UNIVERSIDADE FEDERAL DE MINAS GERAIS

Graduate Program in Electrical Engineering

Pedro Felipe Leite Retes

Regressor Selection for Polynomial NARMAX Models Employing a Randomized Approach

Belo Horizonte

2018

Pedro Felipe Leite Retes

Regressor Selection for Polynomial NARMAX Models Employing a Randomized Approach

Final version

Thesis presented to the Graduate Program in Electrical Engineering of the Universidade Federal de Minas Gerais in partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

Advisor: Luis Antonio Aguirrel

Belo Horizonte

2018

Retes, Pedro Felipe Leite.
R437r Regressor selection for polynomial NARMAX models employing a randomized approach [recurso eletrônico] / Pedro Felipe Leite Retes. – 2018.
1 recurso online (80 f. : il., color.) : pdf.

Orientador: Luís Antônio Aguirre.

Dissertação (mestrado) – Universidade Federal de Minas Gerais, Escola de Engenharia.

Bibliografia: f. 76-80.
Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia elétrica – Teses. 2. Sistemas dinâmicos – Teses. 3. Sistemas não lineares – Teses. 4. Processamento de sinais – Matemática – Teses. 5. Modelos matemáticos – Teses. 6. Métodos de simulação – Teses. 7. Método de Monte Carlo – Teses. I. Aguirre, Luís Antônio. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pelo Bibliotecário Marcio Anderson de Andrade Gomes CRB/6 2812
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

"Regressor Selection for Polynomial Narmax Models Employing a Randomized Approach"

PEDRO FELIPE LEITE RETES

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica.

Aprovada em 29 de junho de 2018.

Por:

gov.br Documento assinado digitalmente
LUIS ANTONIO AGUIRRE
Data: 25/10/2023 15:12:22-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Luis Antonio Aguirre
DELT (UFMG) - Orientador

Prof. Dr. Eduardo Mazoni Andrade Marçal Mendes
DELT (UFMG)

Dimas Abreu Archanjo Dutra

Prof. Dr. Dimas Abreu Archanjo Dutra
DEMEC (UFMG)

gov.br Documento assinado digitalmente
EDUARDO MAZONI ANDRADE MARCAL MENDES
Data: 26/10/2023 09:46:32-0300
Verifique em https://validar.iti.gov.br

Acknowledgments

The author expresses his sincere gratitude to Prof. Luis Aguirre for the support, patience, advising and dedication that made this work possible. The discussions and knowledge sharing were fundamental for this research.

Resumo

Esse documento trata da seleção de regressores de modelos não lineares. Os métodos clássicos de seleção de estruturas de modelos são introduzidos como motivação para o desenvolvimento de métodos contemporâneos de seleção de estruturas capazes de lidar com modelos não lineares utilizando técnicas de Monte Carlo. Uma abordagem para seleção de regressores, baseada em amostragem aleatória, foi implementada no contexto de modelos NARMAX (*nonlinear autoregressive moving average with exogenous input*) como uma extensão do método RaMSS - *Randomized Model Structure Selection* - um algoritmo de seleção de regressores implementado para modelos NARX. O método proposto foi comparado com técnicas clássicas de seleção de regressores como o ERR - *Error Reduction Ratio*, utilizando dados simulados e dados reais. Apesar de que os modelos obtidos com o método proposto nem sempre são os mais compactos, esses modelos normalmente apresentam melhor desempenho seja por uma melhor seleção de regressores ou por contemplar maior flexibilidade de representação, pela inclusão de termos NMA - *Nonlinear Moving Average*.

Palavras-chave: NARMAX, Modelos não lineares, Seleção de regressores, Estrutura de modelos, Identificação de sistemas.

Abstract

This document presents the background, implementation and results for regressor selection in the nonlinear model context. The classical model structure selection techniques are introduced and serve as a motivation to develop model structure selection methods capable of dealing with nonlinear models in a randomized Monte Carlo fashion. A randomized approach for regressor selection was implemented in the context of *nonlinear autoregressive moving average with exogenous input* (NARMAX) model class as the extension of the RaMSS - *Randomized Model Structure Selection* - method. The proposed method is compared to classical regressor techniques such as the ERR - *Error Reduction Ratio* - in both simulated and real data. Although the models selected by the proposed method are not always the more compact ones, they typically exhibit better performance in comparison to the models obtained by other methods. This is due to either better regressor selection performed by the algorithm or by the inclusion of NMA - *Nonlinear Moving Average* - what provides a more flexible system representation.

Keywords: NARMAX, Nonlinear models, Regressor selection, Model structure, System identification.

List of Figures

List of Tables

Symbol List

$\mathbf{0}_N$	Zeros matrix with NxN dimension
$\boldsymbol{\theta}$	General parameter vector
$\hat{\boldsymbol{\theta}}$	Estimated parameter vector
ℓ	NARMAX model degree of nonlinearity
$\boldsymbol{\psi}^{\mathrm{T}}(k-1)$	Regressor vector containing observations up to sample $k-1$
Ψ	Dynamic models regressor matrix
$\mathrm{Be}(\cdot)$	Bernoulli variable random sample
$D(\cdot,\cdot)$	Kullback-Leibler (K-L) divergence
$\mathbb{E}[\cdot]$	Mathematical expectation operator
$e(k)$	Error at time k
$\boldsymbol{e}$	Error vector
$\boldsymbol{\xi}$	Residuals vector
$\mathcal{F}$	The power set of $\mathcal{R}$, such $\mathcal{F}=2^{\mathcal{R}}$
$\boldsymbol{I}_N$	Identity matrix with NxN dimension
$\mathcal{L}[\cdot]$	Likelihood function
$\mathcal{M}_i$	A determined model with index i
$\mathcal{N}$	Normal distribution
N_p	Number of models extracted at each RaMSS and RaMSS-m iteration
n_θ	Parameter vector length
n_e	Maximum lag for noise terms
n_y	Maximum lag for output terms
n_u	Maximum lag for exogenous input terms
$p(\cdot\|\cdot)$	Conditional probability

$\mathcal{R}$	Set of candidate regressors for a model
X	General regressor matrix
X_n	n-th element of a Markov Chain
$y(k),\ y(i)$	Output data at time k or i
$\hat{y}(k),\ \hat{y}(i)$	Predicted output data at time k or i
$\boldsymbol{y}$	Dependent variable vector
$\mathcal{U}$	Uniform distribution
$u(k)$	Input data at time k

Contents

Chapter 1

Introduction

The main objective of system identification is to build models from data [Ljung, 1987; Söderström and Stoica, 1989]. Estimated models can be useful to analyze real system behavior, implement more robust controllers and predict system output based on input configurations. Two important steps in system identification are model selection and parameter estimation. The former one poses special challenges in the case of nonlinear systems.

In nonlinear system identification, Model Structure Selection (MSS) plays a crucial role in the estimated model behavior. The choice of appropriate regressors to compose the model is usually a difficult task. Structure selection has been a key point in nonlinear system identification and several techniques are available [Hong et al., 2008]. Traditional approaches such as the Akaike Information Criterion (AIC) [Akaike, 1974] and the Bayesian Information Criterion (BIC) Schwarz [1978] have been extensively used in determining the number of regressors of linear models. Those methods, due to the extreme importance in the modeling field, are revisited in Chapter 2. Unfortunately such methods are not very useful in helping to choose *which* regressors should compose the model. Hence in model structure selection to determine which regressors – or at least which term clusters [Aguirre and Billings, 1995b] – should compose the model is more critical than to assert the total number of terms [Mendes and Billings, 2001]. Model behavior can be radically different and spurious dynamical regimes can be introduced with different choices of regressor terms [Aguirre and Billings, 1995a].

In some situations where static data are available it is possible to use such auxiliary information to delete from the set of candidate regressors certain term clusters [Aguirre et al., 2000]. This, of course, characterizes a gray-box modeling approach. If no *a priori* information is available, the candidate terms should be chosen in a black-box fashion [Wei and Billings, 2008].

The Error Reduction Ratio (ERR) based algorithms, associated with the orthogonal least squares (OLS) methods have been widely applied in black-box nonlinear system identification. These methods have a particular advantage: the contributions of candidate terms can be decoupled, in such a way that contribution of each term to the model *one-step-ahead prediction* is quantified by the corresponding ERR [Korenberg et al., 1988; Billings et al., 1989]. It was pointed out that OLS-ERR algorithms may occasionally select incorrect or redundant model terms for some wide classes of excitation signals or for noisy data [Mao and Billings, 1997] and a way of overcoming this is to use the model *simulation error* rather than the prediction error in choosing the model structure [Piroddi and Spinelli, 2003; Piroddi, 2008]. This also has problems of its own, probably the greatest one is that it is quite time consuming. To partially circumvent such a limitation alternative procedures have been proposed [Bonin et al., 2010; Farina and Piroddi, 2010].

In a probabilistic vein, the Randomized algorithm for Model Structure Selection (RaMSS) was proposed recently [Falsone et al., 2015]. This method explores the model universe by means of randomly sampling different models and calculating performance indexes. Those indexes, when aggregated in a Monte Carlo fashion, permit regressor importance rank and model structure extraction. The method is quite useful for *linear in the parameters* model representations because of the parameter estimator algorithm that was chosen for the method - ordinary least squares. The RaMSS method has its limitations. It was originally formulated for the NARX (nonlinear autoregressive model with exogenous inputs) model which limits its application to a restrict class of models.

1.1 Aims of this work

The main contribution of this work is to implement an extended version of the RaMSS algorithm (briefly the RaMSS-m) which is intended to cope with a more general NARMAX (nonlinear autoregressive moving average model with exogenous inputs) model. In addition to being able to capture more complex system interaction, the proposed method may improve the estimator accuracy in comparison to the original RaMSS method, feature significant run time reduction by applying a parallel model sampling technique and introduce a way to include prior model information on the regressor selection phase, in a grey box approach.

1.2 Organization of this work

This dissertation is divided in five Chapters, including this introductory one. Chapter 2 reviews some fundamentals of System Identification, ranging from data collection to structure representation. The following subsections are mainly focused on classical model structure selection techniques and step-wise selection approaches. That chapter also introduces an overview on parameter estimation and model validation. Chapter 3 briefly reviews a more contemporary approach on model structure selection, by means of Monte Carlo techniques. The RaMSS [Falsone et al., 2015] algorithm is presented and simple examples are provided. In Chapter 4 rests the main contribution of this dissertation. The RaMSS-m method is introduced as an extension to RaMSS. The new features are exemplified on both simulation and real data. Results are compared the original RaMSS method and to the traditional Error Reduction Ratio method, presented in Sec. 2.2.4.2. The findings on the RaMSS-m method were published in [Retes and Aguirre, 2018]. Final comments are included in Chapter 5.

Chapter 2

Nonlinear Modeling

2.1 Introduction

In the beginning of studies on models, the main idea was to find a mathematical representation that was useful to help to understand how real systems behaves. Science has always tried to explain natural phenomena using a more palatable language for humans. This, most of the times, implies obtaining simplified models though powerful enough to explain natural behavior in bounded conditions.

One example of simplification is to model real, nonlinear data with linear models. The beauty of linear models, although extremely important to provide information on data, has its limitations. Linear models do not allow the required flexibility to mimic real system dynamics in many practical situations. Those properties, observed in the real world, include bifurcations, quasi-periodic dynamics and chaos.

As more information on real systems becomes available and computer power increases, the rise of nonlinear models opened up a new world of possibilities in prediction and of better understanding real world problems. In the past thirty years, several works were dedicated to nonlinear modeling ([Billings, 1980], [Leontaritis and Billings, 1985a,b], [Korenberg et al., 1988; Billings et al., 1989], [Chen and Billings, 1992; Chen et al., 1990], [Aguirre and Billings, 1995a], [Aguirre and Billings, 1995b], [Aguirre et al., 2000], [Mendes and Billings, 2001], [Zhu, 2005], [Wei and Billings, 2008]).

This chapter provides an overview on the system modeling procedures and was the roadmap followed by the author in his nonlinear modeling technique studies. The classical methodology is widely used in various scientific areas and should be present in this dissertation to be compared to newer approaches. Some ideas contained in the following chapters were only developed because of classical method limitations. The traditional methods will provide a basis to introduce a more contemporary approach

using a randomized framework to deal with model structure selection in Chapter 3.

2.2 System Identification

System identification is mainly composed of five steps:

1. data collection and pre-processing;
2. choice of model representation;
3. structure selection;
4. parameter estimation;
5. model validation.

Model complexity and nonlinearities will require that each step is performed in a different way. The following subsections provides an outline of the aforementioned scheme.

2.2.1 Data Collection

Data collection is the first step to system identification. The system behavior must be observed introducing excitation signals, in case of non-autonomous systems. System behavior is observed through direct measurements of output variables (when available) or through observation on state variables.

Input signals should be designed in order to excite the system dynamics and, for this, input signal should have adequate spectral power. In the case of nonlinear models, whereas the amplitude must remain small in order to avoid driving the system outside the range of linear behavior chosen, the input data should be able to explore the system nonlinear characteristics, and, often, that requires exciting the system with greater amplitude signals. Certain conditions should be met to choose the appropriate excitation signal. In practice, filtered white noise and, especially for linear systems, pseudo-random binary sequence (PRBS) are commonly used as the system input.

Finally, data sampling should be performed considering the spectral context of the signals. In what follows, it will be assumed that data was correctly sampled and it is also important to avoid oversampling what could introduce undesirable sample correlation.

2.2.2 Representation

A non-autonomous model describes the output data given an input excitation. A simple model considers that the input-output relation is linear, i.e. the system satisfies the property of superposition. The behavior of the linear system output subjected to a complex input can be described as a sum of responses of simpler inputs. In nonlinear systems, the changes in the output is not proportional to change of the input. Nonlinear models can produce certain dynamic regimes that linear models cannot represent.

The chosen model structure should be complex enough to represent the system nonlinearities of interest. Some classes of those representations include artificial neural networks (Haykin [1994]), radial basis functions networks (Broomhead and Lowe [1988]), Volterra series (Billings [1980]), wavelets (Strang [1989]), rational and polynomial functions (Billings et al. [1989]). Particular data characteristics can, sometimes, drive the choice of structure representation. For example, polynomial and rational functions may be useful to extract non-linear relations between system variables and provide further information on high-level system performance due to input disturbance or variable interaction.

2.2.3 Classical Model Structure Selection Techniques

2.2.3.1 Akaike Information Criterion and AICc

In the model selection task it is desirable to find the optimized model in terms of the bias-variance tradeoff. Finding this model is, most of the times, a difficult task because the model error function is usually unknown. The model should be rich enough to capture and mimic the data behavior but not too flexible so it starts to model noise, overfitting the data. If the model is too simple, with a small number of parameters, it will probably not generalize well on training data. In the other hand, if the model has a huge set of parameters, it will not generalize well in new data. Balancing the correct model between bias and variance, i.e. choosing the model complexity must be performed somehow James et al. [2013].

One way to evaluate the model performance is by the likelihood function (or more conveniently, the log-likelihood function). The principle of maximum likelihood selects the model parameters that make the data most likely. If the model complexity is increased, the likelihood function will be also increased as data will be more likely given the parameters set. The Maximum Likelihood Estimation (see Sec. 2.2.5.3 for more information) is a technique that deals with the bias portion of the MSE function, so the overfitting effect is not accounted for. To cope with the undesirable variance

increase consequence of model flexibility expansion, one idea is to penalize models that are more complex.

In [Akaike, 1974], the author proposes a criterion to compare and select models from a pool of candidates. The general idea is presented next.

Consider two likelihood functions f, being the true likelihood function and f^*, the estimated likelihood function that was obtained from data. In order to compare those probability distribution functions, one of the most common ways is to use the Kullback-Leibler (K-L) divergence:

$$D(p,q) = \sum_x p \log\frac{p}{q}, \tag{2.1}$$

where p and q are distributions over data x.

Let $f_{j\boldsymbol{\theta}^*}$ be the estimated distribution for a particular model j. The loss function, considering the true distribution and the estimated distribution is:

$$\text{loss} = D(f,\ f_{j\boldsymbol{\theta}^*}) \tag{2.2}$$

Considering that $f_{j\boldsymbol{\theta}^*}$ is dependent on the data, to get $f_{j\boldsymbol{\theta}^*}$ closer to f, the expectation of the loss function should be minimized:

$$\mathbb{E}_f[D(f,\ f_{j\boldsymbol{\theta}^*})] \tag{2.3}$$

This way, if the model overfits the data, $f_{j\boldsymbol{\theta}^*}$ will not be close to f and the model is penalized. The same way if the model underperforms on the training data Y, the two distribution functions will be different and the model will be also penalized. This procedure accounts for both bias and variance.

It can be proven that the divergence between f and $f_{j\boldsymbol{\theta}^*}$ is:

$$D(f, f_{j\boldsymbol{\theta}^*}) = c - A(f, f_{j\boldsymbol{\theta}^*}), \tag{2.4}$$

where

$$c = \sum_Y f(Y) \log(f(Y)) \tag{2.5}$$

is the entropy and

$$A(f, f_{j\boldsymbol{\theta}^*}) = \sum_Y f(Y) \log(f_{j\boldsymbol{\theta}^*}(Y)). \tag{2.6}$$

In order to minimize the divergence, Eq. 2.3, one should maximize 2.6. The

challenge in minimizing $A(f, f_{j\boldsymbol{\theta}^*})$ is that the true distribution, f, is unknown. One alternative to solve the optimization problem is to estimate f using the $f_{j\boldsymbol{\theta}^*}$. This choice will lead to a biased estimate. The bias can be shown to be approximately the size of the model, i.e. the model number of parameters k. Shortly, the Akaike Information Criterion (AIC) is the likelihood, $\mathcal{L}$, corrected by the bias, k:

$$\mathbb{E}_f[A(f, f_{j\boldsymbol{\theta}^*})] \approx \log(\mathcal{L}(\boldsymbol{\theta}^*|Y)) - k. \tag{2.7}$$

Considering the maximum likelihood case, the AIC definition becomes:

$$\text{AIC} = -2\log(\mathcal{L}(\theta^*|x)) + 2k. \tag{2.8}$$

When the sample size is small, the probability that AIC will select models with large parameter set is increased. This will lead to model overfitting. In order to address this potential issue, a corrected version of AIC, taking account the sample size was developed [Cavanaugh, 1997]:

$$\text{AICc} = \text{AIC} + \frac{2k^2 + 2k}{n - k - 1}. \tag{2.9}$$

With AIC (and AICc), models can be compared and ranked considering the bias-variance tradeoff. Models with lower AIC should be selected as they are optimum in terms of this criterion. This is possible as long as the models are from the same structure, i.e. the AIC method is applicable to select the number of model parameters, not to select models from alternative structures.

2.2.3.2 Bayesian Information Criterion

When fitting models, it is possible to increase the likelihood function by expanding the number of model parameters, as shown in Sec. 2.2.3.1. Unfortunately, increasing the number of parameters may lead to overfitting. The Bayesian Information Criterion (BIC), also known as Schwarz Criterion was first presented in Schwarz [1978] and uses a Bayesian framework to deal with the model selection problem. Similarly to AIC, the BIC introduces a penalty to the model number of parameters, thus preferring more parsimonious models. The penalty term is larger in BIC than in AIC and it is asymptotically correct based on the assumptions on that data distribution is on the exponential family.

The Bayesian approach to model selection is to maximize the posterior probability of a model ($\mathcal{M}_i$) given the data $Y = \{y_j\}_{j=1}^N$ (Bhat and Kumar [2010]). Applying the Bayes theorem in order to calculate the posterior probability of a model:

$$p(\mathcal{M}_i|Y) = \frac{p(Y|\mathcal{M}_i)p(\mathcal{M}_i)}{p(Y)}, \tag{2.10}$$

where $p(Y|\mathcal{M}_i)P(\mathcal{M}_i)$ is called the marginal likelihood of model $\mathcal{M}_i$.

Considering that all candidate models are equally likely, i.e. the prior distribution of models, $p(\mathcal{M}_i)$, is uniform, maximizing the posterior probability of a model, given the data, is the same as maximizing the marginal likelihood:

$$p(Y|\mathcal{M}_i) = \int_{\boldsymbol{\Theta}_i} p(Y|\boldsymbol{\theta}_i)p(\boldsymbol{\theta}_i|\mathcal{M}_i)d\boldsymbol{\theta}_i = \int_{\boldsymbol{\Theta}_i} \mathcal{L}(\boldsymbol{\theta}_i|Y)p(\boldsymbol{\theta}_i|\mathcal{M}_i)d\boldsymbol{\theta}_i \tag{2.11}$$

where $\boldsymbol{\theta}_i$ is the vector of parameters of $\mathcal{M}_i$, $\mathcal{L}$ is the likelihood function and $p(\boldsymbol{\theta}_i|\mathcal{M}_i)$ is the prior on the parameters.

If it is desired to compare two models, $\mathcal{M}_j$ and $\mathcal{M}_l$, the posterior odds can be used:

$$\frac{p(\mathcal{M}_j|Y)}{p(\mathcal{M}_l|Y)} = \frac{p(\mathcal{M}_j)}{p(\mathcal{M}_l)}\frac{p(Y|\mathcal{M}_j)}{p(Y|\mathcal{M}_l)}, \tag{2.12}$$

where the $\frac{p(\mathcal{M}_j)}{p(\mathcal{M}_l)}$ portion comes from the prior beliefs on the models and the $\frac{p(Y|\mathcal{M}_j)}{p(Y|\mathcal{M}_l)}$ comes from data. The last term is defined as the Bayes factor.

Models can be chosen, by comparing the posteriors in Eq. 2.12. If it is greater than 1, model $\mathcal{M}_j$ is chosen, if it is less than 1, model $\mathcal{M}_l$ is chosen.

By doing some approximations, it can be shown that:

$$\log p(Y|\mathcal{M}_i) \approx \log p(Y|\boldsymbol{\theta}_i^*) - \frac{|\boldsymbol{\theta}_i^*|}{2}\log n, \tag{2.13}$$

where $p(Y|\boldsymbol{\theta}_i^*)$ is the likelihood, $\boldsymbol{\theta}_i^*$ is the maximum likelihood estimate and $|\boldsymbol{\theta}_i^*|$ is the ℓ_0 norm, the number of parameters.

Eq. 2.13 is the definition of the BIC. Choosing the model with the largest BIC, given that the approximations are valid, is equivalent of choosing the model with largest posterior probability.

2.2.3.3 Regularization

The ordinary least squares (OLS) estimator calculates the parameter vector, $\hat{\boldsymbol{\theta}}$, by minimizing the residual sum of squares (RSS):

$$J_{\text{LS}} = \text{RSS} = \sum_{i=1}^{N} \boldsymbol{\xi}(i)^2. \tag{2.14}$$

Ridge regression is very similar to OLS, except that the parameter vector is estimated by minimizing a modified cost function. The Ridge regression parameter vector, $\hat{\boldsymbol{\theta}_{\boldsymbol{R}}}$ is calculated by minimizing:

$$J_{\mathrm{R}} = \sum_{i=1}^{N} \boldsymbol{\xi}(i)^2 + \lambda ||\hat{\boldsymbol{\theta}}_{\mathrm{R}}||^2, \tag{2.15}$$

where $\lambda \geq 0$ is a design parameter. The cost function 2.15 consideres two different criteria. Similarly to least squares, the Ridge regression estimates the parameter vector in order to fit data well, minimizing J_{R}. The second term, $\lambda ||\hat{\boldsymbol{\theta}}_{\mathrm{R}}||^2$, called a *shrinking penalty*, is small when the $\hat{\theta_R^1}...\hat{\theta_R^j}$ are close to zero, so it has the effect of shrinking $\hat{\boldsymbol{\theta}}_{\mathrm{R}}$ towards zero.

Parameter λ controls the relative impact of the RSS and the penalty term in the regression coefficients estimation. When $\lambda = 0$, the Ridge regression parameter estimation is identical to that of the ordinary least squares estimator. However, as λ increases, the penalty term increases and the estimated coefficients will tend to zero. The Ridge regression generates a new set of coefficients estimates for each value of λ, unlike the ordinary least squares estimator which produces a single, optimal set of coefficients calculated by the minimization of the residual sum of squares. The λ parameter thus, can be used to control the model flexibility, i.e. to control the *bias-variance* trade-off.

Even though the Ridge regression can be useful to control model complexity, it will still include all regressors from the candidate regressor set. The penalty $\lambda ||\hat{\boldsymbol{\theta}}_{\mathrm{R}}||^2$ in 2.15 will shrink all coefficients towards zero, but none will be set exactly to zero, unless $\lambda = \infty$. A more recent approach, alternative to the Ridge regression is the *(least absolute shrinkage and selection operator* (LASSO) Tibshirani [2011]. LASSO can perform both regressor selection and regularization in order to control model complexity by minimizing the cost function:

$$J_{\mathrm{L}} = \sum_{i=1}^{N} \boldsymbol{\xi}(i)^2 + \lambda |\hat{\boldsymbol{\theta}}_{\mathrm{L}}|. \tag{2.16}$$

LASSO also includes a penalty term, $\lambda |\hat{\boldsymbol{\theta}}_{\mathrm{L}}|$. This penalty term uses the ℓ_1 norm penalty instead of ℓ_2 penalty. It, just like as in Ridge regression, shrinks coefficients towards zero, however, it also has the effect of forcing some of the coefficients estimates to be exactly zero when the tuning parameter, λ, is large enough. By setting some coefficients to zero, LASSO generate models that involve only a subset of variables, in other words, LASSO performs variable selection.

2.2.4 Forward and Backward Regression-based Approaches

2.2.4.1 Stepwise regression

Often, theory and experience give only a general idea of which pool of candidate variables should be included in the model. In example, if one wants do model the average household energy consumption, the number of individuals in the house is a natural probable explanatory regressor. Specially when dealing with nonlinear modeling, the selection of explanatory variables may not be so evident.

Finding a subset of (independent) regressors involves two antagonistic objectives: to include every possible variable that influences the dependent variable (output) but also include the minimal set of regressors, as the inclusion of irrelevant variables reduces the model prediction ability and decreases the the precision of estimated coefficients.

In linear model selection, forward regression method is used to provide an initial screening of candidate variables of a large pool. The forward regression consists in incrementally selecting variables and rebuilding the model. The start point is to fit models considering one explanatory variable at a time. Then, among all models, the one with the highest R-squared is selected and the explanatory variable associated to that model is included in the "best" model. Next, the procedure is run again, but now, the models include the "best" selected explanatory variable in the previous step and the remaining variables are tested one by one. Again, the model with the highest R-squared value has the next "best" explanatory variable. The method is stopped when the inclusion of explanatory variables does not increase the R-square.

In contrast, backward selection is performed in the opposite direction. The method starts with the model including all explanatory variables. Iteratively, variables are pulled out of the model, respecting the selection rule, e.g. keeping the variables associated with the model with the highest R-squared.

Stepwise selection is a combination between forward selection and backward selection. The method, just like forward selection, incrementally selects explanatory variables but, at each step one variable is added to the model, all variables in the model are checked for their significance. If their significance has been reduced below the specified threshold, the insignificant variables are removed from the model, similarly to the backward selection approach.

Stepwise selection performance on linear models is pretty acceptable, due to the superposition property of linear systems. In nonlinear models, this kind of approach should be used in a more conservative way, as the variable interaction may produce different results in the model in comparison to their individual contribution. Subsec. 2.2.4.2 presents an alternative to decouple inter-variable interaction and account

for unique contribution of each variable to the final nonlinear model.

2.2.4.2 Error Reduction Ratio

Let $\mathcal{R} = \{\psi_1, \psi_2, ..., \psi_m\}$ denote the set of m regressors from a NARMAX structure. In short, the model structure selection problem is to choose from $\mathcal{R}$ a subset of $n_\theta \ll m$ regressors ψ_i to compose the final model. For this, a widely used criterion is the error reduction ratio (ERR) [Billings et al., 1989] which quantifies the reduction in the variance of the residuals, that occurs when a new term is included in the model, normalized with respect to the output variance. Hence, the ERR due to the inclusion of the ith regressor in the model can be written as:

$$[\mathrm{ERR}_1]_i = \frac{\mathrm{MS1PE}(\mathcal{M}_{i-1}) - \mathrm{MS1PE}(\mathcal{M}_i)}{\langle \mathbf{y}, \mathbf{y} \rangle}, \tag{2.17}$$

for $i = 1, 2, \ldots, m$, where $\mathrm{MS1PE}(\mathcal{M}_i)$ stands for the mean square one-step-ahead prediction error of the model with i terms (regressors); m is the number of candidate terms tested for; and $\mathcal{M}$ represents a family of models with nested structures, thus $\mathcal{M}_{i-1} \subset \mathcal{M}_i$. In (2.17) the numerator equals the reduction in variance of the residuals due to the inclusion of the ith regressor. The denominator of (2.17) is the data variance. One of the advantages of the ERR_1 is that it can be represented in compact form as [Billings et al., 1989]:

$$[\mathrm{ERR}_1]_i = \frac{\hat{g}_i^2 \langle \mathbf{w}_i, \mathbf{w}_i \rangle}{\langle \mathbf{y}, \mathbf{y} \rangle}, \qquad i = 1, 2, \ldots, m, \tag{2.18}$$

where $\mathbf{w}_i$ is the ith orthogonal regressor and $\hat{g}_i$ is the corresponding estimated parameter. Thus, at each step, the term with the largest ERR_1 is added to the model.

ERR_1 is easy and fast to use and also it is quite effective, therefore it is widely used. An extention of the ERR criterion using two-step-ahead predictions, the ERR_2, was proposed in [Alves et al., 2012] in order to detect unwanted terms.

A different criterion, called simulation error reduction ratio (SRR), was defined as [Piroddi and Spinelli, 2003]:

$$[\mathrm{SRR}]_i = \frac{\mathrm{MSSE}(\mathcal{M}_{i-1}) - \mathrm{MSSE}(\mathcal{M}_i)}{\langle \mathbf{y}, \mathbf{y} \rangle}, \tag{2.19}$$

for $i = 1, 2, \ldots, m$, where $\mathrm{MSSE}(\mathcal{M}_i)$ stands for the mean square simulation error of the model with i terms (regressors). In (2.19) the *free-run* simulation is used. The SRR can be effective in non-ideal identification conditions and often yields more compact

models. On the other hand, such a criterion requires a significantly large computational effort and hard to apply for time series models (models without input).

When dealing with structure selection from experimental data it is of paramount importance to realize that, strictly speaking, there is no such a thing as "genuine" or "spurious" regressors, although certain regressors might be more likely to be useful than others. Also, it has been argued that, given a set of data of limited length and accuracy different model strcutures might become undistinguishable [Barbosa et al., 2015].

2.2.5 Parameter Estimation

2.2.5.1 Bias and Variance

The identification problem is to define a function f that represents a good approximation of the system by means of learning from training data. Assume that the true system can be represented by the approximation function $\hat{f}$:

$$y \sim \hat{f}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}), \tag{2.20}$$

where $\boldsymbol{x}$ is denoted as the regressor set and $\hat{\boldsymbol{\theta}}$ is denoted as the parameter set. The effectiveness of the proposed model, $\hat{f}$, can be specified as the model generalization capacity. The mean squared error (MSE) can be used as a measure of model generalization, if computed over a validation set of data:

$$\text{MSE} = \mathbb{E}[(y - \hat{f}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}))^2], \tag{2.21}$$

where $\mathbb{E}[\cdot]$ is the expectation operator. As shown in Geman et al. [1992], the MSE can be represented via the bias/variance decomposition:

$$\text{MSE} = \text{Var}(\hat{f}) + [\text{Bias}(\hat{f})]^2 + \text{Var}(\epsilon), \tag{2.22}$$

where $\text{Var}(\epsilon)$ is the variance of the noise.

If, on average $\hat{f}$ is different from $\mathbb{E}[y]$, then $\hat{f}$ is said to be a biased estimator of $\mathbb{E}[y]$.

To minimize the MSE, requires minimizing both the bias term and the variance term, which is, in most practical situations, a conflicting interest. The variance of noise, $\text{Var}(\epsilon)$, cannot be reduced as it is independent of the model and its parameters.

Lets say, for instance, that the output prediction does not depend on the input variables. By neglecting input data, the output may be set to a constant value, regardless of input values. The estimator variance will be definitely minimized, but the

output will be probably way off the real function, i.e. the bias will be excessively large. On the other hand, if the designed model, $\hat{f}$ is tremendously flexible, capable of perfectly interpolating the training data, the estimator bias will be reduced, but model will suffer from a huge increase in variance.

The bias describes how far the model is, on average, to the true process. The variance represents how much the model prediction varies between realizations. The bias-variance tradeoff is, in depth, the over and under-fitting tradeoff. High bias can cause the model to not follow the original system input-output relation, whereas high variance is caused by overfitting, where estimated model tends to model the noise rather than the output.

Defining model complexity can be a challenging task, as there is no analytical way to decide the ideal model size. One possible solution to estimate model complexity is to employ cross-validation techniques, where model size is chosen in order to minimize the prediction error on new data, unseen by the model before.

2.2.5.2 Ordinary Least Squares

Once the model structure is defined, the next step is to estimate model parameters. The mostly employed parameter estimator for linear-on-the parameters regression is the ordinary least squares. The primordial idea of the OLS method may be found in Gauss and Davis [2004].

Consider that the scalar function $y = f(x)$ in applied in the vector case:

$$y = f(\boldsymbol{x}, \boldsymbol{\theta}), \tag{2.23}$$

where $f(\boldsymbol{x}, \boldsymbol{\theta}) : \mathbb{R}^n \to \mathbb{R}$ is dependent on $\boldsymbol{\theta} \in \mathbb{R}^n$.

The function in Eq. 2.23 defines a set of equations (restrictions), which are the result of the application of $f(\boldsymbol{x}, \boldsymbol{\theta})$:

$$\begin{aligned} y_1 &= f(\boldsymbol{x}_1, \boldsymbol{\theta}) \\ y_2 &= f(\boldsymbol{x}_2, \boldsymbol{\theta}) \\ \vdots &= \vdots \\ y_N &= f(\boldsymbol{x}_N, \boldsymbol{\theta}), \end{aligned} \tag{2.24}$$

where y_i is the i^{th} observation of y and $\boldsymbol{x}_i = [x_{1i}\ x_{2i}\ ...\ x_{ni}]^{\text{T}}$ are the i^{th} observations of the n elements of $\boldsymbol{x}_i$.

Considering that f and the parameter vector $\boldsymbol{\theta}$ does not change between restrictions in 2.24, and assuming that f is linear, the restrictions in 2.23 can be rewritten as

$$\boldsymbol{y} = X\boldsymbol{\theta}, \tag{2.25}$$

where

$$X = [x_1 \; x_2 \; ... \; x_N]$$

and

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}.$$

It is easy to infer that if n linearly independent restrictions are taken to determine the n elements of $\boldsymbol{\theta}$, i.e. $N = n$, the parameter vector $\boldsymbol{\theta}$ can be determined by:

$$\boldsymbol{\theta} = X^{-1}\boldsymbol{y}. \tag{2.26}$$

If $N > n$ restrictions were taken, the equation system is overdetermined. This implies that the X matrix is not squared and, therefore cannot be inverted. Equation 2.26 is the only solution that simultaneously satisfies the n restrictions in 2.24. In order to determine the parameter vector $\boldsymbol{\theta}$ that satisfies 2.24, one possible solution is to define a rewritten model of Eq. 2.25, but this time, considering that the solution is not supposed to be exact, thus introducing an error term:

$$\boldsymbol{y} = X\hat{\boldsymbol{\theta}} + \boldsymbol{\xi}, \tag{2.27}$$

where $\boldsymbol{\xi} = [\xi_1 \; \xi_2 \; ... \; \xi_N]^{\mathrm{T}}$ is the error vector generated by the attempt of explaining $\boldsymbol{y}$ by $X\hat{\boldsymbol{\theta}}$. Intuitively, it is desirable that the parameter vector $\hat{\boldsymbol{\theta}}$ is chosen in a way that the error vector, $\boldsymbol{\xi}$, is reduced in some sense. To do so, a loss function, relative to the error vector can be defined as:

$$J_{\mathrm{LS}} = \sum_{i=1}^{N} \xi(i)^2 = \boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{\xi} = ||\boldsymbol{\xi}||^2. \tag{2.28}$$

The estimated parameter vector, $\hat{\boldsymbol{\theta}}_{\mathrm{LS}}$, that minimizes the loss function, Eq. 2.28, can be proven to be:

$$\hat{\boldsymbol{\theta}}_{\mathrm{LS}} = [X^{\mathrm{T}}X]^{-1}X^{\mathrm{T}}\boldsymbol{y}. \tag{2.29}$$

The Eq. 2.29 is defined as the least squares estimator for the parameter vector $\boldsymbol{\theta}$.

2.2.5.3 Maximum Likelihood Estimation

Another class of parameter estimators, based on the probabilistic properties of the data, is the Maximum Likelihood Estimate - MLE. The main idea of this method is to pursue the parameter value that makes data more likely. The MLE is an example of a point estimate, as it outputs a single value for the unknown parameter, rather than a probability or interval. The main challenge of this method is to define the likelihood function to be optimized. This function is dependent on the selected model structure and previous assumptions on data characteristics.

Consider, again, that the data can be modeled in the same way as 2.27:

$$\boldsymbol{y} = X\hat{\boldsymbol{\theta}} + \boldsymbol{\xi}. \tag{2.30}$$

At this point, the model follows exactly the ordinary least squares assumptions.

One should provide information, or premisses, on the error model. In OLS, the objective was to reduce the error vector ℓ_2 norm, $||\boldsymbol{\xi}||^2$, to its minimum by finding an appropriate value of the parameter vector $\hat{\boldsymbol{\theta}}$. This is done despite of the error (residuals) model and it works for either normal or non-normal errors.

In MLE, the idea is to estimate the parameter vector $\hat{\boldsymbol{\theta}}$ that makes data more likely, so it is mandatory that the distribution of errors is known. For a simple linear regression model, it is assumed that:

$$\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}_N, \sigma^2\boldsymbol{I}_N), \text{i.i.d.}, \tag{2.31}$$

where $\boldsymbol{I}_N$ is the identity matrix and $\mathbf{0}_N$ is a vector of zeros. The observations $\boldsymbol{y}$ are independent random variables samples with distribution:

$$\boldsymbol{y} \sim \mathcal{N}(X\hat{\boldsymbol{\theta}}, \sigma^2\boldsymbol{I}_N). \tag{2.32}$$

Under normal errors assumption, the joint probability density function (pdf) of $\boldsymbol{y}$ given $\hat{\boldsymbol{\theta}}$ is:

$$f(\boldsymbol{y}|\hat{\boldsymbol{\theta}}) = \frac{1}{(2\pi\sigma^2)^{N/2}} e^{-\frac{1}{2\sigma^2}(\boldsymbol{y}-X\hat{\boldsymbol{\theta}})^{\mathrm{T}}(\boldsymbol{y}-X\hat{\boldsymbol{\theta}})}, \tag{2.33}$$

and $f(\boldsymbol{y}|\hat{\boldsymbol{\theta}})$ is defined as the likelihood function, $\mathcal{L}(\hat{\boldsymbol{\theta}}|\boldsymbol{y})$.

Usually it is more convenient to deal with the log-likelihood instead of the likelihood itself:

$$\ln \mathcal{L}(\hat{\boldsymbol{\theta}}|\boldsymbol{y}) = -\frac{1}{2\sigma^2}(\boldsymbol{y} - X\hat{\boldsymbol{\theta}})^{\mathrm{T}}(\boldsymbol{y} - X\hat{\boldsymbol{\theta}}) + c, \tag{2.34}$$

where c is a constant that does not depend on $\hat{\boldsymbol{\theta}}$.

The maximum likelihood estimate (MLE) of $\hat{\boldsymbol{\theta}}$ is the estimate that satisfies:

$$\max(\ln \mathcal{L}(\hat{\boldsymbol{\theta}}|\boldsymbol{y})) = \max(-(\boldsymbol{y} - X\hat{\boldsymbol{\theta}})^{\mathrm{T}}(\boldsymbol{y} - X\hat{\boldsymbol{\theta}})) \tag{2.35}$$

Maximizing the likelihood function, which is equal to maximize the log-likelihood function, under the previous assumptions, gives the same result as the OLS estimator, Eq. 2.29, which is to find the parameters that minimize the error vector:

$$\hat{\boldsymbol{\theta}}_{\mathrm{MLE}} = [X^{\mathrm{T}}X]^{-1}X^{\mathrm{T}}\boldsymbol{y}. \tag{2.36}$$

Note that the MLE is a very powerful tool, that accepts a very flexible set of premisses about the data model. Normality was assumed by convenience and to compare the results with the OLS outputs.

2.2.5.4 Extended Least Squares

Considering that the data, $y(k)$, can be described by a polynomial model, the regression equation can be defined as:

$$y(k) = \boldsymbol{\psi}^{\mathrm{T}}(k-1)\boldsymbol{\theta} + e(k), \tag{2.37}$$

where $\boldsymbol{\psi}^{\mathrm{T}}(k-1)$ is the regressor vector (independent variables) taken up to time $k-1$ and $\boldsymbol{\theta}$ is the parameter vector.

From data with p points, the matrix equation can be written:

$$\boldsymbol{y} = \Psi\boldsymbol{\theta} + \boldsymbol{e} \tag{2.38}$$

where

$$\boldsymbol{y} = \begin{bmatrix} y(k) \\ y(k-1) \\ \vdots \\ y(k-p+1) \end{bmatrix}$$

$$\Psi = \begin{bmatrix} \psi(k-1) \\ \psi(k-2) \\ \vdots \\ \psi(k-p) \end{bmatrix}.$$

From the ordinary least squares, the parameter vector $\boldsymbol{\theta}$ can be estimated by:

$$\hat{\boldsymbol{\theta}}_{\mathrm{LS}} = [\Psi^{\mathrm{T}}\Psi]^{-1}\Psi^{\mathrm{T}}\boldsymbol{y}. \tag{2.39}$$

Unfortunately, the Ψ matrix may contain lagged noise terms that are, by definition, unmeasurable. In this case, an alternative approach can be employed to estimate the lagged noise terms. This alternative is to replace the noise terms by the lagged residuals, and reconstruct an "extended" regressor matrix, Ψ^*.

This procedure consists in progressively refine the parameters vector, $\hat{\boldsymbol{\theta}}^*_{\mathrm{ELS}}$ by the following steps:

1) Define an initial random parameter vector:

$$\hat{\boldsymbol{\theta}}^*_{\mathrm{ELS}} = \mathrm{rand}(n_\theta) \tag{2.40}$$

2) Define an initial random residual vector and make $i = 1$:

$$\hat{\boldsymbol{\xi}}_0 = \mathrm{rand}(p) \tag{2.41}$$

3) Reconstruct the extended regressor matrix Ψ^*_i, replacing the noise terms with the estimates $\hat{\boldsymbol{\xi}}_{i-1}$.

4) Update the parameter vector:

$$\hat{\boldsymbol{\theta}}^*_{\mathrm{ELS}} = [\Psi^{*\mathrm{T}}\Psi^*]^{-1}\Psi^{*\mathrm{T}}\boldsymbol{y} \tag{2.42}$$

5) Calculate the current residual vector:

$$\hat{\boldsymbol{\xi}}_i = \boldsymbol{y} - \Psi^*\hat{\boldsymbol{\theta}}^*_{\mathrm{ELS}} \tag{2.43}$$

6) Make $i = i + 1$ and repeat from step 3) until convergence.

In practice, after about 12 iterations the parameter vector $\hat{\boldsymbol{\theta}}^*_{\mathrm{ELS}}$ will converge.

2.2.6 Model Validation

2.2.6.1 Performance Measurements

In order to evaluate the performance of a given model, one should be able to measure the how well the predictions match observed data. In regression models, the most commonly used measure is the *mean squared error* (MSE). In this subsection, a more general definition of MSE will be used and will serve as a basis for particular cases, such as MSPE and MSSE that will be further defined.

The MSE is given by

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y(i) - \hat{y}(i))^2, \tag{2.44}$$

where N is the number of data points, $y(i)$ is the observed data and $\hat{y}(i)$ is the model prediction.

MSE can be applied either to *in sample*-data, where model is evaluated using data already used to train the model or to *out of sample*-data, where MSE is computed for data unseen by the model. The latter is of more interest as the idea of modeling is to provide answers to data that was not available before in the training set. The MSE can be used to *validade* the model, for instance, to measure the model performance on out of sample-data and compare to the in sample-data MSE. A good model should have similar MSE performance for the training set and test set.

Another commonly used perfomance metric is the *mean absolute percentage error* (MAPE), where the measurement is taken on the absolute deviation of prediction to observed data and, usually, expresses accuracy as a percentage:

$$\text{MAPE} = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{y(i) - \hat{y}(i)}{y(i)}\right|\%. \tag{2.45}$$

2.2.6.2 Validation Set

Suppose that it is required to estimate the test error associated with a regression model on a set of observations. The *validation set* technique is a simple, yet powerful, strategy for this job. It consists in dividing observed data in two parts, a *training-set* and a *validation-set*. The model is learned from the training set and the fitted model is used to make predictions on the validation set. A performance index on the validation set, for instance MSE, provides an estimate of the expected error rate on a test set (new data) and, thus, provide information on model quality. One potential

drawback of the validation set approach is that the validation estimate of error can be highly variable, depending on how data is split and which observations are included in training/validating set. A more refined validating procedure will be presented in the next subsection.

2.2.6.3 Cross-validation

Cross-validation follows the same idea presented in Sec. 2.2.6.2, but tries to address the aforementioned drawbacks. Cross-validation, specifically *leave-one-out* cross-validation (LOOCV), consists of splitting the observation set into two parts and have been extensively discussed in [Allen, 1974] and in [Stone, 1974]. Instead of creating two subsets with equal or comparable sizes, the validation set is composed of a single element. The training set is the whole observation set, excluding that element from the validation set. The model is, then, learned from the $N-1$ points set and prediction $\hat{y}_1$ is made for the excluded observation. Since the observation on the validation set was not used for model training, the $\text{MSE}_1 = (y(1) - \hat{y}(1))^2$ provides an approximately unbiased estimate for the test error. Although unbiased, the MSE_1 is a poor estimate because it has high variance, since it is based on a single observation. The underlying idea behind *leave-one-out* cross-validation is to compute several MSE indexes for each N observation, training models on $N-1$ observations and leaving one point for validation. Repeating this approach N times, produces N MSEs: $\text{MSE}_1, ..., \text{MSE}_N$. The LOOCV estimate for the test set MSE is the average computation of those indexes:

$$\text{LOOCV}_N = \frac{1}{N} \sum_{i=1}^{N} \text{MSE}_i. \tag{2.46}$$

Although simple, the LOOCV is known to be asymptotically inconsistent and too conservative as it tends to select an unnecessarily large model[Shao, 1993]. Other methods for model selection, such as the AIC, presented in Sec. 2.2.3.1 and the C_p [Mallows, 2000], are asymptotically equivalent to the LOOCV and, thus, share the same deficiency - they are inconsistent. The problems associated with LOCCV can be suppressed by using an alternative observation division scheme.

The *k-fold* cross-validation approach involves splitting the dataset into k groups, or *folds* of equal size. The first fold is used as the validation set and the remaining $k-1$ ones are used to train the model. The MSE_1 is then computed on the observations on the validation fold. This procedure is repeated k times, each time a group is defined as the validation set. This results in k estimates of the test error, and, by extent, the k-fold estimate is defined as the average of those values:

$$\text{kFCV}_k = \frac{1}{k} \sum_{i=1}^{k} \text{MSE}_i. \tag{2.47}$$

The *k-fold* cross-validation, with proper selection of k, can overcome some of the LOOCV limitations and provide better estimation on model performance over unseen data.

Chapter 3

Randomized Model Structure Selection

3.1 Introduction

In this chapter a more contemporary approach to the MSS task is presented. The increase of computer power available to the general public permits more processing demanding methods, such as Monte Carlo methods, to be applicable and results to be obtained in shorter times. Markov chain is introduced as a basis for the Markov Chain Monte Carlo methods and the Metropolis-Hastings algorithm is presented in the model structure selection context. Recent developed methods, the Reversible Jump Markov Chain Monte Carlo and the RaMSS are presented and simple examples are given in order to illustrate method capabilities.

3.2 Monte Carlo Sampling Approaches

3.2.1 Markov Chains

A sequence $X_1, X_2, ..., X_n$ of random elements from some set is a Markov chain if the conditional distribution of X_{n+1} given $X_1, X_2, ..., X_n$ depends on X_n only. The *state space* of the Markov chain is the set in which X_i assumes values (Geyer [2011]. One of the main properties of the Markov chains, assumed in this section, is the *stationary transition probability*. This means that transition probability of X_{n+1} given X_n does not depend on n and it is the main kind of Markov chain of interest in Markov Chain Monte Carlo, which will be discussed in the next section. The Markov chain process starts in one *state* X_i and moves successively to other state. Those moves are called

steps. The chain current state is X_i, then it moves to state X_j with a probability denoted by p_{ij} - the *the transition probability*. This probability, because of the stationary transition property does not depend on which state the chain was before the current state. The process can stay in the current state with probability p_{ii}. Usually the Markov chain is presented with finite state space and the transition probabilities form a matrix composed of all element-wise transition probabilities. When the state space is uncountable one cannot assume the initial distribution as a vector and transition probability distribution as a matrix. They must be considered to be an unconditional probability distribution and a conditional probability distribution.

3.2.2 Markov Chain Monte Carlo and Metropolis-Hastings

Monte Carlo methods were introduced with the invention of computers. Those methods consists in massive random sampling of data to extract useful information. The first method, developed in the early stages of scientific computing, called Ordinary Monte Carlo (OMC), considers that $X_1, X_2, ..., X_n$ are independent and identically distributed (i.i.d), in which case the Markov Chain is stationary and reversible. This case is the direct application of elementary statistics. Example 3.2.1 shows one simple procedure for mean and variance calculation.

Example 3.2.1. *Mean and variance calculation using OMC*

Suppose that it is possible to simulate $X_1, X_2, ..., X_n$, i.i.d., from the distribution of X. Defining $Y_i = g(X_i)$, then the Y_i are i.i.d with mean

$$\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} g(X_i). \tag{3.1}$$

and variance

$$\sigma^2 = \text{var}\{g(X)\}, \tag{3.2}$$

where $\hat{\mu}_n$ is the sample mean of Y_i and the central limit theorem defines that

$$\hat{\mu}_n \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right). \tag{3.3}$$

The variance can be estimated by

$$\hat{\sigma}_n^2 = \frac{1}{n}\sum_{i=1}^{n}(g(X_i) - \hat{\mu}_n)^2, \tag{3.4}$$

which is the empirical variance of Y_i. □

The theory of OMC follows the theory of basic statistics. The difference here is that, instead of employing real word randomness sampling, the OMC uses the computer simulation to mimic the random process, commonly through pseudorandom sampling. The same statistical properties from the traditional statistics theory apply to the OMC. It is clearly evident that, for instance, the statistical accuracy is inversely proportional to the square root of the sample size. The accuracy of Monte Carlo methods is dependent on the sample size, so to achieve increase in the method accuracy, significant increase in the number of simulated samples is required.

Markov chain Monte Carlo (MCMC) are methods used to approximate the posterior distribution of a parameter by means of sampling. Suppose one wants to sample from a complex distribution p or to approximate the expected value $\mathbb{E}\{f(x)\}$, $f(x) \sim p$ from a complex distribution. The MCMC method draws samples from the target distribution by generating random values (the Monte Carlo part) subject to some rule for determining what makes a good sampled value. The main idea is to define, for a pair of values, which one is most likely to represent the data, given the pior distribution. If a new randomly sampled value is better than the last one, it is added to the chain of values with a certain probability determined by the transition rules (transition probability). This is the Markov chain part. The samples drawn from the posterior distribution can be used to calculate statistics such as mean and variance that reproduces the posterior distribution behavior.

MCMC can be employed to estimate model parameters and it is particularly useful when the model parameters state space is large. Considering that the number of model parameters is fixed, MCMC can be used to sample the parameter values, compose a Markov chain with those values and, after a certain burn-in period of samples, it is expected that the sampled values will converge to the Markov chain stationary distribution. In order to perform MCMC, one should sample from the (joint) posterior distribution. From the Bayes rule, the posterior distribution is described by:

$$p(\boldsymbol{\theta}|Y, \mathcal{M}) = \frac{p(Y|\boldsymbol{\theta}, \mathcal{M})p(\boldsymbol{\theta}|\mathcal{M})}{p(Y|\mathcal{M})}, \tag{3.5}$$

where $\boldsymbol{\theta}$ is the parameter vector, Y is the data, $\mathcal{M}$ is a model definition, $p(\boldsymbol{\theta}|Y, \mathcal{M})$ is the posterior, $p(Y|\boldsymbol{\theta}, \mathcal{M})$ is the likelihood, $p(\boldsymbol{\theta}|\mathcal{M})$ is the prior on the parameters and $p(Y|\mathcal{M})$ is the marginal likelihood. The marginal likelihood plays no role, as it does not depend on $\boldsymbol{\theta}$, and it can be assumed to be a normalization factor.

MCMC is a process that, given data and auxiliar functions, make intelligent or guided guesses on the posterior distribution. MCMC will require the following ingredients to be run:

- Data set
- Model
- Sampler
- Likelihood function
- Prior distribution

The method consistis in sequentially drawing samples from the posterior distribution and, constrained to the update rule, accepting or rejecting the next sample. One of the mostly used samplers for MCMC is the Metropolis-Hastings sampler [Metropolis et al., 1953; Hastings, 1970] and it is considered to be one of the top ten most important algorithms in recent statistic applications.

The Metropolis-Hastings (MH) algorithm proposes a way to construct a Markov chain on a state space $\mathcal{X}$ that is ergodic and stationary with respect to the posterior distribution $\pi(x)$. In other words, if $X_n \sim \pi(x)$, then $X_{n+1} \sim \pi(x)$ and, thus, the chain converges in distribution to $\pi(x)$. While there are other samplers that delivers a Markov chain associated with a target distribution, the MH method is one of the preferred ones, due to its simplicity and versatility. Rather than considering the state space as role, the method constructs a path through the most probable region in the parameter state space in an incremental way, exploring local regions and constructing the posterior distribution by means of the accept/reject schema.

The MH sampler employs an alternative way to draw samples from π, even when the π function is only known to the ratio $\pi(x)/\pi(y)$. This allows the algorithm to be employed without knowing the normalizing factor. The algorithm requires an arbitrary *candidate distribution*, $q(x, y)$, from which one can draw samples. For every point in state space, $q(x, y)$ is a normalized probability density and a sample y can be always taken from $q(x, \cdot)$. Also, at every point of the state space, $q(x, y)$ can be calculated. The Metropolis-Hastings algorithm is presented in Alg. 1:

This acceptance/rejection schema preserves the stationary density π if the chain is irreducible, in other words, if q has a wide enough probability to reach every region of the state space with positive mass under π. Using Eq. 3.5, one can calculate the posteriors ratio $\pi(x)/\pi(y)$ by means of the defined prior, $p(\boldsymbol{\theta}|\mathcal{M})$, and the likelihood, $p(Y|\boldsymbol{\theta}, \mathcal{M})$.

If the the MH algorithms is allowed to draw a large amount of samples ($\sim 10^4$ to 10^6), the initial steps (*burn-in*) can be discarded and the remaining samples are a representation of the posterior distribution. Ex. 3.2.2 will illustrate the MH method to perform parameter estimation in a simple case.

Example 3.2.2. *MCMC Parameter Estimation*

Algorithm 1 Metropolis-Hastings Algorithm

1: Given $X^{(t)}$,
2: Generate $Y_t \sim q(y|X^{(t)})$.
3: Take

$$X^{(t+1)} = \begin{cases} Y_t \text{ with probability } \alpha(X^{(t)}, Y_t), \\ X^{(t)} \text{ with probability } 1 - \alpha(X^{(t)}, Y_t), \end{cases}$$

where

$$\alpha(x, y) = \min\left\{\frac{\pi(y)}{\pi(x)}\frac{q(x|y)}{q(y|x)}, 1\right\}.$$

4: Repeat

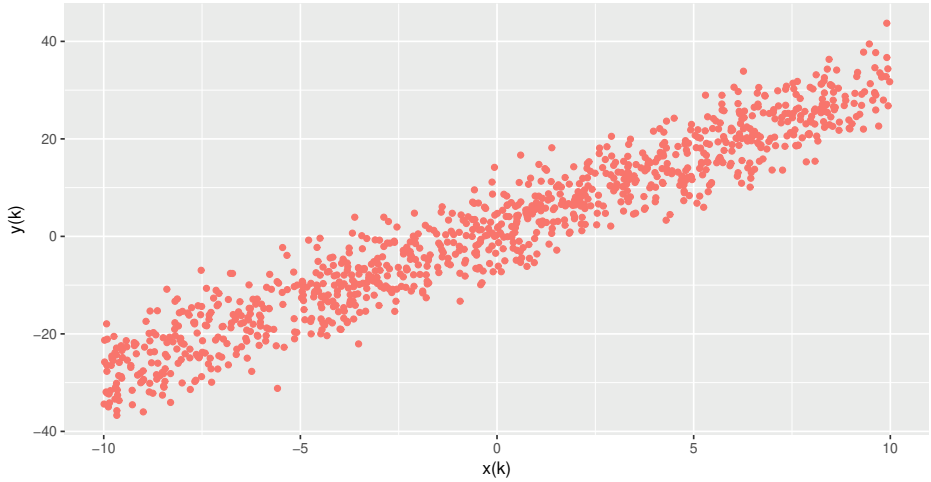


Figure 3.1. Data plot for Ex. 3.2.2.

Consider data generated from the following process:

$$y(k) = \theta_1 x(k) + \theta_2 + e(k), \tag{3.6}$$

where $x(k) \sim \mathcal{U}(-10, 10)$ and $e(k) \sim \mathcal{N}(0, \sigma^2)$.

The process parameters are known: $\theta_1 = 3$, $\theta_2 = 1.5$, $\sigma^2 = 5$. Fig. 3.1 shows the plot of the dependent variable, $y(k)$ against the independent variable $x(k)$.

The next step, following the ingredient list for the MCMC method, is to define a model. For the sake of simplicity, the same model structure that generated data will be used, similarly to what was considered in Eq. 2.30, with unknown parameters:

$$\boldsymbol{y} = X\hat{\boldsymbol{\theta}} + \boldsymbol{\xi}, \tag{3.7}$$

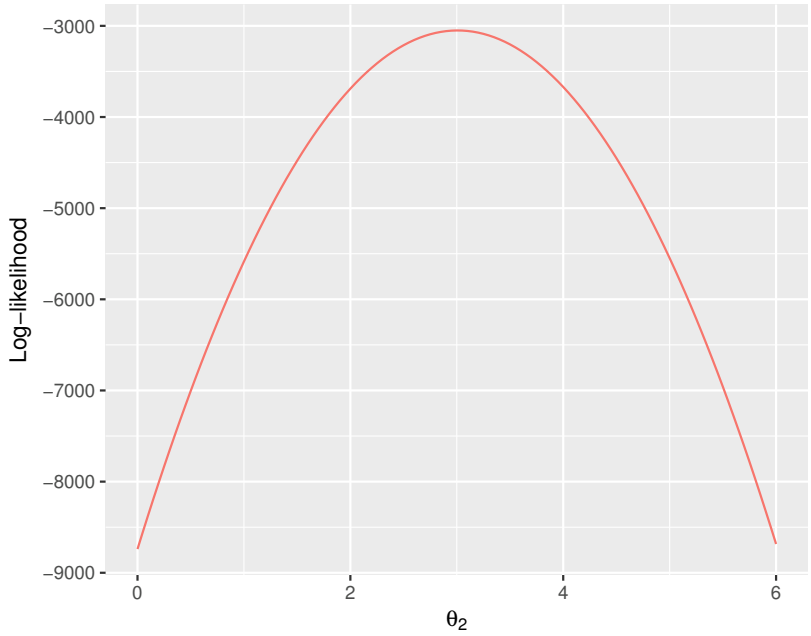


Figure 3.2. Log-likelihood plot for parameter θ_2.

with $\hat{\boldsymbol{\theta}} = [\theta_1 \ \theta_2]^{\mathrm{T}}$, $\boldsymbol{\xi} \sim \mathcal{N}(0, \sigma^2)$.

The likelihood function is the probability which one would expect the data to occur, given the parameters. Considering the model in Eq. 3.7, the likelihood function would be

$$\ln \mathcal{L}(\hat{\boldsymbol{\theta}}, \sigma^2 | \boldsymbol{y}) = -\frac{1}{2\sigma^2}(\boldsymbol{y} - X\hat{\boldsymbol{\theta}})^{\mathrm{T}}(\boldsymbol{y} - X\hat{\boldsymbol{\theta}}). \tag{3.8}$$

Fig. 3.2 shows the log-likelihood function plot for parameters θ_2.

Now, let's define the prior distribution on the parameters. If one has previous beliefs about the parameters, this could be informed to the algorithm as the definition of the priors. A more generic way is to consider *uninformative* priors, for instance, consider a uniform prior, in which the parameter has equal probability in the defined range. In this example, the prior on the parameters will be defined as an uniform distribution for parameter θ_1, a normal distribution for θ_2 and an uniform distribution for parameter σ^2, with arbitrary parameters.

With the log-likelihood and prior definitions, one can define the posterior distribution, by means of Eq. 3.5:

$$p(\hat{\boldsymbol{\theta}}, \sigma^2 | X, \boldsymbol{y}) \propto \mathcal{L}(\hat{\boldsymbol{\theta}}, \sigma^2 | X, \boldsymbol{y}) p(\hat{\boldsymbol{\theta}}, \sigma^2). \tag{3.9}$$

With those definitions in hand, the Metropolis-Hastings algorithm can be run. The procedure jumps around the parameter state space, on a way that regions that

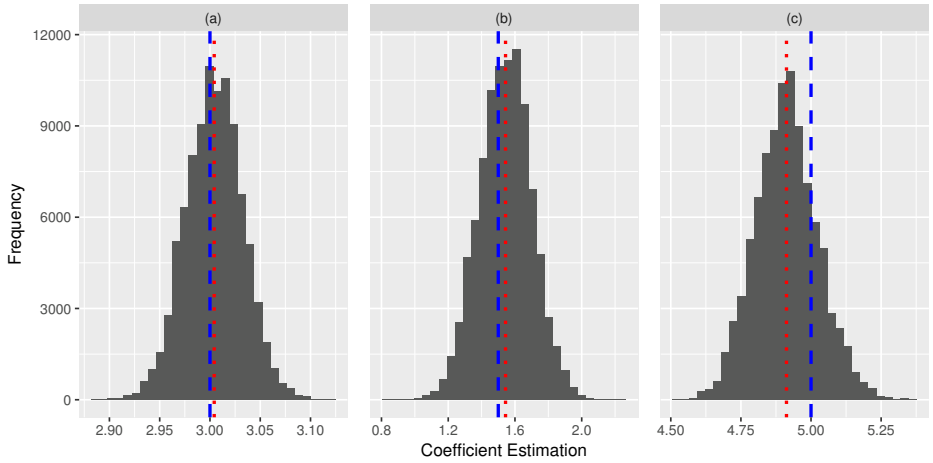


Figure 3.3. Posterior distribution of (a) θ_1, (b) θ_2 and (c) σ^2. True values denoted by the blue dashed line. Mean values appointed by the red red dotted line.

have higher posterior are visited more often. After a *burn-in* period, the Markov chain (the sequence of sampled parameter values) converges to the posterior distribution. Fig. 3.3 shows the final distribution of the parameters.

In comparison to the MCMC method for parameter estimation, a simple, OLS method was employed in the same dataset to perform parameter estimation. Results for the coefficients from the linear regression model (OLS) were $\hat{\theta}_{\text{OLS}} = [3.00326 \;\; 1.54818]^{\text{T}}$. Results from obtained from the MCMC-MH method were (mean values): $\hat{\theta}_{\text{MCMC}} = [3.004216 \;\; 1.543502]^{\text{T}}$. The results were quite similar because the chosen model structure was the same for both methods. MCMC, especially the MH method, has evident advantages over other parameter estimation techniques when the proposed models require a large state space.

□

This subsection presented one of the most famous methods for Bayesian inference, the Markov chain Monte Carlo. Althought powerful, the MCMC method has its limitations. Unfortunately, the method cannot handle state spaces that have varying dimensions. The extended method for this application, the Reversible Jump Markov chain Monte Carlo, is presented in the next subsection.

3.2.3 Reversible Jump Markov Chain Monte Carlo

The Metropolis-Hastings MCMC algorithm was introduced for simulations of posterior distributions on spaces of fixed dimensions. In order to simulate samples from distributions of varying dimensions, the Reversible Jump Markov chain Monte Carlo method was presented in [Green, 1995]. Reversible Jump MCMC can also be

considered a generalization on the Metropolis-Hastings algorithm, where the proposal distribution and the posterior distribution may have densities on spaces of different dimensions. Different problems in which the number of unknown model parameters is itself unknown are extensive. Some of those models include: Change-point models, finite mixtures models, time series models and variable selection [Polasek, 2012]. This subsection summarizes the work presented in [Hastie and Green, 2012], mostly focused on the variable selection problem.

Bayesian model selection involves joint inference about model indicator k and the parameter vector $\boldsymbol{\theta}_k$, where the model indicator determines the dimension of the parameter space n_k, which may vary from model to model. In the Bayesian framework, model indicator and parameter vector, $(k, \boldsymbol{\theta}_k)$, are treated as a joint unknown. The inference on the model is made on the posterior joint $p(k, \boldsymbol{\theta}_k|Y)$.

Suppose that a prior $p(k)$ is specified over models k in a countable set $\mathcal{K}$ and, for each model indicator k, a prior distribution $p(\boldsymbol{\theta}_k)$ and a likelihood $p(Y|k, \boldsymbol{\theta}_k)$, for data Y. The joint posterior

$$p(k, \boldsymbol{\theta}_k|Y) = \frac{p(k, \boldsymbol{\theta}_k)p(Y|k, \boldsymbol{\theta}_k)}{\sum_{k' \in \mathcal{K}} \int p(k', \boldsymbol{\theta}'_{k'})p(Y|k', \boldsymbol{\theta}'_{k'})\mathrm{d}\boldsymbol{\theta}'_{k'}} \tag{3.10}$$

can be factorized as

$$p(k, \boldsymbol{\theta}_k|Y) = p(k|Y)p(\boldsymbol{\theta}_k|k, Y), \tag{3.11}$$

that is, the product of posterior model density and model posterior parameter density.

Inference on the joint posterior, $p(k, \boldsymbol{\theta}_k|Y)$, is usually achieved using simulation techniques on Bayesian approaches, once analytical formulations are mostly limited to toy examples. In MCMC simulations, one can decide to perform *within*-model simulation, in the form of sampling $\boldsymbol{\theta}_k \sim p(\boldsymbol{\theta}_k|k, Y)$ for a given model indicator, which is the case presented in Sec. 3.2.2, and *across*-model simulation, in the form of sampling $(k, \boldsymbol{\theta}_k) \sim p(k, \boldsymbol{\theta}_k|Y)$ which is presented in the current section.

Similarly to what is done in the Metropolis-Hastings MCMC case, the detailed balance condition can be achieved by correctly proposing new states of the chain and accepting those new states with an appropriate probability. Consider a general state space $\mathcal{X}$ and a general target distribution π. To construct a Markov chain with invariant distribution π consider the transition kernel $\mathcal{K}$, such

$$\int_{\mathcal{X}} \pi(\mathrm{d}x)\mathcal{K}(x, \mathrm{d}x') = \pi(\mathrm{d}x'). \tag{3.12}$$

The detailed balance condition is achieved if

$$\int_{\mathcal{A}\mathrm{x}\mathcal{B}} \pi(\mathrm{d}x)\mathcal{K}(x,\mathrm{d}x') = \int_{\mathcal{B}\mathrm{x}\mathcal{A}} \pi(\mathrm{d}x')\mathcal{K}(x',\mathrm{d}x), \tag{3.13}$$

where $\mathcal{A}, \mathcal{B} \subset \mathcal{X}$ are Borel sets.

Consider the current state x and that r random numbers $u \sim g$ are generated. The proposed chain new state, x', can be constructed by a function h, such that $(x', u') = h(x, u)$, where u' are r-dimensional random numbers, sampled from a joint density g' that will be required for the reverse move from x' to x. This is done using the inverse function h' of h. Thus, the detailed balance condition, Eq. 3.13 can be rewritten as:

$$\int_{\mathcal{A}\mathrm{x}\mathcal{B}} \pi(x)g(u)\alpha(x,x')\mathrm{d}x\mathrm{d}u = \int_{\mathcal{B}\mathrm{x}\mathcal{A}} \pi(x')g'(u')\alpha(x',x)\mathrm{d}x'\mathrm{d}u'. \tag{3.14}$$

If transformation h from (x, u) to (x', u') is a diffeomorphism, that is the transformation and its inverse is differentiable, then Eq 3.14 holds if

$$\pi(x)g(u)\alpha(x,x') = \pi(x')g'(u')\alpha(x',x)\left|\frac{\partial(x',u')}{\partial(x,u)}\right|. \tag{3.15}$$

A valid choice for α is

$$\alpha(x,x') = \min\left\{\frac{\pi(x')g'(u')}{\pi(x)g(u)}\left|\frac{\partial(x',u')}{\partial(x,u)}\right|, 1\right\}, \tag{3.16}$$

and, given that the dimensions of x, x', u, u' are, respectively, n, n', r and r', transformation h to be a diffeomorphism, it is required that $n + r = n' + r'$, what is called *dimension matching*.

The Reversible Jump MCMC method is quite general and flexible and its application in model selection (in the trans-dimensional case) may require the transition between states of the following types:

- Birth move:
  Where the parameter vector size is increased
- Update move:
  Where no change is dimension is performed, parameters are updated only
- Death move:
  Where the parameter vector size is decreased

While the idea behind *dimension matching* is fairly simple, one of the main difficulties on the implementation of RJMCMC is the flexible, arbitrary choice of the maps h and h' and the proposal distribution $g(u)$. Since mapping functions ultimately express the relationship of model parameters among different spaces, map function

choices will have a strong effect on the sampler performance [Polasek, 2012].

Example 3.2.3 shows a simple application of the Reversible Jump MCMC method for model selection.

Example 3.2.3. *Model selection using Reversible Jump MCMC*

Consider, again, data generated from the same process of Eq. 3.6

$$y(k) = \theta_1 x_1(k) + \theta_0 + e(k), \tag{3.17}$$

where $x_1(k) \sim \mathcal{U}(-10, 10)$ and $e(k) \sim \mathcal{N}(0, \sigma^2)$. Consider, also a second random variable, $x_2(k) \sim \mathcal{U}(-10, 10)$. Reversible Jump MCMC can be employed to selected among two models, which one is more likely:

$$\mathcal{M}_1 : y(k) = \theta_1 x_1(k) + \theta_0 + e(k)$$
$$\mathcal{M}_2 : y(k) = \theta_1 x_1(k) + \theta_2 x_2(k) + \theta_0 + e(k),$$

The model selection task can be performed employing a model size indicator variable z_2, which will determine if θ_2 should be included in the model. z_2 will be sampled from a Bernoulli prior distribution.

The method was run for a 1000 point data window which is illustrated in Fig. 3.4. After a burn-in period (2000 samples), the chain is expected to converge to the limit distribution, in this case on both parameters and model indicator. Fig. 3.5 shows parameter estimation.

θ_2 parameter model indicator, z_2, is actually part of the chain. From Fig. 3.6, it can be seen that parameter θ_2 is centered around zero, which means that it should not be included in the model. In other words, model $\mathcal{M}_1$ is more likely to be the correct one. Fig. 3.7 shows the histogram for the indicative variable z_2, which, again, corroborates to the hypothesis that model $\mathcal{M}_1$ is more likely to be the correct one.

This example illustrated how the Reversible Jump MCMC algorithm can be employed in order to perform model selection. In this case, model selection and parameter estimation were performed simultaneously.

□

3.3 Randomized Model Structure Selection

Structure selection cannot be practically realized by brute force because the candidate model universe, that comprehends all possible regressor combination, is typically large. This scenario is aggravated in the case of nonlinear models. The addition of noise

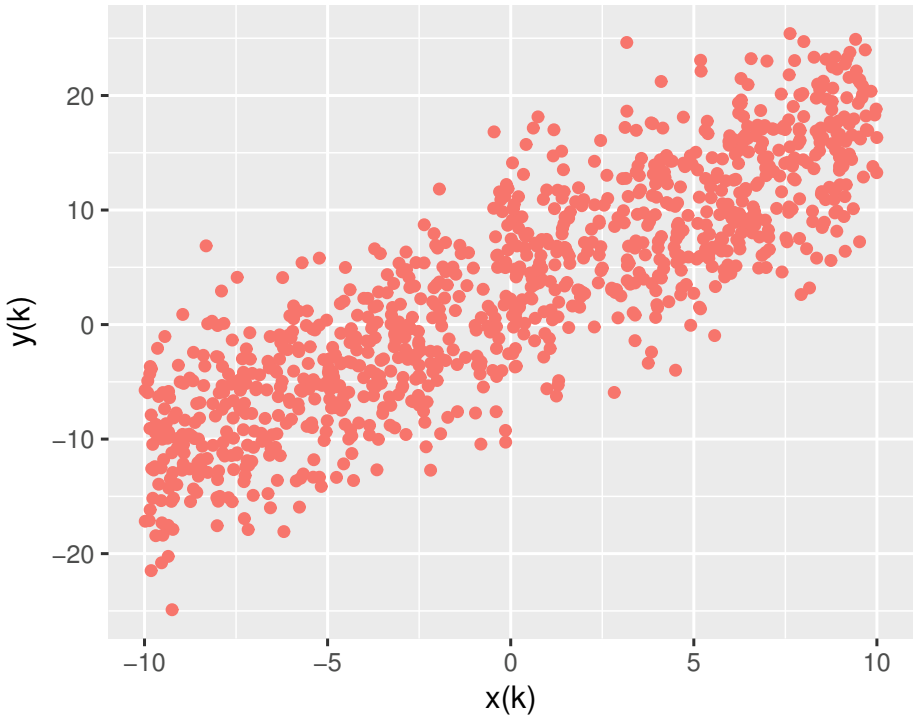


Figure 3.4. Data plot from the process generated by Eq. 3.17.

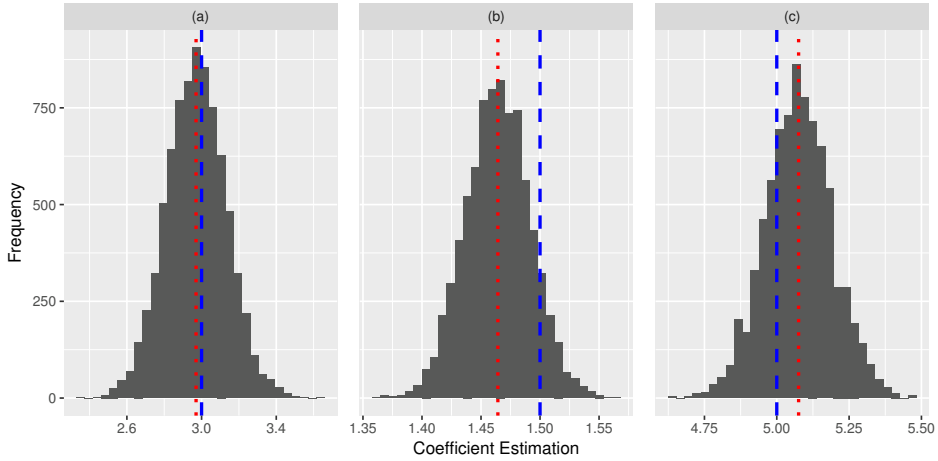


Figure 3.5. Posterior distribution of (a) θ_0, (b) θ_1 and (c) σ^2. True values denoted by the blue dashed line. Mean values appointed by the red red dotted line.

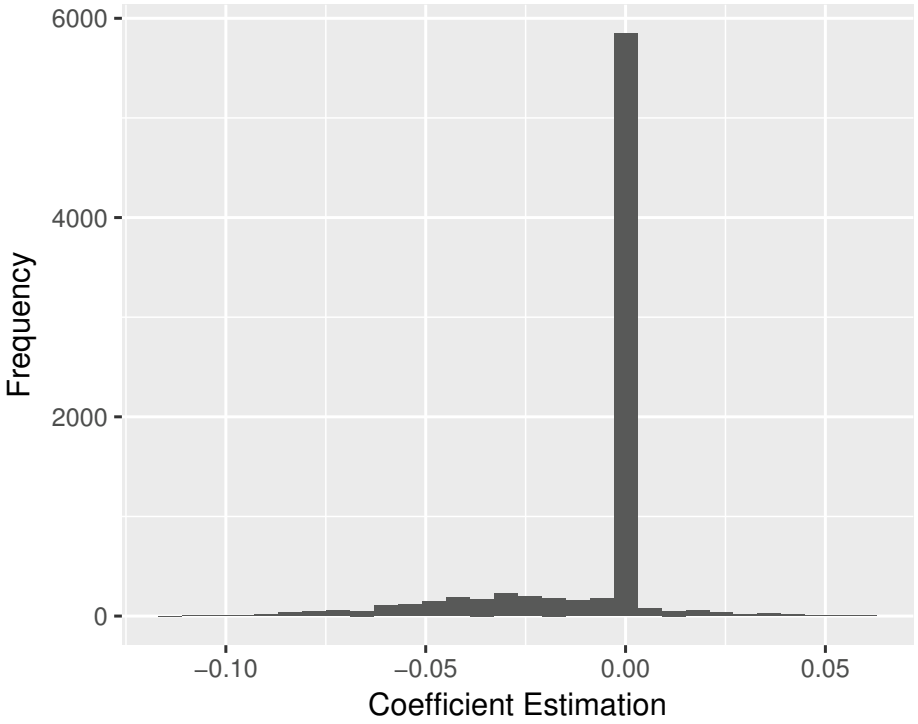


Figure 3.6. Histogram for parameter θ_2.

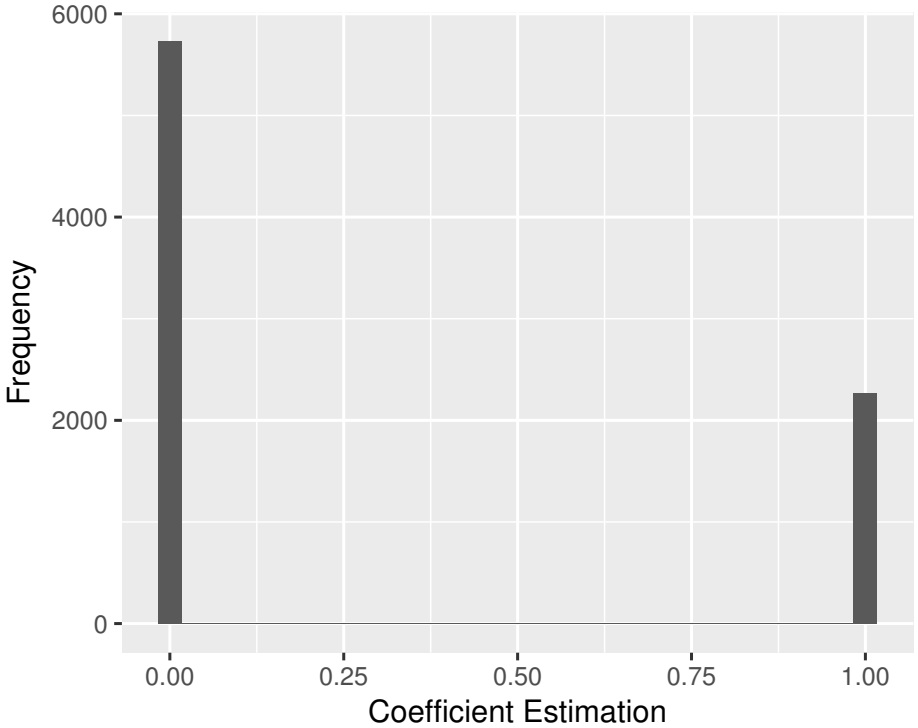


Figure 3.7. Histogram for the indicative variable z_2.

terms typically increases the flexibility of possible models, but also makes the model selection task more difficult. A randomized approach can be employed in order to choose an adequate model structure, i.e. the expected "correct" model structure, by means of model sampling in the model universe set. An interactive randomized algorithm for model structure selection, briefly RaMSS - Randomized Model Structure Selection, has been introduced in [Falsone et al., 2014] and was significantly extended in [Falsone et al., 2015]. The method constructs the structure selection task in a probabilistic framework.

3.3.1 The RaMSS algorithm

The RaMSS method consists of iteratively finding the best subset of regressors, from a set of candidate regressors, in order to maximize the corresponding model prediction accuracy. This is performed by taking model samples from the model universe according to probability functions estimated for each potential regressor, called the regressor inclusion probability (RIP). Models are evaluated, and, based on the performance, the aforementioned functions are updated. Hence, at each iteration, N_p models are drawn from the model universe and performances indexes are computed. The used performance index is based on the model prediction error, simulation error or a combination of both. The performance of a set of models, assuming that any of the N_p models contain the j^{th}-regressor, is computed thus

$$\mathcal{I}^+{}_j = \frac{1}{n_j^+} \sum_{i=1}^{n_j^+} \mathcal{J}^+{}_i \tag{3.18}$$

where n_j^+ is the number of extracted models that contain the j^{th} regressor and $\mathcal{J}^+$ is the vector of performance indexes for the models that contain the j^{th} regressor. The average computed in (3.18) is then attributed as an average *regressor* perfomance and hence can be a used to rank regressor importance.

At each iteration, RIPs are updated. The regressor inclusion rule in a candidate model is modeled through a Bernoulli process, where RIP is the Bernoulli random variable probability of success μ_j. It is expected that, after some iterations, the average model performances that include the correct regressors are significantly larger than the ones that do not include the correct regressors, so "correct" regressors become more likely to be included in the final model.

The following paragraphs explain, in a more mathematical context, the RaMSS method.

The power set of $\mathcal{R}$, i.e. $\mathcal{F} = 2^{\mathcal{R}}$, is the set of all possible model structures. If it is assumed that f^*, denoted as the true model, belongs to $\mathcal{F}$, it should be possible to find such a model by exploring the model combination set and taking the model with best performance. Redundant models, i.e. similar models that include statistically insignificant terms, are removed, and sampling is actually performed on the reduced (non-redundant) model universe $\tilde{\mathcal{F}}$. f^*, the true model, can be presented in an optimization notation, such as

$$f^* = _{\tilde{f} \in \tilde{\mathcal{F}}} \mathcal{J}(\tilde{f}), \tag{3.19}$$

where $\mathcal{J}$ is a performance metric computed for models $\tilde{f}$ taken from the set $\tilde{\mathcal{F}}$.

In order to solve the optimization problem, $\mathcal{J}$ must be estimated. The RaMSS algorithm is capable of estimating the correct model structure, f^*, by sampling the model universe, calculating model performance $\mathcal{J}(\tilde{f})$ and averaging over all models that include a certain regressor. This is carried out for each m regressor in the candidate set. The aforementioned optimization problem can hardly be solved by an exhaustive full-space search approach, given that the number of candidate models, i.e. the model state space size, is exponential in the number of candidate regressors. A typical approach, usually applied to problems where brute force is not feasible, is to consider a frequentist or bayesian framework, adopting sampling as the primary tool to extract information on models or perform data inference.

Consider that model performance metric, $\mathcal{J}$, is such

$$\mathcal{J} = e^{-K \cdot \mathrm{MSPE}}, \tag{3.20}$$

and takes values in the $[0, 1]$ range, where K is an magnifying factor. The expected value of $\mathcal{J}$, considering the probability distribution $\mathcal{P}_\psi$ and a random variable Φ that corresponds to a realization of a model sampling in $\tilde{\mathcal{F}}$, is

$$\mathbb{E}[\mathcal{J}] = \sum_{\tilde{f} \in \tilde{\mathcal{F}}} \mathcal{J}(\tilde{f}) \mathcal{P}_\psi(\Phi = \tilde{f}). \tag{3.21}$$

Eq. 3.21 is the expected value of the combination of performance indexes of all models in $\tilde{\mathcal{F}}$. If $\mathcal{P}_\psi$ is varied over all possible distributions on $\tilde{\mathcal{F}}$, the maximum of 3.21 is obtained by concentrating all the probability mass on the "true" model. The solution of the optimization problem

$$\mathcal{P}_\psi^* = \mathbb{E}[\mathcal{J}(\Phi)] \tag{3.22}$$

is such that $\mathcal{P}_\psi(f^*) = 1$. In other words, solving the optimization problem 3.22 is, essentially, selecting the true model and, thus, providing the same solution of Eq. 3.19.

A key feature of the RaMSS method is to provide a way of estimating P_ψ. To do so, a Bernoulli random variable is associated to each regressor ϕ_j such

$$\rho_j \sim \text{Be}(\mu_j), \tag{3.23}$$

whose possible outcomes are 1, with probability μ_j and 0 with probability $(1 - \mu_j)$, $\mu_j \in [0, 1]$ and $j = 1, ..., m$, m being the number of candidate regressors. Regressor ϕ_j is present in a certain model if $\rho_j = 1$. Random variables ρ_j are assumed, in this case, to be independent, although the results presented in [Bianchi et al., 2016] suggest that a multivariate, conditioned Bernoulli distribution approach provides improvements in terms of accuracy of the model selection process.

The $\boldsymbol{\mu} = [\mu_1, \mu_2, ..., \mu_m]^{\text{T}}$ vector is the Regressor Inclusion Probability (RIP) vector. Setting $\boldsymbol{\mu}$ induces a probability distribution $\mathcal{P}_\psi$ over the models in $\mathcal{F}$. In other words, assuming that $\boldsymbol{\mu}$ is known, the probability of getting a certain model structure f with any subset of n_θ regressors is given by:

$$\mathcal{P}_\psi(\tilde{f}) = \prod_{j:\psi_j \in \tilde{f}}^{n_\theta} \mu_j \prod_{j:\psi_j \notin \tilde{f}}^{m-n_\theta} (1 - \mu_j), \tag{3.24}$$

for any $\tilde{f} \in \tilde{\mathcal{F}}$.

By progressively refining RIP vector $\boldsymbol{\mu}$, it is expected that the mass probability of $\mathcal{P}_\psi$ is concentrated at f^*. To do so, it is necessary to define an update rule for the $\boldsymbol{\mu}$ vector, such for each element μ_j of $\boldsymbol{\mu}$:

$$\mu_j(i + 1) = \mu_j(i) + \gamma \mathcal{I}_j, \tag{3.25}$$

where γ is a design parameter and $\mathcal{I}_j$ is a performance index which will be explained later on.

Let

$$\mathcal{I}_j = \mathbb{E}[\mathcal{J}(\Phi)|\phi_j \in \Phi] - \mathbb{E}[\mathcal{J}(\Phi)|\phi_j \notin \Phi], \tag{3.26}$$

where $j = 1, ..., m$ and the conditional expectations are set to zero if the conditioning event has zero probability to occur. The performance index $\mathcal{I}_j$ compares the average performance of the models containing the j^{th} regressor with the performance of the models that do not contain that specific regressor. The practical computation of the performance index $\mathcal{I}_j$ can be done by estimation, i.e. to take sample models from

the model universe and averaging the regressor performances. Similarly to (3.18), the average performance of models that do not contain the j^{th}-regressor, is defined by

$$\mathcal{I}^{-}{}_{j} = \frac{1}{n_j^-} \sum_{i=1}^{n_j^-} \mathcal{J}^{-}{}_{i} \tag{3.27}$$

where n_j^- is the number of extracted models that do not contain the j^{th} regressor and $\mathcal{J}^-$ is the vector of performance indexes for only the models that do not contain the j^{th} regressor. By setting $\mathcal{I}_j = (\mathcal{I}^{+}{}_{j} - \mathcal{I}^{-}{}_{j})$, a performance index for a specific regressor ϕ_j can be computed and RIP μ_j can be updated using (3.25). If n_j^+ or n_j^- are equal to zero, $\mathcal{I}^{+}{}_{j}$ or $\mathcal{I}^{-}{}_{j}$ are set to zero for numeric reasons.

The main idea is to increase each individual RIP if the average performance of the proposed models that include a specific regressor is greater than the average performance of the models that do not include that regressor. It is expected that the RIPs vector distribution will converge to that associated with the model with the best performance.

The method does not guarantee that the $\boldsymbol{\mu}$ vector will be bounded. A final step keeps the vector elements within the $[0, 1]$ range, saturating values greater than 1 or smaller than 0.

As γ is a design parameter, a bad initial choice can lead either to slow convergence or render the method unstable under iteration. To overcome this problem, an adaptive step size solution was proposed [Falsone et al., 2015]:

$$\gamma = \frac{1}{10(\mathcal{J}_{\max} - \bar{\mathcal{J}}) + 0.1}, \tag{3.28}$$

where $\mathcal{J}_{\max}$ represents the best model performance and $\bar{\mathcal{J}}$ represents the average model performance at the current iteration. The idea behind the adaptive step size is if $\bar{\mathcal{J}}$ is far from the best model performance, γ is kept small to account for the probably great variance of that population of models. On the other hand, if $\bar{\mathcal{J}}$ is close to $\mathcal{J}_{\max}$, the sampled models have low performance variability which can serve as an indicator that model difference should be amplified.

After a certain number of iterations, the $\boldsymbol{\mu}$ vector is expected to converge to the equilibrium distribution. The final model, i.e. the system expected correct model, is composed by the regressors associated with the $\boldsymbol{\mu}$ vector elements greater than a certain threshold. Because it is assumed that the true model $f^* \in \mathcal{F}$, this threshold is set to 1, or a value close to 1.

Algorithm 2 presents the pseudo-code for the RaMSS method and will be used in

the next example, which illustrates a toy case of variable selection in linear regression problem.

Algorithm 2 The RaMSS Algorithm

Require:

$\boldsymbol{y}, N_p, \mu_{\min}, \mu_{\max}, K, \Phi = \{\phi_1, ..., \phi_m\}$

while $iter < iter_{\max}$ **do**

$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu}_0$

for i ***in*** $(1 : N_p)$ **do**

Model sampling

$\boldsymbol{\psi}(k) \leftarrow [\,]$

$\tau \leftarrow 0$

for j ***in*** $(1 : m)$ **do**

$r_j \leftarrow \text{Be}(\mu_j)$ ▷ Sample from the Bernoulli distribution **if** $(r_j = 1)$ **then**

$\boldsymbol{\psi}(k) \leftarrow [\boldsymbol{\psi}^{\mathrm{T}}(k)\ \phi_j(k)]^{\mathrm{T}}$

$\tau \leftarrow \tau + 1$

for h ***in*** $(1 : \tau)$ **do**

$\tilde{\boldsymbol{\psi}}(k) \leftarrow \text{non-redundant}(\boldsymbol{\psi}(k))$ ▷ Remove redundant terms

$\hat{\boldsymbol{y}} \leftarrow \text{Predict}(\tilde{\boldsymbol{\psi}}(k))$

$\mathcal{J}_p[i] \leftarrow e^{-K \cdot \text{MSPE}(\boldsymbol{y}, \hat{\boldsymbol{y}})}$ ▷ Model performance calculation

for j ***in*** $(1 : m)$ **do**

RIP update

$\mathcal{J}^+ \leftarrow 0;\ n^+ \leftarrow 0;\ \mathcal{J}^- \leftarrow 0;\ n^- \leftarrow 0;$ **for** i ***in*** $(1 : N_p)$ **do**

$(\phi_j(k) \in \tilde{\boldsymbol{\psi}}(k))$

$\mathcal{J}^+ \leftarrow \mathcal{J}^+ + \mathcal{J}_p[i];\ n^+ \leftarrow n^+ + 1$ **else**

$\mathcal{J}^- \leftarrow \mathcal{J}^- + \mathcal{J}_p[i];\ n^- \leftarrow n^- + 1$

$\mu_j \leftarrow \mu_j + \gamma \left(\frac{\mathcal{J}^+}{\max(n^+,1)} - \frac{\mathcal{J}^-}{\max(n^-,1)} \right)$

$\mu_j \leftarrow \max(\min(\mu_j, \mu_{\max}), \mu_{\min})$

$=0$

Example 3.3.1. *RaMSS toy example*

This example will provide a more practical approach in how the RaMSS method can be employed to select regressors to be included in a final model. The following

	ψ_1	ψ_2	ψ_3	ψ_4	ψ_5	$\mathcal{J}$
M1	✓	X	X	X	X	0.1
M2	X	✓	X	X	X	0.15
M3	X	X	X	✓	✓	0.5
M4	X	X	✓	X	✓	0.25
M5	X	✓	X	X	X	0.01
n+						
n-						
$\mathcal{I}^+$						
$\mathcal{I}^-$						
$\mathcal{I}$						

Figure 3.8. Sampled models and their respective performances.

explanation exhibits the RIP increase/decrease mechanism. This example is meant for illustrative purposes only, as the candidate regressor set is small and the sampling may not perform well in practice. Consider a candidate regressor set composed by five regressors: $\psi_1, \psi_2, \psi_3, \psi_4, \psi_5$. Five models which contains the regressors marked in green in Fig. 3.8 are sampled and their performance indexes, $\mathcal{J}$, are calculated such in Eq. 3.20.

The next step, as shown in Fig. 3.9 is to identify which models contain that specific regressor and calculate the average model performance that include the current regressor (in this case, ψ_1), ${\mathcal{I}^+}_1$ and the average model performance that do not include that specific regressor, ${\mathcal{I}^-}_1$, such as in, respectively, Eqs. 3.18 and 3.27. The final average performance, associated to that specific regressor is then calculated by $\mathcal{I}_1 = {\mathcal{I}^+}_1 - {\mathcal{I}^-}_1$. This process is held for the remaining regressors, ψ_2, ψ_3, ψ_4, ψ_5 and their respective average performances are calculated, as shown in Fig. 3.10.

The last steps, model sampling through average model performance calculation

	ψ_1	ψ_2	ψ_3	ψ_4	ψ_5	$\mathcal{J}$
M1	✓	X	X	X	X	0.1
M2	X	✓	X	X	X	0.15
M3	X	X	X	✓	✓	0.5
M4	X	X	✓	X	✓	0.25
M5	X	✓	X	X	X	0.01
n+	1					
n-	4					
$\mathcal{I}^+$	0.1					
$\mathcal{I}^-$	0.2275					
$\mathcal{I}$	-0.127					

Figure 3.9. Average model performance that contain the specific regressor ψ_1.

for each regressor, occur in one RaMSS iterate. The next is step is to update de RIP vector, using 3.25. This procedure is repeated until the RIP vector converges.

□

Example 3.3.2. *Linear regression variable selection using RaMSS*

Consider data $y(k)$ was generated from the following process (in a similar fashion from Eq. 3.17, but with four independent variables x_1, x_2, x_3, x_4):

$$y(k) = \theta_4 x_4(k) + \theta_3 x_3(k) + \theta_2 x_2(k) + \theta_1 x_1(k) + \theta_0 + e(k), \tag{3.29}$$

where the four independent variables were generated from $x_1(k), x_2(k), x_3(k), x_4(k) \sim \mathcal{U}(-1, 1)$, i.i.d. and $e(k) \sim \mathcal{N}(0, \sigma^2)$, $\sigma^2 = 0.01$. The coefficients were set to $\theta_0 = 0$, $\theta_1 = 1.5$, $\theta_2 = -3$, $\theta_3 = 1$, $\theta_4 = -0.9$.

Consider, also, 155 other independent variables, $x_5, ..., x_{160}$, that are not present in the data generation rule, i.e. $y(k)$ does not depend on those variables. Let the

	ψ_1	ψ_2	ψ_3	ψ_4	ψ_5	$\mathcal{J}$
M1	✓	X	X	X	X	0.1
M2	X	✓	X	X	X	0.15
M3	X	X	X	✓	✓	0.5
M4	X	X	✓	X	✓	0.25
M5	X	✓	X	X	X	0.01
n^+	1	2	1	1	2	
n^-	4	3	4	4	3	
$\mathcal{I}^+$	0.1	0.08	0.25	0.5	0.375	
$\mathcal{I}^-$	0.2275	0.284	0.19	0.1275	0.087	
$\mathcal{I}$	-0.127	-0.203	0.06	0.375	0.288	

Figure 3.10. Average model performances that contain the specific regressors.

candidate regressor set $\mathcal{R} = [x_1, ..., x_{160}]$ be the pool from which one should consider taking candidate regressors from. Even if the proposed model is as simple as a linear regression, the number of possible models to be evaluated is huge and not feasible in practice, so traditional methods such as AIC and BIC are not applicable. Forward regression methods can be used, although, if the model has inter-regressor dependence, the proposed model may not perform well. A randomized approach, for instance the RaMSS method, may be useful in this situation.

The idea is to build a model that can represent data $y(k)$ well, considering the candidate regressor set $\mathcal{R}$. In this example, a multiple regression model will be estimated for illustration purposes, although the method can be employed in virtually any model family, as long as the proposed models performance can be computed. In this case, the algorithm will pursue the best model in terms of mean squared error, considering that the model is in the linear multiple regression model. Note that no restrictions were made on the model size.

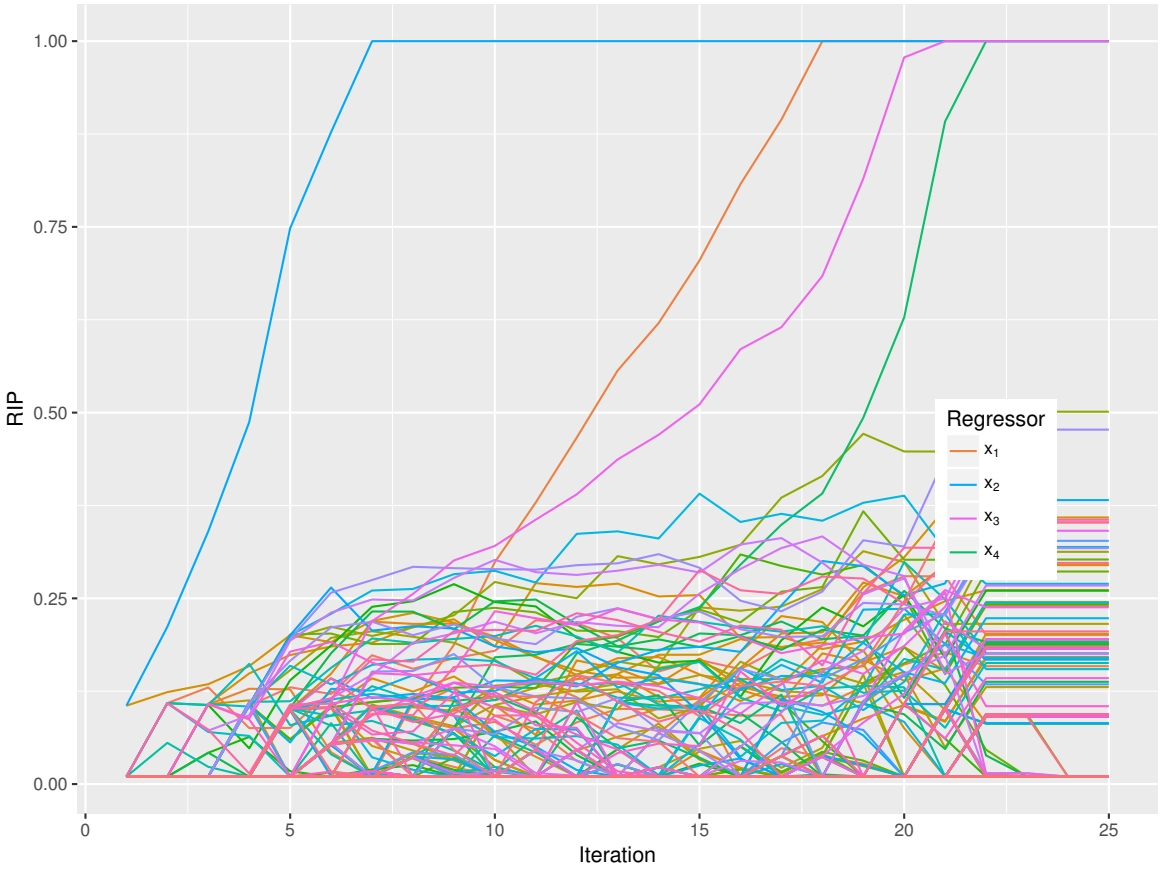


Figure 3.11. RIP evolution for the candidate regressors. True regressors highlighted.

The system represented in Eq. 3.29 was simulated for 1000 points. The method was applied to this data set with parameters $N_p = 30$, $K = 1$, $\mu_{0[1:m]} = 1/m$, $\mu_{\min} = 0.001$, $\mu_{\max} = 1$. As the algorithm runs, the regressor inclusion probability vector is progressively refined, as shown in Fig. 3.11.

Among the clear advantages of the RaMSS method is that it is able to provide regressor importance rank. It becomes evident from Fig. 3.11 that regressor x_2 is strongly related to the output, so the selection of this regressor is performed in the initial stages. This is due to the coefficient value, $\theta_2 = 3$, associated with this regressor, which is the largest value among the coefficients. This property can be useful in identification of partially correct models, where not all "true" regressors can be identified. The RIP vector converges when the evolution of the vector achieves a steady state. Note that not all regressors inclusion probabilities saturate in zero or one, but stay in a middle area. This behavior is associated to regressors that neither increase or decrease model performance and therefore should not be part of the model. The algorithm is capable of identifying the "true" set of regressors in less than 25 iterations in most cases for this candidate set size and data.

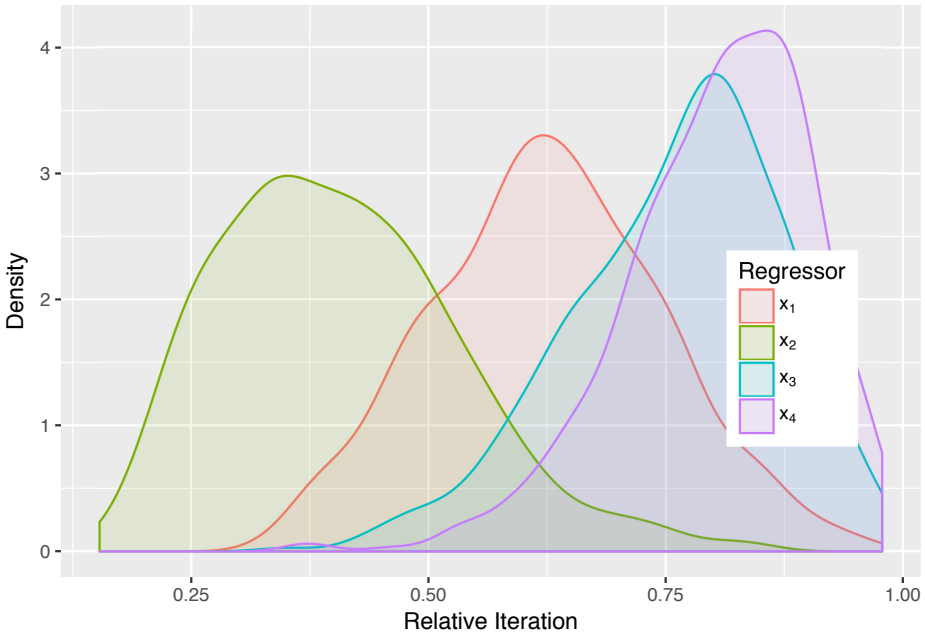
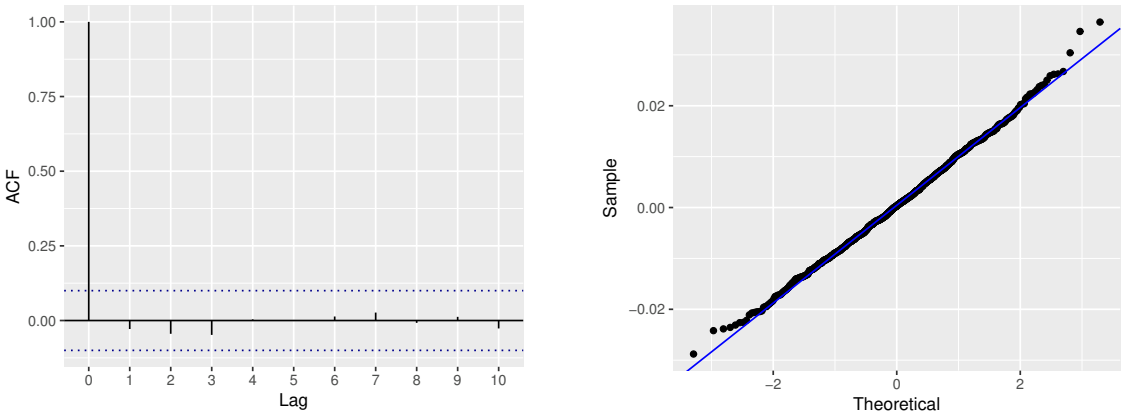


Figure 3.12. Regressor selection distribution over relative iterations for 500 RaMSS runs.

In order to evaluate the method regressor importance ranking, the same dataset was processed by RaMSS 500 times in a Monte Carlo fashion. Fig. 3.12 exhibits the relative number of iterations to the final number of iterations that the respective regressor took to be included in the model for 500 runs. In other words, this graph represents at which point of the RaMSS timeline each "true" regressor is selected. It can be seen that regressor x_2 is selected in the early stages (first iterations). Following the order, regressor x_1 is selected next. This pictures provides a "expected" timeline for regressor importance selection, whereas more important or "stronger" regressors are expected to extracted in early iterations.

Estimated parameter values were $\hat{\theta}_1 = 1.500$, $\hat{\theta}_2 = -2.999$, $\hat{\theta}_3 = 1.000$, $\hat{\theta}_4 = -0.900$, what is compatible with the values used to generate data. Parameters estimation was performed by ordinary least squares. Fig. 3.13 shows the residual plot ACF and residual normal-QQ plot. It can be seen that residuals are uncorrelated and can be assumed normally distributed.

□



(a) Auto Correlation Plot for residuals.

(b) Normal Quantile-Quantile plot for residuals.

Figure 3.13. Analysis of residuals plots.

Chapter 4

Model Selection Task for the NARMAX Model

4.1 Introduction

This chapter presents the model structure selection task in the context of NARMAX models. The NARMAX model is presented in the first subsection and some important definitions are introduced. This chapter reports the main contributions of this work. The RaMSS method was extended to cope with NARMAX models and additional features, such as parallel processing and the possibility of inclusion of prior model information, were also implemented.

4.2 The NARMAX model

4.2.1 Model Definition

The Nonlinear Autoregressive Moving Average with Exogenous Input(NARMAX) model can represent a wide class of nonlinear systems [Leontaritis and Billings, 1985a,b]. It can be defined as:

$$\begin{aligned} y(k) = F\,[&y(k-1), y(k-2), \ldots, y(k-n_y), \\ &u(k-\tau), u(k-\tau-1), \ldots, u(k-n_u), \\ &e(k-1), e(k-2), ..., e(k-n_e)] + e(k), \end{aligned} \tag{4.1}$$

where $y(k)$, $u(k)$, $e(k)$ are the system output, input and noise, respectively and n_y, n_u, n_e are the respective maximum lags and τ is the time delay. F is some nonlinear

function.

As it becomes evident from Eq. 4.1, $e(k)$ represents every effect that cannot be accounted for in the model, in other words, $F[\cdot]$ parameters do not depend on $e(k)$.

Let $\mathcal{R} = \{\psi_1, \psi_2, ..., \psi_{n_\mathcal{R}}\}$ denote the set of $n_\mathcal{R}$ regressors from a NARMAX structure. The dimension of $\mathcal{R}$, i.e. the number of combinations of NARMAX monomial terms is given by the ℓ-combination with repetitions such:

$$n_\mathcal{R} = n_\theta \ell = \binom{n_\theta + \ell - 1}{\ell},$$

where $n_\theta = n_y + (n_u - \tau + 1) + n_e + 1$ and ℓ is the degree of nonlinearity, in the case of polynomial models.

Assuming that $F[\cdot]$ can be represented as a polynomial function, as defined in [Aguirre, 2015], the deterministic part of Eq. 4.1 can be expanded as a non-linear composition lagged input and output terms, with degree varying in the range $1 \leq m \leq \ell$. Each term, ψ_i, with degree m, is composed by a factor with p degree of $y(k-i)$ and a factor with $(m-p)$ degree of $u(k-i)$. Finally, a parameter $c_{p,m-p}(n_1, ..., n_m)$ multiplies the term:

$$y(k) = \sum_{m=0}^{\ell} \sum_{p=0}^{m} \sum_{n_1, n_m}^{n_y, n_u} c_{p,m-p}(n_1, ..., n_m) \prod_{i=1}^{p} y(k-n_i) \prod_{i=p+1}^{m} u(k-n_i), \tag{4.2}$$

where

$$\sum_{n_1, n_m}^{n_y, n_u} \equiv \sum_{n_1=1}^{n_y} ... \sum_{n_m=1}^{n_u} \tag{4.3}$$

The model noise terms, which contain $e(k-n_i)$ terms are formed in a similar manner. The polynomial NARMAX model can be conveniently defined in a more compact notation:

$$y(k) = \boldsymbol{\psi}_{y,u,e}^{\mathrm{T}}(k-1)\hat{\boldsymbol{\theta}} + e(k) \tag{4.4}$$

The following example illustrates the regressor vector formation.

Example 4.2.1. *Regressor set from a polynomial NARMAX model*

Consider a NARMAX model with $n_y = 2$, $n_u = 2$ and $n_e = 2$. For a degree of non-linearity $\ell = 2$, the regressor vector, including the constant term, has the following terms:

$$
\begin{aligned}
\boldsymbol{\psi}(k-1) = [1 \\
& y(k-1)\ y(k-2)\ u(k-1)\ u(k-2)\ e(k-1)\ e(k-2) \\
& y(k-1)^2\ y(k-1)y(k-2)\ y(k-1)u(k-1) \\
& y(k-1)u(k-2)\ y(k-1)e(k-1)\ y(k-1)e(k-2) \\
& y(k-2)^2\ y(k-2)u(k-1) \\
& y(k-2)u(k-2)\ y(k-2)e(k-1)\ y(k-2)e(k-2) \\
& u(k-1)^2\ u(k-1)u(k-2)\ u(k-1)e(k-1)\ u(k-1)e(k-2) \\
& u(k-2)^2\ u(k-2)e(k-1)\ u(k-2)e(k-2) \\
& e(k-1)^2\ e(k-1)e(k-2) \\
& e(k-2)^2]^{\mathrm{T}}
\end{aligned}
$$

□

4.2.2 Term Clustering

It should be noted that the term coefficients in Eq. 4.2 depend on the sampling time, T_s and should be represented as $c_{p,m-p}(T_s, n_1, ..., n_m)$. For simplicity, the argument T_s is omitted. If the sampling time T_s is short enough and the data window is smooth the following approximations can be made

$$
\begin{aligned}
y(k-1) = y(k-2) = ... = y(k-n_y) = \bar{y} \\
u(k-1) = u(k-2) = ... = u(k-n_u) = \bar{u},
\end{aligned}
\tag{4.5}
$$

therefore Eq. 4.2 can be rewritten as

$$
y(k) = \sum_{n_1,n_m}^{n_y,n_u} c_{p,m-p}(n_1, ..., n_m) \sum_{m=0}^{\ell} \sum_{p=0}^{m} y(k-1)^p u(k-1)^{m-p}. \tag{4.6}
$$

From Eq. 4.6, the following definitions arrise.

Definition 1. [Aguirre and Billings, 1995b] "The constants $\sum_{n_1,n_m}^{n_y,n_u} c_{p,m-p}(n_1, ..., n_m)$ that appear on Eq. 4.6 are the *term clusters* $\Omega_{y^p u^{m-p}}$ which contain terms in the form $y(k-i)^p u(k-j)^{m-p}$for $m = 0, ..., \ell$ and $p = 0, ..., m$. Such coefficients are called *cluster coefficients* and represented as $\sum_{y^p u^{m-p}}$."

Definition 2. [Corrêa, 2001] "The set of all terms in the form $y^p(k-d)u^m(k-j)$, $m + p \leq \ell$ is called d-*cluster* and is represented as $\Omega_{y_d^p u^m}$. The sum of respective coefficients

is called d-*coefficient* and is represented as $\sum_{y_d^p u^m}$".

Term cluster is important to describe how model gain varies in various operating points. d-clusters are useful to define how the model dynamics respond to changes in operating points [Aguirre, 2007].

4.3 RaMSS-m Method

The RaMSS method, presented in Sec. 3.3.1, although reported to be modifiable to handle NARMAX models, was originally designed to handle NARX models only. This section is the main contribution of this dissertation and introduces an extension of the RaMSS method. The new method, briefly RaMSS-m, was implemented using the basic idea behind the RaMSS method, but provides additional features beside being able to deal with NARMAX models. The RaMSS-m was first introduced in [Retes and Aguirre, 2018] and is a direct result of this work. Those features include possibly reduced parameter estimation bias, by the inclusion of moving average terms in the model, a parallel processing scheme that allows faster model selection and parameter estimation and a modified regressor inclusion probability rule, what allows the model designer to interfere in the model selection task when prior model information is available. The method innovative aspects will be discussed in the following subsections.

4.3.1 Inclusion of Moving Average Terms

The RaMSS method, presented in details in [Falsone et al., 2015], was applied to NARX models only. The main characteristic of RaMSS that restricts it use in NARMAX models is the employed parameter estimator nature. RaMSS was implemented using ordinary least squares as parameter estimator (see Sec. 2.2.5.2), which is appropriate for NARX models [Billings, 2013], as it fits the *linear in the parameters* model class. The inclusion of moving average (MA) terms, required by the more flexible NARMAX model, renders the final model a *non-linear in the parameters* nature and demands a more flexible parameter estimation algorithm. The proposed RaMSS-m method implements, for the NARMAX case, an extended least squares estimator (ELS) for model parameter estimation. The ELS method, shown in Sec.2.2.5.4, is applicable to a more general model class that includes non-linear moving average (N)MA terms. This estimator provides, in addition to the (N)MA estimation, less susceptibility to parameter estimation bias for colored noise in the regression equation or even white noise added to the output, which is a typical scenario for measurement noise.

When approaching system identification procedure in a black-box fashion, little or no prior information is available apart from the data. One may extract initial information from data, such as the autocorrelation function, in order to produce insights about data behavior. Those methods, although appropriate for linear models, have their limitation when applied with nonlinear models. The inclusion of (N)MA terms in the candidate regressors makes room for wider model dynamic behavior, and also addresses desirable properties verified in linear models, such as parameter estimation bias reduction in case the data is corrupted by correlated noise [Chen et al., 1989; Billings, 2013].

The following example illustrates the application of ELS to reduce parameter estimation bias.

Example 4.3.1. *Parameter Estimation Bias Reduction*

Consider the following system, exposed to strongly correlated noise:

$$\begin{aligned} S_{4.1} : y(k) &= 0.7y(k-1)u(k-1) - 0.5y(k-2) \\ &+0.6u^2(k-2) - 0.7y(k-2)u^2(k-2) + e(k), \end{aligned} \tag{4.7}$$

with $e(k) = 0.5e(k-1) + \nu(k)$, $\nu(k) \sim \mathcal{N}(0, 0.2)$. Data was collected when system achieved equilibrium.

This system data was presented to the original RaMSS method and results were analyzed. Regressors were generated taking the lags of input and output, with maximum lags being $n_y = 4$ and $n_u = 4$. A first used investigation tool was the autocorrelation function plot of the residuals shown in Fig. 4.1. The ACF plot suggests strong residual correlation indicating that not all features of the data were correctly accounted in the model, as the RaMSS method was programmed to identify NARX models only.

Parameter estimation will suffer loss of accuracy (bias) when estimating a model that does not contain MA terms where the data contains correlated noise. To explore this bias induced effect, this analysis was performed using a Monte Carlo approach. System described in Eq. 4.7 was simulated 500 times. Each dataset was fed in the RaMSS algorithm and results from the OLS estimator were computed. Fig. 4.2 exhibits the estimated parameter histograms for the true regressors. In spite of being able to extract the true regressors, models generated from RaMSS lack of inclusion of MA terms induce parameter estimation bias, once the method used for parameter estimation in RaMSS, the OLS, produces biased estimations in this case. It can be noticed that the parameter associated to regressor $y(k-2)$ exhibits the strongest bias. This is an expected result as noise is added to the output y and is not present in the *exogenous*

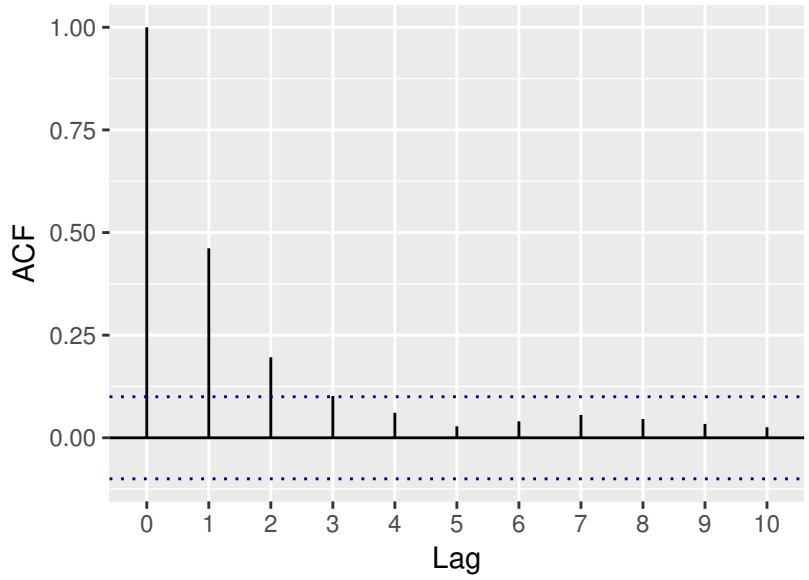


Figure 4.1. Autocorrelation function (ACF) of the residuals of a model estimated using RaMSS from the data produced by system $S_{4.1}$. The horizontal dotted lines indicate the 95% confidence band. The presence of significant correlation indicates bias.

input. The remaining regressors contains *exogenous* input lagged terms, i.e. $u(k-1)$ and $u(k-2)$, what makes bias effect less evident.

The RaMSS-m method is capable of handling a more flexible model and allows (N)MA terms to be present in the final model, which can be useful to reduce parameter bias. The same data were presented to the RaMSS-m method, in a similar Monte Carlo fashion. Fig. 4.3 shows estimated parameter histograms, when allowing the RaMSS-m method to select (N)MA terms up to lag $n_e = 3$. It becomes evident that bias effect is strongly reduced by letting the model selector explore models containing (N)MA terms. Parameter associated to regressor $y(k-2)$ is much less affected by bias.

□

4.3.2 Parallel Processing

Parallel processing or parallel computing is a technique for computing large problems in which calculations are performed in a distributed way, concurrently. Large problems may be subdivided in smaller ones which can be solved faster by using more processors instead of a single one. This is desirable because of processors power consumption and heat generation have become a concern in recent years as processor frequency increase become more difficult by technology limitation. The availability of multicore processors, which, nowadays, is a standard in both the personal computer

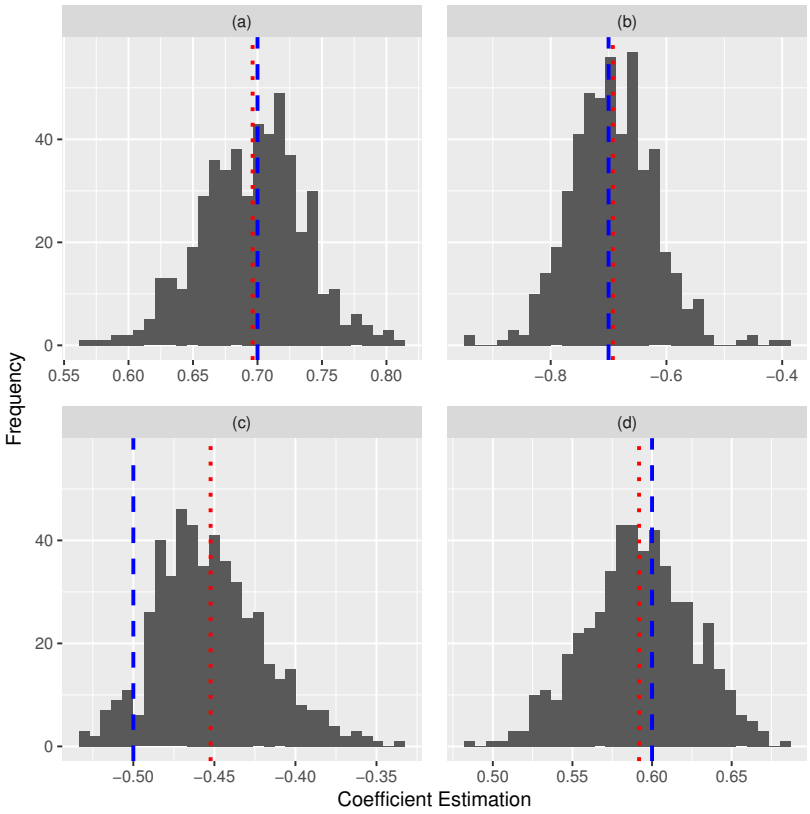


Figure 4.2. Histograms of parameters estimated over 500 Monte Carlo runs using RaMSS. True parameter values are indicated by the dashed blue line, whereas the mean of estimated values is shown as a dotted red line. Parameters of (a) $y(k-1)u(k-1)$, (b) $y(k-2)u^2(k-2)$, (c) $y(k-2)$, (d) $u^2(k-2)$.

and corporate server markets, demands the implementation of algorithms that are programmed considering parallel computation specificities and, thus, take advantage of parallelism.

The parameter estimator algorithm employed in RaMSS-m (Extended Least Squares), due its iterative nature, is more computer demanding than OLS. This can lead to slow code code computation, specially when dealing with large models. To take advantage of RaMSS-m power and, at the same time, get competitive model computation time, the RaMSS-m method was conceived to implement parallel processing. The first step in parallel computing implementation is to identify which piece of code is independent from each other and what information those pieces of code have to exchange.

A distributed randomized regressor selection (dMSS), was presented in [Avellina et al., 2017]. Their main idea is to distribute the model structure selection task among several processors, in a way that each processor executes a small, local, optimization task. Information is shared between the processors and when the communication round

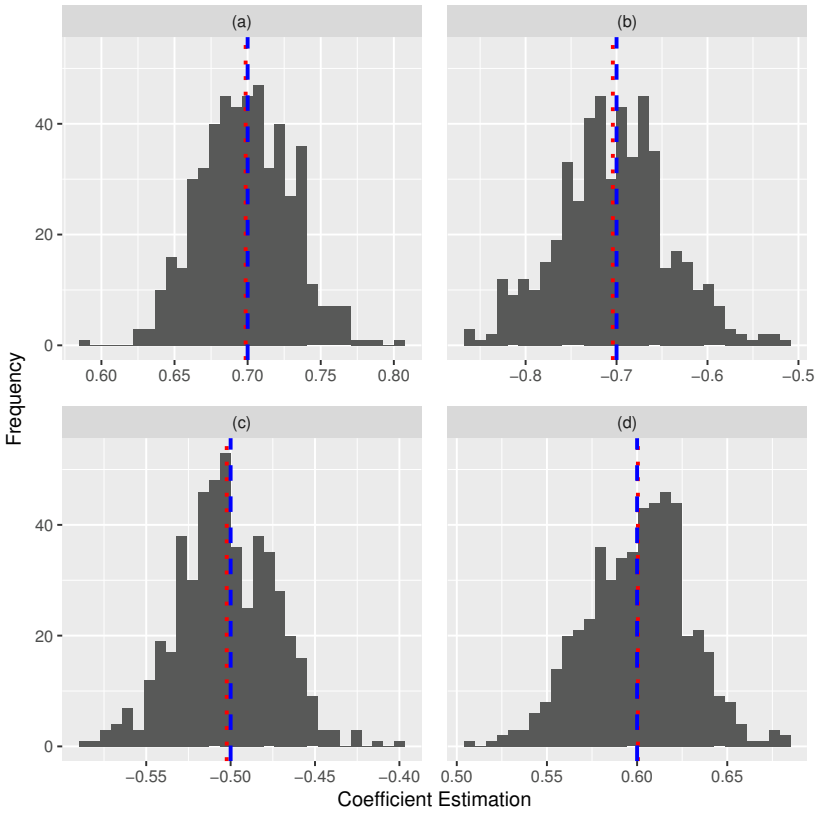


Figure 4.3. Histograms of parameters estimated over 500 Monte Carlo runs using RaMSS-m. True parameter values are indicated by the dashed blue line, whereas the mean of estimated values is shown as a dotted red line. Parameters of regressors (a) $y(k-1)u(k-1)$, (b) $y(k-2)u^2(k-2)$, (c) $y(k-2)$, (d) $u^2(k-2)$.

is finished, each processor verifies that its local solution does not violate any of the shared constraints. If this condition is not satisfied, the processor must rerun the optimization process and update the solution. After a finite number of rounds, the shared constraints are not violated by any of the local solutions and the algorithm stops.

The RaMSS-m parallel processing method followed the idea of distributing the MSS task between processors and information sharing, as presented by [Avellina et al., 2017]. The RaMSS-m method, however, constrains each processor solution by guiding, at each round, the desired model structure. Model sampling and performance index calculations, for each of the N_p models, are independent and, thus, can be computed in parallel, only subjected by the current regressor inclusion probability that is shared among processors. In other words, the individual processor job is to return a sampled model, constrained to the RIP vector, and the performance metric associated to that model.

Unlike the dMSS method, the RaMSS-m implements a omniscient supervisor, re-

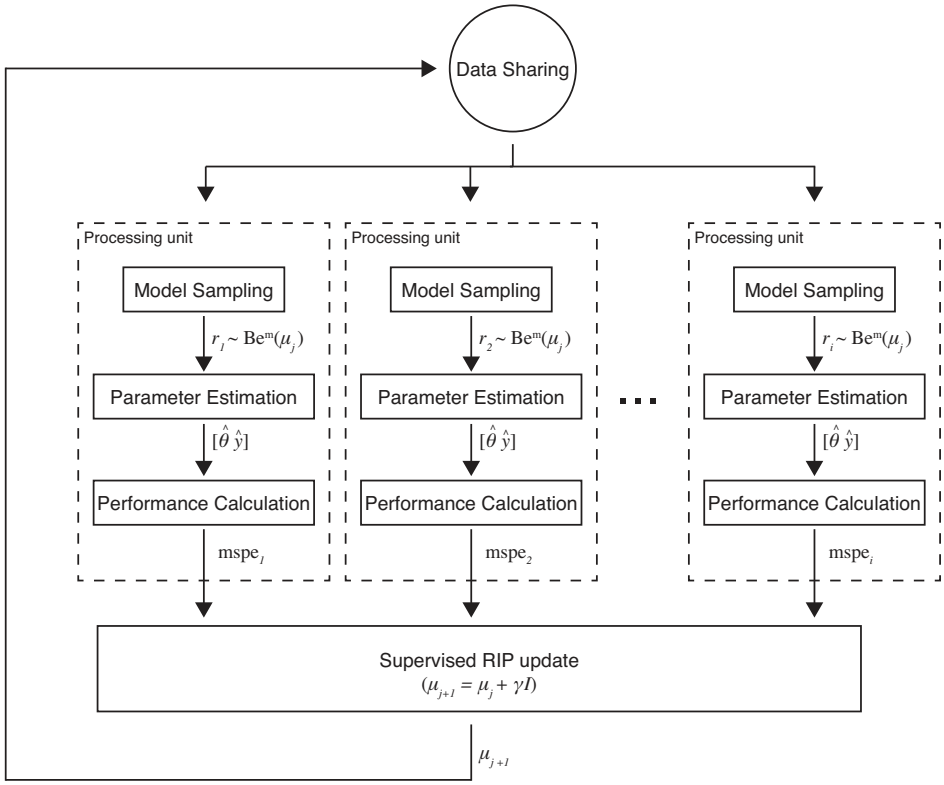


Figure 4.4. Parallel processing scheme adopted in RaMSS-m.

sponsible to bind the resulting models performances and update the RIP vector. The updated RIP vector is then fed again to the processors and new models are sampled at each unit. Processing stops when convergence is achieved when the RIP vector stabilizes or a certain number of iterations is completed. The final model structure from RaMSS-m is obtained in an oversighted fashion, in which the process supervisor is omniscient sharing the desired parameters with the parallel units and then collecting and processing the results accordingly to defined rules, rather than the resulting consensual interaction from the dMSS method.

Additional processing overhead of instantiating and sharing data between multiple processor threads is clearly overcome by the faster aggregate model generation and performance calculation. Fig. 4.4 illustrates the RaMSS-m parallel processing data flow.

4.3.3 Modified RIP Update Rule

If the model designer has prior information regarding model structure, the RaMSS original method provided no means of including this information in the MSS other than setting initial RIP values. In RaMSS-m, the algorithm is capable of incorporating ad-

ditional information about the model structure by means of setting RIPs, at each iteration, to the desired values. RIPs can be set to specific values (e.g. zero or one) to force some of the regressors to be included or excluded in the final model. This approach can be useful, for example, to force MA terms to be present in the parameter estimation phase of a NARX model, and, therefore, reduce parameter estimation bias. The modified RIP rule can also incorporate prior knowledge by forcing or eliminating certain term clusters depending, for instance, on the system static characteristics [Aguirre et al., 2002].

4.3.4 RaMSS-m Algorithm

By incorporating the features presented in previous sections, the RaMSS-m method is capable of selecting models from a wider class of model families and also let the model designer have more control over the model selection task. Alg. 3 shows the method pseudocode.

4.4 Simulation Results

The first system investigated is taken from [Falsone et al., 2015]:

$$\begin{aligned} S4 : y(k) &= 0.7y(k-1)u(k-1) - 0.5y(k-2) \\ &+0.6u^2(k-2) - 0.7y(k-2)u^2(k-2) + e(k), \end{aligned}$$

with $u(k) \sim \mathcal{U}(-1,\, 1)$ and $e(k) \sim \mathcal{N}(0,\, 0.004)$, where $\mathcal{U}(a,\, b)$ indicates a uniform distribution in the range defined from a to b; and $\mathcal{N}(x,\, \sigma^2)$ indicates Gaussian distribution with mean x and variance σ^2.

Two other models where included in the simulations:

$$\begin{aligned} S_{ARMAX} : \quad & 0.5y(k-1) - 0.6y(k-2) + \\ & +0.7u(k-1) + 0.3e(k-1) + e(k) \\ S_{NARMAX} : \quad & y(k) = 0.7y(k-1)u(k-1) - 0.5y(k-2) + \\ & +0.6u^2(k-2) + 0.8y(k-2)e(k-4) + e(k), \end{aligned}$$

with $u(k) \sim \mathcal{U}(-1,\, 1)$, $e(k) \sim \mathcal{N}(0,\, \sigma^2)$. Data was collected when system achieved equilibrium.

In order to evaluate the RaMSS-m correct regressor selection power, each system data was run twenty times through the method. Results are displayed in Tab. 4.1.

Algorithm 3 The RaMSS-m Algorithm

Require:
$\boldsymbol{u}, \boldsymbol{y}, N_p, \mu_{\min}, \mu_{\max}, \boldsymbol{\mu}_{\text{forced}},$
$n_y, n_u, n_e, K, \boldsymbol{\mu_0}, iter_{\max}$
while $iter < iter_{\max}$ **do**
$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu}_0$
parfor i **in** $(1 : N_p)$ **do** ▷ Generate models in parallel
$m \leftarrow n_y + n_u + n_e + 1$
$\boldsymbol{r}_i \leftarrow \text{Be}^m(\boldsymbol{\mu})$ ▷ Model sampling
$[\hat{\boldsymbol{\theta}}\ \hat{\boldsymbol{y}}] \leftarrow \text{ELS\_estimate}(\boldsymbol{u}, \boldsymbol{y}, \boldsymbol{r}_i, m)$
$\mathcal{J}_p[i] \leftarrow e^{-K \cdot \text{MSPE}(\boldsymbol{y}, \hat{\boldsymbol{y}})}$ ▷ Model performance calculation
end parfor
for j ***in*** $(1 : m)$ **do**
Supervised RIP update
$\mathcal{J}^+ \leftarrow 0;\ n^+ \leftarrow 0;\ \mathcal{J}^- \leftarrow 0;\ n^- \leftarrow 0;$ **for** i ***in*** $(1 : N_p)$ **do**
$(\boldsymbol{r}_i[j] = 1)$
$\mathcal{J}^+ \leftarrow \mathcal{J}^+ + \mathcal{J}_p[i];\ n^+ \leftarrow n^+ + 1$ **else**
$\mathcal{J}^- \leftarrow \mathcal{J}^- + \mathcal{J}_p[i];\ n^- \leftarrow n^- + 1$

$\mu_j \leftarrow \mu_j + \gamma \left(\frac{\mathcal{J}^+}{\max(n^+,1)} - \frac{\mathcal{J}^-}{\max(n^-,1)} \right)$
$\mu_j \leftarrow \max(\min(\mu_j, \mu_{\max}), \mu_{\min})$

$\boldsymbol{\mu} \leftarrow \text{update}(\boldsymbol{\mu}, \boldsymbol{\mu}_{\text{forced}})$ ▷ Force RIPs to desired schema

function ELS_ESTIMATE$(\boldsymbol{u}, \boldsymbol{y}, \boldsymbol{r}, m)$
$\hat{\boldsymbol{\theta}} \leftarrow [0\ 0\ 0,\ ...0]^m$ **while** *(stop criterion)* **do**
$\hat{\xi} = \boldsymbol{y} - \boldsymbol{\phi}\hat{\boldsymbol{\theta}}$
$\boldsymbol{\phi} \leftarrow \text{combination\_matrix}(\boldsymbol{u}, \boldsymbol{y}, \hat{\xi}, n_y, n_u, n_e)[\boldsymbol{r}]$
$\hat{\boldsymbol{\theta}} = [\boldsymbol{\phi}^{\mathrm{T}}\boldsymbol{\phi}]^{-1}\boldsymbol{\phi}^{\mathrm{T}}\boldsymbol{y}$

$\hat{\boldsymbol{y}} \leftarrow \boldsymbol{\phi}\hat{\boldsymbol{\theta}}$
return $[\hat{\boldsymbol{\theta}}\ \hat{\boldsymbol{y}}]$
end function

function UPDATE$(\boldsymbol{\mu}, \boldsymbol{\mu}_{\text{forced}})$ **for** j ***in*** $(1 : m)$ **do**
$(\mu_{\text{forced}}[j]\ \neq NULL)$
$\mu[j] \leftarrow \mu_{\text{forced}}[j]$

return $\boldsymbol{\mu}$
end function

Table 4.1. Summary of RaMSS-m algorithm performance.

Model	Noise Level	Correct NAR Terms	Correct NMA terms	Average Model Size
S_4	0.04	100%	-	5.9
S_{ARMAX}	0.3	100%	100%	4.4
S_{NARMAX}	0.3	100%	85%	4.2
S_{NARMAX}	0.4	100%	85%	4.2
S_{NARMAX}	0.5	100%	100%	4.7

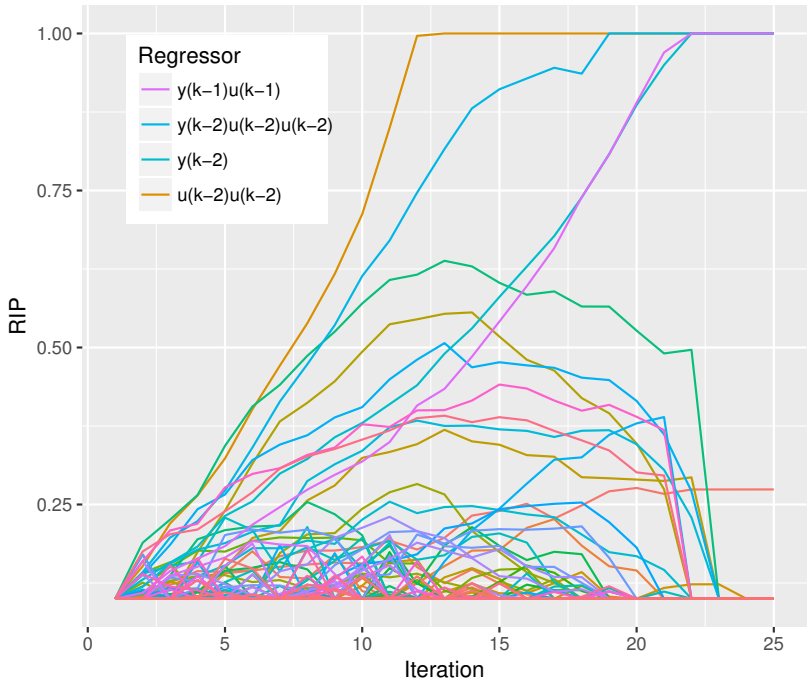


Figure 4.5. Typical RaMSS-m RIP evolution.

Meta-parameters were set to $n_y = n_u = n_e = 4$. The maximum degree of nonlinearity for the NARMAX model was set to $\ell = 3$. The regressor set which the RaMSS-m algorithm will take sample from, $\mathcal{F}$, is composed of 455 terms. The method maximum number of iterations was set to $iter_{\max} = 120$, $N_p = 100$ and $K = 1$. The proposed method was run on the Amazon Elastic Cloud Computing (EC2) platform. The EC2 allowed to instantiate a virtual machine with optimized processing power. The algorithms were implemented in R Statistical Software version 3.3.3. Fig. 4.5 shows a typical RaMSS-m run for system S_4.

The RaMSS-m method could correctly identify all nonlinear AR terms in the evaluated models. For the nonlinear MA terms, the method could identify the correct

structure in 85% of the simulations for lower noise level systems. For the system with higher noise level, the method was able to correctly identify the true model structure in 100% of the times. In order to perform correct identification of MA terms it is required that the noise correlation is enough to excite the algorithm and those terms be distinguished from random noise.

The proposed RaMSS-m method was shown to still perform well under systems that not include (N)MA terms. The correct regressors from the true model were identified for those models. The inclusion of (N)MA terms in the possible regressor universe caused the RaMSS-m method to overestimate model size for simpler systems (S_4 and S_{ARMAX}), with little impact on prediction error.

The ELS method, used as the estimator for the NARMAX model in the modified RaMSS algorithm, generates processing overhead and larger computation times compared to the ordinary least squares estimator. The extended regressor matrix must be updated with MA terms at each ELS iterate. MA terms are hence computed at each ELS iterate using the residuals which are progressively refined. Also at each ELS iterate, a matrix inversion is performed.

One of the most important results of the RaMSS (and, by extension, the RaMSS-m) methods is that regressor importance is progressively refined until convergence. Even when convergence was not achieved yet, the RIP evolution path can be useful. The RIP evolution picture can be very informative on regressor importance and on the interaction between regressors being included or excluded in the models. The RIP evolution picture acts as an average model overview. "Strong" regressors, i.e. regressors that most contribute to model prediction error reduction, easily converge to the maximum, where regressors that increase model prediction error are rapidly set to minimum. Regressors that have little impact on model predict error, in other words, regressors that do not significantly increase or decrease the prediction error, tend to achieve an equilibrium RIP which does not go to the maximum or the minimum.

Finally, the interaction among regressors can also be evaluated by analyzing the RIP evolution. In early iterations, several regressors have high probability of being included in the model. With the inclusion of "strong" regressors in the model and the resulting decrease in model prediction error, regressor interactions become more evident. For instance, in Fig. 4.5, around the 23rd iteration a "true" regressor RIP goes to 1, influencing "spurious" regressors RIP to evolve to the floor level. This happens because the inclusion of a "true" regressor reduces the prediction error which in turn is used to update the RIPs.

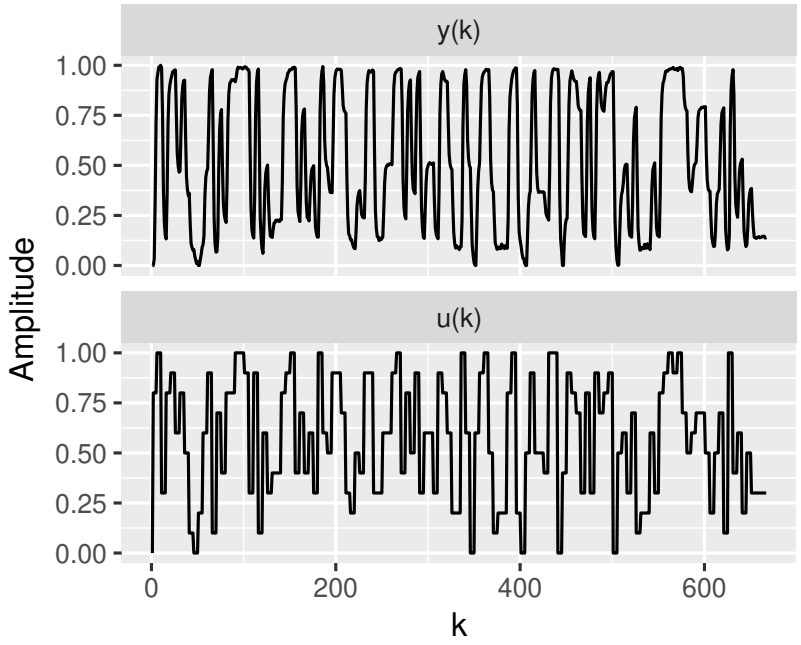


Figure 4.6. Estimation data collected from a tank system, where the output $y(k)$ is the measured flow and the input $u(k)$ is the valve driving signal.

4.5 Real Data Results

In this section, the proposed modified method will be submitted to real world data in order to evaluate the algorithm potential ability to identify nonlinear models from a more uncontrolled environment.

The chosen system data was collected from a control valve installed in a Interacting Tank System, described in details in [Braga, 1994]. The pneumatic valve is driven by an input signal which defines the valve flow. Hence, the input signal, $u(k)$ drives the valve piston position, whereas $y(k)$ is the output flow measured by a turbine sensor. Input and output data were unity-based normalized. The input signal, $u(k)$ was designed so the system was persistently excited. Collected data is composed of 1000 samples. The dataset was split into estimation, test and validation sets, composed, respectively, by $N = 667$, $N = 167$ and $N = 166$ consecutive points. Fig. 4.6 shows the estimation data.

Model performances on validation data were evaluated using e_{MSE} (Eq. 2.44 and e_{MAPE} (Eq. 2.45).

4.5.1 Models obtained with ERR

In the procedures below, models with various process and noise terms were tested. Terms were chosen using the ERR criterion and the model with lowest error index over the *test* data was used on *validation* data to be compared with the model obtained using RaMSS-m.

For $n_y = n_u = n_e = 3$ and $\ell = 1$, the model with lowest e_{MAPE} (and e_{MSE}) on test data was the one with 6 terms. In this case there are only 7 possible process terms and 3 noise terms. On validation data, the 6-process-term model attained $e_{\text{MAPE}} = 7.55\%$.

For $n_y = n_u = n_e = 3$ and $\ell = 2$, the model with lowest e_{MAPE} on test data was the one with 20 process terms. The model with 23 terms had the lowest e_{MSE}. On validation data, the 20-term model attained $e_{\text{MAPE}} = 6.04\%$.

For $n_y = n_u = n_e = 3$ and $\ell = 3$, the model with lowest e_{MAPE} (and e_{MSE}) on test data was the one with 23 terms. On validation data, this model attained $e_{\text{MAPE}} = 3.86\%$ and was the ERR-estimated model with the best performance, using this index.

4.5.2 Models obtained with RaMSS-m

Model performance index was chosen as a mix of *one-step-ahead* prediction error and *free-run*-simulation error. This approach was necessary in order to speed up the algorithm convergence because of the slow varying characteristics of the input.

Chosen method parameters were $K = 500$, $N_p = 100$, $\alpha = 0.5$, where the K parameter is used to highlight the difference between models.

In this case, a different criterion was employed for model performance evaluation. It is argued in [Piroddi and Spinelli, 2003] that using the *free-run*-simulation error as a model evaluation criterion can improve the robustness of the model structure selection process in partial identifiability conditions. In analogy to [Aguirre et al., 2010], a combined performance index will be used:

$$\mathcal{J} = \alpha \mathcal{J}_p + (1 - \alpha)\mathcal{J}_s, \tag{4.8}$$

where

$$\mathcal{J}_p = e^{-K \cdot \text{MSPE}} \tag{4.9}$$

$$\mathcal{J}_s = e^{-K \cdot \text{MSSE}}, \tag{4.10}$$

Table 4.2. Performance of models obtained using ERR and RaMSS-m. e_{MAPE} was computed on validation data.

Model	Process Terms	Noise Terms	Total Model Size	e_{MAPE}
ERR ($\ell = 1$)	6	0	6	7.55%
RaMSS-m ($\ell = 1$)	6	0	6	6.96%
ERR ($\ell = 2$)	20	3	23	6.04%
RaMSS-m ($\ell = 2$)	12	9	21	5.80%
ERR ($\ell = 3$)	18	5	23	3.86%
RaMSS-m ($\ell = 3$)	32	5	37	2.17%

and MSPE is the mean squared prediction error, MSSE is the mean squared simulation error (*free-run*-simulation error). User defined parameter $\alpha \in [0, 1]$ balances the contribution of *one-step-ahead* prediction error and *free-run*-simulation error to the model performance. Meta-parameters were chosen as in Sec. 4.5.1. Tab. 4.2 shows the performance comparison between models obtained with the ERR and the RaMSS-m methods.

4.5.3 Model comparison

The best models, selected by the ERR and the RaMSS-m methods, were the ones with nonlinearity degree $\ell = 3$.

The RaMSS-m method selected 37 terms out of 220 candidate regressors in the best resulting model. ERR selected 23 terms out of 220 candidates. From Tab. 4.2, it can be seen that the RaMSS-m method resulting model outperformed the ERR regressor selection approach in terms of e_{MAPE} in all three scenarios ($\ell = 1$, 2, 3).

Models obtained from both methods have similar performances for $\ell = 1$. Neither RaMSS-m or ERR have selected any noise terms for this scenario. Although both methods have selected the same model size, RaMSS-m has a slightly smaller prediction error than ERR.

For $\ell = 2$, the RaMSS-m method selected a smaller number of process terms than the ERR method, whilst it increased the number of noise terms in the final model to achieve low prediction error.

The most significant result was achieved in the uttermost complex model structure ($\ell = 3$). Although the model size from the one obtained with the RaMSS-m method is greater than the model estimated by the ERR method, the prediction error from

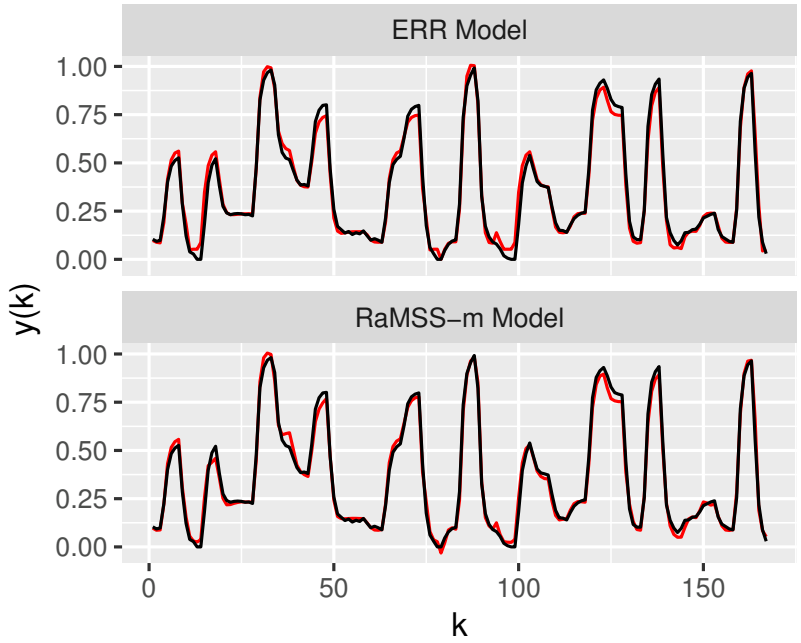


Figure 4.7. Pneumatic valve model prediction ($\ell = 3$) for ERR Model and RaMSS-m Model. Validation output flow data (black), *free-run* simulation (red).

the RaMSS-m model is smaller. Fig. 4.7 shows the validating data and *free-run* best model simulation for both ERR and RaMSS-m methods. Models predicted responses differences are not too evident from the graph as data signal to noise ratio is large.

Models selected by ERR and RaMSS-m methods were evaluated considering term clusters. Tab. 4.3 shows the comparison between models. It can be seen that models obtained from the ERR method and from the RaMSS-m method result in similar clusters. The main difference is that the RaMSS-m method selected terms from the cluster Ω_{y^2} whereas the ERR method did not.

Table 4.3. Comparison between selected term clusters (process terms only) by the ERR and RaMSS-m models. $\sum$ columns represent the cluster coefficient.

Cluster	ERR	$\sum$	RaMSS-m	$\sum$
const	✓	0.0526	✓	0.0227
Ω_u	✓	-0.4804	✓	0.1707
Ω_{u^2}	✓	-0.1258	✓	0.3184
Ω_{u^3}	✓	0.4105	✓	1.0682
Ω_{y^2}	✗	-	✓	1.3276
Ω_{y^2u}	✓	2.0847	✓	5.6499
Ω_{y^3}	✓	-0.1205	✓	-2.7197
Ω_{yu}	✓	-0.6166	✓	0.7003
Ω_{yu^2}	✓	1.0218	✓	-5.5445

Chapter 5

Conclusion

5.1 Final Considerations

This dissertation presented some background material on the main topics of model identification. The model structure selection, the principal topic of this research, was introduced in more details considering the classical approaches and some more recent ones, based on Monte Carlo simulations.

The main contribution of this work is the development of a modified RaMSS method, the RaMSS-m. The new method could correctly identify a wider class of models – the NARMAX models, and also presents additional features such as parallelism and the possibility of prior model information inclusion. The proposed method performance on simpler models, such as the NARX models and ARMAX models, was compatible to classical approaches. In the case of data containing correlated noise, as expected, the RaMSS method suffers not only from parameter bias but also this poses problems in the structure selection step. Those shortcomings are overcome by the RaMSS-m. The parallelization feature may overcome the additional processing time overhead due to the chosen estimator iterative nature.

The RaMSS-m method is also useful to identify the "strong" regressors in the pool as long as the ones with "weak" impact on the model. The RIP evolution clearly segregates those regressors after a "burn-in" number of iterates, i.e. when the model probability distribution is close to the limit one. This "insightful" approach to the structure selection problem seems to be one of the greatest assets of randomized algorithms when compared to more traditional approaches.

The new method was also employed in a real world system, a pneumatic valve. Results showed that although the models obtained with RaMSS-m were not always the ones with less parameters, in all the investigated cases they were those with better

performances when compared with models obtained with the ERR method, specially for the more complex models. This is likely to be a consequence of the use of simulation error as a criterion for updating the regressor inclusion probability functions.

Nonetheless, when data are generated by a real world process, there is no such a thing as "true" model, as it is virtually impossible that all the information of the process that generated data be modeled by a mathematical representation. The uncertainty present in data measurements makes it harder to discriminate among similar model structures [Barbosa et al., 2015]. The randomized approach to model structure selection, such as the one performed by RaMSS-m, provides a convenient framework to deal with a set of possibly equivalent models with rather than a single, "true" model.

5.2 Future Work

The development of a randomized approach for model selection such as the RaMSS-m opens a wide range of possibilities in model analysis and design. Future work that has not been treated in this research phase either due to being out of scope of an initial method implementation or by requiring additional theoretical background reformulation may be listed:

- Non-polynomial NARMAX models evaluation. The RaMSS-m method is a quite general method as it does not impose any restrictions about the model structure. The author finds the proposed method may be a viable approach do structure selection of models that exhibit some degree of correlation among variables. Those models include Vector autoregression (VAR) and its variants - Bayesian vector autoregression and VARMAX.
- Variable selection. Variable selection, although briefly explored in Example 3.3.2, was not the main focus of this document. RaMSS-m may be a proper method to variable selection, specially when dealing with a large set of candidate explanatory variables where a randomized approach may be more appropriate.
- Inclusion of correlation in RIP updates. The regressor inclusion probability model does not take into account the correlation between regressor inclusion probabilities. An approach using a multivariate Bernoulli distribution [Bianchi et al., 2016] may be included in the RaMSS-m method and may provide a more flexible framework.

Bibliography

Aguirre, L. A. (2007). *Introdução à identificação de sistemas: técnicas lineares e não lineares aplicadas a sistemas reais - Terceira Edição*. Editora UFMG, Belo Horizonte.

Aguirre, L. A. (2015). *Introdução à Identificação de Sistemas – Técnicas Lineares e Não Lineares: Teoria e Aplicação, 4a Edição Revista*. Editora UFMG, Belo Horizonte.

Aguirre, L. A., Barbosa, B. H., and Braga, A. P. (2010). Prediction and simulation errors in parameter estimation for nonlinear systems. *Mechanical Systems and Signal Processing*, 24(8):2855–2867.

Aguirre, L. A. and Billings, S. A. (1995a). Dynamical effects of overparametrization in nonlinear models. *Physica* D, 80(1,2):26–40.

Aguirre, L. A. and Billings, S. A. (1995b). Improved structure selection for nonlinear models based on term clustering. *Int. J. Control*, 62(3):569–587.

Aguirre, L. A., Corrêa, M., and Cassini, C. C. S. (2002). Nonlinearities in NARX polynomial models: representation and estimation. *Proc. IEE Part D: Control Theory and Applications*, 149(4):343–348.

Aguirre, L. A., Donoso-Garcia, P. F., and Santos-Filho, R. (2000). Use of *a priori* information in the identification of global nonlinear models — A case study using a Buck converter. *IEEE Trans. Circuits Syst. I*, 47(7):1081–1085.

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Trans. Automat. Contr.*, 19(6):716–723.

Allen, D. M. (1974). The relationship between variable selection and data agumentation and a method for prediction. *Technometrics*, 16(1):125–127.

Alves, M. A., Corrêa, M. V., and Aguirre, L. A. (2012). Use of self-consistency in the structure selection of NARX polynomial models. *Int. J. Modelling, Identification and Control*, 15(1):1–12.

Avellina, M., Brankovic, A., and Piroddi, L. (2017). Distributed randomized model structure selection for narx models. *International Journal of Adaptive Control and Signal Processing*, 31(12):1853–1870.

Barbosa, A. M., Takahashi, R. H. C., and Aguirre, L. A. (2015). Equivalence of non-linear model structures based on pareto uncertainty. *IET Control Theory & Applications*, 9(16):2423–2429.

Bhat, H. S. and Kumar, N. (2010). On the derivation of the bayesian information criterion. *School of Natural Sciences, University of California*.

Bianchi, F., Falsone, A., Prandini, M., and Piroddi, L. (2016). A randomised approach for NARX model identification based on a multivariate bernoulli distribution. *International Journal of Systems Science*, 48(6):1203–1216.

Billings, S. A. (1980). Identification of nonlinear systems — a survey. *IEE Proceedigs Pt. D*, 127(6):272–285.

Billings, S. A. (2013). *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains*. Wiley.

Billings, S. A., Chen, S., and Korenberg, M. J. (1989). Identification of MIMO nonlinear systems using a forward-regression orthogonal estimator. *Int. J. Control*, 49(6):2157–2189.

Bonin, M., Seghezza, V., and Piroddi, L. (2010). NARX model selection based on simulation error minimisation and LASSO. *IET Control Theory and Applications*, 4(7):1157–1168.

Braga, A. R. (1994). Implementation of multiloop and multivariable control strategies (in portuguese). Master's thesis, Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais. Brazil.

Broomhead, D. S. and Lowe, D. (1988). Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom).

Cavanaugh, J. E. (1997). Unifying the derivations for the akaike and corrected akaike information criteria. *Statistics & Probability Letters*, 33(2):201–208.

Chen, S. and Billings, S. A. (1992). Neural networks for nonlinear dynamic system modeling and identification. *Int. Journal of Control*, 56(2):319–346.

Chen, S., Billings, S. A., Cowan, C. F. N., and Grant, P. M. (1990). Practical identification of NARMAX models using radial basis functions. *Int. J. Control*, 52(6):1327–1350.

Chen, S., Billings, S. A., and Luo, W. (1989). Orthogonal least squares methods and their application to nonlinear system identification. *Int. J. Control*, 50(5):1873–1896.

Corrêa, M. V. (2001). Nonlinear dynamic systems identification using narmax rational models: real systems application. Master's thesis, Graduate Program in Electrical Engineering, UFMG.

Falsone, A., Piroddi, L., and Prandini, M. (2014). A novel randomized approach to nonlinear system identification. In *53rd IEEE Conference on Decision and Control*. IEEE.

Falsone, A., Piroddi, L., and Prandini, M. (2015). A randomized algorithm for nonlinear model structure selection. *Automatica*, 60:227–238.

Farina, M. and Piroddi, L. (2010). An iterative algorithm for simulation error based identification of polynomial input-output models using multi-step prediction. *Int. Journal of Control*, 83(7):1442–1456.

Gauss, C. F. and Davis, C. H. (2004). *Theory of the motion of the heavenly bodies moving about the sun in conic sections*. Courier Corporation.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58.

Geyer, C. (2011). Introduction to markov chain monte carlo. *Handbook of Markov Chain Monte Carlo*, 20116022:45.

Green, P. J. (1995). Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732.

Hastie, D. I. and Green, P. J. (2012). Model choice using reversible jump markov chain monte carlo. *Statistica Neerlandica*, 66(3):309–338.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

Hong, X., Mitchell, R. J., Chen, S., Harris, C. J., Li, K., and Irwin, G. W. (2008). Model selection approaches for non-linear system identification: a review. *Int. J. Systems Sci.*, 39(10):925–946.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer New York.

Korenberg, M. J., Billings, S. A., Liu, Y. P., and McIlroy, P. J. (1988). Orthogonal parameter estimation algorithm for nonlinear stochastic systems. *Int. J. Control*, 48(1):193–210.

Leontaritis, I. J. and Billings, S. A. (1985a). Input-output parametric models for nonlinear systems part I: Deterministic nonlinear systems. *Int. J. Control*, 41(2):303–328.

Leontaritis, I. J. and Billings, S. A. (1985b). Input-output parametric models for nonlinear systems part II: Stochastic nonlinear systems. *Int. J. Control*, 41(2):329–344.

Ljung, L. (1987). *System Identification, Theory for the User*. Prentice Hall, New Jersey.

Mallows, C. L. (2000). Some comments on Cp. *Technometrics*, 42(1):87–94.

Mao, K. Z. and Billings, S. A. (1997). Algorithms for minimal model structure detection in nonlinear dynamic system identification. *Int. J. Control*, 68(2):311–330.

Mendes, E. M. A. M. and Billings, S. A. (2001). An alternative solution to the model structure selection problem. *IEEE Trans. on Man and Cybernetics - Part A*, 36(21):597–608.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. Technical report.

Piroddi, L. (2008). Simulation error minimization methods for NARX model identification. *Int. J. Modelling, Identification and Control*, 3(4):392–403.

Piroddi, L. and Spinelli, W. (2003). An identification algorithm for polynomial NARX models based on simulation error minimization. *International Journal of Control*, 76(17):1767–1781.

Polasek, W. (2012). Handbook of markov chain monte carlo edited by Steve Brooks, Andrew Gelman, Galin Jones, Xiao-Li Meng. *International Statistical Review*, 80(1):184–185.

Retes, P. and Aguirre, L. A. (2018). NARMAX model identification using a randomized approach. *International Journal of Modelling, Identification and Control*, (In press).

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

Shao, J. (1993). Linear model selection by cross-validation. *Journal of the American Statistical Association*, 88(422):486–494.

Söderström, T. and Stoica, P. (1989). *System Identification*. Prentice Hall International, London.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society. Series B (Methodological)*, pages 111–147.

Strang, G. (1989). Wavelets and dilation equations: a brief introduction. *SIAM Review*, 31(4):614–627.

Tibshirani, R. (2011). Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(3):273–282.

Wei, H. L. and Billings, S. A. (2008). Model structure selection using an integrated forward orthogonal search algorithm interfered with squared correlation and mutual information. *Int. J. Modelling, Identification and Control*, 3(4):341–356.

Zhu, Q. M. (2005). An implicit least squares algorithm for nonlinear rational model parameter estimation. *Applied Mathematical Modelling*, 29:673–689.