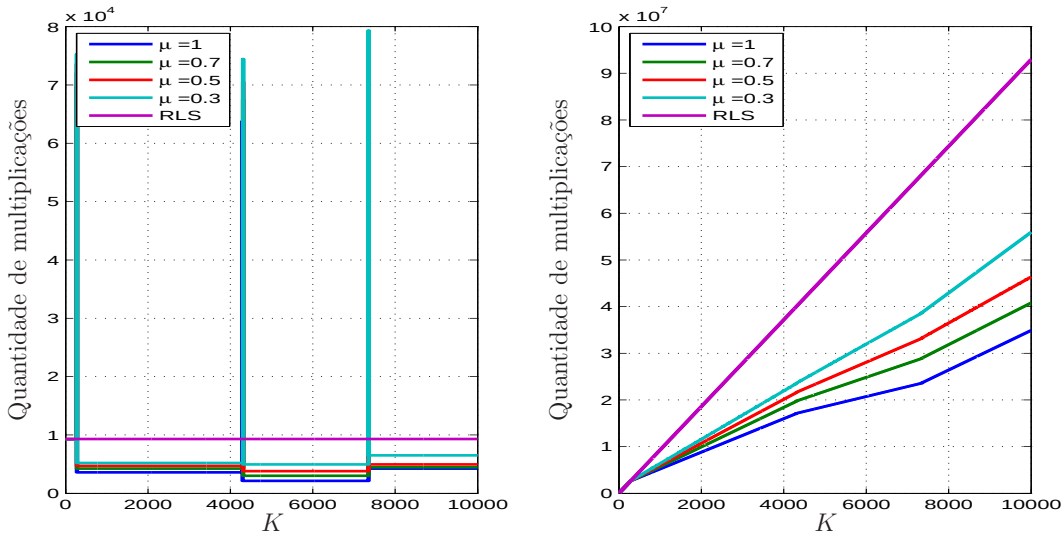
Enquanto $k < 275$, os resultados do algoritmo GLAR-RLS e do algoritmo RLS não são idênticos, pois o algoritmo RLS possui um fator de esquecimento $\lambda = 0{,}99$, enquanto que para o algoritmo GLAR-RLS não há esquecimento ($\lambda = 1$). Por este motivo, o algoritmo GLAR-RLS possui sua curva pouco modificada na Figura 4.3 em $4.000 \leq k \leq 4.320$ e $7.000 \leq k \leq 7.315$ em NL1, assim como em $4.000 \leq k \leq 4.355$ e $7.000 \leq k \leq 7.460$ em NL2. Nestes intervalos, o contador τ está sendo incrementado.

A escolha de μ impacta diretamente a complexidade computacional do algoritmo GLAR-RLS, uma vez que define a quantidade de coeficientes a ser estimada recursivamente. Para comparar a complexidade computacional gerada pelo algoritmo GLAR-RLS com diferentes valores de μ e pelo algoritmo RLS, a complexidade computacional avaliada, calculada a cada instante de instante k, é a soma da quantidade de multiplicações, divisões e potências. A atuação do algoritmo GLAR-RLS gera picos de complexidade computacional, como apresentado nos gráficos a esquerda na Figura 4.4. Entretanto, se o esquema é avaliado dinamicamente e a carga computacional é acumulada ao longo do tempo, devido à quantidade de coeficientes estimada reduzida, o algoritmo GLAR-RLS é mais econômico, guardando energia ao longo do tempo, como observado nos gráficos à direita na Figura 4.4. A inclinação das retas, nos gráficos a direita, é diretamente proporcional a N, quantidade de coeficientes não-nulos estimados determinada pelo algoritmo GLAR.

Uma vez entendida a atuação do algoritmo GLAR-RLS, o conjunto do algoritmo GLAR-RLS em combinação com filtro de Volterra é agora avaliado usando a média de 100 experimentos completos, de $k = 1$ a $k = 10.000$. Para isso, considerando



(a) NL1



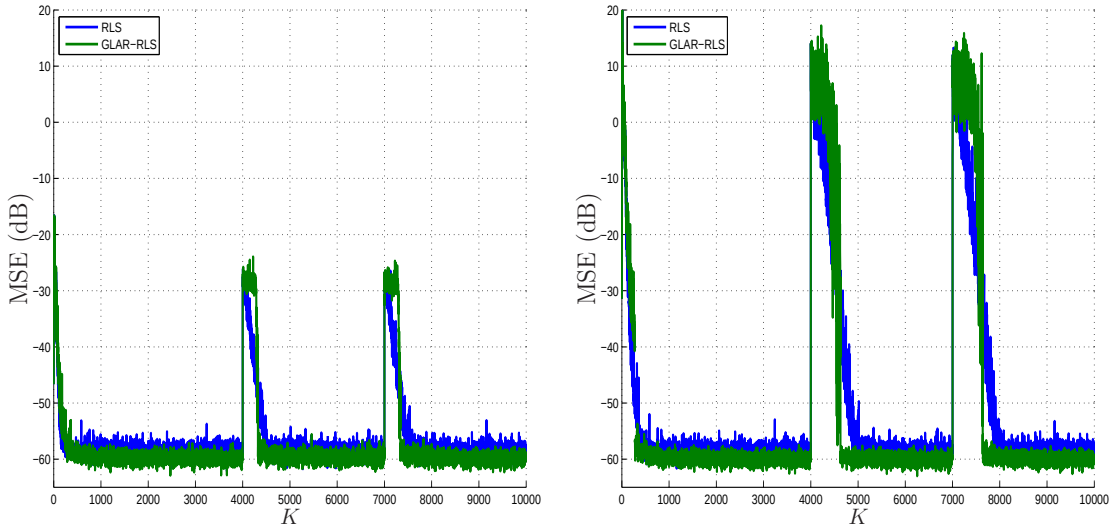
(b) NL2

Figura 4.4: Picos de complexidade computacional ocorrem quando o algoritmo GLAR é acionado (gráficos a esquerda), porém quando analisado ao longo do tempo, o ganho em complexidade computacional é notável (gráficos a direita).

os resultados anteriores, é definido $\mu = 0{,}5$ como o valor ideal para que todos os coeficientes necessários tenham grande probabilidade de serem estimados. O valor $\mu = 0{,}5$ já era esperado devido às conclusões do Capítulo 3.

O resultado de MSE, Figura 4.5, quando utilizado o algoritmo GLAR-RLS com $\mu = 0{,}5$ é comparado com o resultado do algoritmo RLS com fator de esquecimento $\lambda = 0{,}99$. Quando o vetor de coeficientes é modificado, durante o período em que τ está sendo contabilizado, o resultado do algoritmo GLAR-RLS é degradado. Esta perda é compensada, além do ganho em complexidade computacional, por uma convergência mais rápida, melhor observada em NL2.

Durante os períodos estacionários, após a convergência, o algoritmo GLAR-RLS



(a) NL1 (b) NL2

Figura 4.5: MSE, calculado com o erro *a priori*, utilizando o algoritmo GLAR-RLS com $\mu = 0{,}5$ e o algoritmo RLS com fator de esquecimento $\lambda = 0{,}99$.

apresenta um resultado mais satisfatório do que o algoritmo RLS, i.e., o MSE é mais próximo ao ruído de observação imposto pelo sistema. Isto ocorre porque o algoritmo GLAR identifica corretamente quais coeficientes devem ser estimados recursivamente pelo algoritmo GLAR-RLS, enquanto que os coeficientes restantes permanecem iguais a zero. Com isso, o resultado do algoritmo GLAR-RLS tende a ser ligeiramente melhor do que o resultado do algoritmo RLS para sistemas esparsos.

Modificação na função de não-linearidade

Neste cenário, a função de não-linearidade do sistema é caracterizada ora por uma equação polinomial de terceira ordem ora pela função tangente hiperbólica. As modificações ocorrem em $k = 4.000$ e $k = 7.000$. O modelo LNL, conforme apresentado na Figura 4.1, é construído por

$$
\begin{aligned}
L_1 &: r(k) = \mathbf{w}_1^{\mathrm{T}} \tilde{\mathbf{x}}(k) \\
N &: z(k) = \begin{cases} \tanh(r(k)), & k < 4.000 \\ 0{,}1r(k) - 0{,}01r^3(k), & 4.000 \leq k < 7.000 \\ r(k) - r^3(k), & k \geq 7.000 \end{cases} \\
L_2 &: d(k) = \mathbf{w}_2^{\mathrm{T}} \mathbf{z}(k) + n(k)
\end{aligned}
$$

onde

$$
\begin{aligned}
\mathbf{w}_1 &= [0{,}5 \quad 1 \quad 0{,}5]^{\mathrm{T}} \\
\tilde{\mathbf{x}}(k) &= [\tilde{x}(k) \quad \tilde{x}(k-1) \quad \tilde{x}(k-2)]^{\mathrm{T}}
\end{aligned}
$$

$$\mathbf{w}_2 = [0{,}1 \quad -0{,}5 \quad 0{,}1]^{\mathrm{T}}$$
$$\mathbf{z}(k) = [z(k) \quad z(k-1) \quad z(k-2)]^{\mathrm{T}}$$

onde $\mathbf{w}_1$ e $\mathbf{w}_2$ são os vetores do primeiro e segundo filtro, respectivamente, $\tilde{\mathbf{x}}(k)$, $\mathbf{z}(k)$ e $r(k)$ são sinais de entrada como apresentado na Figura 4.1. Em $4.000 \leq k < 7.000$ o sistema NL1 é simulado; em $k \geq 7.000$ o sistema NL2 é simulado. Para o sinal de entrada $\tilde{x}(k)$ é simulado um ruído branco com média nula e variância 10^{-2}.

O filtro de Volterra utilizado, com $l = 3$ e $m = 4$, resulta em 55 coeficientes ($J = 55$) no seu *kernel*. Pelas equações do sistema LNL, para $k \geq 4.000$, 27 destes coeficientes são pertinentes, modificando de amplitude em $k = 7.000$. Para $k < 4.000$ a quantidade de coeficientes não é conhecida. Os primeiros resultados apresentados a seguir possuem um experimento completo de $k = 1$ a $k = 10.000$.

A quantidade de coeficientes sugerida pelo algoritmo GLAR para diferentes valores de μ é apresentada na Figura 4.6. O algoritmo GLAR foi inicializado para todos os valores de μ em $k = 275$ e em $k = 4.285$ e $k = 7.287$. O algoritmo identifica corretamente quando há modificação no sistema em análise.

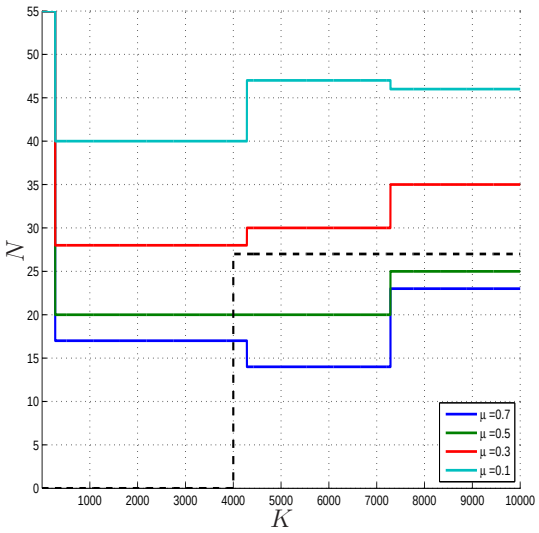


Figura 4.6: Quantidade de coeficientes estimados ao longo do tempo. A reta em pontilhado representa a quantidade total de coeficientes não-nulos conhecida, dada pelo modelo LNL por 27 coeficientes, em $k \geq 4.000$. Para $k < 4.000$, a quantidade de coeficientes não é conhecida.

As descontinuidades nas curvas de μ, Figura 4.6, indicam quando o algoritmo GLAR é aplicado, porém nem sempre aplicar o algoritmo GLAR provoca uma descontinuidade neste tipo de curva. Para $\mu = 0{,}5$ a quantidade de coeficientes N é a mesma antes e depois que o algoritmo GLAR é aplicado próximo a $k = 4.285$, impossibilitando a identificação visual de que o algoritmo GLAR é aplicado. O algoritmo GLAR com $\mu = 0{,}5$ resulta na mesma quantidade de coeficientes estimados ao ser acionado em $k = 275$ e $k = 2.285$, porém diferentes posições no *kernel* Volterra

são escolhidas ou o valor estimado de cada coeficiente é diferente.

Os valores de N neste cenário são inferiores quando comparados aos resultados do cenário anterior com a mesma equação de não-linearidade para o mesmo μ. Isto ocorre porque, neste cenário, o sinal de entrada $\tilde{x}(k)$ é simulado com uma variância muito menor do que no outro cenário, prejudicando a estimação correta do vetor de coeficientes.

Os picos de complexidade computacional identificam quando o algoritmo GLAR é acionado. A descontinuidade, omitida anteriormente para a curva de $\mu = 0{,}5$ na Figura 4.6, é observada no gráfico à esquerda da Figura 4.7.

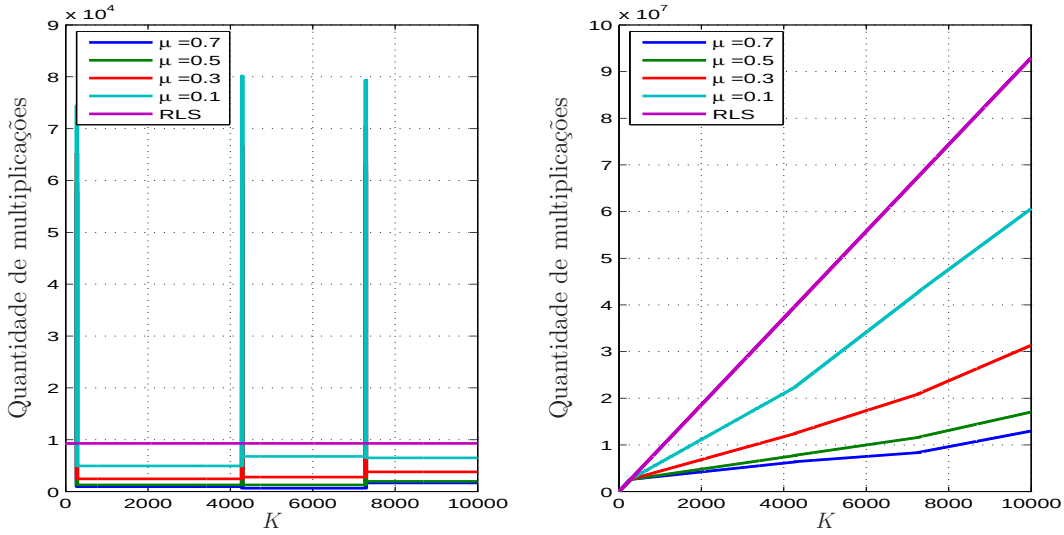
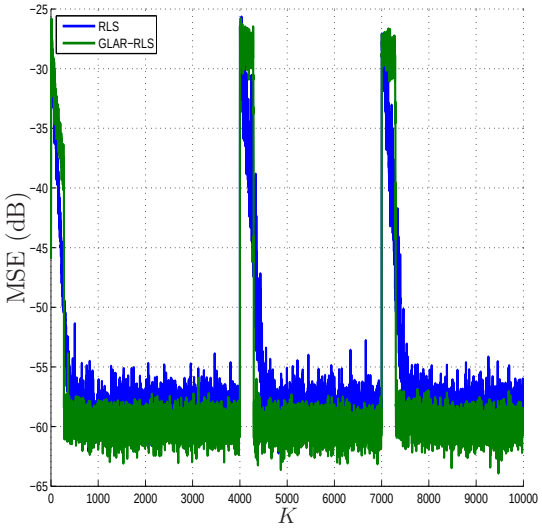


Figura 4.7: Complexidade computacional do algoritmo RLS e do algoritmo GLAR-RLS com diferentes valores de μ. O gráfico à esquerda apresenta a complexidade computacional no instante k. O gráfico à direita apresenta a complexidade computacional acumulada até o instante k.

Quanto menor o valor de μ mais coeficientes são estimados, maior a complexidade computacional. Por isso, a importância da escolha de μ. Para $\mu = 0{,}3$, quando a quantidade de coeficientes mínima é corretamente indicada pelo algoritmo GLAR (Figura 4.6), o ganho em complexidade computacional ao longo do tempo chega a 6×10^7 multiplicações.

O resultado de MSE quando utilizado o algoritmo GLAR-RLS com $\mu = 0{,}3$ é comparado com o resultado do algoritmo RLS com fator de esquecimento $\lambda = 0{,}99$, Figura 4.8. O tempo de convergência de ambos os algoritmos é similar, porém o algoritmo GLAR-RLS se aproxima mais do ruído de observação imposto pelo sistema. O algoritmo GLAR-RLS indica corretamente quais são os coeficientes nulos, melhorando a estimação do vetor de coeficientes.

Para os períodos $4.000 \leq k \leq 4.285$ e $7.000 \leq k \leq 7.287$, o contador τ está sendo incrementado. Nestes períodos, o algoritmo RLS com o fator de esquecimento $\lambda = 0{,}99$ tem uma melhor resposta que o algoritmo GLAR-RLS, que está implementando



(a) NL3

Figura 4.8: MSE, calculado com o erro *a priori*, utilizando o algoritmo GLAR-RLS com $\mu = 0{,}3$ e o algoritmo RLS.

o algoritmo RLS sem o fator de esquecimento ($\lambda = 1$). Uma vez que o algoritmo GLAR é acionado e sua resposta obtida, o algoritmo GLAR-RLS volta a ter uma boa resposta a estimação do novo vetor de coeficientes e, assim, melhora o MSE resultante.

4.3.2 Identificação de um sistema não-linear estacionário por partes usando de um filtro de Volterra de 5^a ordem

Neste cenário, a não-linearidade do sistema é representada ora por uma equação polinomial de quinta ordem ora pela função tangente hiperbólica. A modificação ocorre em $k = 10.000$ e $k = 20.000$. O modelo LNL, conforme apresentado na Figura 4.1, é construído por

$$
\begin{aligned}
L_1 &: r(k) = \mathbf{w}_1^{\mathrm{T}} \tilde{\mathbf{x}}(k) \\
N &: z(k) = \begin{cases} \tanh(r(k)), & k < 10.000 \\ 0{,}1r(k) - 0{,}01r^3(k) + 0{,}01r^5(k), & 10.000 \leq k < 20.000 \\ r(k) - r^3(k) + r^5(k), & k \geq 20.000 \end{cases} \\
L_2 &: d(k) = \mathbf{w}_2^{\mathrm{T}} \mathbf{z}(k) + n(k)
\end{aligned}
$$

onde

$$
\mathbf{w}_1 = [0{,}5 \quad 0{,}5 \quad 1 \quad 0{,}5]^{\mathrm{T}}
$$

$$
\begin{aligned}
\tilde{\mathbf{x}}(k) &= [\tilde{x}(k) \quad \tilde{x}(k-1) \quad \tilde{x}(k-2) \quad \tilde{x}(k-3)]^{\mathrm{T}} \\
\mathbf{w}_2 &= [0{,}1 \quad -0{,}5 \quad 1 \quad -0{,}5]^{\mathrm{T}} \\
\mathbf{z}(k) &= [z(k) \quad z(k-1) \quad z(k-2) \quad z(k-3)]^{\mathrm{T}}
\end{aligned}
$$

onde $\mathbf{w}_1$ e $\mathbf{w}_2$ são os vetores do primeiro e segundo filtro, respectivamente, $\tilde{\mathbf{x}}(k)$, $\mathbf{z}(k)$ e $r(k)$ são sinais de entrada como apresentado na Figura 4.1. O filtro de Volterra utilizado, com $l = 5$ e $m = 6$, resulta em 791 coeficientes ($J = 791$) no seu *kernel*. Pelas equações do sistema LNL, para $k \geq 10.000$, 211 coeficientes são pertinentes, com magnitudes diferentes em cada período ($10.000 \leq k < 20.000$ reproduz o sistema NL1 e $k \geq 20.000$ reproduz o sistema NL2 do Capítulo 3). Para $k < 10.000$, a quantidade de coeficientes no modelo LNL não é definida.

Na Figura 4.9 e na Figura 4.10 são apresentados os resultados de um experimento completo de $k = 1$ a $k = 30.000$.

A Figura 4.9 apresenta a quantidade de coeficientes estimados ao longo do tempo. Em $k < L$, a quantidade de coeficientes calculada pelo algoritmo GLAR-RLS é a quantidade total de coeficientes, $N = J = 791$. Em $k \geq L$, a quantidade de coeficientes é dada pela resposta do algoritmo GLAR, N, modificado a cada vez que o algoritmo GLAR é acionado. Os N coeficientes escolhidos pelo algoritmo GLAR são estimados recursivamente pelo algoritmo RLS.

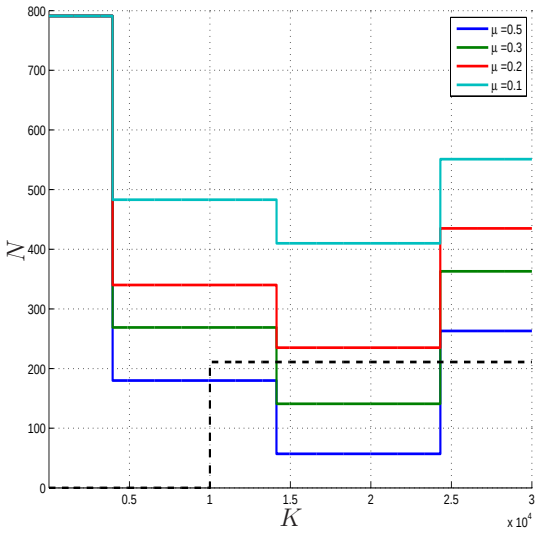


Figura 4.9: Quantidade de coeficientes estimados ao longo do tempo. A reta em pontilhado representa a quantidade total de coeficientes não-nulos conhecida, dada pelo modelo LNL por 211 coeficientes em $k < 10.000$ e $k \geq 20.000$; em $10.000 \leq k < 20.000$, esta quantidade não é conhecida.

As descontinuidades nas curvas da Figura 4.9 mostram que o algoritmo GLAR é acionado em $k = 3.955$, $k = 14.100$ e $k = 24.300$, aproximadamente. O tamanho da janela $L = 5J = 3.955$ e aos momentos de alteração no sistema ($k = 10.000, 20.000$)

justificam esta resposta. O alto valor no tamanho de L é responsável pelo tempo que o algoritmo GLAR leva até ser acionado. O objetivo de testar a consciência situacional do algoritmo GLAR-RLS é atingido com sucesso, o algoritmo GLAR-RLS mostra-se apto a identificar as modificações no sistema em análise.

Para $10.000 \leq k < 20.000$ e $k \geq 20.000$, em que a quantidade de coeficientes do modelo LNL é similar porém com coeficientes de magnitudes diferentes, o valor de N sugerido pelo algoritmo GLAR para o mesmo μ é diferente. Na Seção 3.3.3 foi demonstrado que a magnitude dos coeficientes influencia no valor de N. Porém na Seção 3.3.3, o valor de N resultante para cada μ foi maior do que os resultados aqui apresentados para NL1 e NL2. Isto ocorre porque a variância do sinal de entrada é menor neste cenário do que na Seção 3.3.3. Além do valor de μ, a matriz de dados de entrada e o sinal de saída de referência, utilizados ao aplicar o algoritmo GLAR, influenciam o valor de N sugerido.

As complexidades computacionais do algoritmo RLS e do algoritmo GLAR-RLS com diferentes valores de μ são apresentadas na Figura 4.10. No gráfico à esquerda, os picos de complexidade indicam o instante k em que o algoritmo GLAR é acionado. Estes picos de complexidade mudam a inclinação da reta, diretamente influenciada pelo resultado N, das curvas à direita.

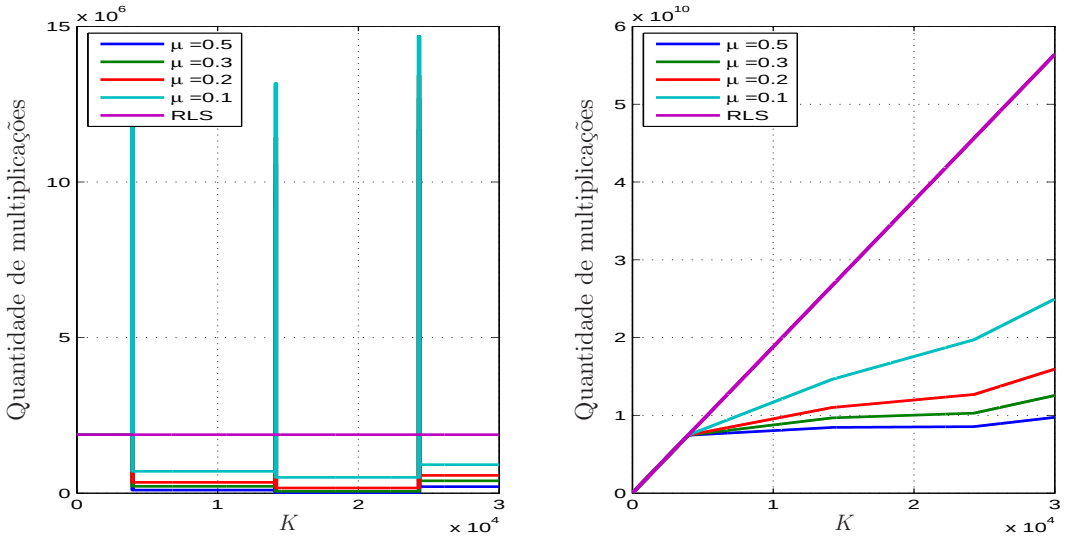


Figura 4.10: O algoritmo GLAR é acionado nos picos de complexidade computacional. Mesmo com mais coeficientes estimados ($\mu = 0{,}1$), o algoritmo GLAR-RLS possui complexidade computacional bem menor do que o algoritmo RLS quando comparado ao longo do tempo.

O algoritmo GLAR-RLS avaliado dinamicamente, com a carga computacional acumulada ao longo do tempo, é bem mais econômico devido à quantidade de coeficientes estimada reduzida. O ganho em complexidade computacional, definido pela diferença entre a complexidade computacional do algoritmo RLS e a complexidade computacional do algoritmo GLAR-RLS, é linear com o tempo. Em um computador

com 2GB de memória, o algoritmo RLS leva em torno de 2 horas para realizar um experimento completo de $k = 1$ a $k = 30.000$, enquanto que o algoritmo GLAR-RLS com $\mu = 0{,}2$ leva em torno de 30 minutos. O menor tempo do algoritmo GLAR-RLS reflete diretamente em economia de energia.

Para $\mu = 0{,}2$ a quantidade de coeficientes é acima daquela conhecida como ideal em $k \geq 10.000$ (Figura 4.9). A combinação do algoritmo GLAR-RLS com $\mu = 0{,}2$ com o filtro de Volterra de quinta ordem é usada para identificar o sistema simulado. A avaliação é feita com a média de 50 experimentos completos, de $k = 1$ a $k = 30.000$.

O resultado de MSE quando utilizado o algoritmo GLAR-RLS com $\mu = 0{,}2$ é comparado com o resultado do algoritmo RLS com fator de esquecimento $\lambda = 0{,}99$, na Figura 4.11. O algoritmo GLAR-RLS converge para o mínimo imposto pelo ruído de observação, enquanto que o algoritmo RLS converge com maior erro (diminuir o fator de esquecimento poderia melhorar o resultado do algoritmo RLS). A alta quantidade de coeficientes prejudica a convergência do algoritmo RLS. O algoritmo GLAR-RLS, após $k \geq L$, estima $N << J$ coeficientes, melhorando a estimação do vetor de coeficientes e diminuindo o MSE resultante.

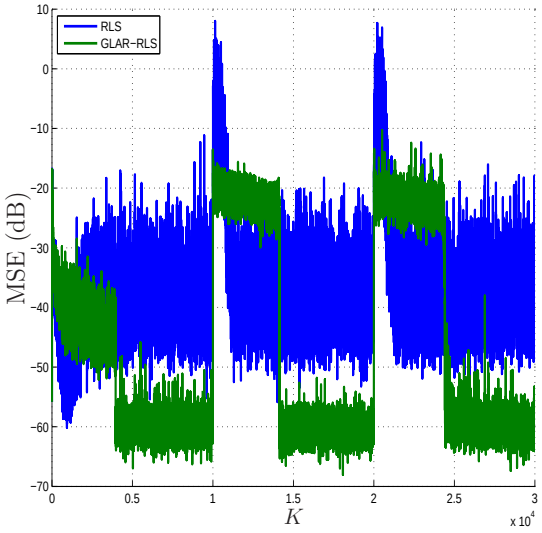


Figura 4.11: MSE, calculado com o erro *a priori*, utilizando o algoritmo GLAR-RLS com $\mu = 0{,}2$ e o algoritmo RLS com fator de esquecimento $\lambda = 0{,}99$.

O período em que o contador τ é contabilizado é facilmente identificado, em $10.000 \leq k \leq 14.100$ e em $20.000 \leq k \leq 24.300$. Neste período, o MSE resultante do algoritmo GLAR-RLS é muito degradado uma vez que o novo vetor de coeficientes ainda não foi estimado. Após o algoritmo GLAR ser acionado, o algoritmo GLAR-RLS é adaptado à nova situação do sistema e o resultado de MSE retorna a um valor próximo ao ruído de observação.

O erro estimado pelo algoritmo GLAR-RLS aproxima-se mais ao ruído de ob-

servação imposto pelo sistema pois indica corretamente quais são os coeficientes nulos, melhorando a estimação do vetor de coeficientes.

Em $k < L$, as respostas do algoritmo RLS e do algoritmo GLAR-RLS não são iguais devido ao fator de esquecimento aplicado ao algoritmo RLS. O algoritmo RLS apresenta maior variação do MSE pois estima $J >> N$ coeficientes, prejudicando o resultado. Com mais experimentos, entretanto, curvas com picos mais suaves são esperadas.

4.4 Conclusão do capítulo

Neste capítulo foi apresentada a segunda contribuição deste trabalho, o algoritmo GLAR-RLS.

O algoritmo GLAR-RLS, apresentado na Seção 4.2, emprega o algoritmo GLAR e o algoritmo RLS alternadamente. Uma vez que o algoritmo GLAR identifica os N coeficientes não-nulos do sistema, o algoritmo RLS é aplicado recursivamente no tempo, estimando $N << J$ coeficientes. O algoritmo GLAR-RLS identifica, com o auxílio de um contador τ, quando o sistema é modificado, acionando o algoritmo GLAR sempre que ocorre uma alteração brusca no vetor de coeficientes.

O algoritmo GLAR identifica corretamente quais coeficientes devem ser estimados recursivamente pelo algoritmo GLAR-RLS, enquanto que os coeficientes restantes permanecem iguais a zero. Isto melhora a estimação do vetor de coeficientes em sistemas esparsos.

O custo do algoritmo GLAR-RLS é visualizado por picos de complexidade computacional no momento em que o algoritmo GLAR é acionado. Avaliado dinamicamente, devido à estimação de $N << J$ coeficientes, o ganho em complexidade computacional do algoritmo GLAR-RLS é temporalmente linear quando comparado ao algoritmo RLS. O ganho em energia é mais evidente quanto maior a ordem do sistema não-linear analisado.

Foram avaliados diferentes cenários para a validação do algoritmo GLAR-RLS. Diversos tipos de modificações foram provocadas: na quantidade de coeficientes nulos no vetor de coeficiente; na equação de não-linearidade (tangente hiperbólica ou equação polinomial); e na magnitude dos coeficientes do sistema. Todas as alterações foram percebidas, acionando o algoritmo GLAR corretamente. O algoritmo GLAR-RLS resultou em um erro de estimação mais próximo ao imposto pelo ruído de observação quando comparado ao algoritmo RLS.

Parte dos resultados aqui apresentados foi submetida em [47].

Capítulo 5

Conclusões

Este trabalho, após uma revisão detalhada do algoritmo *Least Angle Regression*, apresenta duas contribuições. A primeira contribuição é o desenvolvimento de um critério de parada geométrico para o algoritmo LAR, levando à criação do algoritmo GLAR. A segunda contribuição é o algoritmo GLAR-RLS, que combina o algoritmo GLAR e o algoritmo RLS para estimação do vetor de coeficientes.

O critério de parada geométrico, desenvolvido para o algoritmo LAR, é baseado nos ângulos entre os vetores de coeficientes e o vetor de erro de estimação. A cada iteração n os ângulos estão mais próximos a 90^o. Quando todos os J coeficientes são estimados, o princípio da ortogonalidade é satisfeito e o ângulo entre cada vetor de dados de coeficiente e o vetor de erro de estimação é igual a 90^o. A cada iteração do algoritmo LAR, um novo coeficiente é adicionado ao conjunto ativo, i.e., conjunto de coeficientes estimados. Quando o critério de parada geométrico é satisfeito, o algoritmo é interrompido e a quantidade de coeficientes estimada N determinada pela iteração n atual. Assim, o algoritmo GLAR estima $N << J$ coeficientes.

O fator de penalidade do critério de parada geométrico é dado por μ. Nesta tese, o valor de μ foi determinado quando em $n = N$ o MSE era o mínimo imposto pelo ruído de observação. A partir da curva $N \times \mu$, o valor de μ, para o N encontrado, foi então obtido. De modo geral, pode-se dizer que o valor de μ é determinado pela quantidade de dados disponíveis e pela razão entre coeficientes não-nulos e nulos do sistema. Quanto menor a razão $\frac{N}{J-N}$, mais próximo a 1 é o valor de μ. Quanto menor o valor de μ (mais próximo a zero), maior a penalidade do critério de parada geométrico, ou seja, mais coeficientes são estimados.

A tendência a obter sistemas esparsos com o filtro de Volterra incentivou o seu uso em conjunto com o algoritmo GLAR para identificação de sistemas não-lineares. O algoritmo GLAR, em conjunto com filtros de Volterra de terceira ordem e quinta ordem, foi utilizado para identificação de sistemas não-lineares. Modelos LNL simularam sistemas com três tipos de não-linearidades. As funções não-lineares foram

dadas por polinômios de terceira ordem, polinômios de quinta ordem e a função tangente hiperbólica.

Para comparar o desempenho do algoritmo LAR com a quantidade de coeficientes estimada pelo critério de parada geométrico proposto, outros critérios de seleção de modelo amplamente conhecidos foram usados: *Akaike Information Criterion (AIC)*, *Bayesian Information Criterion* (BIC) e *Mallows* C_p. Com as duas ordens do filtro de Volterra utilizadas, a quantidade de coeficientes estimada pelo critério de parada geométrico foi a melhor resposta quando comparada com os critérios de seleção AIC, BIC e C_p. O ruído de observação imposto pelo sistema não prejudicou a determinação correta da quantidade de coeficientes estimada.

Em seguida, o algoritmo GLAR, em conjunto com o filtro de Volterra, foi avaliado quanto à estimação do vetor de coeficientes. Para comparar seus resultados, outros algoritmos foram também avaliados: *Least Squares* (LS), *Constrained Least Squares* (CLS) e *Subset Selection (SSS)*. Com exceção do algoritmo LS, todos os outros algoritmos necessitam de informação da resposta do algoritmo GLAR para saber quantos ou quais coeficientes são iguais a zero. O resultado inferior da norma da diferença entre o vetor de coeficientes estimado pelo algoritmo GLAR e o vetor de coeficientes ideal mostrou-se irrelevante ao ser comparado o MSE resultante.

Foi observado que, com o algoritmo GLAR em conjunto com um filtro de Volterra, é possível identificar os coeficientes mais relevantes do modelo de um sistema não-linear, independente do *kernel* Volterra, permitindo o uso de filtros com ordens mais elevadas. A partir dos resultados, identificou-se que com $K \geq 5J$ o vetor de coeficientes é estimado satisfatoriamente, resultando no MSE mínimo imposto pelo ruído de observação.

A identificação correta dos coeficientes não-nulos em diferentes sistemas incentivou o desenvolvimento do algoritmo GLAR-RLS. O algoritmo GLAR-RLS emprega o algoritmo GLAR e o algoritmo RLS alternadamente. O algoritmo GLAR identifica os N coeficientes não-nulos do sistema, fornecendo esta informação ao algoritmo RLS. O algoritmo RLS é aplicado recursivamente no tempo, estimando $N << J$ coeficientes. Os coeficientes não indicados pelo algoritmo GLAR permanecem iguais a zero, melhorando a estimação do vetor de coeficientes em sistemas esparsos.

O algoritmo GLAR-RLS percebe, com o auxílio de um contador τ, quando o sistema é modificado, acionando o algoritmo GLAR sempre que ocorre uma alteração no vetor de coeficientes. Ao ser acionado, o algoritmo GLAR indica uma nova quantidade de coeficientes N a ser estimada. Desta forma, o algoritmo GLAR-RLS possui consciência situacional e está apto a identificar modificações abruptas no sistema ao longo do tempo.

Diferentes modelos LNL foram simulados para validar o algoritmo GLAR-RLS. As modificações provocadas foram na quantidade de coeficientes nulos, na equação

de não-linearidade (tangente hiperbólica ou equação polinomial) e na magnitude dos coeficientes do sistema. Todas as alterações foram percebidas pelo algoritmo GLAR-RLS, acionando o algoritmo GLAR corretamente. O algoritmo GLAR-RLS resultou em um erro de estimação mais próximo ao imposto pelo ruído de observação quando comparado ao algoritmo RLS com fator de esquecimento $\lambda = 0{,}99$.

O custo do algoritmo GLAR-RLS é produzir picos de complexidade computacional nos momentos em que o algoritmo GLAR é acionado. Avaliado dinamicamente, no caso de sistemas esparsos onde $N << J$ coeficientes, a complexidade computacional do algoritmo GLAR-RLS é muito menor quando comparada ao algoritmo RLS. A diferença entre a complexidade computacional do algoritmo GLAR-RLS e do algoritmo RLS aumenta linearmente (em N) com o tempo. Quanto maior a esparsidade do sistema, melhor a contribuição do algoritmo GLAR-RLS quanto à complexidade computacional. O ganho em energia foi evidente quando o filtro de Volterra de quinta ordem foi utilizado para a identificação de um sistema não-linear, reduzindo em até 75% o tempo de execução.

O algoritmo GLAR-RLS em conjunto com o filtro de Volterra apresentou uma boa resposta na identificação dos sistemas não-lineares apresentados. Aplicar este esquema para a identificação de sistemas não-lineares reais é um tópico de pesquisa natural a ser seguido.

Analisar se é possível usar o algoritmo GLAR com outros filtros além do filtro de Volterra para identificação de sistemas não-lineares é outro tópico de pesquisa. Por exemplo, verificar se o algoritmo GLAR possui também um bom desempenho em conjunto com o filtro FLANN [48, 49] e o Polinômio de Legendre [50].

Alternativamente, o desempenho do algoritmo GLAR em sistemas não esparsos, como os dados da diabetes, também mostrou-se adequado; outras aplicações podem ser avaliadas.

O critério de parada geométrico aqui proposto pode ser testado como critério de parada de outros algoritmos. Algoritmos que podem ser obtidos a partir do algoritmo LAR, como o algoritmo LASSO [5], aqui apresentado, o algoritmo *Stagewise* [5], e o algoritmo *Homotopy* [38], devem obter bons resultados se interrompido com o critério de parada geométrico aqui proposto.

Por fim, uma avaliação matemática de μ pode ser tema de um trabalho futuro.

Referências Bibliográficas

[1] AGUIRRE, L. A. *Introdução à Identificação de Sistemas — Técnicas Lineares e Não-Lineares Aplicadas a Sistemas Reais*. Editora UFMG, 2007. ISBN: 9788570415844.

[2] ENDSLEY, M. "Situation awareness global assessment technique (SAGAT)". In: *Aerospace and Electronics Conference*, v. 3, pp. 789–795, maio 1998. doi: 10.1109/NAECON.1988.195097.

[3] LEE, A. N., LASTRA, J. L. M. "Enhancement of industrial monitoring systems by utilizing context awareness". In: *IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*, pp. 277–284, fev. 2013. ISBN: 978-1-4673-2437-3. doi: 10.1109/CogS.

[4] PREDEN, J. S., MOTUS, L., PAHTMA, R., et al. "Reducing bandwith requirements and optimizing data flow in distributed data acquisitions and processing". In: *IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*, pp. 175–182, fev. 2013. ISBN: 978-1-4673-2437-3. doi: 10.1109/CogSIMA.2013.6523844.

[5] EFRON, B., HASTIE, T., JOHNSTONE, I., et al. "Least Angle Regression". In: *Annals of Statistics, Stanford University*, v. 32, pp. 407 – 409, 2004. doi: 10.1214/009053604000000067.

[6] HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, fev. 2009. ISBN: 978-0387848570.

[7] ELAD, M. *Sparse and Redundant Representations*. Springer, 2010. ISBN: 978-1-4419-7010-7. doi: 10.1007/978-1-4419-7011-4.

[8] BERGHEN, F. V. "LARS Library: Least Angle Regression laSso/Stagewise Library". 2005. Disponível em:

<http://www.applied-mathematics.net/identification/lars.html>. Acessado em: 28 Jul. 2010.

[9] KHAN, J. A., AELST, S. V., ZAMAR, R. H. "Robust linear model selection based on Least Angle Regression", *Journal of the American Statistical Association*, v. 102, n. 480, pp. 1289–1299, 2007. doi: 10.1198/016214507000000950.

[10] HASTIE, T., TAYLOR, J., TIBSHIRANI, R., et al. "Forward stagewise regression and the monotone LASSO", *Electronic Journal of Statistics*, v. 1, pp. 1–29, mar. 2006. doi: 10.1214/07-EJS004.

[11] XI, L., HANG, D., MINGWEN, W. "Two-stage feature selection method for text classification". In: *International Conference on Multimedia Information Networking and Security*, v. 1, pp. 234 – 238, nov. 2009. doi: 10.1109/MINES.2009.127.

[12] KEERTHI, S. S. "Generalized LARS as an effective feature selection tool for text classification with SVMs". In: *22nd International Conference on Machine Learning*, pp. 417 – 424, 2005. doi: 10.1145/1102351.1102404.

[13] LI, X. "Finding deterministic solution from underdetermined equation: large-scale performance modeling by Least Angle Regression". In: *46th ACM/IEEE Design Automation Conference*, pp. 364 – 369, jul. 2009. ISBN: 978-1-6055-8497-3.

[14] XIAO, C. "Two-dimensional sparse principal component analysis for face recognition". In: *2nd International Conference on Future Computer and Communication*, v. 2, pp. 561 – 565, maio 2010. doi: 10.1109/ICFCC.2010.5497525.

[15] CHEN, C., CHANG, Y., RICANEK, K., et al. "Face age estimation using model selection". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 93 – 99, jun. 2010. doi: 10.1109/CVPRW.2010.5543820.

[16] MATHEW, V. J., SICURANZA, G. L. *Polynomial Signal Processing*. John Wiley and Sons, 2001. ISBN: 978-0471034148.

[17] DINIZ, P. S. R. *Adaptive Filtering: Algorithms and Practical Implementations*. Springer, 2008. ISBN: 978-1461441052.

[18] KUECH, F., KELLERMANN, W. "Proportionate NLMS algorithm for second-order Volterra filters and its application to nonlinear echo cancellation".

In: *International Workshop on Acoustic Echo and Noise Control*, Kyoto, Japan, set. 2003.

[19] STENGER, A., TRAUTMANN, L., RABENSTEIN, R. "Nonlinear acoustic echo cancellation with 2^{nd} order adaptive Volterra filters". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing*, v. 2, pp. 877 – 880, mar. 1999. doi: 10.1109/ICASSP.1999.759811.

[20] CHAN, S., STATHAKI, T., CONSTANTINIDES, A. G. "Adaptive weighted Least Squares algorithm for Volterra signal modeling", *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, v. 47, n. 4, pp. 545 – 554, abr. 2000. ISSN: 1057-7122. doi: 10.1109/81.841856.

[21] KAJIKAWA, Y. "The adaptive Volterra filter: its present and future". In: *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, v. 83, pp. 51 – 61, 2000. doi: 10.1002/1520-6440(200012)83:12⟨51::AID-ECJC6⟩3.0.CO;2-K.

[22] VALDMAN, C., DE CAMPOS, M. L. R., APOLINÁRIO JR., J. A. "Nonlinear system identification with LAR". In: *International Workshop on Telecomunications*, pp. 195–200. Instituto Nacional de Telecomunicações, maio 2011.

[23] VALDMAN, C., DE CAMPOS, M. L. R., APOLINÁRIO JR., J. A. "Nonlinear system identification with LAR", *Revista Telecomunicações*, v. 13, n. 02, pp. 12–21, dez. 2011. ISSN: 1516-2338.

[24] AKAIKE, H. "A new look at the statistical model identification", *IEEE Transactions on Automatic Control*, v. 19, n. 6, pp. 716–723, dez. 1974. ISSN: 0018-9286. doi: 10.1109/TAC.1974.1100705.

[25] SCHWARZ, G. "Estimating the dimension of a model". In: *The Annals of Statistics*, v. 6, pp. 461–464, 1978. doi: 10.2307/2958889.

[26] MALLOWS, C. L. "Some comments on C_p", *Technometrics*, v. 15, n. 4, pp. 661–675, nov. 1973. doi: 10.2307/1267380.

[27] ADKINS, L. C., MOOMAW, R. L., TIEN, J.-C. "Bayesian model selection: an application in urban economics". Southern Economics Association Meetings in New Orleans, LA, nov. 1999.

[28] MAZEROLLE, M. J. "Mouvements et reproduction des amphibiens en tourbières perturbées". Internet, jun. 2004. Disponível em: <archimede.bibl.ulaval.ca/archimede/fichiers/21842/apa.html>.

[29] WANG, L., LIBERT, G. A. "Combining patter recognition techniques with Akaike's information criteria for identifying ARMA models", *IEEE Transactions on Signal Processing*, v. 42, n. 6, pp. 1388–1396, jun. 1994. doi: 10.1109/78.286955.

[30] BURNHAM, K. P., ANDERSON, D. R. "Multimodel inference: understanding AIC and BIC in model selection". In: *Sociological methods & research*, v. 33, pp. 261–304, nov. 2004. doi: 10.1177/0049124104268644.

[31] JOHNSON, J. B., OMLAND, K. S. "Model selection in ecology and evolution", *Trends in Ecology and Evolution*, v. 19, n. 2, pp. 101–108, fev. 2004. ISSN: 0169-5347. doi: 10.1016/j.tree.2003.10.013.

[32] VALDMAN, C., DE CAMPOS, M. L. R., APOLINÁRIO JR., J. A. "A geometrical stopping criterion for the LAR algorithm". In: *20^{th} European Signal Processing Conference (EUSIPCO)*, pp. 2104–2108, ago. 2012. ISBN: 978-1-4673-1068-0.

[33] HEGDE, V., RADHAKRISHNAN, C., KRUSIENSKI, D., et al. "Series-cascade nonlinear adaptive filters". In: *45^{th} Midwest Symposium on Circuits and Systems*, v. 3, pp. III–219 – III–222, ago. 2002. doi: 10.1109/MWSCAS.2002.1187010.

[34] DOYLE, P. F. J., PEARSON, R. K., OGUNNAIKE, B. A. *Identification and control using Volterra models*. Springer, 2002. ISBN: 978-1-85233-149-8.

[35] ALDRICH, J. "Doing least squares: perspectives from Gauss and Yule". In: *International Statistical Review*, v. 66, pp. 61–81, 1998. doi: 10.1111/j.1751-5823.1998.tb00406.x.

[36] HAYKIN, S. *Adaptative Filter Theory*. Prentice Hall, 1996. ISBN: 978-0133979855.

[37] TIBSHIRANI, R. "Regression shrinkage and selection via the Lasso", *Journal of the Royal Statistical Society (Series B)*, v. 58, n. 1, pp. 267 – 288, 1996.

[38] DONOHO, D., TSAIG, Y. "Fast solution of l_1-norm minimization problems when the solution may be sparse", *IEEE Transactions on Information Theory*, v. 54, n. 11, pp. 4789–4812, nov. 2008. ISSN: 0018-9448. doi: 10.1109/TIT.2008.929958.

[39] OSBORNE, M. R., PRESNELL, B., TURLACH, B. A. "A new approach to variable selection in least squares problems", *IMA journal of numerical*

analysis, , n. 3, pp. 389–403, 2000. ISSN: 0272-4979. doi: 10.1093/imanum/20.3.389.

[40] OSBORNE, M. R., PRESNELL, B., TURLACH, B. A. "On LASSO and its dual", *Journal of Computational and Graphical Statistics*, v. 9, n. 2, pp. 319–337, 2000. doi: 10.2307/1390657.

[41] SEWELL, M. "Model selection". Internet. Disponível em: <http://www.modelselection.org/model-selection.pdf>. Acessado em: 15 Abr. 2007.

[42] BEAL, D. J. "Information criteria methods in SAS for multiple linear regression models". In: *SESUG 2007: SouthEast SAS Users Group*, n. SA05. Institute for Advanced Analytics, 2007.

[43] DRAKOS, N. "Matrix inverse in block form". University of Leeds, mar. 1998. Disponível em: <http://www.cs.nthu.edu.tw/~jang/book/addenda/matinv/matinv/>. Acessado em: 13 Jan. 2014.

[44] KREYSZIG, E. *Advanced Engineering Mathematics*. John Wiley and Sons, Inc., 1972.

[45] WEISSTEIN., E. W. "Hyperbolic tangent". MathWorld–A Wolfram Web Resource. Disponível em: <http://mathworld.wolfram.com/HyperbolicTangent.html>. Acessado em: 23 Dez. 2013.

[46] VALDMAN, C., DE CAMPOS, M. L. R., APOLINÁRIO JR., J. A. "A geometrical stopping criterion for the LAR algorithm applied to nonlinear systems identification", *IEEE Signal Processing Letters*, 2014. ISSN: 1070-9908. Artigo submetido.

[47] VALDMAN, C., DE CAMPOS, M. L. R., APOLINÁRIO JR., J. A. "RLS sparse system identification using LAR-based situtational awareness". In: *22nd European Signal Processing Conference (EUSIPCO)*, 2014. Artigo submetido.

[48] SICURANZA, G. L., CARINI, A. "A generalized FLANN filter for nonlinear active noise control", *IEEE Transactions on Audio, Speech, and Language Processing*, v. 19, n. 8, pp. 2412–2417, nov. 2011. ISSN: 1558-7916. doi: 10.1109/TASL.2011.2136336.

[49] DAS, D. P., PANDA, G. "Active mitigation of nonlinear noise processes using a novel filtered-s LMS algorithm", *IEEE Transactions on Speech and Audio Processing*, v. 12, n. 3, pp. 313–322, maio 2004. ISSN: 1063-6676. doi: 10.1109/TSA.2003.822741.

[50] PATRA, J. C., MEHER, P. K., CHAKRABORTY, G. "Nonlinear channel equalization for wireless communication systems using Legendre neural networks", *Signal Processing*, v. 89, n. 11, pp. 2251–2262, 2009. ISSN: 0165-1684. doi: 10.1016/j.sigpro.2009.05.004.