POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell' Informazione
Corso di Laurea Magistrale in Ingegneria Informatica

A randomized approach for NARX model identification based on a multivariate Bernoulli distribution

Relatore: Prof. Luigi Piroddi
Correlatore: Dott. Alessandro Falsone

Tesi di Laurea di:
Federico Bianchi, matricola 819133

Anno Accademico 2014-2015

POLITECNICO DI MILANO

Sommario

Scuola di Ingegneria Industriale e dell' Informazione
Corso di Laurea Magistrale in Ingegneria Informatica

A randomized approach for NARX model identification based on a multivariate Bernoulli distribution

by Federico BIANCHI

Nell'ambito dell'identificazione di sistemi, uno degli obiettivi primari è quello di ottenere un modello matematico del sistema atto a sviluppare strumenti di analisi e sintesi.
La famiglia dei modelli non lineari NARX (Nonlinear Autoregressive models with eXogenous variables) basati su espansione polinomiale è stata ampiamente studiata nella letteratura di settore e molti metodi di identificazione sono stati proposti. Tali metodi sono tipicamente basati su una costruzione incrementale del modello per cui i termini sono progressivamente selezionati come candidati all'inclusione nel modello. Il limite principale di questi metodi è la difficoltà nel calcolare per ogni termine la giusta significatività, cosa che può pregiudicare la corretta selezione del modello.
Per superare questo limite, negli ultimi anni il problema dell'identificazione è stato riformulato in termini probabilistici. Sono stati quindi definiti metodi randomizzati basati sul campionamento di distribuzioni di probabilità definite sullo spazio dei possibili modelli. Tra questi, di particolare interesse è il metodo RaMSS che è basato sul campionamento progressivo di distribuzioni Bernoulliane indipendenti, ognuna associata ad un termine. Tali distribuzioni sono poi aggiornate sulla base delle performance dell'intera popolazione di modelli estratti.
In questa tesi è formulata una variante del metodo RaMSS, che tiene conto di eventuali dipendenze tra i termini del modello. I termini sono descritti da una distribuzione Bernoulliana multivariata e il campionamento di un termine è quindi condizionato all'esito dell'estrazione degli altri.
I risultati ottenuti su alcuni sistemi analitici di diversa natura confermano le aspettative: la nuova variante del RaMSS riesce maggiormente nell'identificazione del modello corretto anche per quei sistemi per i quali il miglior modello selezionato dall'algoritmo base non corrisponde a quello corretto. Inoltre, rispetto all'algoritmo base, la variante proposta identifica modelli più compatti nel caso di sistemi con ingressi lenti. Il nuovo algoritmo è stato validato anche rispetto a metodi classici, quali il FROE.

POLITECNICO DI MILANO

Abstract

Scuola di Ingegneria Industriale e dell' Informazione
Corso di Laurea Magistrale in Ingegneria Informatica

A randomized approach for NARX model identification based on a multivariate Bernoulli distribution

by Federico BIANCHI

In the field of system identification, one of the primary goal is to obtain a mathematical model of a system useful to develop formal tools of analysis and synthesis.
The polynomial Nonlinear Autoregressive with eXogenous inputs (NARX) model family has been widely studied in the literature and several methods have been proposed. These methods are typically based on incremental model building techniques that progressively select from a candidate set the terms to include in the model. The main limitation of these methods concerns the difficulty to correctly assess the significance of each term which can lead to wrong model selections.
To overcome this limit, the identification task has been recently restated in a probabilistic fashion and new randomized methods have been proposed. These methods are based on the sampling from probability distributions defined over the space of possible model structures. Particularly interesting is the RaMSS method which progressively samples from independent Bernoulli distributions associated to the terms. Then, the method updates the distributions according to the performances of all the sampled model structures.
In this thesis a variant of the RaMSS is presented. It takes into account the dependences between model terms. The model terms are described by means of a multivariate Bernoulli distribution and the sampling of a term is thus conditioned by the sampling of the others.
The new algorithm is tested on several analytic systems and the obtained results confirm the expectations: the proposed variant is capable of consistently identifying the correct model even for those systems for which the RaMSS occasionally fails to select the correct model structure. Furthermore, the identified models in the case of system excited with slowly varying input signals are more compact than those selected by the RaMSS. The new algorithm is validated over classical methods such as the FROE as well.

Acknowledgements

Foremost, I would like to thank my advisor Prof. Luigi Piroddi for his willingness, the opportunity to work with him on a such interesting topic, and the continuous support during the preparation of this thesis.
I would like to thank also:

- my assistant advisor Dott. Alessandro Falsone, for his support offered during the preparation of my thesis,
- all the professors and colleagues that I met in these five years and with whom I had the pleasure of working,
- all my friend.

Lastly, but most importantly, I would like to thank my family who always supported me and believed in me, and Elena for constant support and motivation.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AIK	**A**kaike **I**nformation **C**riterion
AMS	**A**verage **M**odel **S**ize
ARX	**A**uto **R**egressive with e**X**ogenous input
CLF	**C**onditional **L**inear **F**amily
EA	**E**volutionary **A**lgorithm
ERR	**E**rror **R**eduction **R**atio
FPE	**F**inal **P**rediction **E**rror
FROE	**F**orward **R**egression **O**thogonal **E**stimator
FIR	**F**inite **I**mpulse **R**esponse
GA	**G**enetic **A**logorithm
LS	**L**east **S**quares
LTI	**L**inear **T**ime **I**nvariant
MDL	**M**inimum **D**escription **L**ength
MSE	**M**ean **S**quared **E**rror
MSPE	**M**ean **S**quared **P**rediction **E**rror
MSSE	**M**ean **S**quared **S**imulation **E**rror
MSS	**M**odel **S**tructure **S**election
NARX	**N**onlinear **A**uto **R**egressive with e**X**ogenous input
NN	**N**eural **N**etworks
OLS	**O**rthogonal **L**east **S**quares
PEM	**P**rediction **E**rror **M**inimization
RaMSS	**Ra**ndomized **M**odel **S**tructure **S**election
RBF	**R**adial **B**asis **F**unction
RIP	**R**egressor **I**nclusion **P**robability
RJMCMC	**R**eversible **J**ump **M**arkov **C**hain **M**onte **C**arlo
SRR	**S**imulation error **R**eduction **R**atio
SVD	**S**ingular **V**alue **D**ecomposition
VSM	**V**ector **S**pace **M**odel

To everyone who have allowed me to get here

Chapter 1

Introduction

The concepts of *system* and *model* are the basis of the engineering disciplines, where one is generally interested in a quantitative assessment of the behavior of a dynamical system. It is necessary, therefore, to obtain a mathematical description of it. Starting from the model it is possible to develop formal tools of analysis and synthesis. Basically, there are two ways of constructing mathematical models:

- *Mathematical modeling*. This is an analytic approach. Prior knowledge and physical insight about the system are used to describe the dynamic behavior of a system.
- *System identification*. This is an experimental approach. Some experiments are performed on the system in order to learn a model from the collected data by means of *parameter estimation* (Ljung, 1987; Söderström and Stoica, 1989).

The nonlinear auto-regressive (moving average) with exogenous inputs (NAR[MA]X) model (Leontaritis and Billings, 1985a; Leontaritis and Billings, 1985b) is an input-output description often used in nonlinear system identification, where the current output is obtained by means of a nonlinear functional expansion of lagged inputs, outputs (and possibly noise) elementary terms. The NAR[MA]X class potentially reduces the number of model terms - combinations of elementary terms - w.r.t. other parametric classes such as the Volterra series and Wiener's models. This peculiarity is very important when the model structure that represents the dynamical nonlinear system is unknown since it allows to reduce the number of possible candidate structures amongst which look for the best one (*model structure selection*). The simplest way to select the best model structure is to generate all the possible structures and choose between them by comparing their performance. Although this could be in principle done in the linear case (AR[MA]X models), it is not feasible in the nonlinear one since the number of possible model terms, and therefore the number of possible models, increases rapidly with the number, maximum lag and degree of elementary terms (*curse of dimensionality*). Therefore, we need to develop alternative search methods for the nonlinear case (Sjöberg et al., 1995), keeping in mind that over-parametrization leads to very accurate models at the expense of a lower capability to provide approximation to the true system output for unseen input data (loss of the *generalization* property) (Hong et al., 2008; German, Bienenstock, and Doursat, 1992; Aguirre and Billings, 1995).

Polynomial NAR[MA]X models have been widely used since they are *linear- in-the- parameters* and can thus be identified with least squares (LS) algorithms in a prediction error minimization (PEM) framework. However, PEM methods require a precise matching of the model structure with that of the underlying system in order to guarantee unbiased parameter estimates.
In the linear case, information criteria such as the final prediction error (FPE), the Akaike information criterion (AIK) and the minimum description length (MDL) are used to prevent over-parametrization since they allow to estimate the correct model size taking into account the model accuracy, which decreases as the number of parameters increases, and the number of parameters itself. In (Palumbo and Piroddi, 2000) it is shown that these indices are not easily applicable to the nonlinear case since in this context there is no a simple relation between model accuracy and model size.
In (Billings, Chen, and Korenberg, 1989) a forward-regression orthogonal estimator (FROE) based on the error reduction ratio (ERR) criterion has been presented. This identification algorithm represents a milestone and several variants of this method have been proposed in the literature (Mao and Billings, 1999; Bonin, Seghezza, and Piroddi, 2010; Piroddi and Spinelli, 2003). The ERR is used to express the capability of a term to improve the model accuracy if it were added to the model. The idea behind the FROE method is to decouple the estimation of additional parameters from that of the parameters already included in the model by means of an orthogonalization technique, the *orthogonal least square* (OLS). In this way, at each step the significance of each candidate regressor can be separately evaluated by computing the ERR criterion. The main drawbacks are the incremental nature of the model building procedure and the fact that this method is based on the PEM approach, that leads to a procedure that is initialization dependent. Therefore, the overall selection procedure is not guaranteed to converge to the correct structure. Additionally, methods based on forward/forward-backward regression do not provide an indication of the uncertainty in the model structure.
In (Baldacchino, Anderson, and Kadirkamanathan, 2013) the problems of model structure selection and parameter estimation have been dealt with in a unified Bayesian framework, that is suitable for describing uncertainty since this approach does not obtain a single description, but rather, a distribution over models. These posterior distributions are obtained using the Reversible Jump Markov Chain Monte Carlo (RJMCMC) procedure which is based on three operations: sampling of unselected terms to include in the model, sampling of previously selected terms to remove from the model and updating of the existing parameters. At each step only one of these operations is randomly attempted and its result is accepted or not according to an acceptance ratio. The Markov Chain requires a burn-in period (many iterations) to converge to the desired stationary distribution (it is initialization independent) and this is the main drawback of RJMCMC. Nonetheless, the stochastic nature of the RJMCMC algorithm ensures to a global search of term space.
The RJMCMC algorithm is analogous in some regards to the genetic algorithm (GA) procedure (Rodriguez-Vazquez, Fonseca, and Fleming, 2004; Mao and

Billings, 1999): both approaches use random sampling methods. In detail, a GA is a search heuristic that mimics the process of natural selection. GAs belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.
In (Falsone, Piroddi, and Prandini, 2014) a novel iterative randomized algorithm for model structure selection (RaMSS) has been introduced focusing on NARX models only. This method has some features in common with both RJMCMC and GA: as RJMCMC it defines distributions (Bernoulli distributions) over model terms, describing the probability that the term is present in the "true" model. As GA it is based on the concept of population. In fact, at each iteration a new population of models is generated according to the Bernoulli distributions. The performances of the whole population of extracted models are then used to update the mean of each individual Bernoulli distribution. More precisely, the mean performance of the models that include a specific term is compared with those that do not. The result of this comparison is then used to increase or decrease the term's probability taking into account also the dispersion of the values of the model's evaluation criterion. As the number of iterations increases, the number of different explored models decreases, converging to a specific model structure. A deeper analysis of the inclusion mechanism of regressors into the model has emphasized that often regressors tend to enter/disappear jointly in a model. So, although the regressors are extracted independently, often they are somehow related. This fact has motivated the work that will be presented in this thesis.
The idea is to introduce second order information about relations among regressors in the updating of the Bernoulli distributions, in order to improve the performance of RaMSS algorithm, in terms of correctness, number of explored models and number of iterations required to achieve convergence. Unlike the classic RaMSS, this time the regressors will be not independent but they will be correlated with a given marginal mean vector and a correlation matrix. To simulate these correlated binary variables, a family of multivariate binary distributions, the Conditional Linear Family (CLF), will be used. This family was introduced in (Qaqish, 2003). The basic idea behind CLF is to simulate correlated binary variables with specified mean vector and correlation matrix without fully specifying the joint distribution since this becomes impractical as the number of variables grows. Since in the RaMSS each regressor is independent from the others, the algorithm, as is, lacks the correlation matrix needed by CLF. To fill this gap, an approximation of the correlation matrix will be obtained from the *cosine similarity matrix*, computed considering each regressor as a vector in the Euclidean space of the models. This kind of similarity measure is widely use in the Information Retrieval field to assess the relevance of a document to a given query (Berry, Drmac, and Jessup, 2006).

The work is organized as follows:

- *Chapter* 2 provides the basic framework and notation for nonlinear system identification of NARX models and briefly reviews the main approaches in the literature, with particular interest in the RaMSS method that has been used as starting point for this work.
- *Chapter* 3 presents the necessary tools to introduce dependence information among the regressors into the RaMSS. Finally, the new algorithm is presented.
- *Chapter* 4 reports the results obtained on several systems taken from the literature. The performed analysis aims to assess the effectiveness of the new algorithm w.r.t. the RaMSS and classical methods as the FROE.
- Finally, in *Chapter* 5 the main considerations are recapped and possible future works are proposed.

Chapter 2

Review of the State of the Art

In this section we review the basic framework and notation used in the nonlinear system identification problem, the NARX model class and finally we analyze the state-of-the-art methods for the identification of these models. Many algorithms have been proposed in the literature that combine model structure selection and parameter estimation in a single optimization procedure. It is usually required to specify only the set of candidate regressors for inclusion in the model. By combining the elements of this set the identification procedure selects the most significant terms to include in the model, based on a quantitative evaluation of the quality of the estimated model. The following methods define the path for the presented work: from the most popular (FROE), moving to randomized methods (RJMCMC and GA), to finally consider the starting point of the thesis (RaMSS).

2.1 Nonlinear system identification framework

System identification starts from observed inputs $u(t)$ and outputs $y(t)$ taken from some experiments performed on a dynamical system

$$u^t = [u(1), u(2), \cdots, u(t)] \tag{2.1}$$

$$y^t = [y(1), y(2), \cdots, y(t)] \tag{2.2}$$

and the objective it to look for a relationship between these past observations $[u^{t-1}, y^{t-1}]$ and future outputs $y(t)$:

$$y(t) = g(u^{t-1}, y^{t-1}) + e(t) \tag{2.3}$$

The additive term $e(t)$ accounts for the fact that there is an *unpredictable* part in the behavior of the system, i.e. the next output $y(t)$ will not be an exact function of past data. In other words, we will think of the available observations as being a finite length realization of a stochastic process. This stochastic nature is explicitly represented by $e(t)$. Therefore, the goal of the identification task must be that $e(t)$ is small, so that $g(u^{t-1}, y^{t-1})$ is a good prediction/approximation of $y(t)$ given the observations. When the model structure is unknown the identification task can be decomposed in two subtasks:

- *model structure selection* (MSS);

- *parameter estimation*.

The aim of MSS is to find the form of nonlinear dependence between data, within a family of functions which is usually parametrized with a finite-dimensional *parameter vector* θ:

$$g(u^{t-1}, y^{t-1}; \theta). \tag{2.4}$$

Amongst all the possible parametrizations of the chosen family, we are looking for the one that provides the best approximation of $y(t)$. The parameter estimation task deals with this approximation problem, minimizing w.r.t. θ a cost function of the following type:

$$J(\theta) = \sum_{t=1}^{N} \|y(t) - g(u^{t-1}, y^{t-1}; \theta)\|^2 \tag{2.5}$$

Since the past observations are potentially infinite, we have to restrict the observability scope, obtaining a finite-dimensional vector

$$x(t) = [y(t-1), \cdots, y(t-n_y), u(t-1), \cdots, u(t-n_u)] \tag{2.6}$$

where n_y and n_u are suitable maximum lags. A mapping φ is applied on this vector that projects the observations into a finite-dimensional space called *basis functions space*:

$$\varphi(t) = \varphi(y(t-1), \cdots, y(t-n_y), u(t-1), \cdots, u(t-n_u)) \tag{2.7}$$

Finally, this finite-dimensional vector $\varphi(t)$ is used as input to the nonlinear mapping $g(\varphi(t))$ that returns the final output values relative to the provided basis functions set. So, the MSS can be seen in turn as the composition of two subproblems:

- how to combine past lagged observations in order to obtain the vector $\varphi(t)$;
- how to define the nonlinear mapping $g(\varphi(t))$ between basis functions and output values.

Often, the nonlinear mapping $g(\varphi(t))$ is a nonlinear functional expansion, provided that the chosen expansion is a *universal approximating function*, i.e. it should be capable of approximating any continuous function at whatever precision level, on a compact domain, provided it is endowed with sufficient degrees of freedom. The following are universal approximating functions: polynomials, splines, multi-layer perceptron neural networks (NN), radial basis function (RBF), wavelets.

2.2 The NARX model class

A popular choice is the *polynomial* functional expansion which is a linear combination of all monomials of $x(t)$ up to a given order.

Example 1. Consider $x(t) = [x_1, x_2, x_3]$. Then, a full cubic expansion contains the following basis functions:

order 0 1 (1 term)

1^{st} **order** x_1, x_2, x_3 (3 terms)

2^{nd} **order** $x_1^2, x_1 \cdot x_2, x_1 \cdot x_3, x_2^2, x_2 \cdot x_3, x_3^2$ (6 terms)

3^{nd} **order** $x_1^3, x_1^2 \cdot x_2, x_1^2 \cdot x_2, x_1 \cdot x_2 \cdot x_3, x_1 \cdot x_2^2, x_1 \cdot x_3^2, x_2^3, x_2^2 \cdot x_3, x_3^3, x_3^2 \cdot x_2$ (10 terms)

This functional expansion allows us to represent the chosen parametrized family 2.3 as a linear regression:

$$y(t) = \varphi^T(t)\theta + e(t) \tag{2.8}$$

In this case the basis functions are called *regressors* and the vector $\varphi(t)$ is referred as *regression vector*. Linear-in-the-parameters models are convenient from the computational point of view since their parameters can be estimated by simple LS algorithms which have well known variants suitable also to the on-line identification problem - problems in which the data are not given as a batch but they are collected during the execution of the algorithm - and adaptive learning. Also, linearity provides an easier model interpretation (the capability to interpret the properties of the process that the model represents and to extract the knowledge of the underlying system). In fact, the terms in a linear model are often associated to physical aspects of the system and so the parameters act as importance factors for the relative physical phenomenon. This peculiarity does not hold for all the functional expansions: for example Artificial NNs - which are inspired by biological neural networks, composed by networks of simple computing units that perform very simple operations according to an "activation" function - are very powerful estimators which, however, do not provide transparent models. Neither the single unit nor an entire layer has a direct interpretation in terms of the system properties. Finally, in a polynomial model all the nonlinearity is confined in the regressors.

2.3 The over-parametrization problem

Consider Example 1. Notice that the total number of monomials up to order k for a set of n variables equals $\frac{(n+k)!}{n!k!}$, and grows rapidly with n and k (this problem is often referred as *curse of dimensionality*). Therefore, the MSS task assumes a key role in the identification procedure provided that it is not too time-consuming. In fact, the use of over-parametrized models arises from the inability to find selection criteria that do not heavily impact on the computational performance of the overall procedure and, moreover, from the fact that the prediction error usually decreases as the model accuracy increases. However, this accuracy improvement does not necessarily lead to models that are more effective in capturing the dynamics of the underlying system than simpler ones, due to loss in generalization. The mean squared error (MSE) can be used as a measure of model generalization. Denoting $\hat{y} = g(u^{t-1}, y^{t-1}; \theta)$ - *one-step-ahead prediction of* $y(k)$, the MSE is defined as

$$E[(y - \hat{y})^2] \tag{2.9}$$

According to the *bias/variance dilemma* (German, Bienenstock, and Doursat, 1992), the MSE can be decomposed into two components, the *bias* and the *variance*, as

$$E[(y - \hat{y})^2] = Bias[\hat{y}]^2 + Var[\hat{y}] + \sigma^2 \tag{2.10}$$

where

$$Bias[\hat{y}] = E[\hat{y}] - y \tag{2.11}$$

$$Var[\hat{y}] = E[(\hat{y} - E[\hat{y}])^2] \tag{2.12}$$

and σ^2 is the variance of the white noise $e(t)$, which is an irreducible error. Suppose we repeat the whole model building process more than once: each time we gather new data and run a new analysis creating a new model. Due to randomness in the underlying data sets, the resulting models will have a range of prediction performance. The bias measures how far off in general the model predictions are from the correct value, while the variance describes how much the predictions vary between different realizations of the model (see Figure 2.1 where each hit represents an individual realization of our model and the center of the target is a model that perfectly predicts the correct values).

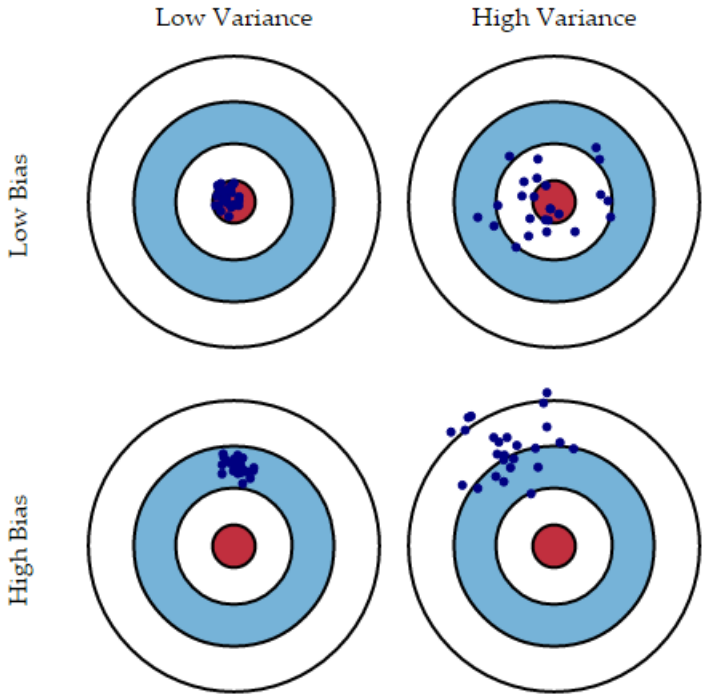


FIGURE 2.1: Graphical illustration of bias and variance.

At its root, dealing with bias and variance is really about dealing with *over- and under-fitting*. High bias can cause a model to miss the relevant relations between inputs and target outputs (underfitting). High variance can cause overfitting: modeling the random noise in the training data, rather than the intended outputs. Hence, there is a tradeoff between bias and variance, i.e. the minimization w.r.t. model size, since the bias is reduced and variance is increased in relation to model complexity, see Figure 2.2.
MSS is therefore particularly challenging since there is not an analytical way to find the optimal model complexity. Instead we must use an accurate measure of prediction error, as MSE, and explore differing levels of model complexity and then choose the complexity level that minimizes the overall error on the validation dataset (*Cross Validation*). Let's see what this looks like in practice with a simple example.

Example 2. Suppose that one wants to implement a wealth and happiness model as a linear regression. We can start with the simplest regression possible:

$$Happiness = a + b \cdot Wealth + e \tag{2.13}$$

and then we can add polynomial terms to model nonlinear effects. Each polynomial term we add increases the model complexity. For example:

$$Happiness = a + b \cdot Wealth + c \cdot Wealth^2 + e \tag{2.14}$$

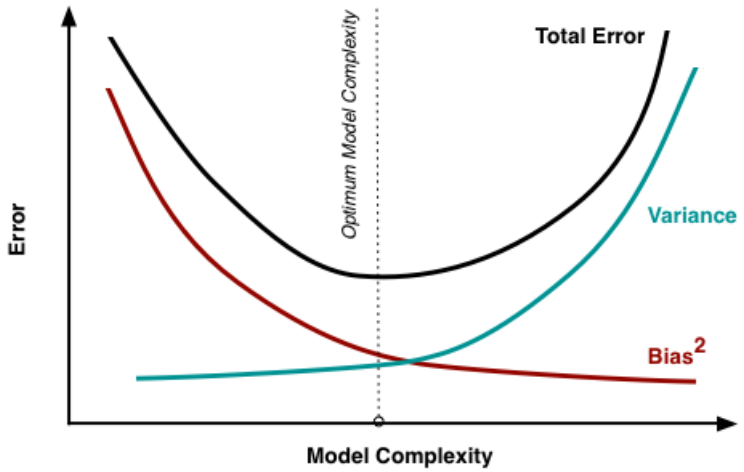


FIGURE 2.2: Bias-variance tradeoff.

The Figure 2.3 represents the training and prediction error curves of the model for different levels of model complexity.

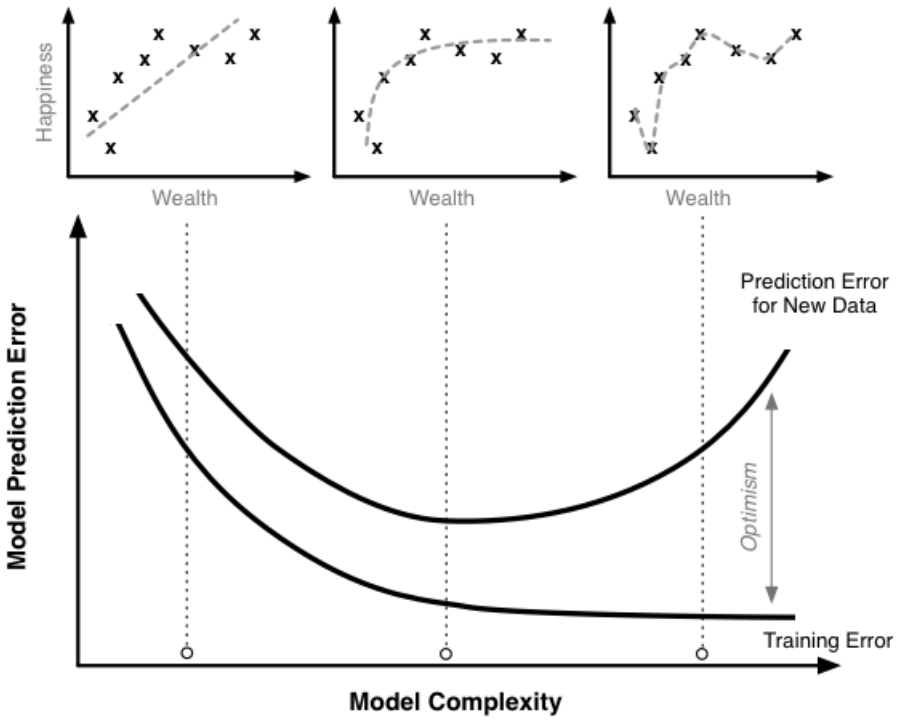


FIGURE 2.3: Training and prediction error.

Information criterion, such as FPE, AIK and MDL have been also applied to the general problem of model selection as model generalization metrics. All

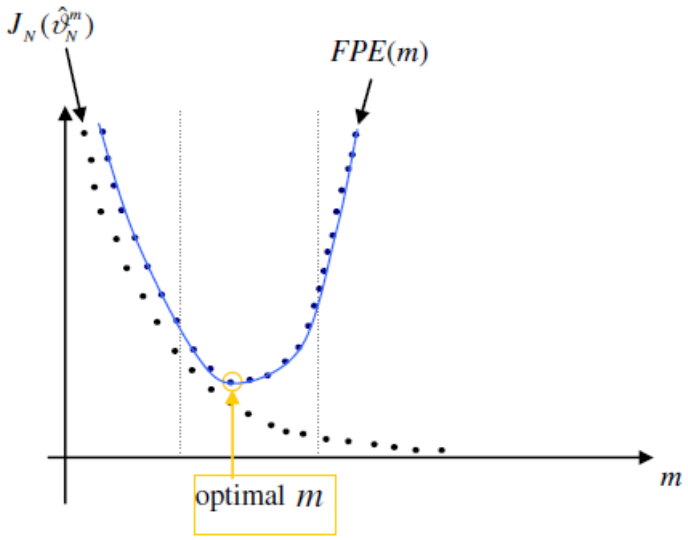


FIGURE 2.4: FPE error curve.

three indices are related to each others. Consider as reference the FPE

$$FPE(m) = \frac{N+m}{N-m} \cdot J_N(\hat{\theta}_N^m)) \tag{2.15}$$

where N is the number of data, m is the number of parameters.
$\frac{N+m}{N-m}$ is increasing with m, while $J_N(\hat{\theta}_N^m)$ is decreasing with m. Hence, there is a unique minimum, see Figure 2.4. The optimal m indicates the number of parameters for which the model starts to over-fit the data, losing in generalization. These indices are widely used in both the linear and nonlinear cases, but in the latter they rarely yield conclusive proofs in favour of a specific model structure as pointed out in (Piroddi and Spinelli, 2003).

2.4 The FROE for polynomial NARX identification

A common approach for solving the MSS problem is based on an iterative model construction - *forward regression* - where at each iteration a regressor is chosen from a candidate set, according to some criterion, and it is added to the current model. Sometimes, pruning methods are also applied. This procedure continues until a chosen level of accuracy is reached. Alternatively, some heuristics are applied, such as the *elbow method*: one should choose to stop the procedure, if adding another regressor does not improve much the model. The iterative model construction approach is thus based on a greedy selection policy, therefore it is sub-optimal, although it works fine in most cases. The usage of some criterion for selecting the regressor to add, is needed since a complete trial-and-evaluate approach is not feasible due to the numerous repetitions of parameter estimation and model performance evaluation. In FROE, parameters are estimated by means of *orthogonal least squares* (OLS), while the structure selection is based on the error reduction reduction ratio (ERR) criterion which measures the increment in the output explained variance. The idea behind the FROE method is to project the regressors (and parameters) in a new space in which they are mutually independent. In so doing, the relevance of each regressor can be independently evaluated from that of other regressors, by means of the ERR criterion:

$$[ERR]_i = \frac{\hat{g}_i^2 \sum_{t=1}^{N} w_i^2(t)}{\sum_{t=1}^{N} y^2(t)} \tag{2.16}$$

where w_i is the *i*-th *auxiliary* orthogonal regressor and $\hat{g}_i$ the corresponding estimated *auxiliary* parameter. For a better explanation of the OLS, see Appendix A.1. The regressor with the highest value of the ERR is added to the model. At the end of the procedure the regressors and parameters in the original space can be recovered from the auxiliary ones in a recursive way. The whole forward regression selection procedure based on OLS is:

1. Test all $\varphi_i(t)$, $i = 1, \ldots, n$, for inclusion in the model as $w_1(t)$. Precisely, set $w_1^{(i)}(t) = \varphi_i(t)$ and compute $[ERR]_1^{(i)}$ according to Equation 2.16.
   Let $j = arg\,max_i([ERR]_1^{(i)})$. Then, select $w_1(t) = w_1^{(j)}(t) = \varphi_j(t)$, $\hat{g}_1 = \hat{g}_1^{(j)}$ and $[ERR]_1 = [ERR]_1^{(j)}$.

2. Test all $\varphi_i(t)$, $i = 1, \ldots, n$, $i \neq j$ for inclusion in the model as $w_2(t)$. Precisely, set $w_2^{(i)}(t) = \varphi_i(t) - a_{12}^{(i)} w_1(t)$, where $a_{12}^{(i)} = \frac{\sum_{t=1}^{N} w_1(t)\varphi_i(t)}{\sum_{t=1}^{N} w_1(t)^2}$, and calculate $[ERR]_2^{(i)}$. Let $j = arg\,max_i([ERR]_2^{(i)})$. Then, select $w_2(t) = w_2^{(j)}(t) = \varphi_j(t)$, $\hat{g}_2 = \hat{g}_2^{(j)}$ and $[ERR]_2 = [ERR]_2^{(j)}$.

3. Iterate the procedure until a sufficient portion of the output variance has been explained, i.e. until $1 - \sum_{i=1}^{ns} [ERR]_i$, where ns is the number of selected terms, is less than a prescribed tolerance.

There are some known drawbacks related to the FROE procedure, due to the PEM based MSS task. Indeed, in the PEM framework the parameter estimates

are unbiased only if the model structure is exact. This condition does not hold in the early stages, when the model structure is at least incomplete due to the incremental nature of the selection procedure, leading to bad estimations. These wrong estimates make the order in which the regressors are added very important, especially because the relative importance of a regressor changes as the model building process goes on, as pointed out in (Piroddi and Spinelli, 2003). In addition, to have correct estimates, PEM requires the input signal to have sufficiently exciting properties. This condition becomes essential especially due to the usage of the ERR criterion which measures the increment in the output explained variance and not the output explained variance itself. In fact, if the input is a slowly varying process - the relative outputs are therefore similar, $y(t) \simeq y(t-1)$ - the ERR tends to encourage autoregressive terms, regardless of the actual underlying system dynamics. In conclusion, there is no guarantee that the algorithm will converge to the optimal solution, i.e. the true model. To deal with this issue, pruning techniques can be adopted to eliminate wrong terms at the expense of an increasing computational complexity. Such as an example, consider the pruning method proposed in (Piroddi and Spinelli, 2003). In the proposed approach, each iteration starts with the inclusion of a new regressors in the current model. Then, an iterative pruning procedure is applied:

1. Consider each sub-model obtained after the elimination of a previously selected regressors. Re-estimate its parameters and compute the corresponding performance index.

2. Amongst all the sub-models, consider the one associated to the best value of performance index. The candidate regressor to elimination is the one that has lead to the generation of the selected sub-model.

3. If the selected sub-models is better than the current model, the regressors is actually eliminated.

4. If an elimination has occurred, go to step 1, otherwise the iteration ends.

The idea behind this pruning mechanism is to encourage smaller models if these have the same performance of bigger ones. It is a greedy policy and it may occasionally eliminate a correct regressor, but this will be typically reintroduced in later stage.
In (Piroddi and Spinelli, 2003) is also introduced a variant of the FROE based on the simulation error reduction ratio (SRR) criterion which is defined as:

$$[SRR]_j = \frac{MSSE(M_i) - MSSE(M_{i+1})}{\frac{1}{N}\sum_{t=1}^{N} y^2(t)} \tag{2.17}$$

where:

- $MSSE$ is the Mean Squared Simulation Error defined as:

$$MSSE = \frac{1}{N}\sum_{t=1}^{N}(y(t) - \hat{y}_s(t))^2 \tag{2.18}$$

where $\hat{y}_s(t)$ denotes the simulated output of the models,

- M_i is the model obtained at the i-th iteration,
- M_{i+1} is the candidate model at the subsequent iteration, with the inclusion of the j-th regressor.

The ERR (2.16) can be rewritten similarly to SRR as:

$$[ERR]_j = \frac{MSPE(M_i) - MSPE(M_{i+1})}{\frac{1}{N}\sum_{t=1}^{N} y^2(t)}. \tag{2.19}$$

The Mean Squared Prediction Error(MSPE) is defined as:

$$MSPE = \frac{1}{N}\sum_{t=1}^{N}(y(t) - \hat{y}(t))^2 \tag{2.20}$$

where $\hat{y}(t)$ denotes the one-step-ahead prediction of $y(k)$.
A peculiarity of the SRR criterion is its capability to generally assign comparable values to regressors belonging to the same cluster. As argued in (Piroddi and Spinelli, 2003), due to this feature, an error in the MSS is less critical since at least the cluster is the right one. Moreover, if the input signal is a slowly varying process the one-step-ahead predictor tends to be short-sighted, and the simulation error can provide more accurate results.

2.5 The RJMCMC for polynomial NARX identification

Methods based on forward/backward regression do not provide an indication of the uncertainty in the model structure which, however, could be useful to better determine the relevance of an identified model. In (Baldacchino, Anderson, and Kadirkamanathan, 2013) the problems of model structure selection and parameter estimation are treated with a Bayesian inference approach which, by definition, derives the posterior probability over a population, in our case the set of all possible models, from observed data drawn from the population itself. Bayesian inference computes the posterior probability according to Bayes' theorem:

$$P(\theta|X) = \frac{P(X|\theta) \cdot P(\theta)}{P(X)} \tag{2.21}$$

where:

- X is a set of n observed data points.
- θ is the parameter vector.
- The prior distribution is the distribution of the parameter before any data is observed, i.e. $P(\theta)$.
- The sampling distribution is the distribution of the observed data conditional on its parameters, i.e. $P(X|\theta)$. This is also termed the likelihood.
- The posterior distribution is the distribution of the parameters after taking into account the observed data, i.e. $P(\theta|X)$.

The main difficulty in using Bayesian inference in system identification is the computation of the sampling distribution which requires to analytically solve an integral. To fill this gap, several sampling methods have been proposed to numerically compute posterior distributions, such as *inverse sampling*, *accept/reject sampling*, *importance sampling* and *Markov chain Monte Carlo*. One method which has been derived from the Markov Chain Monte Carlo method is the *Metropolis-Hastings* (MH) algorithm which defines the base for the RJMCMC. For a better explanation of the MH algorithm and others sampling methods see Appendix A.2 and A.3. The peculiarity of the RJMCMC based identification method is that it does not provide a single description, but rather, a distribution over models. The basic idea of this method is to move between the states of a Markov chain, which represent models of k terms. Three base operations have been defined to move between states:

- Birth move: sampling of unselected terms to include in the current model, i.e. $k' = k + 1$.
- Death move: sampling of previously selected terms to remove from the current model, i.e. $k' = k - 1$.

- Update: updating the variance of the parameters relative to existing terms, i.e. $k' = k$.

At each iteration one of this operations is randomly attempted according to some probabilities and its result is accepted or rejected according to an acceptance ratio. The probabilities of performing such operations have been updated according to the likelihood that the size of the real model is larger or smaller than the current model.
This method has some well known features:

- it solves jointly both the model structure selection and the parameter estimation problems;
- thanks to its markovian nature it is initialization independent. In fact the Markov chain will converge to the same steady-state from any initial conditions after some transient behavior;
- it naturally includes a pruning method to remove incorrect terms;
- finally, its stochastic nature lends itself to a global search of the term space.

The main drawbacks are:

- the Markov chain requires a burn-in period (many iterations) to converge to the desired stationary distribution, so it is not competitive with FROE in terms of speed
- the joint identification of both structure and parameters could be very difficult since the importance of a regressor, and therefore its parameter value, can greatly vary according to the structure of the models in which it appears, so the parameters distribution over models could be very complex.

2.6 The GA for polynomial NARX identification

The RJMCMC algorithm is analogous in some regards to the genetic algorithm procedure: both approaches use random sampling to ensure global searching. A GA is a search heuristic that mimics the process of natural selection. GAs belong to the larger class of evolutionary algorithms (EAs), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. These techniques are used to evolve an initial random population of candidate solutions toward a better one (Figure 2.5). A typical GA requires a genetic representation of the solution domain and a fitness function to evaluate the solution domain. A standard representation of each candidate solution is a binary string. In the system identification context we can think of a population of candidate solutions as a set of random models represented as binary strings.

Example 3. Assuming for example to have a candidate regressors set composed by 10 terms, i.e. $\varphi = \{\varphi_1, \varphi_2, \ldots, \varphi_{10}\}$.
The model $y(t) = a_1 \cdot \varphi_1 + a_3 \cdot \varphi_3 + a_7 \cdot \varphi_7 + a_{10} \cdot \varphi_{10}$ can be represented by the following binary string

$$1010001001 \tag{2.22}$$

Therefore, GAs address the MSS problem, working with a coding of the parameters rather then with the parameters themselves. The binary string can be modified by:

- *Crossover*: two strings are randomly selected from a pool previously defined by means of a selection (*tournament selection*, *roulette wheel selection* and others) on the candidate population. The bit position from which the two string will be splitted is randomly selected and the obtained substrings finally combined. For example, consider the following strings: i)1010001001 and ii) 1110101011. Assuming to split from the fifth bit, the two resulting strings will be: i)1010001001 and ii)1110101001.

- *Mutation*: randomly flip a bit.

The fitness function assumes a key role in the selection phase: individual solutions are selected through a fitness-based process, where fitter solutions are typically more likely to be selected. In other words, the idea is to combine good solutions hoping to find a better one. In this way, the population will be move toward better solutions. A reasonable choice for the fitness function could be the MSE.
GAs are simple to implement, but there are some limitations:

- fitness function evaluations, and so the parameters estimation, are performed many times;

- they do not scale well with complexity. As the number of regressors increases, the algorithm has to handle binary strings that become longer and longer;

- the second problem of complexity is the issue of how to protect parts that have evolved to represent good solutions from further destructive mutation or crossover operations. So, there is a tradeoff between exploration of the search space and exploitation of the current good results.

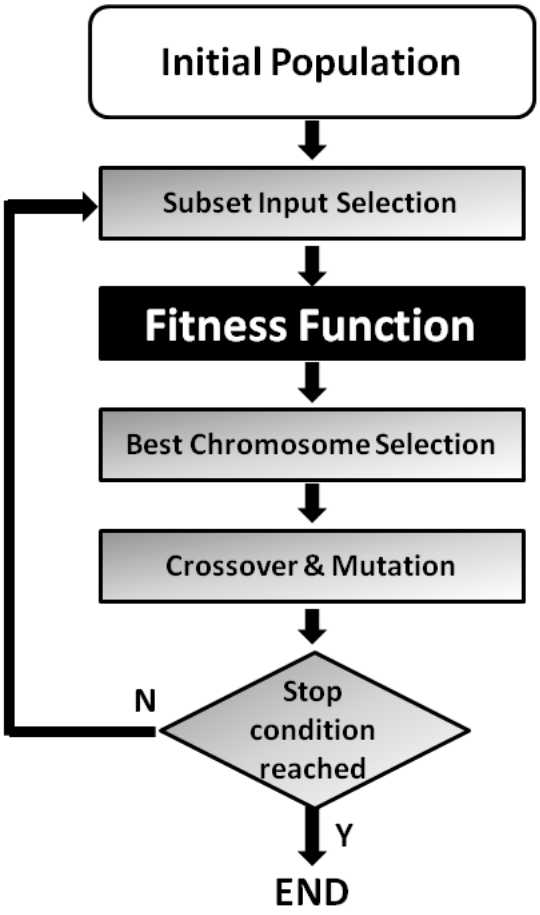


FIGURE 2.5: Pipeline of a Genetic Algorithm.

2.7 RaMSS

An iterative randomized algorithm for MSS has been introduced in (Falsone, Piroddi, and Prandini, 2014) . This method has some features in common with GAs and the RJMCMC based method:

- As the RJMCMC based method, the RaMSS introduces a probability distribution over models and iteratively updates it. Indeed, each regressor has its own Bernoulli distribution that describes the probability that the term is present in the "true" model.
- As GAs, RaMSS is based on the concept of population. In fact, at each iteration a new population of models is generated according to the Bernoulli distributions and, at the end, the entire population is used to decide if to encourage or not the extraction of a term in the next iteration.

Obviously, there are also some important differences w.r.t. the others two methods:

- The RaMSS does not perform a joint identification of both structure and parameters (as done by the RJMCMC based method), but solves instead the MSS problem in a probabilistic framework while the parameter estimation task is performed along the PEM framework.
- GAs evolve an initial random population of candidate solutions toward a better one, while in RaMSS at each iteration a new population is generated from scratch. All the information about previous populations is implicitly present in the Bernoulli distributions whose parameters are updated according to an aggregate analysis on the entire population.

In detail, at each iteration NP models are generated according to the regressors' probabilities $\rho_j \sim Be(\mu_j)$, for $j = 1 \dots m$, where the μ_j are referred to as Regressor Inclusion Probabilities (RIPs). A simple check is done to make sure that there are no empty models. Each extracted model undergoes a *t-student* statistical test to remove redundant terms. Then, the model parameters are estimated by means of a LS procedure and the performance of each model is evaluated according to:

$$J = e^{-K \cdot MSPE} \tag{2.23}$$

K being a tuning parameter that allows to further enlarge or restrict the range of J values, while MSPE is the mean square prediction error, Equation 2.20. The exponential mapping on the MSPE emphasizes the difference between model with high probabilities, in order to improve the selection capabilities of the algorithm when it goes toward its convergence point. Once the performances have been evaluated, they are used to update the RIPs parameters. Considering the entire population, for each regressor the aggregate mean performance of models that include that term has been computed. The same is done also for those models that do not include that term. Finally, a comparison between

these two quantities is done to decide if encourage or discourage the extraction of that regressor. More in details, the j-th RIP is updated according to:

$$\mu_j(i+1) = \mu_j(i) + \gamma \cdot \frac{\partial E_P[J]}{\partial \tilde{\mu}_j} \tag{2.24}$$

where:

- the adaptive step size γ is a design parameter defined as:

$$\gamma = \frac{1}{10 \cdot (J_{max} - J_{mean}) + 0.1} \tag{2.25}$$

  where J_{max} is the highest performance amongst all the others in the current population and J_{mean} is the average value of the model performances. It is a relevance measure for the update factor, in relation to the dispersion of the J values.

- $\frac{\partial E_P[J]}{\partial \tilde{\mu}_j}$ is the update factor defined as:

$$\frac{\partial E_P[J]}{\partial \tilde{\mu}_j} = E_P[J|\tilde{\rho}_j = 1] - E_P[J|\tilde{\rho}_j = 0] \tag{2.26}$$

  For a better explanation see Appendix A.4.

Finally a thresholding is applied to the obtained RIP values to make sure that they are still probabilities, i.e. $\mu_j \in [0, 1]$. The algorithm goes on as long as a chosen stopping criterion is met.
Figures 2.6 and 2.7 show a typical run of the RaMSS on the following analytic system:

$$y(k) = 0.7y(k-1)u(k-1) - 0.5y(k-2) - 0.7 * y(k-2)u(k-2)^2 + 0.6u(k-2)^2 + e(k)$$

with $u(k) \sim WUN(-1, 1)$, $e(k) \sim WGN(0, 0.04)$.
The RIPs associated to correct regressors are consistently increasing until they reach 1, while all other RIPs tend eventually to 0. One can note that a spurious regressor initially has a larger RIP than those associated to correct terms, but the algorithm is able to discard this term as it proceeds collecting more information on the true regressors. The average model size converges to the actual value, implying that very small models are generated and tested.

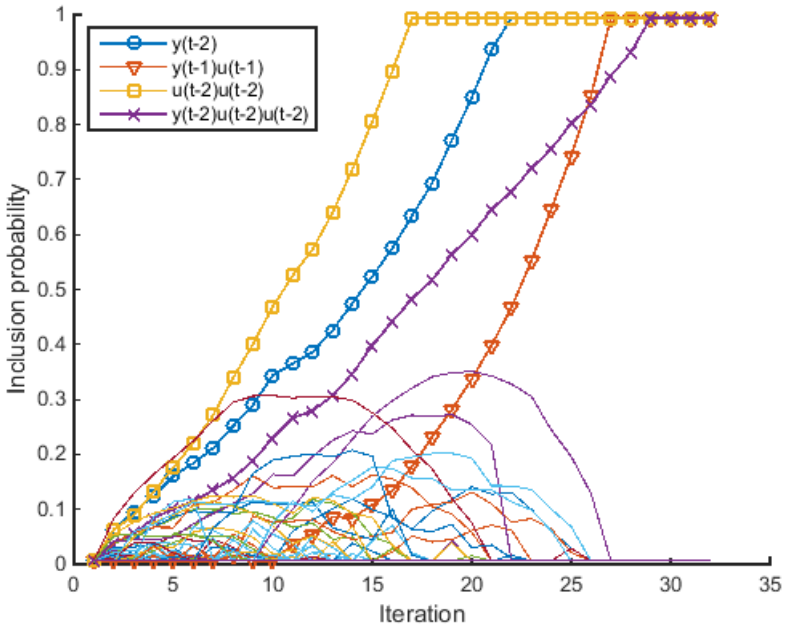


FIGURE 2.6: A typical run of the RaMSS - RIP.

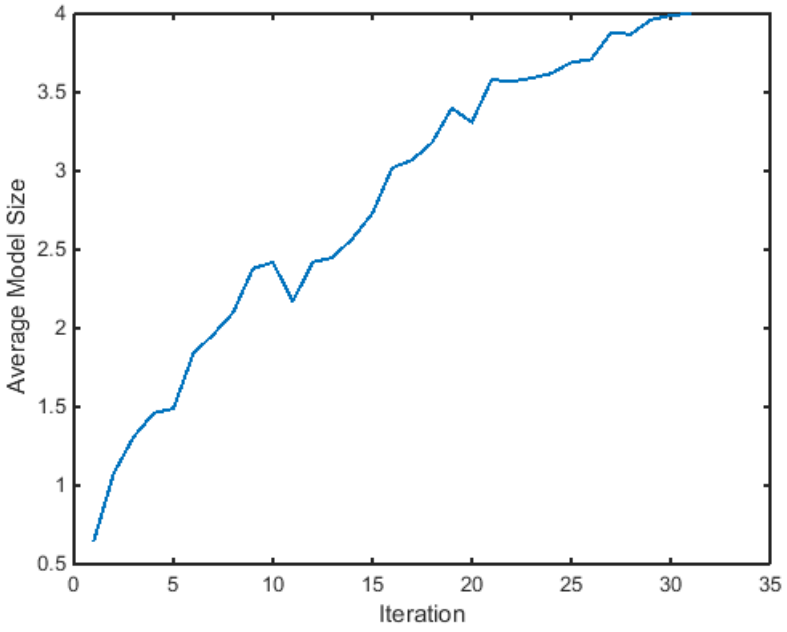


FIGURE 2.7: A typical run of the RaMSS - Average Model Size (AMS).

The RaMSS method is very attractive because of its simplicity and the encouraging results obtained on the analytic systems on which it has been tested. These facts have led us to further investigate the potential of this method. The

RaMSS is based on an independence assumption between regressors, whereby each regressor is extracted independently from the others. However, this independence is only conceptual since each RIP is updated according to the performance of the entire population of models and so, implicitly, each regressor is related somehow to the others. To better understand this concept, consider a simple example.

Example 4. We have the following system:

$$y(t) = y(t-1) + u(t-1)^2 + e(t).$$

Set $n_y = 1$, $n_u = 1$ and the maximum degree equal to 2. Thus, the set of all possible regressors will be

$$\{y(t-1), y(t-1)^2, y(t-1)u(t-1), u(t-1), u(t-1)^2\}$$

Suppose one extracts 5 models, thus obtaining the following models-regressors matrix:

$$\mathrm{M} = \begin{array}{c} \\ m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \end{array} \begin{array}{c} \begin{array}{ccccc} r_1 & r_2 & r_3 & r_4 & r_5 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \tag{2.27}$$

The fourth model matches exactly the system and therefore will have the highest performance. The performances of models m_3 and m_5 will be high since both models include only correct terms, but they will be worse than those of m_4 in which both terms occur. Accordingly, suppose that the performance vector J is as follows:

$$J = (0.974, 0.977, 0.986, 0.999, 0.983)^T \tag{2.28}$$

The RIPs of r_1 and r_5 will benefit from the presence of m_4 in the extracted population since the performance of this model will increase significantly the update factor, defined in Equation 2.26, of these two regressors. Conversely, the RIPs of r_2, r_3 and r_4 will not benefit from the presence of m_4. Indeed,

	considering m_4	without considering m_4
r_1	0.009	0.003
r_2	-0.009	-0.004
r_3	-0.984	-0.98
r_4	-0.012	-0.008
r_5	0.006	0

TABLE 2.1: Update factor values for the RIPs

Thus, we can say that the presence of model m_4 clusters the regressors in two groups: one containing the regressors which benefit from the the presence of m_4 and a second one containing the remaining ones. The model m_4 is defining a relation between regressors.

This fact agrees with an experimental evidence: often, as the algorithm progresses, the regressors tend to enter/disappear jointly in a model. Accordingly we can conclude that the regressors are somehow mutually related. This conclusion has motivated the work that will be presented in this thesis. The idea is to introduce second order information about relations among regressors in the MSS phase. In this new scenario, a Bernoulli distribution is associated to each regressor as before, which defines the probability of extraction of that term. However, unlike the classic RaMSS, this time the regressors are not independent but they are correlated with a given marginal mean vector and a correlation matrix.

Chapter 3

CLF based RaMSS

Correlated random variables are described by means of *multivariate distributions* a.k.a. *joint distributions*, which are parameterized by a mean vector μ and a covariance matrix V. The components of μ are the means of the different variables. The (i, j)-th component of V is the covariance between the i-th and j-th variable (the diagonal of V gives the variances of the variables). If these distributions are known, it is straightforward to simulate the associated correlated random variables by means of sampling methods, as described in Appendix A.2. However, fully specifying these distributions becomes impractical as the number of variables increases. In the literature, several methods have been proposed to simulate correlated random variables with specified mean vector and correlation matrix without fully specifying the joint distribution (Qaqish, 2003). The simulation method introduced in (Qaqish, 2003) is here used to simulate correlated *binary* variables which represent the regressors in the RaMSS. To ensure that the extracted models converge to the correct model we need a mechanism to update the correlation matrix as well, besides that for the mean vector (Equation 2.24). This second update rule must take into account the model performances.
The chapter is organized as follows. First, we introduce and discuss a mechanism to simulate correlated random variables. Then we define the update rule for the correlation matrix and finally we discuss how to modify the RaMSS.

3.1 The Conditional Linear Family

In (Qaqish, 2003) a method is introduced to simulate correlated binary variables, such as the Bernoulli variables associated to the regressors. Let Y denote an $[n \times 1]$ vector of Bernoulli random variables $(Y_1, \ldots, Y_n)^T$, with

- $E(Y) = (\mu_1, \ldots, \mu_n)^T = \mu$,
- $corr(Y) = \{r_{ij}\} = R$,
- $cov(Y) = \{v_{ij}\} = V$.

Note that it is easy to switch between (μ, V) and (μ, R) since for Bernoulli variables $var(Y) = v_{ii} = \mu_i(1 - \mu_i)$, so that specifying the pair (μ, V) is equivalent to specifying the pair (μ, R). Indeed, the correlation and the covariance are related

according to

$$r_{ij} = \frac{v_{ij}}{v_{ii}v_{jj}}, \qquad v_{ij} = r_{ij}v_{ii}v_{jj}. \tag{3.1}$$

So, if one knows the correlation matrix and the variances v_{ii}, the covariance matrix can be easily computed. The same holds if the roles were reversed.
For $i = 2, \ldots, n$ define

- $X_i = (Y_1, \ldots, Y_{i-1})^T$,
- $\theta_i = E(X_i)$,
- $G_i = cov(X_i)$,
- $s_i = cov(X_i, Y_i)$.

Generally, for $i = 2, \ldots, n$, the conditional distribution of Y_i given X_i can be expressed in the form

$$pr(Y_i = 1|X_i = x_i) = E(Y_i|X_i = x_i) = \mu_i + \sum_{j=1}^{i-1} \sum_{a \in A_{ij}} k_{ij}(a) \prod_{t=1}^{j} (y_{a_t} - \mu_{a_t}), \tag{3.2}$$

where A_{ij} is the set of integer j-vectors $\{a : 1 \leq a_1 < \cdots < a_j \leq i-1\}$ and each k_{ij} is an

$$\binom{i-1}{j} \times 1$$

vector conveniently indexed by elements of A_{ij}. What will be referred to as the *conditional linear family* (CLF) is obtained from Equation 3.2 by setting $k_{ij} = 0$, for $i = 2, \ldots, n$ and $j = 2, \ldots, i-1$, leading to

$$E(Y_i|X_i = x_i) = \mu_i + k_{i1}^T(x_i - \theta_i) \tag{3.3}$$

Furthermore, for a given (μ, V), the vector of $(i-1)$ parameters is given in closed form as

$$k_{i1} = G_i^{-1}s_i = b_i \ \textit{for}\ i = 2, \ldots, n, \tag{3.4}$$

and the corresponding CLF is given by

$$\lambda_i = \lambda_i(x_i; \mu, V) = E(Y_i|X_i = x_i) = \mu_i + b_i^T(x_i - \theta_i) = \mu_i + \sum_{j=1}^{i-1} b_{ij}(y_j - \mu_j) \tag{3.5}$$

where $y_j \in [0, 1]$ is the simulation's result of the variable Y_j.
The simulation algorithm proceeds as follows. First, simulate Y_1 as a Bernoulli with mean μ_1, and then, for $i = 2, \ldots, n$, simulate Y_i as a Bernoulli with conditional mean λ_i given by Equation 3.5. It then follows that the vector Y obtained has the required mean vector and covariance matrix. This algorithm allows unequal means and both positive and negative correlations but there are restrictions on the reproducibility of a given (μ, V), which holds if $\lambda_i \in [0, 1]$. First, V should be positive definite in order to compute its inverse needed in Equation

3.4. This restriction can be easily overcome if one considers the *pseudoinverse* of V, rather than the inverse, and thus also positive semidefinite covariance matrices can be considered. Details on the pseudoinverse of a matrix and on how to compute it iteratively are in Appendix A.5. Secondly, natural restrictions on r_{ij} are imposed by μ_i and μ_j, i.e. the marginal means limit the ranges of covariances and correlations. Specifically,

$$max(0, \mu_i + \mu_j - 1) \leq E(Y_i Y_j) \leq min(\mu_i, \mu_j) \ (i \neq j) \tag{3.6}$$

or equivalently,

$$max(-\psi_i \psi_j, \frac{-1}{\psi_i \psi_j}) \leq r_{ij} \leq min(\frac{\psi_i}{\psi_j}, \frac{\psi_j}{\psi_i}) \ (i \neq j) \tag{3.7}$$

where, $\psi_i = (\mu_i/(1-\mu_i))^{\frac{1}{2}}$.

3.2 Update rule for the correlation matrix

As presented previously, the simulation algorithm requires the mean vector and the correlation (or covariance) matrix. While the mean vector and its update rule are already defined in the RaMSS, the method lacks a corresponding rule for the correlation matrix. The idea is to iteratively construct this matrix starting from an identity matrix. The updating at the $(i+1)$-th iteration takes the form:

$$r_{ij}^{(i+1)} = \nu \cdot r_{ij}^{(i)} + (1-\nu) \cdot \textit{update factor} \tag{3.8}$$

where, r_{ij} is the (i,j)-th element of the correlation matrix R, ν is the learning factor and *update factor* is the increment for r_{ij}. The update factor must be a measure of the similarity between the two regressors, taking into account the performance of the models extracted at the i-th iteration. Therefore, first we have to define when two regressors are similar each other. Two regressors have an high similarity when the presence of one of them in a good model implies the presence of the other one. Vice versa, two regressors have a low similarity when the presence (or absence) of one of them in a good model implies the absence (or presence) of the other one. According to these definitions, we need to discover efficiently when two regressors appear jointly in the same models. An easy way to do that, is to use a *vector space model* (VSM) defined on regressors and models. A VSM is an algebraic model for representing objects as vectors of identifiers, in our case regressors as vectors of models. Specifically, consider a *NP-dimensional* vector space, where *NP* is the number of models extracted at each iteration of the algorithm. Into this vector space each model is identified by a unit vector

$$m_i = \left(\ 0, 1, 0, \ldots, 0 \ \right)$$

pointing in the direction of the i-th axis. The set of vectors m_i, $i = 1, \ldots, NP$, forms a canonical basis of the Euclidean space $\mathbb{R}^{NP}$.
Any regressor vector r_j can be represented by a canonical basis expansion

$$r_j = \sum_{i=1}^{NP} w_{ij} m_i.$$

The importance weights w_{ij} are defined as

$$w_{ij} = \begin{cases} J_i, & \text{if regressor } j \text{ appears in model } i \\ 0, & \text{otherwise} \end{cases}$$

where J_i is the performance of the i-th model. The entire VSM can be described by means of a $[m \times NP]$ *weighted regressor-model matrix* $\tilde{M}$, where m is the number of regressors and the $\tilde{M}_{ji}$ component is given by w_{ij}. The $\tilde{M}$ matrix can be viewed as:

$$\tilde{M} = M\tilde{J},$$

where:

- M is the binary regressor-model matrix, i.e. $w_{ij} = 1$ if regressor j appears in model i, 0 otherwise,
- $\tilde{J} = diagonal(J_1, J_2, \ldots, J_{NP})$.

Regressors are thus described in terms of the performance of the models in which they appear. Therefore, two regressors are similar if the associated vectors in the defined VSM are close. So, we need to quantify the distance between vectors. A widely used distance measure in a VSM is the *cosine* of the angle between vectors. Specifically, given two regressors r_i and r_j, the cosine similarity is defined as:

$$CosSim(r_i, r_j) = \frac{\sum_k r_{i_k} r_{j_k}}{\sqrt{\sum_k r_{i_k}^2} \sqrt{\sum_i r_{j_k}^2}} = \frac{\langle r_i, r_j \rangle}{\|r_i\| \|r_j\|}. \tag{3.9}$$

It is bounded between 0 and $+1$ since r_i and r_j are non-negative. In general, the cosine similarity is bounded between -1 and $+1$.
The cosine similarity measure defined on $\tilde{M}$ is a good candidate to be the update factor in Equation 3.8 in that it quantifies the co-occurrences of regressors in the models, taking into account implicitly also the model performances. This choice is motivated also by the following consideration. Correlation and cosine similarity are strongly related since both can be viewed as variants on the *inner product*. In general, given two vectors x and y, the *inner product* is defined as:

$$Inner(x, y) = \sum_i x_i y_i = \langle x, y \rangle$$

The inner product is unbounded. One way to make it bounded between -1 and $+1$ is to divide by the vectors' L_2 norms, giving the cosine similarity as in

Equation 3.9. Cosine similarity is not invariant to shifts. What is invariant is the *Pearson correlation*:

$$Corr(x,y) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2}\sqrt{\sum_i (y_i - \bar{y})^2}} = \frac{\langle (x_i - \bar{x}), (y_i - \bar{y}) \rangle}{\|(x_i - \bar{x})\| \|(y_i - \bar{y})\|}$$

where $\bar{x}$ and $\bar{y}$ are the respective means. The correlation is the cosine similarity between centered versions of x and y, again bounded between -1 and $+1$. The correlation is invariant to both scale and shift changes of x and y.
The update rule in Equation 3.8 can be thus expressed as:

$$r_{ij}^{(i+1)} = \nu \cdot r_{ij}^{(i)} + (1 - \nu) \cdot CosSim(r_i, r_j). \tag{3.10}$$

Defining

$$W = \tilde{M}\tilde{M}^T$$

$$K = [k_{ij}], \text{where } k_{ij} = \|r_i\| \|r_j\|$$

$$C = \left[\frac{w_{ij}}{k_{ij}}\right]$$

the Equation 3.10 takes the matrix form:

$$R^{(i+1)} = \nu \cdot R^{(i)} + (1 - \nu) \cdot C. \tag{3.11}$$

Matrix $R^{(i+1)}$ is symmetric by construction. However, a correlation matrix is a symmetric positive semidefinite matrix with unit diagonal. So we need a mechanism to compute the nearest correlation matrix to a given symmetric matrix. In (Higham, 2002) the problem considered by the author is, for arbitrary symmetric matrix $A \in \mathbb{R}^{n \times n}$, that of computing the distance

$$\gamma(A) = min\{||A - X|| : X \text{ is a correlation matrix}\}$$

and calculating the matrix that achieves this minimum distance. The norm is a weighted version of the Frobenius norm, i.e. $||A||_F = ||W^{\frac{1}{2}} A W^{\frac{1}{2}}||_F$, where W is a symmetric positive definite matrix.
Define the sets

$$S = \{Y = Y^T \in \mathbb{R}^{n \times n} : Y \geq 0\}$$

$$U = \{Y = Y^T \in \mathbb{R}^{n \times n} : y_{ii} = 1,\ i = 1, \ldots, n\}$$

We are looking for a matrix in the intersection of S and U that is closest to A in a weighted Frobenius norm. Since S and U are both closed convex sets, so is their intersection. This problem concerns a projection from the symmetric matrices onto the correlation matrices, with respect to a weighted Frobenius form. To find such matrix we might iteratively project by repeating the operation

$$A \longleftarrow P_U(P_S(A))$$

where, $P_U(A)$ and $P_S(A)$ are respectively, the projection of the matrix A onto space U and S.

Theorem 1. (Higham, 2002)

$$P_U(A) = A - W^{-1}diag(\theta_i)W^{-1},$$

where $\theta = [\theta_1, \ldots, \theta_n]^T$ is the solution of the linear system

$$(W^{-1} \cdot W^{-1})\theta = \phi$$

where $\phi = [a_{ii} - 1]$ for $i = 1, \ldots, n$.

In the case where W is diagonal we can write, more simply,

$$P_U(A) = (p_{ij}),\ p_{ij} = \begin{cases} a_{ij}, & (i \neq j) \\ 1, & (i = j) \end{cases}$$

For $P_S(A)$, we need to define two auxiliary matrices. Given a symmetric matrix $A \in R^{n \times n}$ with *spectral decomposition* $A = QDQ^T$, where $D = diag(\lambda_i)$ and Q is an orthogonal matrix, let

$$A_+ = Qdiag(max(\lambda_i, 0))Q^T,\ A_- = Qdiag(min(\lambda_i, 0))Q^T$$

Theorem 2. (Higham, 2002)

$$P_U(S) = W^{-\frac{1}{2}}((W^{\frac{1}{2}}AW^{\frac{1}{2}})_+)W^{-\frac{1}{2}}$$

Using the defined projections $P_U(A)$ and $P_U(S)$, one can define the following general algorithm:

Algorithm 3.1 Nearest Correlation Matrix

1: $\Delta S_0 = 0$; ▷ Correction term
2: $Y_0 = A$; ▷ Starting real symmetric matrix
3: **for** k=1,2,... **do**
4: $R_k = Y_{k-1} - \Delta S_{k-1}$;
5: $X_k = P_S(R_k)$;
6: $\Delta S_k = X_k - R_k$;
7: $Y_k = P_U(X_k)$;
8: **end for**

The result is a correlation matrix, i.e. a symmetric positive semidefinite with unit diagonal matrix.

Example 5. Suppose one has the following unweighted regressor-model matrix:

$$\begin{array}{c} \\ r_1 \\ r_2 \\ r_3 \end{array} \begin{array}{c} \begin{array}{ccc} m_1 & m_2 & m_3 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \end{array}$$

r_1 co-occures with r_2 in the m_2, r_2 co-occures with r_3 in the m_1 and finally, r_1 co-occures with r_3 in the m_3. Without importance weights, all the co-occurrences are equally important. In fact, the cosine similarity measures are:

$r_1 - r_2$**:** 0.5,

$r_1 - r_3$**:** 0.5,

$r_2 - r_3$**:** 0.5.

Consider the following model performances vector:

$$J = (1, 0.2, 1)^T.$$

The resulting weighted regressor-model matrix is:

$$\begin{array}{c} \\ r_1 \\ r_2 \\ r_3 \end{array} \begin{array}{c} \begin{array}{ccc} m_1 & m_2 & m_3 \end{array} \\ \begin{bmatrix} 0 & 0.2 & 1 \\ 1 & 0.2 & 0 \\ 1 & 0 & 1 \end{bmatrix} \end{array}$$

and the similarity measures become:

$r_1 - r_2$**:** 0.0385,

$r_1 - r_3$**:** 0.6934,

$r_2 - r_3$**:** 0.6934.

As one can note, the $r_1 - r_2$ similarity is now very low since it concerns a co-occurrence in a bad model (m_2). Meanwhile, the $r_1 - r_3$ similarity is increased, since the two regressors co-occur in a good model (m_3) and do not co-occur in a bad model (m_2). The same holds for the pair $r_2 - r_3$.
Consider now the following models performances vector:

$$J = (1, 0.2, 0.2)^T.$$

The resulting weighted regressor-model matrix is:

$$\begin{array}{c} \\ r_1 \\ r_2 \\ r_3 \end{array} \begin{array}{c} \begin{array}{ccc} m_1 & m_2 & m_3 \end{array} \\ \begin{bmatrix} 0 & 0.2 & 0.2 \\ 1 & 0.2 & 0 \\ 1 & 0 & 0.2 \end{bmatrix} \end{array}$$

and the similarity measures become:

$r_1 - r_2$**:** 0.1387,

$r_1 - r_3$**:** 0.1387,

$r_2 - r_3$**:** 0.9615.

The $r_1 - r_2$ similarity is slightly increased since now it takes into account also the no co-occurrence in the bad model m_3. The $r_1 - r_3$ similarity is fell down since now there is anymore a co-occurrence in a good model. The highest similarity regards the regressors r_2 and r_3 which co-occur in the good model m_1 and do not co-occur in both bad models m_2 and m_3.
To sake of completeness, consider the case in which all the models are poor:

$$J = (0.2, 0.2, 0.2)^T.$$

The resulting weighted regressor-model matrix is:

$$\begin{array}{c} \\ r_1 \\ r_2 \\ r_3 \end{array} \begin{array}{c} \begin{array}{ccc} m_1 & m_2 & m_3 \end{array} \\ \begin{bmatrix} 0 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0 \\ 0.2 & 0 & 0.2 \end{bmatrix} \end{array}$$

and the similarity measures become:

$r_1 - r_2$**:** 0.5,

$r_1 - r_3$**:** 0.5,

$r_2 - r_3$**:** 0.5.

These results are reasonable in that, if there are no truly good models, the poor ones become the current good ones.
Therefore, including the models performance in the VSM is essential to correctly quantify the importance of the co-occurrences.

Remark 1. The VSMs are widely used in the Information Retrieval field where the goal is to retrieve a set of relevant documents to a given query. In this field, queries and documents are defined as vectors of components. Each component of the vector reflects a particular concept, key word, or *term* associated with the given document. The value assigned to that component reflects the importance of the term in representing the semantics of the document. Typically, the value is a function of the frequency with which the term occurs in the document or in the document collection as a whole. An often used function is the *td-idf* statistic whose value increases proportionally to the number of times a term appears in the document, but is offset by the frequency of the term in the collection, which helps to adjust for the fact that some terms appear more frequently in general. A collection of d documents described by t terms is represented as a $[t \times d]$ *term-by-document matrix*. The d vectors representing the d documents form the columns of the matrix. Thus, the matrix element $A[i, j]$ is the weighted frequency at which term i occurs in document j.

Query matching consists in finding the documents most similar to a given query. In the VSM, the documents selected are those geometrically closest to the query according to some measure, such as the cosine similarity.

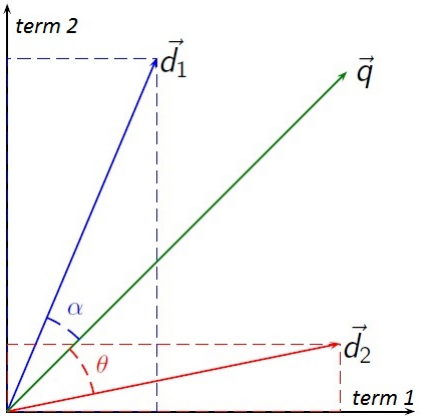


FIGURE 3.1: Vector space model in the Information Retrieval filed.

Example 6. Suppose one has the following set of five documents:

d_1 : *Romeo and Juliet.*

d_2 : *Juliet: O happy dagger!*

d_3 : *Romeo died by dagger.*

d_4 : *"Live free or die", that's the New-Hampshire's motto.*

d_5 : *Did you know, New-Hampshire is in New-England.*

and a search query: *die, dagger*.
Let A be the term-document matrix, where $A[i,j] = a$ if term i occurs a times in the document j:

$$\begin{array}{c} \\ romeo \\ juliet \\ happy \\ dagger \\ live \\ die \\ free \\ new{-}hampshire \end{array} \begin{array}{c} \begin{array}{ccccc} d_1 & d_2 & d_3 & d_4 & d_5 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

Clearly, d_3 should be ranked top of the list since it contains both *die, dagger*. Then d_2 and d_4 should follow, each containing a word of the query. Indeed, the query vector is given by

$$q = (0, 0, 0, 1, 0, 1, 0, 0)^T,$$

and the cosine similarity measures are, in order, 0, 0.4082, 0.8165, 0.3536 and 0. However, what about d_1 and d_5? Should they be returned as possibly relevant document to this query? As humans we know that d_1 is quite related to the query. On the other hand, d5 is not so much related to the query. Thus, we would like d_1 to be ranked higher than d_5. Several approaches have been developed to solve these failures of the basic VSM, such as Latent Semantic Analysis.

3.3 The C-RaMSS algorithm

The CLF tool presented in the Section 3.1 and the update rule for the correlation matrix defined in the Section 3.2 can be exploited to define a variant of the RaMSS algorithm that will be referred as the C-RaMSS algorithm (*Conditioned-RaMSS*). More in detail, at each regressor φ_i is associated a Bernoulli random variable ρ_i which is used to establish if φ_i belongs to the model that we are extracting. The regressors are dependent on each other and the conditional probability of extraction of each regressor is computed according to Equation 3.5. The RIP vector and the correlation matrix R are used by the CLF method as reference statistics for the new population of extracted models. The m regressors are initially uniformly distributed while the correlation matrix R is initialized as an identity matrix. At each iteration a population of NP models is extracted. The linear coefficients b_i depend only on the covariance matrix V, so they are constant for all models, while the conditional probabilities depend also on the models due to $y_j \in [0, 1]$. So, for each regressor we can simultaneously simulate the extraction for all the models, considering for each model, its relative λ_i. A check is done to verify that no model is empty. If this happens, we randomly switch a regressor. Once the models have been extracted, parameter estimates are carried out with LS. A statistical *t-test* is performed in order to remove redundant regressors and then the parameters are re-estimated. The significance confidence level α for the statistical test is a tuning parameter. The performance of each model is evaluated through the index defined in Equation 2.23, where factor K is a tuning parameter. Finally, the correlation matrix and the RIPs are updated according to the rules in Equations 3.11 and 2.24. The learning factor ν is a tuning parameter. Notice that the tuning rule for the RIPs does not ensure that the new parameters μ_j remain in the [0, 1] interval. Therefore, suitable saturation thresholds μ_{min} and μ_{max} must be introduced to keep the μ_j in the mentioned interval. The Algorithm 3.2 summarizes the whole model identification procedure.

Algorithm 3.2 C-RaMSS algorithm

Input:$\{(u(k), y(k)), k = 1, \cdots, N\}, \Re = \{\varphi_j(k), j = 1, \cdots, m\}$,
$NP, K, \nu, \alpha, \mu, \mu_{min}, \mu_{max}$
Output:μ
$\mu_i \leftarrow \frac{1}{m}$;
$R \leftarrow$ *identity matrix*;
repeat
    Compute the covariance matrix V;
    $\psi_p(k) = []$;
    $\tau_p = 0$;
    for p=1,...,NP **do** ▷ Generate first regressor
        Extract $r_{p,1}$ from $Be(\mu_1)$;
        if $r_{p,1} = 1$ **then** ▷ Add regressor
            $\psi_p(k) \leftarrow \varphi_1(k)$;
            $y_{p,1} \leftarrow 1$;
            $\tau_p \leftarrow \tau_p + 1$;
        else
            $y_{p,1} \leftarrow 0$;
        end if
    end for
    for i=2,...,m **do** ▷ Generate regressors
        $G_i \leftarrow V[1, \cdots, i-1; 1, \cdots, i-1]$;
        $s \leftarrow V[i; 1, \cdots, i-1]$;
        $b_i \leftarrow pseudoinverse(G_i) \cdot s$; ▷ Generate linear coefficients
        for p=1,...,NP **do**
            $\lambda_i \leftarrow \mu_i + \sum_{j=1}^{i-1} b_{ij}(y_{p,j} - \mu_j)$; ▷ Compute conditional probabilities
            Extract $r_{p,i}$ from $Be(\lambda_i)$;
            if $r_{p,i} = 1$ **then** ▷ Add regressor
                $\psi_p(k) \leftarrow [\psi_p(k)^T \varphi_i(k)]^T$;
                $y_{p,i} \leftarrow 1$;
            else
                $y_{p,i} \leftarrow 0$;
            end if
        end for
    end for
    Check for empty model;

for p=1,...,NP **do**
  $\hat{\theta} \leftarrow \left(\sum_{k=1}^{N} \psi_p(k)\psi_p(k)^T\right)^{-1} \sum_{k=1}^{N} \psi_p(k)y(k)$; ▷ Estimation
  $VAR \leftarrow \left(\sum_{k=1}^{N} \psi_p(k)\psi_p(k)^T\right)^{-1}$;
  $\hat{\sigma_e^2} \leftarrow \frac{1}{N-\tau_p} \sum_{k=1}^{N} (y(k) - \psi_p(k)^T\hat{\theta})$;
  for h=1,...,τ_p **do** ▷ Remove redundant terms
    $\sigma_h^2 \leftarrow \sigma_e^2 \cdot VAR_{h,h}$;
    if $|\hat{\theta_h}| \leq \hat{\sigma_h} \cdot t_{\alpha,N-\tau_p}$ **then**
      Remove regressor $\psi_{p,h}(k)$ from $\psi_p(k)$;
    end if
  end for
  $\hat{\theta} \leftarrow \left(\sum_{k=1}^{N} \psi_p(k)\psi_p(k)^T\right)^{-1} \sum_{k=1}^{N} \psi_p(k)y(k)$; ▷ Re-estimation
  $J_p \leftarrow e^{-K \cdot PE_p}$; ▷ Model evaluation
end for
for j=1,...,m **do** ▷ Update RIPs
  $J^+ \leftarrow 0$; $n^+ \leftarrow 0$; $J^- \leftarrow 0$; $n^- \leftarrow 0$;
  for p=1,...,NP **do**
    if $\varphi_j \in \psi_p(k)$ **then**
      $J^+ \leftarrow J^+ + J_p$; $n^+ \leftarrow n^+ + 1$;
    else
      $J^- \leftarrow J^- + J_p$; $n^- \leftarrow n^- + 1$;
    end if
  end for
  $\mu_j \leftarrow \mu_j + \gamma \left(\frac{J^+}{max(n^+,1)} - \frac{J^-}{max(n^-,1)}\right)$;
  $\mu_j \leftarrow max\left(min\left(\mu_j, \mu_{max}\right), \mu_{min}\right)$;
end for
$M \leftarrow (\psi_1(k), \ldots, \psi_{NP}(k))$; ▷ Define unweighted regressor-model matrix
$\tilde{M} \leftarrow M \cdot diag(J_1, \ldots, J_{NP})$; ▷ Define weighted regressor-model matrix
$W \leftarrow \tilde{M} \cdot \tilde{M}^T$;
$K_{i,j} \leftarrow \|\varphi_i\|\|\varphi_j\|$;
$C_{i,j} \leftarrow \frac{W_{i,j}}{K_{i,j}}$;
$R \leftarrow \nu R + (1-\nu)C$; ▷ Update correlation matrix R
Compute the nearest correlation matrix to R according to Algorithm 3.1;
Check constraints on R values according to Equation 3.7;
until Stopping criterion on μ_i is not met ▷ Stopping criterion

Chapter 4

Simulation results

In this chapter several simulation examples are discussed to show the effectiveness of the C-RaMSS algorithm. First, a typical run of the algorithm has been shown to illustrate the behavior of the proposed algorithm. Then an analysis of the algorithm performance has been carried out considering different values of the several tuning parameters. Finally, a comparative analysis w.r.t. the RaMSS, the FROE and the RJMCMC algorithms has been carried out to assess the goodness of the C-RaMSS method. This analysis has been performed considering some systems taken from the literature (Wei and Billings, 2008; Baldacchino, Anderson, and Kadirkamanathan, 2013; Bonin, Seghezza, and Piroddi, 2010; Piroddi and Spinelli, 2003; Mao and Billings, 1997; Aguirre, Barbosa, and Braga, 2008):

S_1:

$$y(k) = -1.7y(k-1) - 0.8y(k-2) + u(k-1) + 0.8u(k-2) + e(k),$$

with $u(k) \sim WUN(-2, 2)$, $e(k) \sim WGN(0, 0.01)$

S_2:

$$y(k) = 0.7y(k-1)u(k-1) - 0.5y(k-2) - 0.7 * y(k-2)u(k-2)^2 + 0.6u(k-2)^2 + e(k),$$

with $u(k) \sim WUN(-1, 1)$, $e(k) \sim WGN(0, 0.04)$

S_3:

$$y(k) = 0.8y(k-1) + 0.4u(k-1) + 0.4u(k-1)^2 + 0.4u(k-1)^3 + e(k),$$

with $u(k) \sim WGN(0, 0.3)$, $e(k) \sim WGN(0, 0.01)$

S_4:

$$y(k) = 0.5y(k-1) + 0.8u(k-2) + u(k-1)^2 - 0.05y(k-2)^2 + 0.5 + e(k),$$

with $u(k) \sim WGN(0, 0.3)$, $e(k) \sim WGN(0, 0.01)$

S_5:

$$\begin{aligned} y(k) = {} & 0.2y(k-1)^3 + 0.7y(k-1)u(k-1) - 0.5y(k-2) - \\ & -0.7y(k-2)u(k-2)^2 + 0.6u(k-2)^2 + e(k), \end{aligned}$$

with $u(k) \sim WUN(-1, 1)$, $e(k) \sim WGN(0, 0.01)$

S_6**:**

$$y(k) = 0.75y(k-2) + 0.25u(k-1) - 0.2y(k-2)u(k-2) + e(k),$$

with $u(k) \sim WGN(0, 0.25)$, $e(k) \sim WGN(0, 0.02)$

where $WGN(\eta, \sigma^2)$ is a white noise with a Gaussian distribution with mean η and standard deviation σ, while $WUN(a, b)$ is a white noise with Uniform distribution in the interval $[a, b]$.

All the tests have been performed in MATLAB 2014b enviroment, on an HP ProBook 650 G1 CORE i7-4702MQ CPU @2.20 GHz with 8GB of RAM.

4.1 Typical run of the C-RaMSS algorithm

A typical run of the C-RaMSS algorithm is analyzed considering the system S_2. The candidate regressor set contains all the monomials obtained as combination of the the lagged input and output signals with maximum lags, n_u and n_y, equal to 4 and maximum degree equal to 3 for a total of $m = 165$ regressors. The number of models generated at each iteration, NP, is set to 200 and the initial RIPs equal to $\mu_j = \frac{1}{m}$. By doing so, all the regressors have initially the same probability of being extracted and the early models have typically a small size. The initial correlation matrix R is initialized as an identity matrix in order not to influence the model extraction at the first iteration. The tuning parameter K in the performance index J is set to 1 and the learning factor ν is set to 0.1.

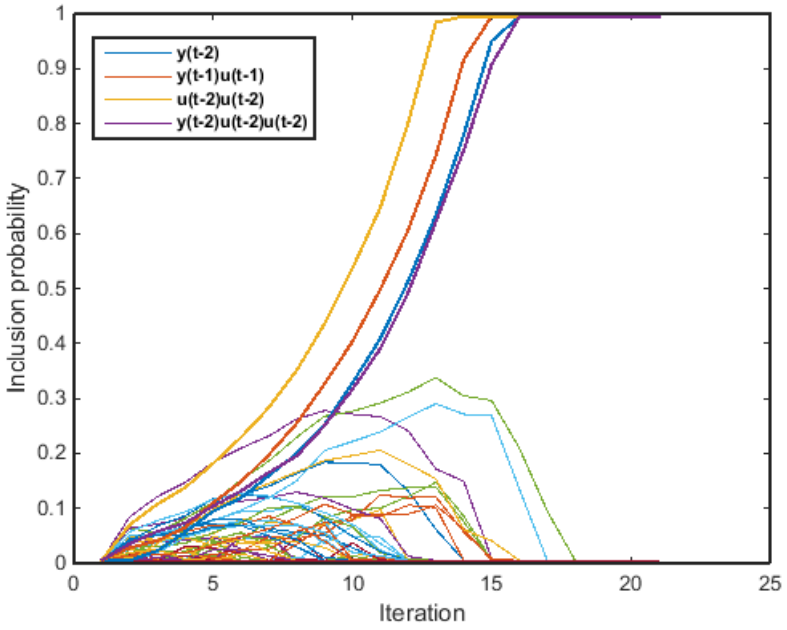


FIGURE 4.1: A typical evolution of the RIPs.

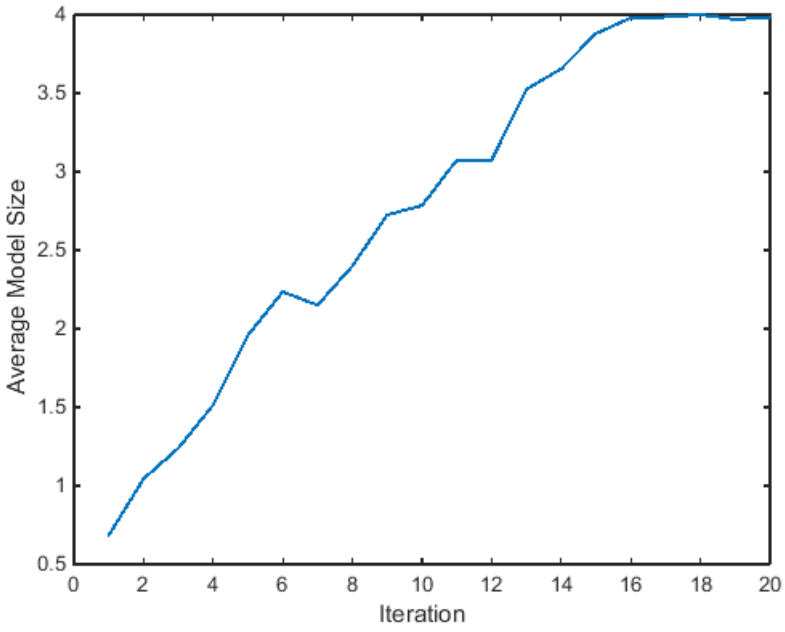


FIGURE 4.2: A typical evolution of the Average Model Size value.

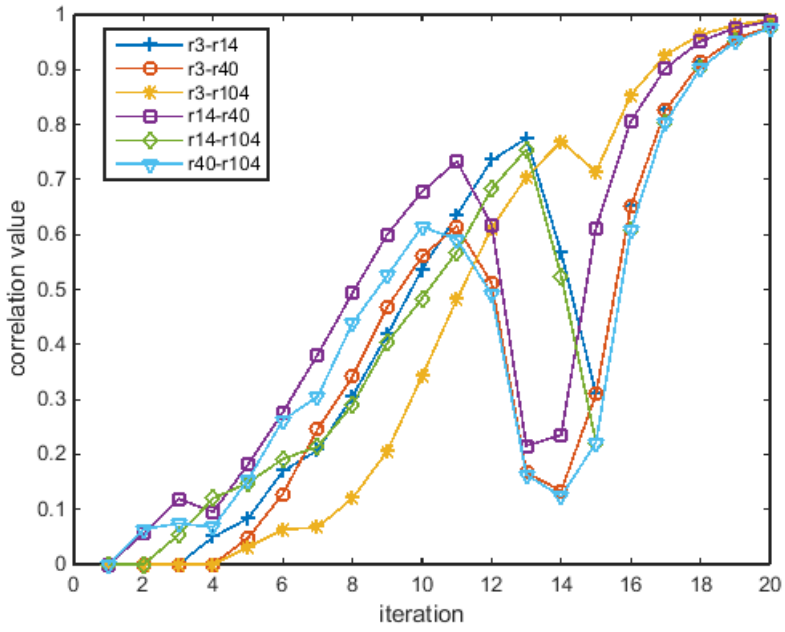


FIGURE 4.3: Evolution of the correlation associated to true regressors - S_2

Figures 4.1, 4.2 and 4.3 illustrate respectively the evolution of the RIPs, that of the *average model size* (AMS) value and that of the correlation value associated to pairs of true regressors. The RIPs associated to the correct (*true*) regressors are gradually increasing until they reach 1. On the other hand the RIPs associated to the other regressors, after a transient, tend to 0. The AMS is globally monotonically increasing to the correct value. This emphasizes the incremental nature of the selection procedure which starts from very small models and progressively increases the (average) size of the explored models. The correlation values gradually reach 1 but the evolution is not monotonically increasing. Why does it happen? One can note in Figure 4.1 that around the fifteenth iteration some RIPs associated to spurious regressors are still quite high, leading to the generation of models with both these regressors and the true ones. This implies that the corresponding correlation values are different from zero. Thanks to the update rule 2.24 the RIPs associated to the spurious regressors progressively decrease and these regressors tend to disappear from models. The presence into models of true regressors correlated with the spurious ones is slightly affected as well, according to Equation 3.5. This leads to a temporary reduction of the similarity between true regressors. This side effect is temporary since the RIPs associated to true regressors increase anyway thanks to the significant gap between them and those associated to the spurious regressors, which ensures their inclusion in at least few models. The correlation between true terms, which is significant despite its reduction, helps as well to ensure the inclusion of these terms in models. The correlation values, not reported in Figure 4.3, associated to pairs of spurious regressors or pairs of regressors that do not co-occur frequently tend to zero.
For the sake of completeness, the parameters estimate of the identified model is reported in Table 4.1.

	real value	estimated value
θ_1	-0.5	-0.5049
θ_2	0.7	0.7013
θ_3	0.6	0.5897
θ_4	-0.7	-0.6699

TABLE 4.1: Estimated parameters

Other model selection problems may turn out to be more difficult to tackle compared to the previous example. Consider e.g. Figures 4.4, 4.5 and 4.6 which refer to a run of the algorithm on system S_4. The final model selection is correct, but this time some spurious regressors have a long transient and the relative RIPs are quite high for some time. This implies that these regressors are present in several intermediate models. Another interesting fact is that at a certain point, roughly at around the 25-th iteration, there is a sort of restart of the procedure. This can be noted from all the evolution curves. In fact, in Figure 4.4 it is evident how at a certain point some RIPs associated to spurious regressors, that until then had a rising trend, suddenly converge to 0. In the meanwhile, some of the other regressors, including one of the true terms which is missing,

start to become relevant. In Figure 4.5, at around the same point, there is a sudden rise of the model size towards the true value. This is due to the random nature of the algorithm which allows to exit from a local optimum point of the search space, leading to the correct model structure selection. Therefore, these jumps encourage a global search in the solution space. In Figure 4.6 one can note that around the 25-th iteration some true regressors start to be correlated, according to the stable inclusion of these regressors into the models.

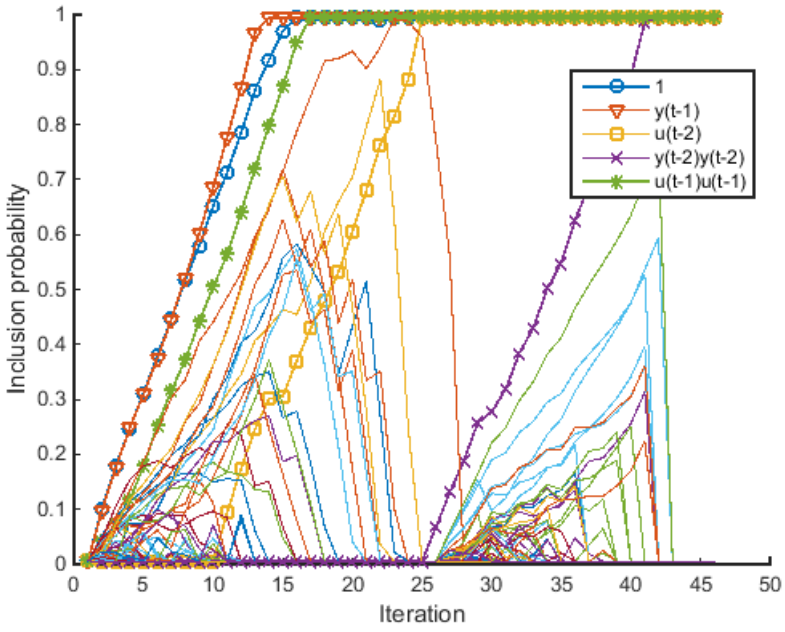


FIGURE 4.4: A not straightforward evolution of the RIPs.

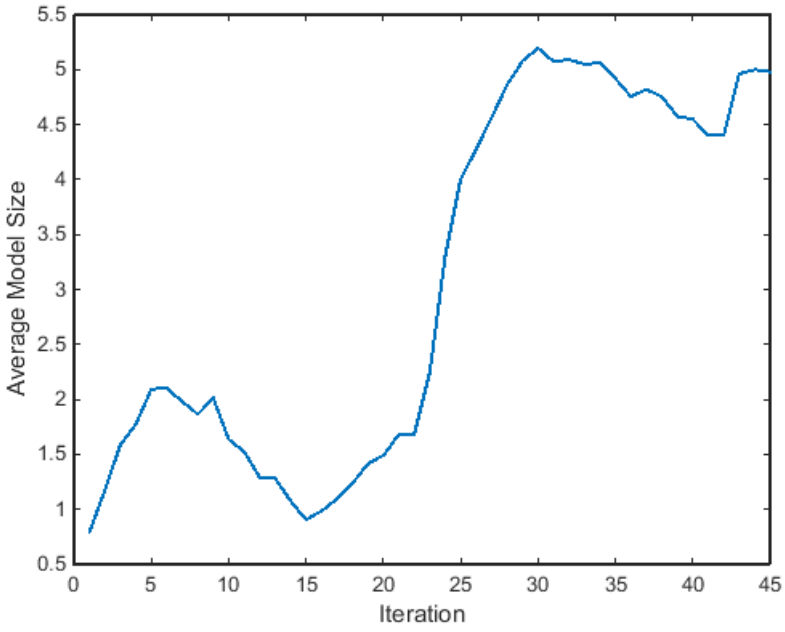
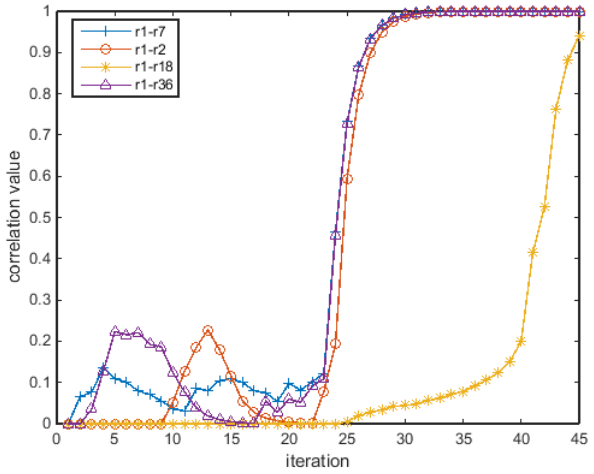
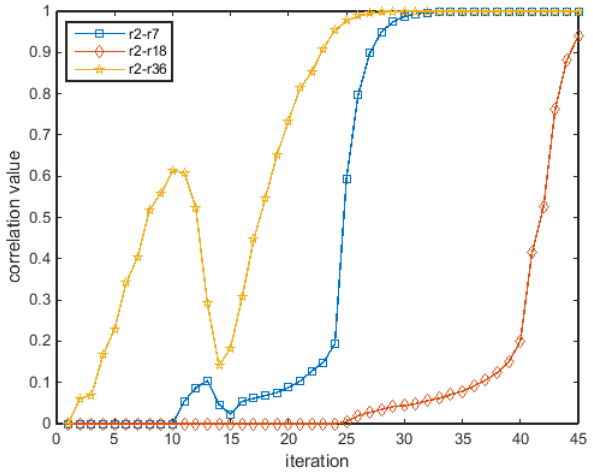


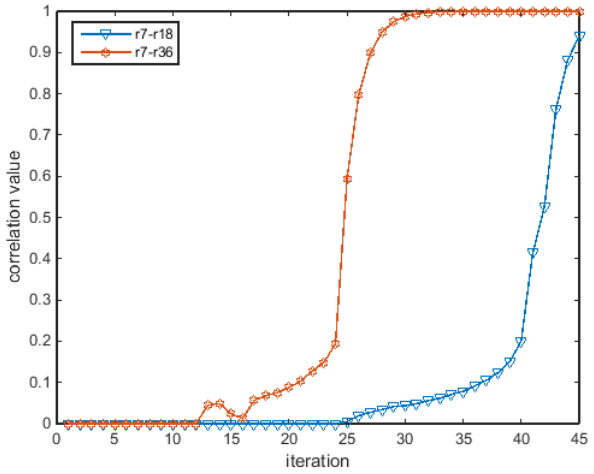
FIGURE 4.5: A not straightforward evolution of the Average Model Size value.



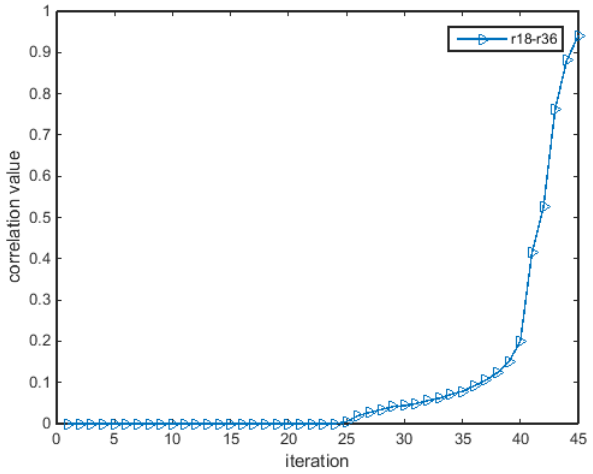
r_1 w.r.t. r_2, r_7, r_{18} and r_{36}



r_2 w.r.t. r_7, r_{18} and r_{36}



r_7 w.r.t. r_{18} and r_{36}



r_{18} w.r.t. r_{36}

FIGURE 4.6: Evolution of the correlation associated to true regressors - S_4

4.2 C-RaMSS algorithm performance

To collect some statistics about the algorithm performance, 300 runs of the C-RaMSS algorithm have been carried out for each system and the results have been aggregated. For each system, the data-set is composed of 500 randomly generated input/output pairs. To ensure the reproducibility and consistency of the results the random number generator is always initialized with the same seed. Several statistics have been studied:

number of iterations: number of performed algorithm iterations,

elapsed time: time required to obtain the final model,

correctness: percentage of exact model selections,

explored models: number of distinct models explored by the algorithm,

maximum AMS: maximum model size amongst all the mean sizes computed at each iteration,

final AMS: average model size of the last iteration.

The aggregated results in Table 4.2 are obtained with the following configuration of the tuning parameters: $n_u = n_y = 4$, maximum order of non linearity is set to 3, $NP = 200$, $K = 1$, $\nu = 0.1$ and $\alpha = 0.997$.

	S_1	S_2	S_3	S_4	S_5	S_6
Correct selection	88.3%	100%	100%	91.3%	100%	100%
# of Iterations	29.57	20.41	19.6	58.77	26.32	98.66
Elapsed Time [sec]	15.63	10.56	10.5	31.92	16.96	48.52
Maximum AMS	4.65	4.02	4.0	5.11	5.01	2.68
Final AMS	4.08	3.99	3.98	4.93	4.98	2.66
Explored Models	562.6	643.0	584.6	796.1	839.0	377.91

TABLE 4.2: C-RaMSS - Overall view

Each cell of Table 4.2 reports the average value over 300 runs of the corresponding statistic. Regardless of the correctness of the selected models, the final values for the AMS show that the algorithm tends to select models which have the actual number of regressors. Actually, these final values for the AMS are slightly different from the actual model size values due to the thresholds imposed on the RIP values. Indeed, $\mu_{min} > 0$ and $\mu_{max} < 1$ and therefore there is no absolute certainty that a specific regressor is or is not extracted. Therefore, in the population of extracted models there may be some "outliers" that cause these statistics to drift from their actual values. Considering the maximum AMS values, one can note how close they are to their relative final values, except for system S_1, whose maximum AMS value is somewhat bigger than the final. This implies that the algorithm mainly explores solutions of size smaller or equal to the actual one. The large gap between the two statistics of system

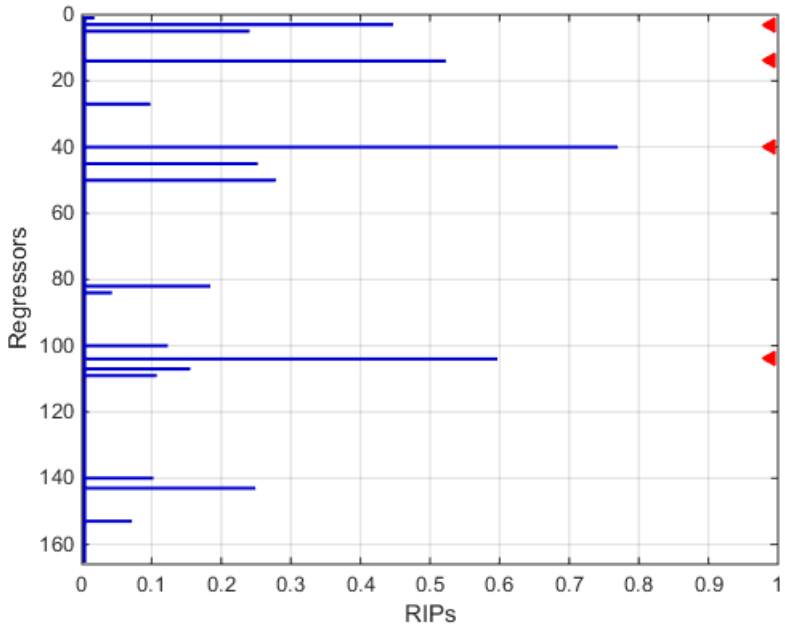


FIGURE 4.7: RIP value for each regressor. The red triangular markers are used to identify the correct regressors.

S_1 means that the algorithm remains stalled for some time before converging to the true model on some local optimum points of the search space which represent models of larger size. As pointed out in (Falsone, Piroddi, and Prandini, 2014) this aspect may be due to the fact that the chosen family is largely over-parametrized with respect to the simple linear structure of S_1. Notice how the MSS process explores an infinitesimal fraction of the $(2^{165} - 1)$ possible models. Nonetheless, this tiny fraction is enough to drive the MSS procedure toward the correct model (or an equivalent one) thanks to the update rules for the RIPs and the correlation matrix which exploit the information extracted from all the partial models, so that no information is wasted. As an example, consider Figure 4.7 that illustrates the RIPs at the tenth iteration of the algorithm on system S_2. The gap between the RIPs relative to the correct regressors and the remaining ones demonstrates how this small set of information is enough to select the correct structure after few iterations.

We next analyze the impact of the learning factor ν, Equation 3.11, on the selection procedure.

4.2.1 Learning factor

Consider the following configuration of the tuning parameters: $n_u = n_y = 4$, maximum order of non linearity is set to 3, $NP = 200$, $K = 1$ and $\alpha = 0.997$. The algorithm has been tested on several values of the learning factor. Notice that it does not make sense to take $\nu > 0.5$ since then older data would be more influential than recent ones.

	S_1	S_2	S_3	S_4	S_5	S_6
Correct selection	85.3%	100%	100%	90.7%	100%	100%
# of Iterations	28.78	20.62	20.35	56.4	27.41	96.38
Elapsed Time [sec]	15.63	11.43	11.73	31.56	18.3	47.59
Maximum AMS	4.6	4.01	3.99	5.14	5.01	2.66
Final AMS	4.11	3.98	3.98	4.97	4.98	2.64
Explored Models	551.2	668.6	593	808.4	894.4	378.63

TABLE 4.3: C-RaMSS - $\nu = 0.3$.

	S_1	S_2	S_3	S_4	S_5	S_6
Correct selection	84.3%	100%	100%	92.3%	100%	100%
# of Iterations	29.59	21.13	20.02	58.55	28.52	86.05
Elapsed Time [sec]	17.87	11.54	12.22	36.43	16.37	43.16
Maximum AMS	4.6	4.0	4.0	5.12	5.02	2.57
Final AMS	4.10	3.98	3.989	4.93	4.98	2.55
Explored Models	573	730	599.7	840.3	965.6	394.09

TABLE 4.4: C-RaMSS - $\nu = 0.5$.

Regarding $\nu = 0.1$ the aggregated results are those reported in Table 4.2. Considering Tables 4.2, 4.3 and 4.4 one can immediately note that amongst all the statistics the number of explored models is the most sensitive to ν while the others are less sensitive to it. Why does the number of explored models slightly increase as ν increases? At the early iterations the small model size leads to many zero regressor vectors and thus the cosine similarity between two of them is set to zero, according to the definition of similarity between regressors which is based on the concept of co-occurrences (Section 3.2). As ν increases, the correlation update rule takes progressively into greater account the past correlation values and this helps to have denser correlation matrices despite the actual density of the cosine similarity matrix. Denser correlation matrices influence more significantly the extraction of the models, through the CLF method, possibly leading to the inclusion of regressors despite of the real value of the associated RIPs. This causes a greater exploration of the solutions space. Then, as the algorithm proceeds, these past correlation values become always smaller due to the learning factor less than 1, and eventually go to zero if the associated pairs of regressors no longer occur.
The wider exploration of the solution space does not imply an increase in the maximum and final AMSs. This supports the thesis that the wider exploration

affects mainly the early iterations in which the extracted models have always size smaller than the correct one (Figure 4.2).
Systems S_2, S_3, S_5 and S_6 are not affected by the increase in the learning factor. In fact, the algorithm is always able to find the correct model for these systems. Instead, this does not happen with system S_1 for which the algorithm fails progressively more as ν increases. As said before, this may be due to the fact that S_1 has a simple linear structure. The algorithm fails to identify the correct model as well on system S_4. This time there is not a strong relation between ν and the correctness value, the case $\nu = 0.3$ being slightly worse than $\nu = 0.1$ while $\nu = 0.5$ is the best.
To recap, the learning factor has a direct effect on the number of explored models, which slightly increases as ν increases. There does not appear to be a strong relation between ν and the correctness. All the other statistics are not influenced by ν.

We next analyze the impact of the number of extracted models at each iteration, NP, on the selection procedure.

4.2.2 Number of extracted models

Consider the relation between the analyzed statistics and NP, the number of models extracted at each iteration. Intuitively, the elapsed time may be strongly correlated to the increase in NP since the time required to generate all the models increases. Also the number of explored models may be affected by NP since as NP increases, the probability to extract distinct models increases, mostly in the early iterations. The results presented here have been obtained by setting $n_u = n_y = 4$, maximum order of non linearity equal to 3, $K = 1$, $\alpha = 0.997$. The value of ν has been chosen according to the analysis carried on the learning factor. Specifically, the values $\nu = 0.1$ for system S_2 and $\nu = 0.5$ for system S_4 have been chosen.

	$NP = 25$	$NP = 50$	$NP = 75$	$NP = 100$	$NP = 200$	$NP = 300$
Correct selection	93.7%	100%	100%	100%	100%	100%
# of Iterations	54.33	33.72	26.54	24.23	20.41	19.33
Elapsed Time [sec]	15.18	11.4	9.67	10.1	10.55	13.85
Maximum AMS	5.12	4.44	4.18	4.13	4.02	4.0
Final AMS	4.13	3.98	3.98	3.98	3.99	3.99
Explored Models	182.7	235.3	299.5	376.4	642.95	815.57

TABLE 4.5: C-RaMSS - Analysis on S_2 of the effect of NP.

	$NP = 25$	$NP = 50$	$NP = 75$	$NP = 100$	$NP = 200$	$NP = 300$
Correct selection	67%	75.3%	76.3%	85.3%	92.3%	93.3%
# of Iterations	70.14	49.96	51.43	52.21	58.55	54.31
Elapsed Time [sec]	22.22	18.14	20.13	24.08	36.43	41.24
Maximum AMS	6.94	5.94	5.72	5.49	5.12	5.09
Final AMS	5.45	5.3	5.28	5.15	4.93	4.99
Explored Models	269.32	362.83	493.92	580.23	840.28	1040.4

TABLE 4.6: C-RaMSS - Analysis on S_4 of the effect of NP.

The aggregated results in Tables 4.5 and 4.6 confirm what one can expect regarding the number of explored models. In fact, it increases significantly as NP increases. Instead, the elapsed time does not necessarily increase because of a simultaneous reduction of the number of iterations required to achieve convergence. Indeed, the algorithm collects progressively more information in a single iteration as NP increases and thus it requires fewer iterations to converge to a specific model structure.
Regarding S_2, the decreasing trend of the maximum AMS value is simply due to the fact that, as NP increases, the importance of the biggest models decreases thanks to the averaging operator. This occurs also for S_4 when NP is greater than 100. Instead, for NP less than 100 the AMS value is significantly conditioned by the low percentage of exact model selections.
Regarding S_4, the interesting fact is that the wider exploration importantly affects the correctness of the selection procedure. In fact the percentage of exact model selections monotonically increases as NP increases. This is reasonable

since a wider exploration helps the algorithm to collect information from a bigger set of intermediate models. As previously said, ν affects the number of explored models as well. However, this strong relation between explored models and correctness was not so evident since the increase in the number of visited models was not significant compared to that caused by the increase in NP. One can ask if the monotonic increasing trend can lead to a 100% of correctness. The aggregated results in Table 4.7 show that there is a limit in the correctness achievable by increasing NP. Note that the number of required iterations remains almost constant leading to a continuous increase of the elapsed time because of the greater effort required to the models generation.

	$NP = 400$	$NP = 500$
Correct selection	96%	90.5%
# of Iterations	53.05	55.33
Elapsed Time [sec]	61.331	82.61
Maximum AMS	5.12	5.13
Final AMS	5.0	5.04
Explored Models	1234.7	1382.1

TABLE 4.7: C-RaMSS - Analysis on S_4 of the effect of increasingly big NP.

For the sake of completeness, consider what happens with system S_1 when moving toward bigger populations of extracted models. As said before, S_1 is a linear system and thus the chosen family is largely over-parametrized with respect to its simple linear structure. As pointed out by the analysis on the learning factor, a wider exploration causes a degradation of the algorithm performance when the procedure is tested on S_1. Indeed, one can note how the correctness is reduced as NP increases (Table 4.8).

	$NP = 25$	$NP = 50$	$NP = 75$	$NP = 100$	$NP = 200$	$NP = 300$
Correct selection	78.3%	93.3%	91.7%	90%	88.3%	83%
# of Iterations	52.13	40.15	34.0	32.75	29.57	27.19
Elapsed Time [sec]	21.04	18	14.98	13.94	15.63	19.564
Maximum AMS	6.58	5.39	4.98	4.92	4.65	4.44
Final AMS	4.25	4.06	4.08	4.07	4.08	4.06
Explored Models	252.86	329.5	374.7	448.9	562.6	590.4

TABLE 4.8: C-RaMSS - Analysis on S_1 of the effect of NP.

4.3 Comparative analysis

For each system, a comparative analysis between the C-RaMSS, the RaMSS and the FROE has been performed to assess the effectiveness of the C-RaMSS. Finally, our results on system S_2 have been compared with those of the RJMCMC method. The C-RaMSS was used the following configuration for the tuning parameters: $n_u = n_y = 4$, maximum order of non linearity equal to 3, $K = 1$, $\alpha = 0.997$. The analyzed tuning parameters are set as follows:

S_1**:** $NP = 50$ and $\nu = 0.1$,

S_4**:** $NP = 400$ and $\nu = 0.5$,

$other$**:** $NP = 200$ and $\nu = 0.1$.

4.3.1 RaMSS vs C-RaMSS

	RaMSS - $NP = 100$	RaMSS - $NP = 200$	C-RaMSS
Correct selection	90.3%	78.3%	93.3%
# of Iterations	57.2	59.51	40.15
Elapsed Time [sec]	3.27	6.25	18
Maximum AMS	5.23	4.93	5.39
Final AMS	4.08	4.05	4.06
Explored Models	864.7	1056.1	329.5

TABLE 4.9: Comparative analysis - S_1

	RaMSS - $NP = 100$	RaMSS - $NP = 200$	C-RaMSS
Correct selection	100%	100%	100%
# of Iterations	33.16	31.92	20.41
Elapsed Time [sec]	1.67	2.77	10.56
Maximum AMS	3.99	3.99	4.02
Final AMS	3.98	3.98	3.99
Explored Models	662.7	1091.3	643.0

TABLE 4.10: Comparative analysis - S_2

	RaMSS - $NP = 100$	RaMSS - $NP = 200$	C-RaMSS
Correct selection	100%	100%	100%
# of Iterations	25.53	24.18	19.6
Elapsed Time [sec]	1.24	2.07	10.5
Maximum AMS	3.99	3.99	4
Final AMS	3.98	3.98	3.98
Explored Models	483.4	728.14	584.6

TABLE 4.11: Comparative analysis - S_3

	RaMSS - $NP = 100$	RaMSS - $NP = 200$	C-RaMSS
Correct selection	78%	71.33%	96%
# of Iterations	69.4	74.36	53.05
Elapsed Time [sec]	4.57	9.14	61.33
Maximum AMS	5.53	5.43	5.12
Final AMS	5.31	5.37	5.0
Explored Models	744.6	1038.2	1234.7

TABLE 4.12: Comparative analysis - S_4

	RaMSS - $NP = 100$	RaMSS - $NP = 200$	C-RaMSS
Correct selection	100%	100%	100%
# of Iterations	46.82	45.53	26.32
Elapsed Time [sec]	2.59	4.42	16.96
Maximum AMS	5.0	4.98	5.01
Final AMS	4.98	4.98	4.98
Explored Models	951.41	1619.6	839.0

TABLE 4.13: Comparative analysis - S_5

	RaMSS - $NP = 100$	RaMSS - $NP = 200$	C-RaMSS
Correct selection	66%	82%	100%
# of Iterations	105.89	103.36	98.66
Elapsed Time [sec]	6.66	9.16	48.52
Maximum AMS	2.55	2.33	2.68
Final AMS	2.52	2.32	2.67
Explored Models	287.96	437.84	377.91

TABLE 4.14: Comparative analysis - S_6

The C-RaMSS outperforms (or at least equals) the RaMSS for all the systems. The most significant results are obtained for S_4 and S_6 for which the gain in correctness is 18% .
The second significant result concerns the minor number of iterations required to obtain the final model (Table 4.15).

	RaMSS - $NP = 100$	RaMSS - $NP = 200$
S_1	29.8	32.5
S_2	38.4	36.1
S_3	23.2	18.9
S_4	23.6	28.7
S_5	43.8	42.2
S_6	6.8	4.55

TABLE 4.15: Gain in the number of required iterations [-%].

Another interesting result concerns the number of explored models. Indeed, the C-RaMSS outperforms the RaMSS although it explores less (regarding S_4 the greater number of explored models by the C-RaMSS is due to NP which is the double that used by the RaMSS). This seems at odds with what has been previously said about the benefits of having a wider exploration. Actually, that consideration still holds. The new evidence simply says that the C-RaMSS explores better than the RaMSS the solution space thanks to the extra information contained in the correlations between regressors which constrain the exploration.
The maximum and final AMS are not affected by the new method. Regarding these statistics, the differences between the RaMSS and its variant on systems S_4 and S_6 are a direct consequence of the increase in correctness.
All these improvements come at the cost of an increasing elapsed time due to the extra computation required to generate models according to the CLF method.

4.3.2 RaMSS vs C-RaMSS with colored noise as input

Assume that the input signal $u(t)$ is a filtered white noise a.k.a. colored noise. The MSS process is generally sensitive to the excitation characteristics of the input, and particularly to slowly varying input signals (Piroddi and Spinelli, 2003). The aim of this analysis is to assess how the C-RaMSS behaves w.r.t. the RaMSS when the signal is not fully exciting. Therefore, an $AR(2)$ process is generated starting from $\xi(\cdot) \sim WGN(0, \sigma^2)$ as:

$$u(t) = a_1 u(t-1) + a_2 u(t-2) + \xi(t).$$

Using the backward shift operator z^{-1}, the $AR(2)$ process takes the form:

$$u(t) = a_1 u(t-1) + a_2 u(t-2) + \xi(t) = a_1 z^{-1} u(t) + a_2 z^{-2} u(t) + \xi(t)$$
$$\rightarrow u(t) = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2}} \xi(t) = \frac{z^2}{z^2 - a_1 z - a_2} \xi(t)$$

where the backward shift operator is defined as $z^{-1}u(t) = u(t-1)$. Therefore, an AR process is always generated through a minimum phase filter (a filter with all zeros lying at the origin) excited with a white noise.
The idea is to vary the position of the two poles in order to generate different slowly varying input signals. As the poles move from the origin to the border of the unit circle, the filtering is greater and thus slower input signals are generated (Figure 4.8).
Both the RaMSS ($NP = 200$) and the C-RaMSS ($NP = 200, \nu = 0.1$) have been tested on system S_2 excited with several slowly varying input signals, the aggregated results being reported in Tables 4.16 and 4.17. Both methods get into trouble as the input signal becomes slower and slower. This is to be expected of being PEM based methods. Indeed, parameter estimation is performed along the PEM framework, which requires exact model matching and the input signal to have sufficiently exciting properties to have correct estimates. Unreliable estimates affect the RIPs update process leading to wrong MSS. Thus, it is interesting to analyze the selected structure in terms of model size and identified terms - are they completely wrong or some of them are somehow equivalent to the correct ones? - rather than absolute correctness. Accordingly, considering the maximum and final AMS values relative to slow signals, one can note how the C-RaMSS tends to explore mostly models of size smaller or equal to the actual one while the RaMSS explores bigger ones. Consider the case $p_1 = 0.8, p_2 = 0.75$ corresponding to the signal:

$$u(t) = 1.55u(t-1) - 0.6u(t-2) + \xi(t).$$

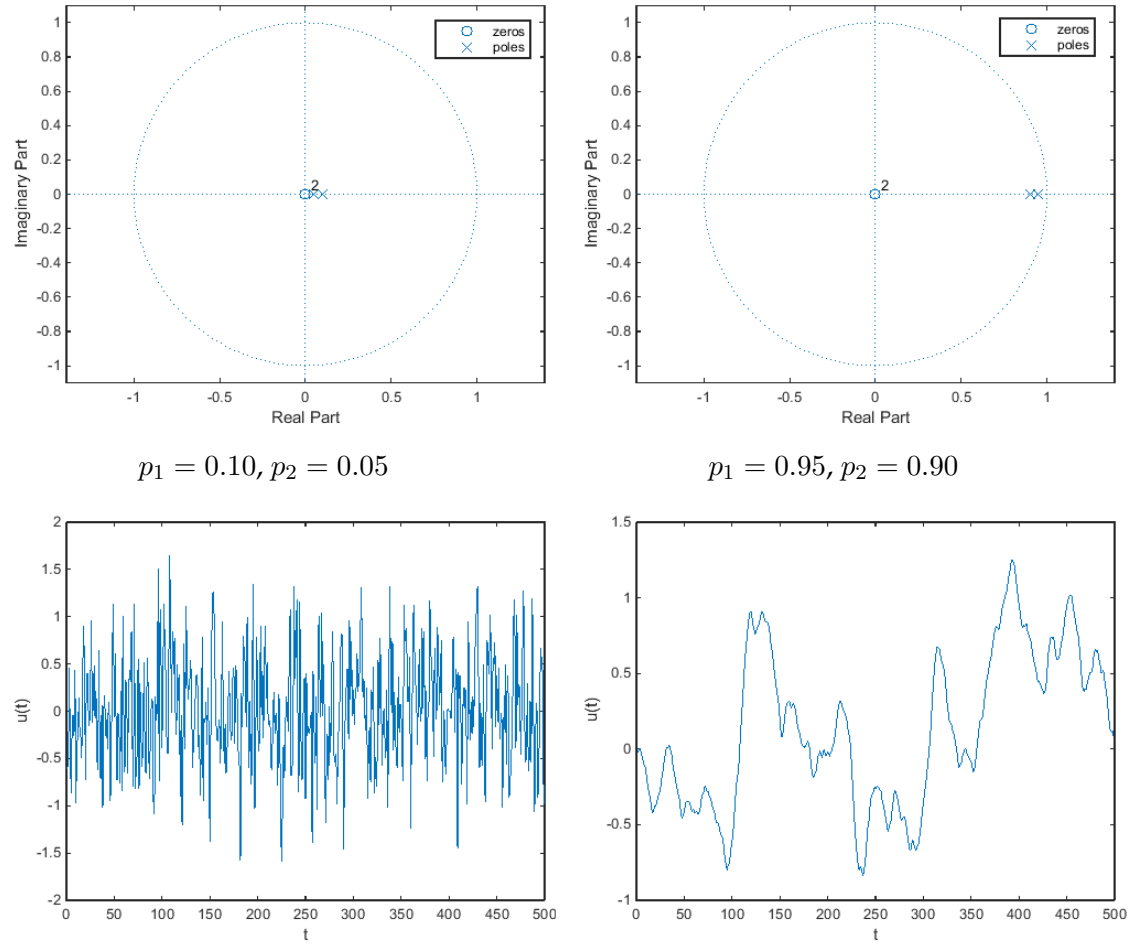
Amongst all the wrong models identified by the RaMSS, more than half have model size grater than 10, while the actual one is 4 ($\{y(t-2), y(t-1) \cdot u(t-1), y(t-2) \cdot u(t-2)^2, u(t-2)^2\}$). Instead with the C-RaMSS, the most frequently identified model structures are:

model size equal to 4: $\{y(t-2), y(t-1) \cdot u(t-1), u(t-2)^2, y(t-2) \cdot u(t-2) \cdot u(t-3)\}$,

model size equal to 5**:** $\{y(t-2), y(t-1)\cdot u(t-1), u(t-2)\cdot u(t-3), u(t-3)^2, y(t-2)\cdot u(t-2)^2\}$,

model size equal to 5**:** $\{y(t-2), y(t-1)\cdot u(t-1), u(t-2)^2, y(t-2)\cdot u(t-2)\cdot u(t-3), y(t-2)\cdot u(t-3)^2\}$.

As one can note, most of the identified terms are correct, the remaining ones belonging to the correct cluster, where a cluster is a set of regressors of the same order of nonlinearity which have been obtained combining the same type of elementary regressors, e.g. $\{y(t-1)\cdot y(t-2), y(t-1)\cdot y(t-3), \ldots, y(t-2)^2, y(t-2)y(t-3), \ldots\}$ or $\{y(t-1)\cdot u(t-1), y(t-1)\cdot u(t-2), \ldots, y(t-2)\cdot u(t-1), \ldots\}$. Sometimes, the lack of a true regressor is filled with more than one equivalent regressor.



$p_1 = 0.10, p_2 = 0.05$ $p_1 = 0.95, p_2 = 0.90$

$u(t) = 0.15u(t-1) + 0.005u(t-2) + \xi(t)$ $u(t) = 0.185u(t-1) + 0.855u(t-2) + \xi(t)$

FIGURE 4.8: Slowly varying input signals

	Correctness	Performance	# Iterations	Elapsed Time [sec]	Max AMS	Final AMS	Explored models
$p_1 = 0.1, p_2 = 0.05$	100%	0.99494	28.24	2.50	3.99	3.99	731.8
$p_1 = 0.6, p_2 = 0.55$	100%	0.99458	45.02	5.0	5.21	3.99	2641.9
$p_1 = 0.65, p_2 = 0.6$	98%	0.99415	80.98	11.50	7.13	4.12	3382.6
$p_1 = 0.7, p_2 = 0.65$	84%	0.99326	117.32	21.80	7.13	4.62	3654.7
$p_1 = 0.75, p_2 = 0.7$	62%	0.99130	168.8	34.90	8.18	6.52	3899
$p_1 = 0.8, p_2 = 0.75$	26%	0.98873	180.66	37.91	8.37	7.66	3157.8
$p_1 = 0.85, p_2 = 0.8$	8%	0.98936	121.78	20.74	5.57	5.30	2367.2
$p_1 = 0.9, p_2 = 0.85$	0%	0.99333	84.02	11.65	4.26	4.13	1849.7
$p_1 = 0.95, p_2 = 0.9$	2%	0.98617	84.68	11.09	3.80	3.64	1089.9

TABLE 4.16: RaMSS - Slowly varying input signals

	Correctness	Performance	# Iterations	Elapsed Time [sec]	Max AMS	Final AMS	Explored models
$p_1 = 0.1, p_2 = 0.05$	100%	0.99498	18.54	9.69	4.01	3.98	427.26
$p_1 = 0.6, p_2 = 0.55$	100%	0.99432	24.54	14.05	4.04	3.98	1324
$p_1 = 0.65, p_2 = 0.6$	98%	0.99438	35.56	20.83	4.38	4.03	1591.6
$p_1 = 0.7, p_2 = 0.65$	90%	0.99367	50.3	28.88	4.63	4.08	1769.5
$p_1 = 0.75, p_2 = 0.7$	60%	0.99141	64.4	37.56	4.99	4.22	2078.6
$p_1 = 0.8, p_2 = 0.75$	34%	0.98889	60.9	34.47	4.44	4.33	1505.6
$p_1 = 0.85, p_2 = 0.8$	10%	0.99111	65.22	37.20	4.26	4.17	1577.7
$p_1 = 0.9, p_2 = 0.85$	8%	0.99344	68.18	38.53	4.22	4.06	1431.3
$p_1 = 0.95, p_2 = 0.9$	0%	0.98366	53.7	29.83	3.52	3.46	869.3

TABLE 4.17: C-RaMSS - Slowly varying input signals

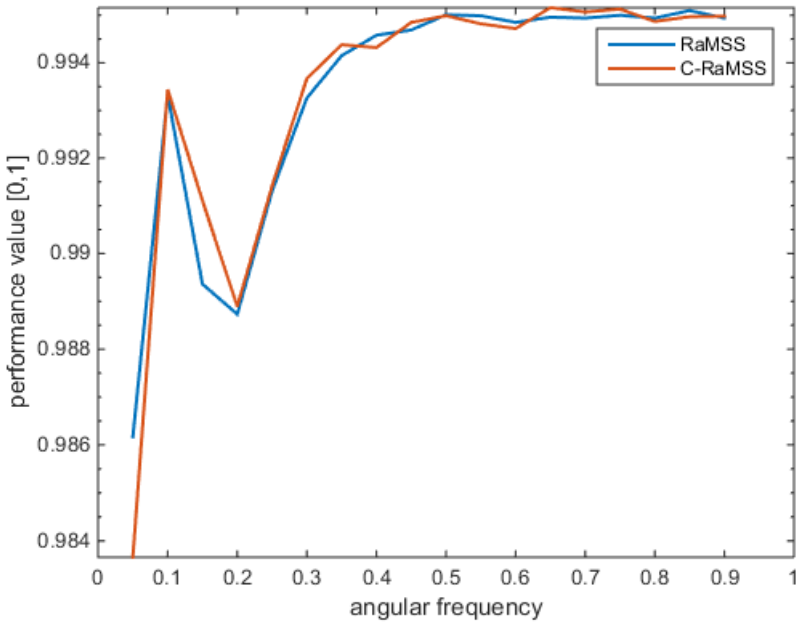


FIGURE 4.9: RaMSS vs C-RaMSS - Performances with slowly varying inputs

In Figure 4.9 the average model performances for both the methods are compared as the cutoff frequency increases (as the input signal becomes more exciting). As one can note both the methods perform well also with slowly varying signals, in fact all the performance values are greater than 0.98 on a range of $[0, 1]$. This leads to an important consideration: though the C-RaMSS tends to identify more compact modesl than its rival, the performances of the extracted models are very similar.
Therefore, one can conclude that, when the input signal is not fully exciting, the second order information of relations between regressors, used by the C-RaMSS, constrains the exploration of the solution space leading to more compact models than the RaMSS without loss of efficiency. This is very important, since compact models are typically more robust and more easily interpretable and thus they are useful for understanding the behavior of the underlying system, which is the primary goal of an identification system procedure.

4.3.3 FROE vs C-RaMSS

The FROE method has been tested on all the systems. Only the model structures of systems S_3 and S_6 are correctly identified. Two regressors out of four are correctly selected for system S_1 while the remaining ones are lost due to the wrong selection at the third iteration which leads to a misleading MSS. For system S_2 a constant term is added to the model. The algorithm fails also for system S_4 for which the term $y(t-2)$ is wrongly included instead of the term $y(t-2)^2$. Finally, for system S_5 a constant term and the autoregressive term $y(t-1)$ are included, while the cubic term $y(t-1)^3$ is left out.

Iteration	Regressor	Correct
1	$y(t-1)$	yes
2	$u(t-1)$	yes
3	$u(t-3)$	no
4	$y(t-3)$	no

TABLE 4.18: FROE results - S_1

Iteration	Regressor	Correct
1	$y(t-2)$	yes
2	1	no
3	$u(t-2)^2$	yes
4	$y(t-1)u(t-1)$	yes
5	$y(t-2)u(t-2)^2$	yes

TABLE 4.19: FROE results - S_2

Iteration	Regressor	Correct
1	$y(t-1)$	yes
2	$u(t-1)$	yes
3	$u(t-1)^3$	yes
4	$u(t-1)^2$	yes

TABLE 4.20: FROE results - S_3

Iteration	Regressor	Correct
1	1	yes
2	$u(t-1)^2$	yes
3	$u(t-2)$	yes
4	$y(t-2)$	no
5	$y(t-1)$	yes

TABLE 4.21: FROE results - S_4

Iteration	Regressor	Correct
1	$y(t-2)$	yes
2	1	no
3	$y(t-1)u(t-1)$	yes
4	$u(t-2)^2$	yes
5	$y(t-2)u(t-2)^2$	yes
6	$y(t-1)$	no

TABLE 4.22: FROE results - S_5

Iteration	Regressor	Correct
1	$y(t-2)$	yes
2	$u(t-1)$	yes
3	$y(t-2)u(t-1)$	yes

TABLE 4.23: FROE results - S_6

4.3.4 RJMCMC vs C-RaMSS

The analyzed system S_2 has been taken from (Baldacchino, Anderson, and Kadirkamanathan, 2013), so we can compare directly our results with those reported by the author in the paper. The RJMCMC was run 10 times on the same input-output data set. The maximum a posteriori model was correct on 7 occasions and the maximum a posteriori model from the aggregated results corresponded to the true model (Table 4.24). Instead, the C-RaMSS (and the RaMSS) is always able to identify the correct model for this system. The C-RaMSS outperforms the RJMCMC also in terms of computational efficiency: the method proposed in (Baldacchino, Anderson, and Kadirkamanathan, 2013) requires a huge amount of iterations. In fact, it was run with $30,000$ iterations and a burn in period of $5,000$ iterations.

Model	MAP model					Prob	Correct
	Term1	Term2	Term3	Term4	Term5		
1	$y(t-1)u(t-1)$	$y(t-2)$	$y(t-2)u(t-2)^2$	$u(t-2)^2$	–	0.999	yes
2	$y(t-1)u(t-1)$	$y(t-2)^3$	$y(t-4)u(t-2)^2$	$u(t-2)^2$	$y(t-3)u(t-3)$	0.415	no
3	$y(t-1)u(t-1)$	$y(t-2)$	$y(t-2)u(t-2)^2$	$u(t-2)^2$	–	0.946	yes
4	$y(t-1)u(t-1)$	$y(t-2)$	$y(t-2)u(t-2)^2$	$u(t-2)^2$	–	0.775	yes
5	$y(t-1)u(t-1)$	$y(t-4)$	$y(t-2)u(t-2)^2$	$u(t-2)^2$	$u(t-4)^2$	0.796	no
6	$y(t-1)u(t-1)$	$y(t-2)$	$y(t-2)u(t-2)^2$	$u(t-2)^2$	–	0.961	yes
7	$y(t-1)u(t-1)$	$y(t-2)$	$y(t-2)u(t-2)^2$	$u(t-2)^2$	–	0.995	yes
8	$y(t-1)u(t-1)$	$y(t-2)$	$y(t-2)u(t-2)^2$	$u(t-2)^2$	–	0.870	yes
9	$y(t-1)u(t-1)$	$y(t-2)$	$y(t-2)u(t-2)^2$	$u(t-2)^2$	–	0.997	yes
10	$y(t-3)u(t-1)$	$y(t-2)u(t-4)^2$	$y(t-2)u(t-2)^2$	$u(t-2)^2$	$u(t-1)u(t-3)^2$	0.76	no
Agg.	$y(t-1)u(t-1)$	$y(t-2)$	$y(t-2)u(t-2)^2$	$u(t-2)^2$	–	0.673	yes

TABLE 4.24: RJMCM results - S_2

Chapter 5

Conclusions and future work

A variant of the randomized algorithm RaMSS denoted C-RaMSS is proposed for nonlinear system identification based on the NARX model family. This field has been dominated by methods based on iterative model construction techniques which progressively select the regressors to include into the model from a candidate set, according to some significance criterion. Among all, the FROE is a milestone and it is often used as a reference method for the validation of the new ones. These methods often yield unsatisfactory models due to the difficulty to correctly assess the significance of each regressor.
The RaMSS recasts the MSS problem in a probabilistic framework where a model distribution is defined in terms of the probabilities of each regressor to be present in a sampled model. Then, these regressor probabilities are updated according to the performance of the entire population of extracted models. The algorithm proceeds untill it locally converges to a limit model distribution.
The novel variant introduces a multivariate Bernoulli distribution to formalize the dependences among regressors, as opposed to the RaMSS which is based on the assumption of independence between them. More specifically, the CLF tool is used to simulate these correlated binary variables with specified mean vector and correlation matrix without fully specifying the joint distribution. In so doing, the probability of extracting each regressor depends on the simulation of the previously extracted ones. The RIPs are updated as in the RaMSS, and a new update rule is introduced for the correlation matrix. It takes into account the cosine similarity measures calculated on a VSM where each regressor is defined in terms of performances of those models in which it appears. Therefore, two regressors are similar if they appear jointly in good models and do not appear concurrently in bad models.
The new algorithm is validated over different systems taken from the literature. The obtained results show its advantages in terms of robustness and reliability with respect to the RaMSS at the expense of the computational efficiency, due to the new heavier simulation mechanism. Nonetheless, this loss in efficiency is acceptable since the algorithm works on a batch of collected system observations, rather than on on-line data. Furthermore, the new algorithm tends to identify more compact models than the RaMSS, when the exciting signal slowly varies. This is an appreciable result since compact models are more robust and easier to interpret. The C-RaMSS outperforms classical methods as the FROE and also competitor probabilistic methods as the RJMCMC.

An interesting continuation on this line-up would be to test the algorithm over NARMAX models, thus including also the identification of the noise process.
Furthermore, it would be useful to better investigate the impact of slowly varying input signals, in order to improve the algorithm robustness also in such situation.
Finally, it would be nice to apply the proposed method to the general problem of features selection that has become an essential task in order to apply data mining algorithms effectively in real-world scenarios. A challenge in this area is the feature selection with sparse data matrices. For example, in a business context, many individual transactions are recorded in the application such as market basket analysis, direct-mail marketing, insurance, and health care. These types of data collections have a sparse matrix with a large number of attributes. This is a promising scenario for the C-RaMSS. Therefore, the idea is to test the C-RaMSS on a real data-set and compare its results with those of well known feature subset selection techniques, such as Stepwise Regression, LASSO, Decision Trees, Regularized Trees. The goal is to understand if the new algorithm can be considered an alternative to existing methods in this area and if not, try to adapt the algorithm to this scenario.

Appendix A

Appendix

A.1 Orthogonal Least Squares

Recall that

$$S : y(t) = \varphi^T(t)\theta + e(t) \qquad e(t) \sim WN(0, \sigma^2) \tag{A.1}$$

Define

$$Y = \begin{bmatrix} y(1) \\ \vdots \\ y(N) \end{bmatrix} \qquad \Psi = \begin{bmatrix} \varphi_1(1) & \dots & \varphi_m(1) \\ \vdots & \dots & \vdots \\ \varphi_1(N) & \dots & \varphi_m(N) \end{bmatrix} \qquad \Theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_m \end{bmatrix} \qquad E = \begin{bmatrix} e(1) \\ \vdots \\ e(N) \end{bmatrix} \tag{A.2}$$

where N is the number of observations and m the number of basis functions. Then, S can be expressed in matrix form as

$$Y = \Psi\Theta + E \tag{A.3}$$

Assume that Ψ has full column rank. Then $\Psi^T\Psi \succ 0$, and can be decomposed as:

$$\Psi^T\Psi = A^T D A \tag{A.4}$$

where A is an upper triangular matrix with unitary diagonal, and D is a diagonal matrix with positive diagonal elements.
Using A one can rewrite the original linear regression in a different form:

$$Y = \Psi\Theta + E = \Psi(A^{-1}A)\Theta + E = (\Psi A^{-1})(A\Theta) + E = Wg + E \tag{A.5}$$

where

- the columns of $W = \Psi A^{-1}$ are the *auxiliary regressors*
- $g = A\Theta$ is the vector of the *auxiliary parameters*

This form is convenient, because it turns out that matrix W is *orthogonal*, indeed

$$W^TW = (\Psi A^{-1})^T(\Psi A^{-1}) = (A^{-1})^T\Psi^T\Psi A^{-1} = (A^T)^{-1}A^T D A A^{-1} = D \tag{A.6}$$

which implies that $\sum_{t=1}^{N} w_i(t)w_j(t) = 0$, $i \neq j$, i.e. the auxiliary regressors are *mutually orthogonal*.

The auxiliary parameter vector can be estimated with LS on the linear regression $Y = \Psi\Theta + E$, which yields:

$$\hat{g} = (W^T W)^{-1} W^T Y = D^{-1} W^T Y \tag{A.7}$$

$$\begin{bmatrix} \hat{g}_1 \\ \vdots \\ \hat{g}_m \end{bmatrix} = \begin{bmatrix} d_1^{-1} & 0 & \dots & 0 \\ 0 & d_2^{-1} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & d_m^{-1} \end{bmatrix} \begin{bmatrix} w_1(1) & \dots & w_1(N) \\ \vdots & \dots & \vdots \\ w_m(1) & \dots & w_m(N) \end{bmatrix} \begin{bmatrix} y(1) \\ \vdots \\ y(N) \end{bmatrix} \tag{A.8}$$

Thanks to orthogonality of the auxiliary regressor matrix, we can compute each auxiliary parameter *independently* of the others:

$$\hat{g}_i = \frac{\sum_{t=1}^{N} w_i(t) y(t)}{\sum_{t=1}^{N} w_i(t)^2} \tag{A.9}$$

Both original regressors and parameters can be easily calculated starting from the auxiliary form:

$$W = \Psi A^{-1} \rightarrow \Psi = WA = WA + (W - W) \rightarrow W = \Psi - W(A - I) \tag{A.10}$$

$$g = A\Theta = A\Theta + (\Theta - \Theta) \rightarrow \Theta = g - (A - I)\Theta \tag{A.11}$$

The auxiliary regressors can be calculated recursively as follows:

$$w_1(t) = \Psi_1(t),\ w_2(t) = \Psi_2(t) - a_{12} w_1(t),\ \dots,\ w_k(t) = \Psi_k(t) - \sum_{i=1}^{k-1} a_{ik} w_i(t),\ k = 2, \dots, m. \tag{A.12}$$

As well, the auxiliary parameters can be calculated as follows:

$$\hat{\theta}_m = \hat{g}_m,\ \hat{\theta}_{m-1} = \hat{g}_{m-1} - a_{m-1,m}\hat{\theta}_m, \dots, \hat{\theta}_i = \hat{g}_i - \sum_{k=i+1}^{m} a_{ik}\hat{\theta}_k,\ i = m-1, \dots, 1. \tag{A.13}$$

A.2 Sampling methods

As described in (Devroye, 1986) several techniques have been introduced to solve the sampling and integration problems.

A.2.1 Inverse Sampling method

The easiest way to generate random values starting from a distribution, whose inverse is known and computable, is the *inverse sampling method* which is based on the following theorem:

Theorem 3. Let F be a continuous distribution function on $\mathbb{R}$ with inverse F^{-1} defined by

$$F^{-1}(u) = Inf\{x : F(x) = u \ , 0 < u < 1\}. \tag{A.14}$$

If U is a uniform $[0, 1]$ random variable, then $F^{-1}(U)$ has distribution function F. Also, if X has distribution function F, then $F(X)$ is uniformly distributed on $[0, 1]$.

Thus, the inverse sampling procedure is the following:

Algorithm A.1 Inverse Sampling

1: $n = 1$; ▷ number of samples
2: **repeat**
3: $u \sim Uniform(0, 1)$;
4: $x = F^{-1}(u)$;
5: $n = n + 1$;
6: **until** $n \leq threshold$

A.2.2 Rejection Sampling method

Often, the inverse sampling method is not applicable since it is difficult to compute the distribution function or its inverse. An alternative is the *rejection sampling method*, providing that it is possible evaluate the density function $p(x)$ for each possible x. The idea is to define a proposal density $q(x)$ from which you can sample easily and, such that, $c \cdot q(x) \geq p(x)$ for some $c > 1$ and for each x. Then, the rejection sampling method works as follows:

1. sample a point (an x-position) from the proposal density;
2. draw a vertical line at this x-position, up to the curve of the proposal density;
3. sample uniformly along this line. If the sampled value is greater than the value of the desired distribution at this vertical line, return to step 1.

Algorithmically,

Algorithm A.2 Rejection Sampling

1: $n = 1$; ▷ number of samples
2: **repeat**
3: $x \sim q(x)$;
4: $u \sim Uniform(0, 1)$;
5: **if** $u < \frac{p(x)}{c \cdot q(x)}$ **then**
6: *accept* x;
7: $n = n + 1$;
8: **else**
9: *reject* x;
10: **end if**
11: **until** $n \leq threshold$

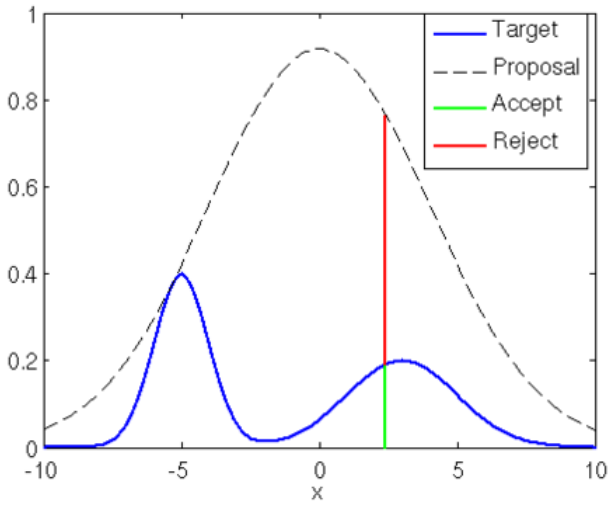


FIGURE A.1: Graphical illustration of rejection sampling method.

A.2.3 Importance Sampling method

Unlike inverse and rejection sampling methods, the importance sampling is not a method to sample from $p(x)$. Rather, it is a method to compute the Monte Carlo estimator of the expectation of a function $f(x)$

$$\mu = E_p[f(x)] = \int f(x)p(x)dx \simeq \frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{A.15}$$

Therefore, the goal is to numerically compute $I(f) = \int f(x)p(x)dx$. As for the rejection sampling method, if there is a density $q(x)$ which is easy to sample from, one can sample $x^{(i)} \sim q(x)$. Define the importance weight as:

$$w(x^{(i)}) = \frac{p(x^{(i)})}{q(x^{(i)})}. \tag{A.16}$$

These weights account for the bias due to the sampling from the proposal density rather than the desired one. Consider the *weighted* Monte Carlo sum:

$$\begin{aligned}\frac{1}{N}\sum_{i=1}^{N} f(x^{(i)}) \cdot w(x^{(i)}) &= \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)}) \cdot \frac{p(x^{(i)})}{q(x^{(i)})} \\ \xrightarrow{a.s.} \int (f(x) \cdot \frac{p(x)}{q(x)}) \cdot q(x)dx \quad &\textit{(Law of Large Numbers)} \\ &= \int f(x)p(x)dx\end{aligned}$$

In principle, one can sample from any distribution $q(x)$. In practice, one would like to choose $q(x)$ as close as possible to $|f(x)| \cdot w(x)$ to reduce the variance of the Monte Carlo estimator. To better understand the importance of choosing a good proposal density, consider this example.

Example 7. One wants a Monte Carlo approximation to $\int_0^1 g(x)dx$ for the $g(x)$ shown in the figure below. Note that $g(x) = 0$ for $x < 0$ and $x > 1$.

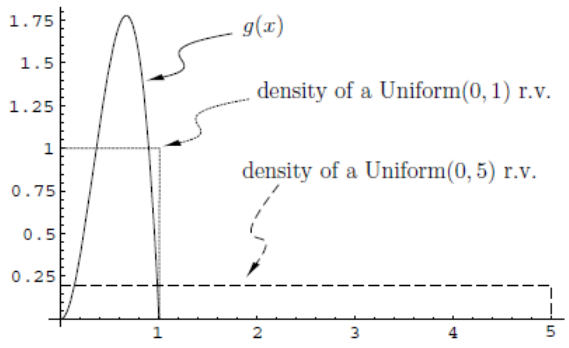


FIGURE A.2: Importance sampling method example.

If one has $U \sim Uniform(0, 1)$, then one can cast the integral as the expectation with respect to U:

$$\int_0^1 g(x)dx = E[g(U)], \tag{A.17}$$

so one may approximate it by a Monte Carlo estimator. This would work reasonably well. The figure, however, suggests another possibility. One could use $W \sim Uniform(0, 5)$ giving:

$$\int_0^1 g(x)dx = 5 \cdot E[g(W)]. \tag{A.18}$$

However this possibility doe not make sense because, on average, 80% of the realized w_i would tell you nothing substantial about the integral of $g(x)$ since $g(x) = 0$ for $1 < x < 5$. This would be *barely relevant sampling*.

A.3 Monte Carlo techniques

The distribution of the parameter vector θ conditional on the data X, called *posterior* distribution, can be computed by Bayes theorem as:

$$P(\theta|X) = \frac{P(X|\theta) \cdot P(\theta)}{P(x)} = \frac{P(X|\theta) \cdot P(\theta)}{\int P(X|\theta) \cdot P(\theta) d\theta} \tag{A.19}$$

We know $P(\theta|X) \propto P(X|\theta) \cdot P(\theta)$, but we cannot easily evaluate the normalization factor $\int P(X|\theta) \cdot P(\theta) d\theta$ due to the integral.
Integration can be solved numerically by means of Monte Carlo methods which are used to approximate an expectation by the sample mean of a function of simulated random variables. Indeed,

$$P(X) = \int P(X|\theta) \cdot P(\theta) d\theta \tag{A.20}$$

is a marginal distribution and it can be always written as an expectation:

$$P(X) = \int P(X|\theta) \cdot P(\theta) d\theta = E_\theta[P(X|\theta)]. \tag{A.21}$$

Let $f(\theta) = P(X|\theta)$. Then, Monte Carlo integration makes the approximation:

$$E[f(\theta)] \simeq \frac{1}{n} \sum_{t=1}^{n} f(\theta^t) \tag{A.22}$$

where θ^t are random draws from $P(\theta)$. When the samples are independent, the law of large numbers ensures that the approximation can be made as accurate as desired by increasing the number of samples n. In general, sampling directly from $P(\theta)$ is hard. One way of doing this is through a Markov chain having $P(\theta)$ as its stationary distribution. This combined technique is called *Markov Chain Monte Carlo*. A Markov chain is a random process that undergoes transitions from one state to another on a state space. It must satisfy the *Markov property*: the probability distribution of the next state $P(\theta^{t+1}|\theta^t)$ depends only on the current state and not on the sequence of events that preceded it. So, we can generate a sequence of samples simply by considering the visited states, starting from an initial state θ^0. As t increases, thanks to the Markov property, the chain will forget its initial state and $P(\cdot|\theta^0)$ will converge to the stationary distribution. Thus as t increases, the sampled points will look like samples taken from $P(\theta)$ and can be used to solve Monte Carlo integration. The main drawback is that the chain requires a long *burn-in* period to converge to its stationary distribution.
The *Metropolis-Hasting* (MH) algorithm allows to construct a Markov chain such that its stationary distribution is exactly the desired distribution, i.e. in our case $P(\theta)$. For the MH algorithm, the next state θ^{t+1} is chosen by first sampling a candidate point y from a proposal distribution $Q(\cdot|\theta)$ which is easier to be sampled w.r.t. $P(\theta)$. Then with some probability, the candidate y is either accepted

- in which case the candidate value is used as next state in the next iteration - or rejected - the current state is reused in the next iteration. The acceptance probability is:

$$\alpha(\theta^t, y) = min(1, \frac{P(y)}{P(\theta^t)} \frac{Q(\theta^t|y)}{Q(y|\theta^t)}) \tag{A.23}$$

Thus, the entire MH algorithm is:

Algorithm A.3 Metropolis-Hasting

Initialize θ^0;
$t = 0$;
loop
    Sample a point y *from* $Q(\cdot|\theta^t)$;
    Sample a $Uniform(0, 1)$ *random variable* U
    if $U \leq \alpha(\theta^t, y)$ **then**
        $\theta^{t+1} = y$;
    else
        $\theta^{t+1} = \theta^t$;
    end if
    $t = t + 1$;
end loop

A.4 RaMSS - RIPs update rule

If A is a Bernoulli random variable and B is a random variable that depends on A, it holds that

$$E_P[B] = P(A = 1)E_P[B|A = 1] + P(A = 0)E_P[B|A = 0] \tag{A.24}$$

According to Equation A.24, we can compute $E_P[J]$ as:

$$E_P[J] = \tilde{\mu_j}E_P[J|\tilde{\rho_j} = 1] + (1 - \tilde{\mu_j})E_P[J|\tilde{\rho_j} = 0] \tag{A.25}$$

where:

- P is the probability distribution over all possible models obtained as combination of the candidate regressors;
- J is the performance index defined as $J = e^{-K \cdot MSPE}$;
- $\rho_j \sim Be(\mu_j)$ is a regressor;
- $\tilde{\rho_j} \sim Be(\tilde{\mu_j})$ is a candidate regressor, i.e. a non-redundant term.

Deriving Equation A.25 w.r.t. $\tilde{\mu_j}$, one obtains

$$\frac{\partial E_P[J]}{\partial \tilde{\mu}_j} = E_P[J|\tilde{\rho_j} = 1] - E_P[J|\tilde{\rho_j} = 0] \tag{A.26}$$

We can then use the Equation A.26 to update the parameter μ_j

$$\mu_j(i + 1) = \mu_j(i) + \gamma \cdot \frac{\partial E_P[J]}{\partial \tilde{\mu}_j} \tag{A.27}$$

A.5 Pseudoinverse

The pseudoinverse $A^\dagger$ of a matrix A is a generalization of the inverse matrix. The most widely known type of pseudoinverse matrix is the Moore–Penrose one. A common use of this matrix is to compute a LS solution to a system of linear equations that lacks a unique solution (overdetermined system). Another use is to find the minimum norm solution to a system of linear equations with multiple solutions (undetermined system). For a real matrix A, it is defined as:

overdetermined system: $A^\dagger = (A^T A)^{-1} A^T$

underdetermined system: $A^\dagger = A^T (A A^T)^{-1}$

The pseudoinverse matrix can be computed using the SVD of A.

Theorem 4. Given $A \in \mathbb{R}^{m \times n}$ of rank r, there exist

- $V \in R^{mxr}$ with $V^T V = I$,
- $U \in R^{nxr}$ with $U^T U = I$,
- $\Sigma = diag(\sigma_1, \cdots, \sigma_r)$ with $\sigma_1 \geq \cdots \geq \sigma_r > 0$

such that

$$A = V \cdot \Sigma \cdot U^T. \tag{A.28}$$

This formula is called Singular Value Decomposition of A and the σ_i are called singular values of A.

If A is a skinny full rank matrix (number of columns less than number of rows), which corresponds in a linear matrix equation $Ax = b$ to an overdetermined system, then

$$A^\dagger = (A^T A)^{-1} A^T = (U \Sigma V^T V \Sigma U^T)^{-1} U \Sigma V^T = (U \Sigma^2 U^T)^{-1} U \Sigma V^T \tag{A.29}$$

Since U is a square orthogonal matrix $U^T = U^{-1}$ then

$$(U \Sigma^2 U^T)^{-1} U \Sigma V^T = U \Sigma^{-2} U^T U \Sigma V^T = U \Sigma^{-1} V^T \tag{A.30}$$

If A is a fat full rank matrix (number of columns grater than number of rows), which corresponds to an underdetermined system, then

$$A^\dagger = A^T (A A^T)^{-1} = U \Sigma V^T (V \Sigma U^T U \Sigma V^T)^{-1} = U \Sigma V^T (V \Sigma^2 V^T)^{-1} \tag{A.31}$$

Since V is a square orthogonal matrix $V^T = V^{-1}$, one has that

$$U \Sigma V^T (V \Sigma^2 V^T)^{-1} = U \Sigma V^T V \Sigma^{-2} V^T = U \Sigma^{-1} V^T \tag{A.32}$$

A recursive computation method can be applied as well. Le $A^\dagger$ be the pseudoinverse of A and define the matrix:

$$M = \begin{bmatrix} A & c \\ b^* & d \end{bmatrix},$$

where b and c are columns and d is a scalar. According to the *full* SVD formula, for any complex $[m \times n]$ matrix A, of rank r, there exist unitary matrices U and V such that:

$$V^* \cdot A \cdot U = \left[\begin{array}{c|c} \Sigma & 0 \\ \hline 0 & 0 \end{array}\right], \tag{A.33}$$

where V^* is the complex conjugate of V, Σ is defined as in Equation A.28. If we partition U and V as

$$U = [U_1, U_2], \qquad V = [V_1, V_2],$$

with $U_1 = [u_1, \cdots, u_r]$ and $V_1 = [v_1, \cdots, v_r]$, then it can be proved that the problem of finding $M^\dagger$ reduces to finding the Moore-Penrose inverse of the following partitioned block matrix:

$$M = \left[\begin{array}{cc|c} \Sigma & 0 & q_1 \\ 0 & 0 & q_2 \\ \hline p_1^* & p_2^* & d \end{array}\right],$$

where:

- $p_1^* = b^* U_1$,
- $p_2^* = b^* U_2$,
- $q_1 = V_1^* c$,
- $q_2 = V_2^* c$.

There are 5 cases that can occur, namely:

- $rank(M) = r + 2 \Leftrightarrow p_2 \neq 0 \wedge q_2 \neq 0$,
- $rank(M) = r + 1 \Leftrightarrow \begin{cases} p_2 = 0 \wedge q_2 \neq 0 \\ p_2 \neq 0 \wedge q_2 = 0 \\ p_2 = 0 \wedge q_2 = 0 \wedge z \neq 0 \end{cases}$,
- $rank(M) = r \Leftrightarrow p_2 = 0 \wedge q_2 = 0 \wedge z = 0$,

where $z = d - p_1^* \Sigma^{-1} q_1 = d - b^* A^\dagger$. Since we are working with the correlation/covariance matrix which is symmetric and with unit diagonal, $rank(M) = r + 1$ and $U = V \Rightarrow p_2 = q_2$, and the only useful case is that which is referred as *case 1* in (Hartwig, 1976), i.e $rank(M) = r + 1 \Leftrightarrow p_2 = 0 \wedge q_2 = 0 \wedge z \neq 0$. For this case, since $m = n = r$, the problem reduces to finding the Moore-Penrose inverse of the following partitioned block matrix:

$$M = \left[\begin{array}{cc} \Sigma & q_1 \\ p_1^* & d \end{array}\right],$$

Accordingly, the Moore-Penrose inverse is computed as:

$$M^{\dagger} = \begin{bmatrix} A^{\dagger} + \frac{k \cdot h^{*}}{z} & \frac{-k}{z} \\ \\ \frac{-h^{*}}{z} & \frac{1}{z} \end{bmatrix}, \tag{A.34}$$

where:

- $k = A^{\dagger}c$,
- $h = A^{*^{\dagger}}b$,

In this way we are able to compute the pseudoinverse of matrix G_i based on the pseudoinverse of G_{i-1} and the added row and column, without needing to compute the SVD of G_i. The only needed SVD is that of matrix G_1 which is, actually, a scalar value. A comparison, in terms of elapsed time required to compute all the b_i coefficients has been done between the illustrated iterative method and the MATLAB pinv function which computes the pseudoinverse by means of SVD. Considering different matrix dimensions n, a random $[n \times n]$ sparse symmetric matrix with unit diagonal has been generated and the b_i coefficients have been computed according to Equation 3.4. The aggregated results on 500 matrices for each n, are reported in Figure A.3. Apparently, as the matrix dimension increases, the iterative method slightly outperforms the SVD based method.

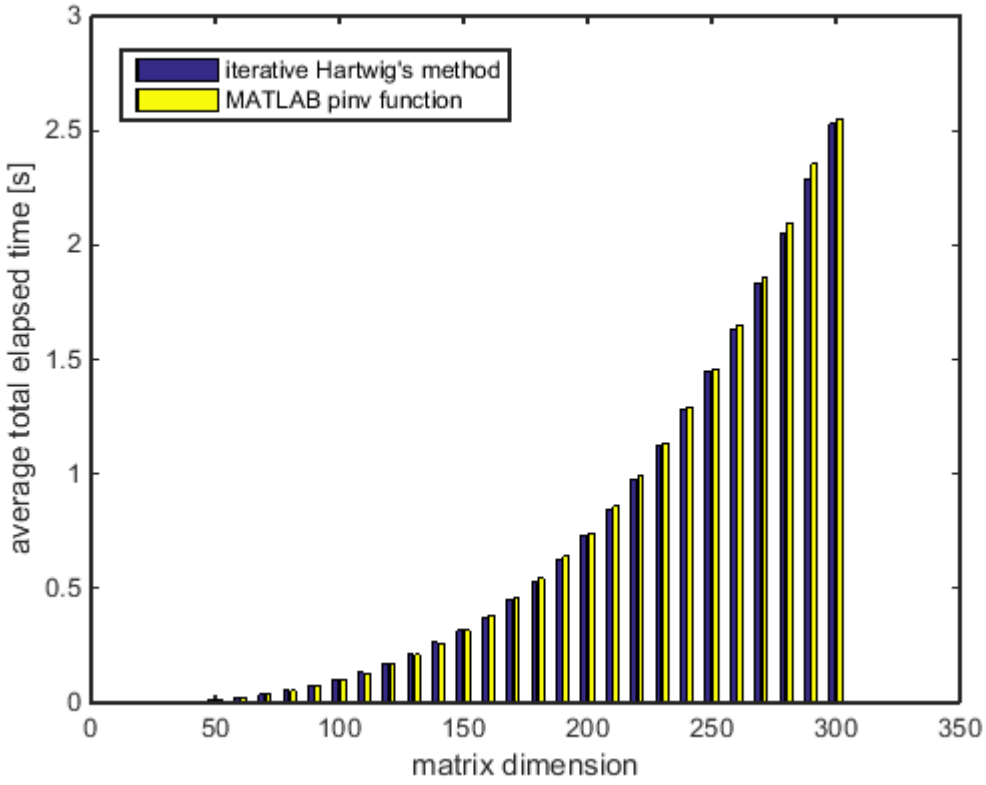


FIGURE A.3: Comparison between methods of computation of the pseudoinverse.

Bibliography

Aguirre, L. A., B. H. G. Barbosa, and A. P. Braga. In: *Mechanical Systems and Signal Processing, Month = , Number = 24, Numpages = , Pages = 2855-2867, Title =Prediction and simulation errors in parameter estimation for nonlinear systems, Volume = , Url = , Year = 2010*.

Aguirre, L. A. and S. A. Billings (1995). "Dynamical effects of overparametrization in nonlinear models". In: *Physica* 80.4, pp. 26–40.

Baldacchino, T., S. R. Anderson, and V. Kadirkamanathan (2013). "Computational system identification for Bayesian NARMAX modeling". In: *Automatica* 49, pp. 2641–2651.

Berry, M. W., Z. Drmac, and E. R. Jessup (2006). "Matrices, Vector Spaces, and Information Retrieval". In: *SIAM Review* 41.2, pp. 335–362.

Billings, S. A., S. Chen, and M. Korenberg (1989). "Identification of MIMO non-linear systems using a forward-regression orthogonal estimator". In: *Internal Journal of Control* 49.6, pp. 2157–2189.

Bonin, M., V. Seghezza, and L. Piroddi (2010). "NARX model selection based on simulation error minimisation and LASSO". In: *IET Control Theory & Applications* 4.7, pp. 1157–1168.

Devroye, L., ed. (1986). *Non-uniform Random Variate Generation*. New York: Springer-Verlag.

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

German, S., E. Bienenstock, and R. Doursat (1992). "Neural Networks and the Bias/Variance Dilemma". In: *Neural Computation* 4, pp. 1–58.

Hartwig, R. E. (1976). "Singular Value Decomposition and the Moore-Penrose Inverse of Bordered Matrices". In: *SIAM Journal on Applied Mathematics* 31.1, pp. 31–41.

Higham, J. (2002). "Computing the nearest correlation matrix - a problem from finance". In: *IMA Journal of Numerical Analysis* 22.3, pp. 329–343.

Hong, X. et al. (2008). "Model selection approaches for non-linear system identification: a review". In: *International Journal of Systems Science* 39.10, pp. 925–946.

Leontaritis, I. J. and S. A. Billings (1985a). "Input-output parametric models for non-linear systems part I: deterministic non-linear system". In: *International Journal of Control* 48.1, pp. 303–328.

— (1985b). "Input-output parametric models for non-linear systems part II: stochastic non-linear system". In: *International Journal of Control* 48.1, pp. 303–328.

Ljung, L., ed. (1987). *System Identification: Theory for the User*. London: Prentice-Hall.

Mao, K. Z. and S. A. Billings (1997). "Algorithms for minimal model structure detection in nonlinear dynamic system identification". In: *International Journal of Control* 68.2, pp. 311–330.

— (1999). "Variable selction in non-linear systems modelling". In: *Mechanical Systems and Signal Processing* 13.2, pp. 351–366.

Palumbo, P. and L. Piroddi (2000). "Seismic behaviour of bruttress dams: nonlinear modeling of a damaged buttress based on ARX/NARX models". In: *Journal of Sound and Vibration* 239, pp. 405–422.

Piroddi, L. and W. Spinelli (2003). "An identification algorithm for polynomial NARX models based on simulation error minimization". In: *International Journal of Control* 76.17, pp. 1767–1781.

Qaqish, B. F. (2003). "A family of multivariate binary distributions for simulating correlated binary variables with specified marginal means and correlations". In: *Biometrika* 90.2, pp. 455–463.

Rodriguez-Vazquez, K., C. M. Fonseca, and P. J. Fleming (2004). "Identifying the structure of nonlinear dynamic systems using multiobjective genetic programming". In: *IEEE Transactions on Systems, Man & Cybernetics, Part A (Systems & Humans)* 34.4, pp. 1157–1168.

Sjöberg, J. et al. (1995). "Nonlinear black-box modeling in system identification: a unified overview". In: *Automatica* 31.12, pp. 1691–1724.

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

Wei, H. L. and S. A. Billings (2008). "Model Structure Selection Using an Integrated Forward Orthogonal Search Algorithm Assisted by Squared Correlation and Mutual Information". In: *International Journal of Modeling, Identification and Control* 3.4, pp. 341–356.