Journal homepage http://revistas.unitru.edu.pe/index.php/SSMM

SELECCIONES MATEMÁTICAS
Universidad Nacional de Trujillo
ISSN: 2411-1783 (Online)
2022; Vol. 9(2): 312-322.

Subspace identification of Wiener model and linear systems

Identificação por subespaços para modelo de Wiener e sistemas lineares

Santos D. M. Borjas

Received, Jun. 15, 2022 *Accepted, Nov. 13, 2022*

How to cite this article:
Borjas SD. *Subspace identification of Wiener model and linear systems*. Selecciones Matemáticas. 2022;9(2):312–322.
http://dx.doi.org/10.17268/sel.mat.2022.02.08

Abstract

This article presents an overview of the problem of identifying linear and nonlinear systems (Wiener type) operating in open loop. There are several algorithms that solve this problem, among them are the subspace methods (MOESP and N4SID). In order to evaluate the performance of the MOESP and N4SID methods, a simulated example is presented to compare the performance of these algorithms.

Keywords. System identification, structure of Wiener, subspace identification.

Resumo

Neste artigo é apresentada uma visão geral sobre o problema da identificação de sistemas lineares e não lineares (tipo Wiener) operando em malha aberta. Existem diversos algoritmos que solucionam este problema, entre eles encontram-se os métodos por subespaços (MOESP e N4SID). Com o objetivo de avaliar o desempenho dos métodos MOESP e N4SID, um exemplo simulado é apresentado para comparar o desempenho destes algoritmos.

Palavras-chave . Identificação de sistemas, estrutura de Wiener, identificação por subespaços.

1. Introdução. Na atualidade, cada vez mais, o trabalho de um engenheiro consiste na obtenção de modelos matemáticos dos processos estudados [8]. O campo de utilização desses modelos é muito amplo, destacando-se aplicações como: controle, supervisão, predição, simulação, otimização, etc. As técnicas de identificação de sistemas têm evoluído bastante, procurando atender essa demanda por modelos cada vez mais precisos. Existem métodos de identificação como métodos de predição do erro (Prediction Error Methods - PEM) e os métodos das variáveis instrumentais (Instrumental Variable Methods - IVM) que são muitos populares. Recentemente, os métodos de identificação por subespaços têm emergido como uma alternativa para os métodos tradicionais. Nos métodos de identificação de sistemas dinâmicos por subespaços são tratados modelos de sistemas lineares invariantes no tempo em espaço de estados operando em tempo discreto. Pelas restrições citadas, pode parecer uma classe altamente restrita de modelos (especialmente por serem lineares), no entanto é bastante surpreendente como muitos processos industriais podem ser descritos com precisão por este tipo de modelo [3], [4], [5] e [6] [11]. A maioria dos processos industriais é não linear, mas a planta pode ser identificada com modelos lineares, se eles forem linearizados em torno de um ponto de operação. No entanto, há sistemas não lineares que têm dinâmica complexa, não sendo possível aproximá-los por sistemas lineares. Um modo de contornar este problema é restringir a operação do processo em certa faixa, em que o sistema tenha comportamento linear. Um caso típico são as válvulas de controle. A necessidade de descrever de forma mais precisa o comportamento não linear de sistemas reais levou à busca de representações não lineares usando métodos de séries funcionais, como as séries de Volterra. A desvantagem delas é a complexidade computacional, a dificuldade de incorporar informações a priori e de interpretar e estimar características físicas do processo a partir do modelo [2]. Como opção para esses problemas, Billings (1980) propôs os sistemas orientados a blocos. Billings (1980) sugeriu novas

*Departamento de matemática, Universidade do Rio Grande do Norte, Natal, Brasil. (santos.miranda@ccet.ufrn.br).

pesquisas para simplificar e estender a aplicação de tais métodos. A identificação de sistemas para modelos não lineares orientados a blocos do tipo Hammerstein ou Wiener vem sendo pesquisada nos últimos anos [1] e [9]. Recentemente, os métodos de identificação por subespaços têm sido propostos para identificar modelos do tipo Wiener e Hammerstein [7].

Este artigo busca avaliar o desempenho dos métodos MOESP (Multivariable Output-Error State sPace) e N4SID (Numerical algorithms for Subspace State Space System Identification), na identificação de sistemas determinísticos operando em malha aberta para o caso linear e não linear.

2. Identificação Determinística de Sistemas Lineares Usando o Método de Subespaços. Considere um sistema linear discreto invariante no tempo dado por:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k), \\ y(k) &= Cx(t) + Du(k), \end{aligned} \tag{2.1}$$

onde $u(k) \in \mathbb{R}^m$ e $y(k) \in \mathbb{R}^l$ são, respectivamente os valores medidos das entradas e saídas no instante k dos processos com m entradas e l saídas. $x(k) \in \mathbb{R}^n$ é o vetor de estados do processo em tempo discreto no instante k. A, B, C, e D são matrizes de dimensões apropriadas..

2.1. Problema de identificação. Suponha que os dados de entrada e saída $u(k) \in \mathbb{R}^m$ e $y(k) \in \mathbb{R}^l$ sejam dados. O problema é identificar a dimensão n do sistema e as matrizes (A, B, C, D) do sistema (2.1)[10].

2.2. Solução Ideal. O sistema (2.1) pode ser representado na forma matricial [3]:

$$\begin{bmatrix} X_{k+1} \\ Y_k \end{bmatrix} = \Theta \begin{bmatrix} X_k \\ U_k \end{bmatrix}, \tag{2.2}$$

onde $\Theta = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ corresponde aos parâmetros desconhecidos. A equação (2.2) pode ser interpretada como um modelo de regressão. Se na equação (2.2) as matrizes X_{k+1}, Y_k, X_k e U_k são dadas, então o parâmetro desconhecido Θ pode ser calculado pelo método dos mínimos quadrados, isto é:

$$\hat{\Theta} = \begin{bmatrix} \hat{A} & \hat{B} \\ \hat{C} & \hat{D} \end{bmatrix} = \min_{A,B,C,D} \left|\left| \begin{bmatrix} X_{k+1} \\ Y_k \end{bmatrix} - \Theta \begin{bmatrix} X_k \\ U_k \end{bmatrix} \right|\right|_F^2; \tag{2.3}$$

onde $\hat{\Theta}$ denota a estimativa de Θ e $||.||_F^2$ denota a norma de Frobenius de uma matriz. Da equação (2.3), resulta:

$$\hat{\Theta} = \begin{bmatrix} X_{k+1} \\ Y_k \end{bmatrix} \begin{bmatrix} X_k \\ U_k \end{bmatrix}^T \left[\begin{bmatrix} X_k \\ U_k \end{bmatrix} \begin{bmatrix} X_k \\ U_k \end{bmatrix}^T \right]^{-1}. \tag{2.4}$$

Então em um caso ideal, quando se têm os dados de entrada, saída e a sequência de estados para dois instantes de tempo sucessivos k e $k+1$, a identificação do parâmetro Θ na equação (2.2) é trivial. No entanto, na prática, X_k e X_{k+1} não são conhecidos e têm que ser estimados a partir dos dados de entrada e saída. Isto é um ponto importante nos métodos de identificação por subespaços. A diferença entre estes métodos reside na forma como obter a sequência de estados estimados.

2.3. Equações matriciais por subespaços. Fazendo iterações sucessivas no sistema (2.1) obtém-se a seguinte equação matricial:

$$Y_f = \Gamma_i X_f + H_i U_f, \tag{2.5}$$

onde a matriz U_f é definida de forma similar à matriz Y_f. As matrizes Y_f e H_i são definidas por:

$$Y_f = \begin{bmatrix} y_i & y_{i+1} & \cdots & y_{i+j-1} \\ y_{i+1} & y_{i+2} & \cdots & y_{i+j} \\ \vdots & \vdots & \ddots & \vdots \\ y_{2i-1} & y_{2i} & \cdots & y_{2i+j-2} \end{bmatrix}, \tag{2.6}$$

$$H_i = \begin{bmatrix} D_i & 0 & \cdots & 0 \\ CB & D & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{i-2}B & CA^{i-3} & \cdots & D \end{bmatrix}. \tag{2.7}$$

O número de colunas em Y_f e U_f é $j = N - 2i + 1$, onde N representa a quantidade de dados e i é o número de linhas definido pelo usuário (por exemplo $i = 10$). $\Gamma_i \in \mathbb{R}^{lixn}$ é a matriz de observabilidade estendida e é definida por:

$$\Gamma_i^T = \begin{bmatrix} (C)^T & (CA)^T \dots (CA^{i-1})^T \end{bmatrix}^T, \tag{2.8}$$

onde $(*)^T$ denota a transposta da matriz $(*)$. $X_p = X_0 = \begin{bmatrix} x_0, & \cdots & x_{j-1} \end{bmatrix}$ e $X_f = X_i = \begin{bmatrix} x_i, & \cdots & x_{i+j-1} \end{bmatrix}$ representam os estados passados e futuros, respectivamente, sendo que o símbolo p denota dados passados e f dados futuros.

2.4. Projeção ortogonal e projeção oblíqua. A projeção ortogonal do espaço linha de A sobre o espaço linha de B é [10]:

$$A/B = AB(BB^T)^\dagger B, \tag{2.9}$$

onde $(*)^\dagger$ denota a pseudo-inversa da matriz $(*)$.
A projeção oblíqua do espaço de linhas de G no espaço de linhas de H sobre o espaço de linhas de J é [10]:

$$G/_H J = [G/H^\perp].[J/H^\perp]^\dagger.J, \tag{2.10}$$

onde $(*)^\perp$ denota o complemento ortogonal da matriz $(*)$. Propriedades da projeção ortogonal e projeção oblíqua:

$$A_x/A_x^\perp = 0, \tag{2.11}$$

$$A_x/_{A_x} C_x = 0. \tag{2.12}$$

Para a prova, ver [10].

2.5. Método MOESP. A seguir é apresentado o método MOESP básico, do qual um grande número de variações foi criado para diferentes tipos de problemas [12]. O objetivo deste método é computar Γ_i a partir da equação (2.5), então aplicando a projeção ortogonal da equação (2.5) sobre o espaço linha de $U_f^\perp$ resulta:

$$Y_f/U_f^\perp = \Gamma_i X_f/U_f^\perp + H_i U_f/U_f^\perp. \tag{2.13}$$

Pela equação (2.11) tem-se $U_f/U_f^\perp = 0$. Então a equação (2.13) pode ser simplificada:

$$Y_f/U_f^\perp = \Gamma_i X_f/U_f^\perp. \tag{2.14}$$

A equação (2.14) indica que o espaço coluna de Γ_i pode ser estimado pela SVD de $Y_f/U_f^{\perp}$ [12]. Γ_i pode ser estimada da fatoração LQ a partir dos dados de entrada e saída, na forma:

$$\begin{bmatrix} U_k \\ Y_k \end{bmatrix} = \begin{bmatrix} R_{11} & 0 \\ R_{21} & R_{22} \end{bmatrix} \begin{bmatrix} Q1 \\ Q2 \end{bmatrix}. \tag{2.15}$$

A projeção ortogonal do lado esquerdo de (2.14) pode ser computada a partir da matriz R_{22}. A SVD da matriz R_{22} é [12]:

$$\begin{bmatrix} R_{22} \end{bmatrix} = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} Sn & 0 \\ 0 & S_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} = USV. \tag{2.16}$$

A ordem n do sistema é igual ao número de valores singulares não nulos em S. O espaço coluna de U_1 aproxima consistentemente a matriz Γ_i [12]:

$$\Gamma_i \approx U_1. \tag{2.17}$$

As matrizes A e C são obtidas de $C = \Gamma_i(1:l, 1:n)$ e $\Gamma_i(1:l(i-1), 1:n)A = \Gamma_i(l+1:il, 1:n)$, respectivamente. Para obter as matrizes B e D ver [12]. Este método pode ser resumido no seguinte algoritmo.

Algoritmo MOESP (Determinístico)

1. Construir as matrizes de Hankel U_k e Y_k.
2. Calcular a fatoração LQ dos dados de entrada e saída, equação (2.15).
3. Calcular a SVD da matriz R_{22} dada na equação (2.16).
4. Determine a ordem do sistema por inspeção dos valores singulares em S_n e particionar o SVD para obter U_1.
5. Determine as matrizes A e C das equações $C = \Gamma_i(1:l, 1:n)$ e $\Gamma_i(1:l(i-1), 1:n)A = \Gamma_i(l+1:il, 1:n)$, respectivamente.
6. As matrizes B e D são determinadas da equação, (com $K = \left(U_1^{\perp}\right)^T R_{21} R_{11}^{-1}$)

$$\begin{bmatrix} K(:, 1:m) \\ K(:, m+1:2m) \\ \vdots \\ K(:, m(i-1)+1:mi) \end{bmatrix} = \begin{bmatrix} U_1^{\perp}(1:l,:)^T & \cdots & U_1^{\perp}(l(i-1)+1:li,:))^T \\ U_1^{\perp}(1+l:2l,:)^T & \cdots & 0 \\ \vdots & \vdots & \vdots \\ U_1^{\perp}(l(i-1)+1:li,:)^T & 0 & 0 \end{bmatrix} \begin{bmatrix} I_l & 0 \\ O & U_1 \end{bmatrix} \begin{bmatrix} D \\ B \end{bmatrix}.$$

Neste trabalho, a entrada $u(k)$ do sistema é considerada um sinal de entrada persistentemente exitante de ordem 2i ver [8]. As matrizes D e B existem se $posto(U_1^{\perp}(l(i-1)+1:li,:)) = l$ ver [12]

2.6. Método N4SID. O método N4SID soluciona o problema de identificação determinística recuperando os estados passados e futuros do sistema desconhecido. A sequência de estados X_f pode ser expressa por $X_f = L_p W_p$ como combinação linear das entradas passadas e saídas passadas [10]. Logo, substituindo-se $X_f = L_p W_p$ na equação (2.5) tem-se:

$$Y_f = \Gamma_i L_p W_p + H_i U_f. \tag{2.18}$$

Aplicando-se a projeção ortogonal da equação (2.18) sobre o espaço linha de $U_f^{\perp}$ resulta:

$$Y_f/U_f^{\perp} = \Gamma_i L_p W_p/U_f^{\perp}. \tag{2.19}$$

Multiplicando-se a equação (2.19) por $[W_p/U_f^{\perp}]^{\dagger}.W_p$ de ambos os lados,

$$Y_f/U_f^{\perp}[W_p/U_f^{\perp}]^{\dagger}.W_p = \Gamma_i L_p W_p/U_f^{\perp}[W_p/U_f^{\perp}]^{\dagger}.W_p. \tag{2.20}$$

Como $W_p = W_p/U_f^{\perp}[W_p/U_f^{\perp}]^{\dagger}$ e usando a equação (2.10) tem-se a projeção oblíqua Θ_i definida por [10];

$$\Theta_i = Y_f/_{U_f^{\perp}} W_p = \Gamma_i X_f. \tag{2.21}$$

De forma similar é obtido Θ_{i-1}, ver [10]. Θ_i dado na equação (2.21) pode ser computado da fatoração LQ a partir dos dados de entrada e saída, colocados na forma $[U_p^T U_f^T Y_p^T Y_f^T]^T$. A equação (2.21) indica que o espaço coluna de Γ_i pode ser estimado pela SVD de Θ_i, então resulta:

$$\Gamma_i \approx U_1 S_1^{1/2}. \tag{2.22}$$

Usando a equação (2.22) computa-se $X_i = \Gamma_i^{\dagger}\Theta_i$ e $X_{i+1} = \Gamma_{i-1}^{\dagger}\Theta_{i-1}$, logo as matrizes do sistema são estimadas da equação (2.23) [10]:

$$\begin{bmatrix} X_{i+1} \\ Y_i|_i \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} X_i \\ U_i|_i \end{bmatrix}. \tag{2.23}$$

Este método pode ser resumido no seguinte algoritmo.

Algoritmo N4SID (Determinístico)

1. Construir as matrizes de Hankel U_k e Y_k.
2. Calcular as projeções oblíquas $\Theta_i = Y_f/_{U_f^{\perp}} W_p$ e $\Theta_{i-1} = Y_f^-/_{U_f^-} W_p^+$.
3. Calcule a SVD da projeção oblíqua ponderada $W_1\Theta_i W_2 = USV^T$.
4. Determine a ordem do sistema por inspeção dos valores singulares em S e encontre U_1 e S_1, através do particionamento adequado da SVD.
5. Determine Γ_i e Γ_{i-1} onde $\Gamma_i = W_1^{-1}U_1 S_1^{1/2}$ e $\Gamma_{i-1} = \underline{\Gamma_i}$, onde $\underline{\Gamma_i}$ denota a matriz Γ_i sem as l linhas passadas de Γ_i.
6. Determine $X_i = \Gamma_i^{\dagger}\Theta_i$ e $X_{i+1} = \Gamma_{i-1}^{\dagger}\Theta_{i-1}$.
7. Solucione o sistema de equações lineares, equação (2.23), para obter as matrizes A,B,C e D.

No método N4SID é necessário computar duas projeções oblíquas para recuperar as matrizes do sistema, portanto, é necessário ter duas condições inicias, isto pode levar a problemas de polarização. O número de linhas i da matriz de Hankel dos dados de entrada e saída, usados no método MOESP e N4SID, é dado pelo usuário e esta diretamente relacionada com o número de dados coletados. Idealmente, se os dados são infinitos, o sistema identificado simula muito bem o sistema real, mas na prática isso não acontece.

3. Identificação Determinística de Sistemas não Lineares Usando o Método por Subespaços. Os modelos por blocos interconectados são estruturas eficientes na modelagem de sistemas não lineares. Uma estrutura de Wiener é composta por um bloco linear dinâmico e um bloco não linear estático, conforme indicado na Figura 3.1.

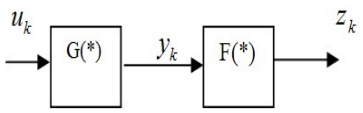

Figura 3.1: Modelo de Wiener.

Onde u_k representa o sinal de entrada, z_k representa o sinal de saída do sistema, $G(*)$ representa o bloco dinâmico linear e $F(*)$ o bloco estático não-linear. O modelo em espaços de estados de um sistema de Wiener é representado pela equação (2.1) e pela equação $z_k = F(y_k) + v_k$. v_k é ruído branco com média zero. Dado um conjunto de dados de entradas u_k e saídas z_k, o problema da identificação não linear consiste em determinar uma estimativa para $G(*)$ e $F(*)$.

4. Simulações e resultados.

4.1. Caso Linear: Modelo em Espaço de Estados com Matrizes Aleatórias. Nesta seção apresenta-se um modelo simulado usado na identificação por subespaços aplicado a um sistema linear. A função drss do Matlab 7.0 permite gerar um modelo linear discreto em espaços de estados, com matrizes do sistema A, B, C e D na forma aleatória. A ordem escolhida do modelo é 4, que é igual ao posto (A). Gerou-se um sistema MIMO de duas entradas e duas saídas, na forma:M=idss(drss(4,2,2)); $[A, B, C, D]$=th2ss(M); (recuperando as matrizes do sistema); u=idinput([1000,2],'prbs',[0 0.3]); y=sim(M,u). Para o modelo M com sinal de entrada u, foram coletados 1000 dados, dos quais 700 foram aplicados para identificação e o restante para validação. Os sinais pré-tratados através do comando "detrend"do Matlab usados na identificação são mostrados na Figura 4.1.

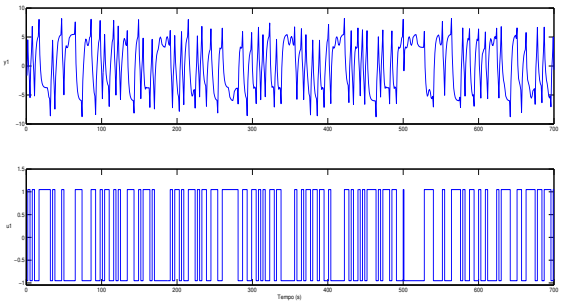

Figura 4.1: Sinal de saída y_1 e entrada u_1 usadas na identificação.

Neste trabalho é usado dois modelos MOESP e N4SID. O modelo N4SID se encontra implementado no Toolbox do Matlab e o método MOESP é implementado por Michael Verhaegen. O passo seguinte é encontrar o melhor modelo que simule o processo M. Isto é mostrado na Tabela 4.1. A ordem $n = 6$ do sistema identificado é dada pelos valores singulares mais significativos da matriz S, a qual é obtida da decomposição em valores singulares (SVD) de certas projeções oblíquas ou ortogonais, dependendo do algoritmo sendo aplicação [3]. Para se avaliar a qualidade do modelo, aplicam-se indicadores de desempenho. Neste trabalho empregou-se a média da variância relativa (MVAF), definido por:

$$MVAF(\%) = \frac{1}{l}\sum_{i=1}^{N}(1 - \frac{Var(z - \hat{z})}{Var(z)}).100.$$

Onde z é a saída real e $\hat{z}$ é a saída estimada pelo modelo obtido. O índice MVAF é usado pelo SMI toolbox do Matlab. Este índice de desempenho é empregado para se avaliar a qualidade do modelo produzido por cada algoritmo, como mostra a Tabela 4.1.

Algoritmo	Tempo (s)	Validação (%)
MOESP	0.297	100
N4SID	0.844	100

Tabela 4.1: Resultados numéricos do Desempenho dos algoritmos.

Analisando-se os valores da Tabela 4.1, todos os modelos tiveram um bom desempenho em termos de validação. Verifica-se que o tempo de processamento para a obtenção do modelo é menor para o método MOESP, motivo pelo qual se optou por esse método para identificar o processo. A ordem do sistema real não é necessariamente a mesma do modelo obtido, portanto em geral as matrizes do sistema real e do modelo obtido têm diferentes dimensões.

4.2. Caso não Linear. O detalhamento deste exemplo simulado pode ser encontrado em [13]. a função $G(*)$ e $F(*)$ é dada por:

$$G(z) = \frac{0.1578z + 0.1379}{z^2 - 1.375z + 0.6703},$$

$$F(y) = \begin{cases} 1.5 & y > 1.5, \\ y & 0.5 \leq 0 \leq 1.5, \\ 0.5 & y < 0.5; \end{cases}$$

Dada as funções $G(*)$ e $F(*)$, o passo seguinte é coletar os dados de entrada e saída. Estes dados são gerados usando os seguintes comandos do Matlab:
N=1800; (número de dados a coletar); $u = 1 + randn(n, 1); y = dlsim(num, dem, u);$
$z = (y < .5) * .5 + ((y >= .5)\ \&\ (y <= 1.5)). * y + (y > 1.5) * 1.5.$

Foram coletados 1800 pontos de entrada u e saída y, dos quais 1400 foram usados na identificação e o restante na validação. Agora deve-se encontrar um modelo que determine a melhor estimativa para a função $G(*)$. Neste trabalho são usados os modelos MOESP, N4SID e NARX. NARX se encontram no Toolbox do Matlab. A ordem do sistema é dada pelos valores mais significativos da matriz S, a qual é obtida da SVD de certas projeções oblíquas ou ortogonais, dependendo do algoritmo sendo aplicado [3]. O passo seguinte é estimar a função não linear $F(*)$, usando a saída estimada do sistema linear e a saída z_k do modelo de Wiener. Esta estimativa é dada pelos polinômios de Chebyshev, para $n = 13$.

Esta metodologia pode ser resumido nos seguintes passos:

1. Dado um conjunto de dados de entrada u_k e saída z_k determine uma estimativa para a função $G(*)$.
2. Dada a sinal de entrada u_k e uma estima do sistema linear $G(*)$, determine uma estimativa para $y_k \approx \widetilde{y_k}$.
3. Dado o sinal estimado $\widetilde{y_k}$ e o sinal de saída z_k, determine uma estimativa para função não linear $F(*)$.

O índice MVAF é usado para avaliar a qualidade do modelo produzido por cada algoritmo, como mostra a Tabela 4.2.

Algoritmo	Autovalidação (%)	Validação Cruzada (%)
MOESP	99.6716	99.6416
N4SID	99.4703	99.4137
NARX	97.8184	96.779

Tabela 4.2: Resultados numéricos do desempenho dos algoritmos.

Através dos dados apresentados na Tabela 4.2, percebe-se que o modelo MOESP superou os modelos N4SID e NARX na autovalidação e na validação cruzada, motivo pelo qual se utiliza este modelo para identificar o sistema de Wiener. A relação entre a saída estimada linear e a não linear $F(*)$ é mostrada na Figura 4.2.

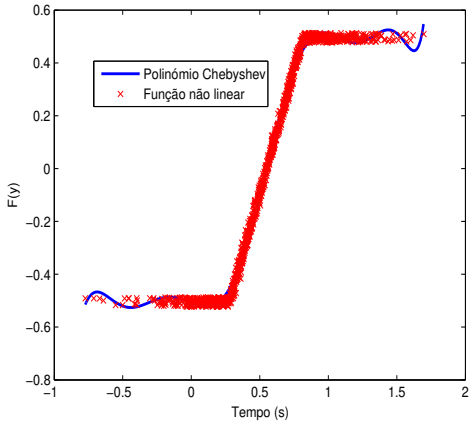

Figura 4.2: Sinal de saída y_1 e entrada u_1 usadas na identificação.

A linha contínua azul representa o polinômio de Chebyshev para n=13. Da Figura 4.2 observa-se que

o modelo identificado reproduz muito bem as principais características do processo. Foram consideradas condições iniciais nulas.

5. Conclusões. Neste trabalho foi mostrada uma visão geral do problema da identificação por subespaços de sistemas lineares e não lineares (caso determinístico). Dois exemplos simulados, um para o caso linear e outro para o caso não linear são apresentados. Para estes exemplos simulados, o método MOESP teve melhor desempenho na identificação em relação ao modelo N4SID, de acordo com o critério MVAF.

ORCID and License
Santos D. M. Borjas https://orcid.org/0000-0003-3001-9266

This work is licensed under the Creative Commons - Attribution 4.0 International (CC BY 4.0)

Referências

[1] Biagiola S, Figueroa L. Identification of uncertain MIMO Wiener and Hammerstein models, Computers & Chemical Engineering. 2011; 35(11): 1-9.
[2] Billings S. Identification of nonlinear systems: a survey. Proc. IEEE, Part D, 1980; 272-285.
[3] Borjas SD, Garcia C. Subspace identification using the integration of MOESP and N4SID methods applied to the Shell benhmark of a distillation column, TEMA. 2011; 12(3):183-194.
[4] De Moor B, Van Overschee P, Favoreel M. Algorithms for subspace state space system identification - an overview, Appl. and comp. control, signal and circuits. 1999; 1:247-311.
[5] Garg, A, Mhaskar, P. Subspace identification-based modeling and control of batch particulate processes. Ind. Eng. Chem. Res. 2017; 56(26):7491
[6] Jin N, Dertimanis V, Chatzi E, Dimitrakopoulos E, Katafygiotis L. Subspace identification of bridge dynamics via traversing vehicle, Journal of Sound and Vibration. 2022; 523(116690):1-20.
[7] Lawrynczuk M. Nonlinear predictive control for Hammerstein Wiener systems. ISA Transactions. 2015; 55:49-62.
[8] Ljung L. System Identification it Theory for the User, 2a. Ed., Prentice Hall Englewood Cliffs: NJ, 1999.
[9] Saha P, Krishnan S, Rao V, Patwardhan S. Modeling and predictive control of MIMO nonlinear systems using Wiener-Laguerre models. Chem. Eng. Communication. 2004; 191:1083-1119.
[10] Van Overschee P, De Moor B. Subspace Identification for Linear Systems. Kluwer Academic Pub, 1996.
[11] Viberg M. Subspace methods in systems identification, 10th IFAC Symposium on System Identification, SYSID. 94, Copenhagen, Denmark, Proceedings. 1994; 1:1-12.
[12] Verhaegen M, Dewilde P. Subspace model identification. part i: the output-error state-space model identification class of algorithms, Int. J. of Control, 1992; 56(1):1187-1210.
[13] Wigren T. Convergence analysis of recursive identification algorithms based on the nonlinear Wiener model. IEEE Trans. AC, 1994; 39:2191-2206.

Apêndice A. Algoritmo MOESP (caso determinístico).

```
%*****************************************************
%                ALGORITMO LAZO ABERTO
%                    ORDINARIO MOESP
%*****************************************************
% Sistema               x(k+1) = A x(k) + B u(k)
%                         y(k) = C x(k) + D u(k)

% Dados de entrada u=[u1 u2...uN];   m = numero de entradas
% Dados de saida   y=[y1 y2 ...yN];  L = numero de saidas
% i tem que ser i > n  onde n = ordem do sistema
% N: Numero de dados coletados

M=idss(drss(4,2,2));
% [A,B,C,D]=th2ss(M); matrizes do sistema real simulado
u=idinput([1000,2],'prbs',[0 0.3]);  % dados de entrada coletados
y=sim(M,u);                          % dados de saida coletados

u=detrend(u); y=detrend(y);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     Graficando os  dados de entrada e saida
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plotando os dados de entrada e saida usados na identificação
idplot([y(1:700,1) u(1:700,1)]) ;  % entrada e saida 1
```

```
idplot([y(1:700,2) u(1:700,2)]);  % entrada e saida 2

[m,N]=size(u); if m > N; u=u'; end; [m,N]=size(u);
[L,N]=size(y); if L > N; y=y'; end; [L,N]=size(y);

N=length(y); %
i=10;       % Numero definido pelo usuario
j=N-2*i+1;  % numero de colunas da matriz da Hankel

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% matriz de Hankel das entradas UT  e Saidas YT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
UT=zeros(2*i*L,j);
for s=1:2*i;
    UT((s-1)*L+1:s*L,:)=u(:,s:s+j-1);
end

YT=zeros(2*i*m,j);
for s=1:2*i;
    YT((s-1)*m+1:s*m,:)=y(:,s:s+j-1);
end

U=UT(m*i+1:2*m*i,:);   %Entradas futuras
Y=YT(L*i+1:2*L*i,:);   %saidas futuras
im=size(U,1); iL=size(Y,1);
r=triu(qr([U;Y]'))';

 L11=r(1:im,1:im);
 L21=r(im+1:im+iL,1:im);
 L22=r(im+1:im+iL,im+1:im+iL);
 [Un,ss,Vn]=svd(L22);
 SS=diag(ss);
 SS=SS(1:i); semilogy(SS,'rx'); title('valores singulares');

 n = input('   Ingrese ordem do sistema '); %observe o grafico para definir 'n'

 un1=Un(1:(i-1)*L,1:n); un2=Un(L+1:i*L,1:n);
 A=un1\un2;    % computando a matriz A do sistema identificado
 C=un1(1:L,:);  % computando a matriz C do sistema identificado
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %  Recuperando as matrizes B e D
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 u2=Un(:,n+1:i*L);
 K=u2'*L21*inv(L11);

 qL=L*i-n;
 sig=zeros(i*qL,m);
for k=1:i
    sig((k-1)*qL+1:k*qL,:)=K(:,(k-1)*m+1:k*m);
end
  Q=u2';

  QQ=zeros(qL*i,i*L);
 for k=1:i
      QQ((k-1)*qL+1:k*qL,1:(i-(k-1))*L)=Q(:,(k-1)*L+1:L*i);
 end

 IU=[eye(L) zeros(L,n);zeros(L*(i-1),L) un1];
 % solucionando no sentido de minimos quadrados
  DB=(QQ*IU)\sig;
```

```
  D=DB(1:L,:);      % computando a matriz D do sistema identificado
  B=DB(L+1:L+n,:);  % computando a matriz B do sistema identificado

  A,  B, C, D
  %  FIM DO ALGORITMO
```

Apêndice B. Algoritmo N4SID (caso determinístico).

```
%*****************************************************
%               ALGORITMO LAZO ABERTO
%                  ORDINARIO n4sid
%*****************************************************
% Dados de entrada u=[u1 u2...um];   m = numero de entradas
% Dados de saida   y=[y1 y2 ...yL];  L = numero de saidas
% i tem que ser i > n  onde n = ordem do sistema, i=10

M=idss(drss(4,2,2));
% [A,B,C,D]=th2ss(M); matrizes do sistema real simulado
u=idinput([1000,2],'prbs',[0 0.3]);  % dados de entrada coletados
y=sim(M,u);                          % dados de saida coletados
% tratamento de dados
 u=detrend(u); y=detrend(y);
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Graficando os  dados de entrada e saida
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plotando os dados de entrada e saida usados na identificação
 idplot([y(:,1) u(:,1)]) ;  % entrada e saida 1
 idplot([y(:,2) u(:,2)]);  % entrada e saida 2

[m,N]=size(u); if m > N; u=u'; end; [m,N]=size(u);
[L,N]=size(y); if L > N; y=y'; end; [L,N]=size(y);

N=length(y); %
i=10;       % Numero definido pelo usuario
j=N-2*i+1;  % numero de colunas da matriz da Hankel

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% matriz de Hankel das entradas U  e Saidas Y
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
U=zeros(2*i*L,j);
for s=1:2*i;
    U((s-1)*L+1:s*L,:)=u(:,s:s+j-1);
end

Y=zeros(2*i*m,j);
for s=1:2*i;
    Y((s-1)*m+1:s*m,:)=y(:,s:s+j-1);
end

h=[U;Y];
R = triu(qr(h'))';
R=R(1:2*(m+L)*i,1:2*(m+L)*i);      % Trucamento de R;

clear U;Y;
% Computando Projeçao obliqua
Up = R(1:m*i,:);                        % Entradas Pasadas
Uf = R(m*i+1:2*m*i,:);                  % Entradas futuras
Yp = R(2*m*i+1:2*m*i+L*i,:);            % Saidas Pasadas
Yf = R(2*m*i+L*i+1:(2*m*i+2*L*i),:);    % Saidas futuras
```

```
Wp = [Up;Yp];

AB=Yf -Yf*(Uf')*pinv(Uf*(Uf'))*Uf;
CB=Wp -Wp*(Uf')*pinv(Uf*(Uf'))*Uf;
Oi= AB*pinv(CB)*Wp;
clear AB CB
% Computando a outra projecao obliqua
Uf_    = R(m*(i+1)+1:2*m*i,:);      Yf_     = R(i*(2*m+L)+L+1:2*i*(m+L),:);
Upmas  = R(1:m*(i+1),:);            Ypmas   = R(2*m*i+1:i*(2*m+L)+L,:);
Wpmas  = [Upmas;Ypmas];

AB=Yf_ -Yf_*(Uf_')*pinv(Uf_*(Uf_'))*Uf_;
CB=Wpmas -Wpmas*(Uf_')*pinv(Uf_*(Uf_'))*Uf_;
Oi_= AB*pinv(CB)*Wpmas;
clear AB CB
[U,S,V] = svd(Oi);
 ss=diag(S);
 ss=ss(1:i); semilogy(ss,'rx'); title('valores singulares');

 n = input('   Ingrese ordem do sistema '); %observe o grafico para definir 'n'

% Encontrando U1 e S1
U1 = U(:,1:n);  % Determine U1
% ************************************
%               Determine ri e Ri_
% ************************************
ri = U1*diag(sqrt(ss(1:n)));
ri_= ri(1:L*(i-1),:);
% e suas seudos inversas
Xi     = pinv(ri)*Oi;
Xi_mas = pinv(ri_)*Oi_;
%====================================
%   determine as matrizes A,B,C,D
%====================================
   Uii = R(m*i+1:m*i+m,:);
   Yii = R((2*m+L)*i+1:(2*m+L)*i+L,:);
   K = [Xi;Uii];
   J = [Xi_mas;Yii];

X=J/K;
A=X(1:n,1:n)
B=X(1:n,n+1:n+m)
C=X(n+1:n+L,1:n)
D=X(n+1:n+L,n+1:n+m)

 %  FIM DO ALGORITMO
```