UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA

MICHAEL SANTOS DUARTE

# ON CORRENTROPY-BASED MACHINE LEARNING MODELS FOR NONLINEAR SIGNAL PROCESSING: ADDRESSING SPARSITY, RECURSIVE ESTIMATION AND OUTLIER-ROBUSTNESS

FORTALEZA
2021

MICHAEL SANTOS DUARTE

# ON CORRENTROPY-BASED MACHINE LEARNING MODELS FOR NONLINEAR SIGNAL PROCESSING: ADDRESSING SPARSITY, RECURSIVE ESTIMATION AND OUTLIER-ROBUSTNESS

Tese apresentada ao Programa de Pós-Graduação em Engenharia de Teleinformática do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Engenharia de Teleinformática. Área de Concentração: Sinais e Sistemas

Orientador: Prof. Dr. Guilherme de Alencar Barreto

FORTALEZA

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

D873o Duarte, Michael Santos.
On Correntropy-Based Machine Learning Models for Nonlinear Signal Processing : Addressing Sparsity, Recursive Estimation and Outlier-Robustness / Michael Santos Duarte. – 2021.
168 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia de Teleinformática, Fortaleza, 2021.
Orientação: Prof. Dr. Guilherme de Alencar Barreto.

1. Correntropy. 2. Kernel method. 3. Non-Gaussian noise. 4. Nonlinear system identification. 5. Recurrent neural network. I. Título.

CDD 621.38

MICHAEL SANTOS DUARTE

# ON CORRENTROPY-BASED MACHINE LEARNING MODELS FOR NONLINEAR SIGNAL PROCESSING: ADDRESSING SPARSITY, RECURSIVE ESTIMATION AND OUTLIER-ROBUSTNESS

A thesis presented to the Graduate Program on Teleinformatics Engineering of the Center of Technology at Federal University of Ceará in fulfillment of the the requirement for the degree of Doctor in Teleinformatics Engineering.
Area of Concentration: Signals and systems

Approved on:

EXAMINING COMMITTEE

A meu amado filho, Eron.

**ACKNOWLEDGEMENTS**

To Prof. Dr. Guilherme de Alencar Barreto for guiding me in my doctoral thesis. I thank him for granting me the opportunity to grow as a researcher.

To Prof. Dr. José C. Príncipe and colleagues from University of Florida (UF) who kindly welcomed me into their research group during the internship period at Computational NeuroEngineering Laboratory (CNEL). I am especially grateful to the researcher Dr. Kan Li who warmly welcomed me into the group and was a great inspiration for my studies.

To my many friends from PPGETI and Centaurinho, for all the conversations and exchange of ideas. I am especially grateful to friends Renan Bessa and Allan Kelvin Mendes de Sales, with whom I had valuable discussions without any fear of judgement.

To the Ph.D. student in Electrical Engineering, Ednardo Moreira Rodrigues, and his assistant, Alan Batista de Oliveira, a graduate student in Electrical Engineering, for the adequacy of the template used in this work so that it could follow the standards of the library Federal University of Ceará (UFC).

I thank all the teachers in the program for providing me with not only rational knowledge, but also the manifestation of the character in the professional training process.

To IFCE, UFC, and CAPES, for the promotion of the interinstitutional doctorate program.

To my mother for constantly encouraging studies and human development.

Finally, my wife Michely Hellen, who in my absence dedicated to higher education, has always made herself understood that the future is made from the constant dedication in the present!

As invenções são, sobretudo, o resultado de um trabalho teimoso, em que não deve haver lugar para o esmorecimento.

(Santos Dumont)

## RESUMO

Esta tese investiga métodos de aprendizado de máquina baseados em teoria da informação, espaços de Hilbert de reprodução e redes neurais recorrentes para tratar problemas de modelagem de sistemas dinâmicos não lineares, tais como identificação de sistemas e predição de séries temporais. Argumenta-se que tais métodos são capazes de prover o correto tratamento a dados gerados por sistemas não lineares e não estacionários, muitas vezes imersos em cenários contaminados com ruído não gaussiano. Tais características conferem maior complexidade ao problema de construção do modelo preditivo, principalmente quando o processo de estimação de parâmetros deve ser executado online. Isto posto, esta tese propõe usar o conceito de correntropia, que é basilar dentro do arcabouço de aprendizado baseado em teoria da informação, em conjunção com métodos de filtragem adaptativa kernel e arquiteturas de redes neurais recorrentes para desenvolver novos algoritmos dotados da capacidade de aprendizado online e robusto em tarefas de modelagem de sistemas dinâmicos não lineares. Dentre as contribuições constantes nesta tese destacam-se as seguintes. $(i)$ Introdução de uma variante esparsa do modelo kernel de aprendizado por correntropia (CKL, *correntropy kernel learning*) que melhora o modelo CKL original através da introdução do critério de esparsidade por dependência linear aproximada e pela computação recursiva de matrizes de kernel. $(ii)$ Introdução de uma segunda variante do modelo CKL, também esparsa e online, porém agora dotado de um dicionário completamente adaptativo, ou seja, que pode crescer ou diminuir de tamanho à medida que o tempo passa. $(iii)$ Uma nova solução do modelo CKL no primal que é baseada em características aleatórias de Fourier (*random Fourier features*, RFF) que são usadas como uma função de kernel definida positiva para induzir um espaço de Hilbert de reprodução de dimensão pré-definida. $(iv)$ Proposição de um método alternativo de construção de dicionários através da divergência Kullback–Leibler, método este que é aplicado em redes regularizadas no espaço de Hilbert de reconstrução. $(v)$ Introdução de uma nova abordagem de aprendizado online chamado estados de eco com kernel recursivo de máxima correntropia, cuja distinção se dá pelo mapeamento espaço-temporal usando computação de reservatório. $(vi)$ Por fim, é proposta uma rede neural recorrente com algoritmo de treinamento baseado em correntropia para modelar a dinâmica de séries temporais caóticas. Todos os modelos preditivos resultantes das seis propostas são avaliados em cenários desafiadores usando uma variedade de conjunto de dados de pequena e larga escala, para diferentes níveis de contaminação por outliers. Os resultados alcançados revelam que os modelos propostos são de fato robustos a outliers, sendo capazes de manter alto poder preditivo mesmo sob regime de aprendizado online e não estacionário.

**Palavras-chave:** Correntropia. Identificação de sistemas não-lineares. Processamento de sinais não-Gaussianos. Aprendizado de máquina online. Métodos de kernel. Redes neurais recorrentes. Esparsificação.

## ABSTRACT

This thesis investigates machine learning methods based on information theory, reproduction kernel Hilbert spaces and recurrent neural networks to address nonlinear dynamical system modeling problems, such as system identification and time series prediction. It is argued that such methods are capable of providing the correct treatment to data generated by non-linear and non-stationary systems, often immersed in scenarios contaminated with non-Gaussian noise. These characteristics make the problem of model building considerably more complex, especially when the parameter estimation process must be performed online. That said, this thesis proposes to use the concept of correntropy, which is fundamental within the information-theoretic learning framework, in conjunction with kernel adaptive filtering methods and recurrent neural network architectures to develop new outlier-robust algorithms endowed with online learning capability to be applied to nonlinear dynamical system modeling tasks. Among the contributions in this thesis, the following ones are highlighted. $(i)$ Introduction of a sparse variant of the kernel correntropy learning (CKL) model that improves on the original CKL model by using the approximate linear dependence sparsity criterion and recursive computation of kernel matrices. $(ii)$ Introduction of a second variant of the CKL model, also sparse and online, but now equipped with a fully adaptive dictionary, that is, it can grow or shrink in size as time passes. $(iii)$ A new solution of the primal CKL model that is based on random Fourier features (RFF) which are used as a positive definite kernel function to induce a reproducing kernel Hilbert space with predefined dimensionality. $(iv)$ Proposition of an alternative method of constructing dictionaries through the Kullback–Leibler divergence, a method that is applied to regularized networks in the reproducing kernel Hilbert space. $(v)$ Introduction of a new approach to online learning called echo states with a recursive kernel of maximum correntropy, whose distinction is to put forward a spatiotemporal mapping using reservoir computing. $(vi)$ Finally, a recurrent neural network with a training algorithm based on correntropy is proposed to model the dynamics of chaotic time series. All predictive models resulting from the six proposals are evaluated in challenging scenarios using a variety of small and large-scale data sets, for different levels of outlier contamination. The results achieved reveal that the proposed models are in fact robust to outliers, being able to maintain high predictive power even under online and non-stationary learning.

**Keywords:** Correntropy. Nonlinear system identification. Non-Gaussian signal processing.

Online machine learning. Kernel methods. Recurrent neural networks. Sparsification.

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ALGORITHMS

## LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| ALD | approximate linear dependence |
| ARMAX | autoregressive moving average with exogenous inputs |
| ARX | autoregressive with exogenous inputs |
| AWGN | additive white Gaussian noise |
| CFC | correntropy-based feedback criterion |
| CKL | correntropy kernel learning |
| CRONOS | correntropy-based regularized online network with sparsity |
| CYBERNET | correntropy-based Elman recurrent network |
| ECC | error correntropy criterion |
| ES-KRMC | echo state kernel recursive maximum correntropy |
| ES-RLM | robust echo state recursive M-estimate |
| ESN | echo state networks |
| EX-KRLS | extended KRLS |
| FNN | feedforward neural network |
| FTDNN | focused time-delay neural network |
| GM | geometric mean |
| GP | Gaussian process |
| GQ | Gaussian quadrature |
| ITL | information-theoretic learning |
| KAF | kernel adaptive filtering |
| KAPA | kernel affine projection algorithms |
| KL | kernel learning |
| KLMS | kernel least mean squares |
| KMC | kernel maximum correntropy |
| KRLS | kernel recursive least squares |
| KRMC | kernel recursive maximum correntropy |
| LMS | least mean squares |
| LS-SVM | least squares support vector machine |
| MCC | maximum correntropy criterion |
| ML | maximum likelihood |
| MLP | multilayer perceptron |

| | |
|---|---|
| MSE | mean squared error |
| NARMAX | nonlinear ARMAX |
| NARX | nonlinear ARX |
| NFIR | nonlinear finite impulse response |
| NMSE | normalized mean squared error |
| NOE | nonlinear output error |
| OS-CKL | online sparse CKL |
| OS-CKL-NC | OS-CKL model using the novelty criteria |
| OSA | one-step ahead |
| PEM | prediction error method |
| RC | reservoir computing |
| RFF | random Fourier features |
| RFF2 | RFF with complex exponentials |
| RKHS | reproducing kernel Hilbert spaces |
| RLS | recursive least square |
| RMSE | root mean squared error |
| RN | regularized network |
| RNN | recurrent neural network |
| ROB-KRLS | robust KRLS |
| SVM | support vector machine |
| SVR | support vector regression |
| TDFNN | time-delay single-layer feedforward neural network |
| TS | Taylor series |
| WLS-SVM | weighted least squares support vector machine |

## LIST OF SYMBOLS

| | |
|---|---|
| $\boldsymbol{x}_t$ | $t$-th input regression vector |
| $y_t$ | $t$-th real system output |
| $\hat{y}_t$ | $t$-th predicted system output |
| $\boldsymbol{y}$ | Vector of system outputs |
| $\boldsymbol{\theta}$ | Parameters vector |
| $p$ | Parameters vector size $\boldsymbol{\theta}$ |
| $f(\cdot)$ | Regression function |
| $\phi(\cdot)$ | Nonlinear mapping or function activation |
| $d$ | Dimensionality of the input space |
| $u_t$ | $t$-th real system input |
| $L_u$ | Memory order of the system input |
| $L_y$ | Memory order of the system output |
| $\epsilon_t$ | $t$-th random noise AWGN |
| $\mathcal{D}_t$ | $t$-th stream of training samples |
| $N$ | Number of data samples |
| $\boldsymbol{w}$ | Model's parameter vector |
| $b$ | Bias term |
| $\mathcal{D}^{sv}$ | Dictionary of support vectors |
| $Z$ | data set |
| $Z_{est}$ | Training data set |
| $Z_v$ | Validation data set |
| $J(\cdot)$ | Cost function (Learning criterion) |
| $\gamma, \lambda$ | Regularization parameter or moment term |
| $f_t(\cdot)$ | $t$-th regression function |
| $\alpha_t$ | $t$-th Lagrange multiplier or $t$-th coefficient of the solution vector |
| $\boldsymbol{\alpha}$ | Vector of Lagrange multipliers |

| | |
|---|---|
| $k(\cdot)$ | Kernel function |
| $\beta_k$ | Scale parameter |
| $\boldsymbol{b}_k$ | Location parameters |
| $\rho(\cdot)$ | Objective function or loss function for M-Estimators |
| $\Psi(\cdot)$ | Score function for M-estimators |
| $e_t$ | $t$-th prediction error |
| $\varpi(\cdot)$ | Weighting function |
| $\sigma$ | Gaussian kernel bandwidth |
| $z^{-1}$ | Unit delays |
| $\breve{y}_t$ | $t$-th noiseless system output |
| $\boldsymbol{w}^*$ | True model's parameter vector |
| $V$ | Correntropy |
| $E$ | Expected value |
| $\hat{V}$ | Sample estimator of correntropy |
| $L(\cdot)$ | Lagrangian function |
| $\boldsymbol{K}$ | Kernel matrix |
| $\boldsymbol{\Omega}$ | Diagonal matrix or high-dimensional weight vector |
| $\boldsymbol{V}, \boldsymbol{Q}$ | Diagonal matrix |
| $\boldsymbol{P}$ | Inverse kernel matrix |
| $\boldsymbol{H}$ | Kernel matrix plus diagonal matrix |
| $m_t$ | Size of support vectors dictionary |
| $\boldsymbol{a}$ | Coefficients vector |
| $\upsilon$ | ALD threshold |
| $\tilde{\boldsymbol{K}}$ | Kernel matrix calculated with the dictionary samples |
| $\tilde{\boldsymbol{k}}$ | Kernel function calculated with the dictionary samples |
| $\tilde{\boldsymbol{\Omega}}$ | Diagonal matrix calculated with the dictionary samples |
| $\tilde{\boldsymbol{P}}$ | Inverse kernel matrix calculated with the dictionary samples |
| $\tilde{\boldsymbol{H}}$ | Kernel matrix plus diagonal matrix calculated with the dictionary samples |

| | |
|---|---|
| $\tilde{\boldsymbol{\theta}}$ | Parameters vector with the dictionary samples |
| $\tilde{\boldsymbol{x}}$ | Input regression vector with the dictionary samples |
| $\tilde{\boldsymbol{\alpha}}$ | Lagrange multiplier vector with the dictionary samples |
| $\boldsymbol{A}$ | Coefficients matrix |
| $\delta$ | Result of ALD test |
| $\epsilon_{update}$ | Update matrix threshold to CKL models |
| $\mathcal{N}$ | Gaussian function |
| $\boldsymbol{y_d}$ | Vector of system outputs correspondents to the dictionary |
| $\boldsymbol{\Phi}$ | Matrix with the nonlinearity projected input vectors |
| $\Lambda(\cdot)$ | Convex, continuous, but not necessarily differentiable regularization function |
| $Prox$ | Proximal operator |
| $\beta$ | Proximal operator weights |
| sign$(\cdot)$ | Signum function |
| $\xi$ | Small positive constant to avoid division by zero |
| $\delta_1$ | Novelty criterion threshold |
| $dist$ | Mallest distance from the input vector |
| $D, d_E$ | Embedding dimension |
| $\hat{\phi}(\cdot)$ | Explicit expression for $\phi$ |
| $\tilde{\phi}(\cdot)$ | Explicit expression for $\phi$ with the dictionary samples |
| $\mathcal{H}$ | Hilbert space |
| $\varepsilon, \delta_2$ | Error criterion threshold |
| $\boldsymbol{W}$ | Weight matrix of internal network connections |
| $\boldsymbol{W}_{in}$ | Input weight matrix |
| $\boldsymbol{W}_{out}$ | Trained output weights |
| $\boldsymbol{W}_{back}$ | Output feedback weight matrix |
| $h(\cdot)$ | Neuron activation function |
| $\boldsymbol{\psi}_t$ | $t$-th concatenation of the input vector and the internal state |
| $N_x$ | Size vector of reservoir neuron activations |

| | |
|---|---|
| $N_u$ | Size input vector fed to the network |
| $N_\psi$ | Size of readout vector to the network |
| $N_y$ | Size of output vector to the network |
| $\vartheta$ | Threshold for outlier-detection |
| $\tau$ | Embedding delay |
| $\delta$ | Attractor size |
| $\alpha$ | Learning rate |
| $u_i$ | $i$-th neuron activation of the first hidden layer |
| $w_{ij}$ | Synaptic weights of input layer to first hidden layer |
| $w_{il}^c$ | Synaptic weights of first hidden layer to input layer (First context layer) |
| $\nu_i$ | $i$-th neuron output of the first hidden layer |
| $q_1$ | Number of neurons in the first hidden layer |
| $q_2$ | Number of neurons in the second hidden layer |
| $\eta$ | Correntropy propagation threshold |
| $u_s$ | $s$-th neuron activation of the second hidden layer |
| $b_{si}$ | Synaptic weights of first hidden layer to the second hidden layer |
| $b_{sl}^c$ | Synaptic weights of second hidden layer to first hidden layer (Second context layer) |
| $\nu_s$ | $s$-th neuron output of the second hidden layer |
| $u_k$ | $k$-th neuron activation of the output layer |
| $m_{ks}$ | Synaptic weights of the second hidden layer to output layer |
| $y_k$ | $k$-th system output |
| $\delta_k$ | $k$-th local gradient of the output layer |
| $\delta_s$ | $s$-th local gradient of the second hidden layer |
| $\delta_i$ | $i$-th local gradient of the first hidden layer |

CONTENTS

# 1 INTRODUCTION

This thesis looks into fresh approaches about kernel-based learning machines in which model-building and, hence, parameter estimation are continuous processes and executed on the fly for each new incoming sample through the amalgamation of different learning frameworks which leads to *adaptive, nonlinear, robust, online* and *sparse* models.

In practical applications dynamical systems modeling tasks often have to deal with nonlinear phenomena, non-Gaussian noise, and time-varying operational states. In such scenarios, the central problem involves function estimation to model a dynamic system from a finite collection of noisy and output samples.

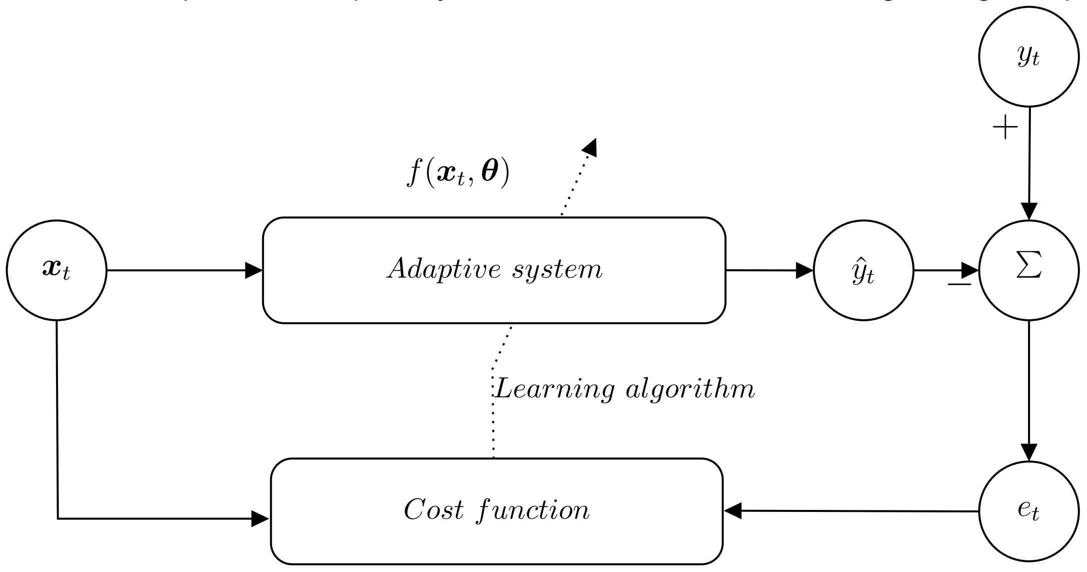
In this context, mathematical models for adaptive filtering and machine learning have been used to approximate the relationship between system inputs and outputs by observing pairs of its samples $(\boldsymbol{x}_t, y_t)$ produced by a given process or phenomenon of interest. For this adaptive system, $\boldsymbol{x}_t \in \mathbb{R}^d$ denotes the input regression vector of dimension $d$ at time $t$ and its components are given as a function of the relevant variables of the system. $y_t \in \mathbb{R}$ denotes the output at time $t$. Fig. 1 illustrates the concept of an adaptive system used for function approximation under a supervised learning scheme.

A learning machine or adaptive system with a set of free parameters $\boldsymbol{\theta}$ is constructed to receive the data $\boldsymbol{x}_t$ and produce a predicted output $\hat{y}_t$. The process of estimating parameters from data, called learning or adaptation makes the output $\hat{y}_t$ of the adaptive system to converge (i.e. to become similar) to the observed output $y(t)$, according to a predefined cost function that is optimized by changing the parameters $\boldsymbol{\theta}$ by means of a systematic procedure. In the end, an approximated model is obtained that, when activated with new data from the source $\boldsymbol{x}_t$, predicts the output of the system with $\hat{y}$.

The procedure just outlined builds implicitly a model for the relationship between the system input and output (PRINCIPE, 2010). This is a system identification problem and two approaches have been used to solve it. Firstly, a parametric approach in which a finite-dimensional vector $\boldsymbol{\theta}$ of size $p$ is introduced to parameterize the unknown function, e.g., using a polynomial model (AKAIKE, 1974). Secondly, a nonparametric approach in which the unknown target function $f(\cdot)$ is assumed to belong to a high-dimensional space. This approach leads to predictor estimation that is the core of the classical system identification paradigm (LJUNG, 1999) and the machine learning philosophy (BISHOP, 2006).

From a machine learning perspective, non-parametric methods based over repro-

Figure 1 – Example of an adaptive system, in which a model is building through adaptation.



Source: adapted from Principe (2010).

ducing kernel Hilbert spaces (RKHS) (CHIUSO; PILLONETTO, 2019), called kernel learning methods, which include powerful approaches like kernel adaptive filtering (KAF), regularized network (RN), support vector machine (SVM), and Gaussian process (GP) have gained popularity in the system identification community.

Kernel methods provide to the learning model the ability to deal with nonlinear systems via the kernel trick, which transforms data from the original input space to a possibly infinite-dimensional RKHS without the need of explicitly computing inner products in that space. For nonlinear system identification tasks, several algorithms have been proposed, using, e.g., KAF-type models (SANTOS; BARRETO, 2017), GP-type models (MATTOS, 2017), and SVM-type models (LIU; CHEN, 2013).

An alternative to deal with nonlinear dynamical systems is to use a recurrent neural network (RNN). Standard single-layer feedforward neural network (FNN) architectures, such as the multilayer perceptron (MLP), are incapable of modeling long-term temporal characteristics of nonlinear dynamic systems (KREMER, 2001). An FNN learns a static mapping in which the outputs of the network depend solely on the current inputs, thereby neglecting the dependency of data points at neighboring time steps. In order to allow kernel methods to handle such long term temporal dependencies, authors have incorporated the reservoir computing (RC) framework into the KAF algorithm (ZHOU *et al.*, 2018; SHI; HAN, 2007). The reservoir is used typically in echo state networks (ESN) (LUKOSEVICIUS; JAEGER, 2009) and provides the KAF algorithm with the ability to perform a spatiotemporal mapping that transforms the past inputs into a high-dimensional reservoir state space in which the readout layer is learned by simple linear regression methods. As the reservoir is randomly generated and stays fixed during all learning processes the dynamic reservoir has a fixed recurrent topology and thus only the output weights need to be solved. ESN is a well-known class of RNN. It is important to observe that to develop a successful training procedure for RNN we must solve problems related to the output fed back into a network during training (JAEGER; HAAS, 2004).

Although kernel-based models and recurrent neural networks can handle nonlinear systems, during the model building, the presence of outliers or impulsive noise in the data deteriorates considerably the resulting predictors of performance. In order to tackle this problem it is necessary that the cost function and the parameter estimation have adequate mechanisms to deal with non-Gaussian noise present in real-world data sets.

Linear parameter estimation methods, such as, least mean squares (LMS) and recursive least square (RLS) (HAYKIN, 1996), and their nonlinear counterparts, such as, kernel least mean squares (KLMS) (LIU *et al.*, 2008) and kernel recursive least squares (KRLS) (ENGEL *et al.*, 2004) rely on 2nd order statistics, such as the mean squared error (MSE) and hence are not robust to non-Gaussian noise. An efficient alternative is using information-theoretic learning (ITL) framework (PRINCIPE, 2010).

ITL has been successfully applied to address the issues of nonlinear and non-Gaussian signal processing, especially in system identification in which one significant challenge is that the input-output samples often contain different kinds of outliers (PEARSON, 2002). ITL is a broader learning framework for the implementation of the concept of adaptive information filtering, which replaces the cost function or any other second-order statistics, with an information theoretic descriptor. For instance, a learning criterion in ITL, the maximum correntropy criterion (MCC) (SANTAMARIA *et al.*, 2006; LIU *et al.*, 2007) used for developing robust adaptive filters, such as the kernel maximum correntropy (KMC) (ZHAO *et al.*, 2011), the correntropy kernel learning (CKL) (LIU; CHEN, 2013) and the kernel recursive maximum correntropy (KRMC) (WU *et al.*, 2015).

From the exposed so far, the present thesis uses the correntropy framework and machine learning paradigms as a foundation to develop new robust sequential learning machines for nonlinear dynamic system modeling in challenging scenarios, typically found in real-world applications. The new explorable approach allows continual lifelong learning to propose alternatives to deal with challenges of system identification of a machine learning perspective.

## 1.1 Thesis Statements

### *1.1.1 Research Problem*

System identification is quite a mature area, but many problems remain. It involves issues as nonlinear adaptive signal processing, convexification, model reduction, online learning, and large-scale data sets (LJUNG, 2008). Therefore, the most recent applications in science and engineering have been based on model design and optimization that represent highly complex systems with high demands on autonomy, and adaptation. The models used internally in these systems also need to be maintained and updated throughout life autonomously calling for data-driven models (LJUNG *et al.*, 2011). Thus, the big challenge in system identification, especially in an online continual learning regime, is how we can achieve high predictive power with low computational complexity in a real-world that involves nonlinear phenomena, time-varying operational states, and non-Gaussian noise.

### *1.1.2 Key Ideas*

This thesis proposes using the correntropy framework and machine learning paradigms to provide a significant push forward to the problem of outlier-robust learning in dynamical system modeling to lead the idea of active sequential learning.

The idea of robust active sequential learning here is to construct an accurate model with significantly fewer parameters than standard kernel-based adaptive filters. For this purpose, we rely on efficient techniques, such as, techniques as random projection, recursively matrix computation, and criteria for sparsification to obtain a subset of data for efficient training and sparse representation.

## 1.2 Objectives

The overall objective of this thesis is to propose new learning machines for system identification with online and continual learning that deal with nonlinear, time-varying, and non-Gaussian signals with low computational cost. In order to accomplish that, we shall specifically follow the next steps:

1. Introduce a new correntropy kernel learning model, for online system identification in outlier-contaminated scenarios;

2. Propose fully adaptive dictionaries of support vectors (SVs) so that it can either grow (as most of the kernel models to date do) or shrink if some of the SVs become obsolete with time;
3. Develop a primal solution of the correntropy kernel learning model capable of operating in an online learning regime;
4. Assign the ability of spatiotemporal mapping to outlier-robust KAF-type model;
5. Use nontraditional schemes based on correntropy to learn the behavior of dynamic systems;
6. Evaluate all the proposed learning machines in modeling of dynamic systems, in scenarios with long-term recursive prediction in the presence of outliers, using synthetic and real-world data sets, including large-scale data sets.

## 1.3 Associated Publications

The following articles have been published by the author throughout the development of this thesis. It includes articles directly related to the main subjects of the present thesis, as well as articles related to the cooperation on projects between different researchers. The articles related to the thesis are listed below.

1. **Michael S. Duarte** and Guilherme A. Barreto (2019). *Online Sparse Correntropy Kernel Learning for Outlier-Robust System Identification*. 12th IFAC Symposium on Dynamics and Control of Process Systems (DYCOPS-CAB). Florianópolis, SC, Brazil. vol. 52, p. 430-435, to appear (https://doi.org/10.1016/j.ifacol.2019.06.100).
2. **Michael S. Duarte** and Guilherme A. Barreto (2019). *Uma Rede Regularizada Esparsa e Robusta para Identificação Recursiva de Sistemas Dinâmicos*. XIV Congresso Brasileiro de Inteligência Computacional (CBIC). Belém, PA, Brazil. vol. 1, p. 1-8, to appear (http://dx.doi.org/10.21528/CBIC2019-76).
3. **Michael S. Duarte** and Guilherme A. Barreto (2021). *Fully Adaptive Dictionary for Online Correntropy Kernel Learning Using Proximal Methods*. Expert Systems with Applications, vol. 178, p. 1-10, to apper (https://doi.org/10.1016/j.eswa.2021.114976).
4. **Michael S. Duarte**, Guilherme A. Barreto, Kan Li and Jose C. Principe. *Robust SVR Online Primal Solution Based on Correntropy*. IEEE Transactions on Neural Networks and Learning Systems, in preparation.
5. **Michael S. Duarte**, Renan Bessa and Guilherme A. Barreto. *An Extremily Outlier-*

*Robust Echo State Kernel Recursive Maximum Correntropy Algorithm for Nonlinear System Identification*. Nonlinear Dynamics, in preparation.

6. **Michael S. Duarte** and Guilherme A. Barreto. *Robust Learning of Chaotic Dynamics using Correntropy-based Recurrent Neural Network*. Nonlinear Dynamics, in preparation.

## 1.4 Outline and Contributions

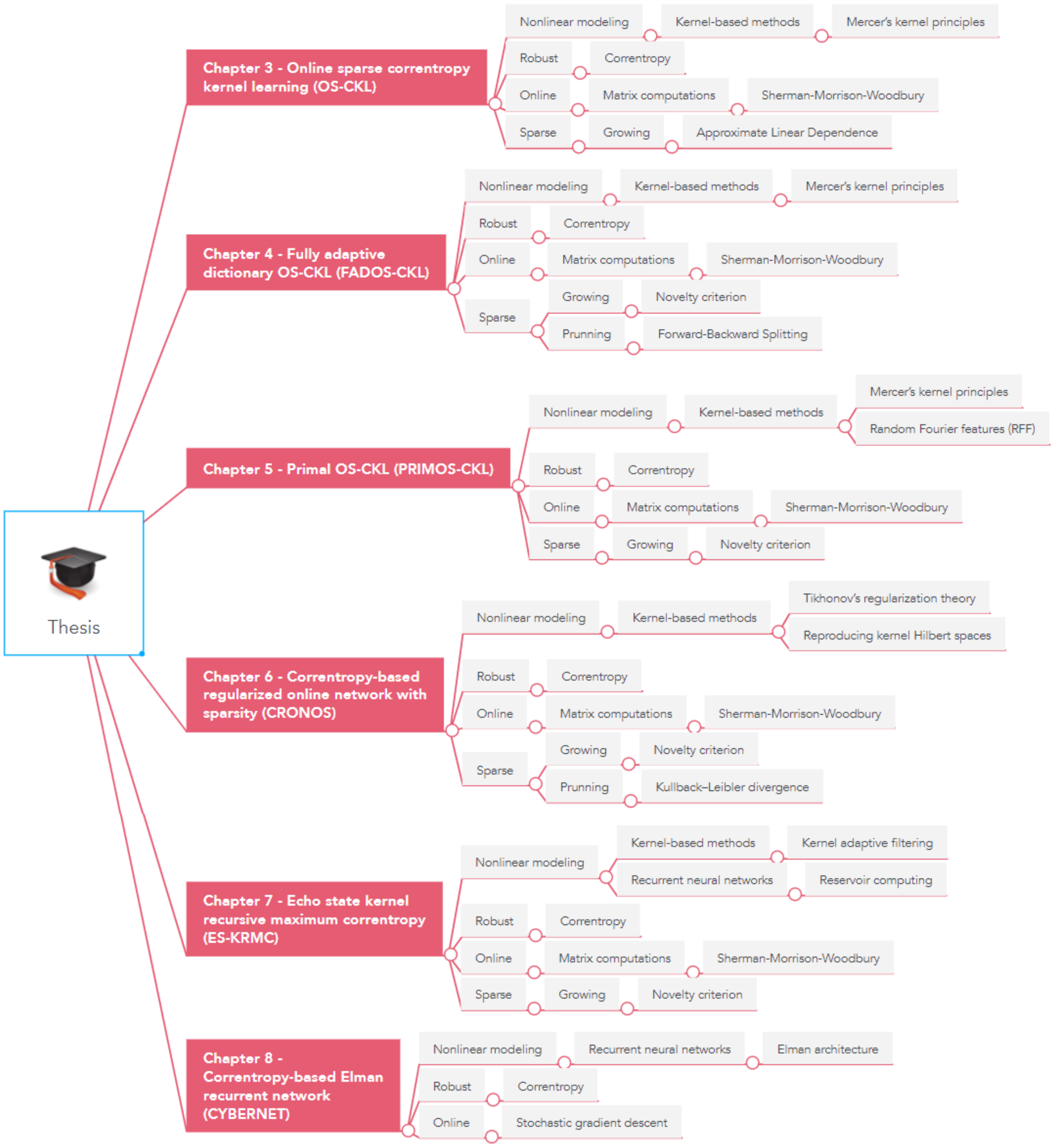
The remaining chapters of this thesis and their main contributions are detailed in the following.

Chapter 2 details the fundamentals of system identification and signal processing exploiting machine learning concepts. It includes the core of system identification paradigms around the concepts of information, estimation, and validation. Also, some challenges of system identification from a practical perspective are discussed, such as identification of nonlinear models, and robust estimation involving non-Gaussian noise.

From Chapters 3 to 8, new methods set for supervised learning are described and in Fig. 2 a diagram summarizes the contributions and fundamentals used in this thesis. Original contributions in these chapters include the following ones.

- In Chapter 3, it is proposed a sparse and online CKL model through approximate linear dependence (ALD) and by kernel matrices recursively;
- In chapter 4, it is proposed a new method for building fully adaptive dictionary of support vectors in online and sparse learning models through the proximal methods, and by computing matrices and submatrices recursively;
- In chapter 5, it is proposed a new method to develop an online SVR-type model in primal space using random Fourier features with an explicit feature-map reproducing kernel.
- In chapter 6, it is proposed an alternative method for the formulation of an adaptive dictionary of support vectors in online and sparse learning models through Kullback–Leibler divergence (also called relative entropy), and by computing matrices and submatrices recursively;
- In Chapter 7, it is proposed an outlier-robust learning model based on the KRMC algorithm and reservoir computing which is capable of spatiotemporal mapping;
- In Chapter 8, correntropy is inserted into a recurrent neural network; firstly, as a learning criterion to robust learning and secondly as a correntropy propagation parameter to ensure the learning of nonlinear correlations.

Figure 2 – Diagram with contributions summary and fundamentals used in this thesis.


Thesis
Chapter 3 - Online sparse correntropy kernel learning (OS-CKL)
Nonlinear modeling
Kernel-based methods
Mercer's kernel principles
Robust
Correntropy
Online
Matrix computations
Sherman-Morrison-Woodbury
Sparse
Growing
Approximate Linear Dependence
Chapter 4 - Fully adaptive dictionary OS-CKL (FADOS-CKL)
Nonlinear modeling
Kernel-based methods
Mercer's kernel principles
Robust
Correntropy
Online
Matrix computations
Sherman-Morrison-Woodbury
Sparse
Growing
Novelty criterion
Prunning
Forward-Backward Splitting
Chapter 5 - Primal OS-CKL (PRIMOS-CKL)
Nonlinear modeling
Kernel-based methods
Mercer's kernel principles
Random Fourier features (RFF)
Robust
Correntropy
Online
Matrix computations
Sherman-Morrison-Woodbury
Sparse
Growing
Novelty criterion
Chapter 6 - Correntropy-based regularized online network with sparsity (CRONOS)
Nonlinear modeling
Kernel-based methods
Tikhonov's regularization theory
Reproducing kernel Hilbert spaces
Robust
Correntropy
Online
Matrix computations
Sherman-Morrison-Woodbury
Sparse
Growing
Novelty criterion
Prunning
Kullback–Leibler divergence
Chapter 7 - Echo state kernel recursive maximum correntropy (ES-KRMC)
Nonlinear modeling
Kernel-based methods
Kernel adaptive filtering
Recurrent neural networks
Reservoir computing
Robust
Correntropy
Online
Matrix computations
Sherman-Morrison-Woodbury
Sparse
Growing
Novelty criterion
Chapter 8 - Correntropy-based Elman recurrent network (CYBERNET)
Nonlinear modeling
Recurrent neural networks
Elman architecture
Robust
Correntropy
Online
Stochastic gradient descent

Source: prepared by the author (2021).

The thesis is concluded in Chapter 9 with a discussion about the work and directions for future research on related topics.

All proposed approach were implemented from the scratch using MATLAB scripting language running on a Lenovo Ideapad 320 notebook with a 2.70 GHz Intel Core i7-7500U processor, 16 GB of RAM, and Windows 10 Home operating system. All codes implemented in this thesis are publicly available on the following repository: http://github.com/michaelsduarte/.

# 2 SYSTEM IDENTIFICATION AND SIGNAL PROCESSING

System identification is about building mathematical models of dynamic systems from observed input–output data. It is an old and mature area with its origin in automatic control, which started in the late 1950s (ZADEH, 1956). It can be seen as the interface between the real world of applications, the mathematical world of control theory, and model abstractions. As such, it is an ever-present necessity for successful applications. The characteristics of the models to be estimated, e.g., linear, nonlinear, hybrid, and/or non-parametric, determine the technique that will be used in the system identification procedure, making it a very large topic. Simultaneously, the area can be characterized by a small number of fundamental principles, e.g., to look for parsimonious representation by proper decisions in the following triangle: model complexity, information contents in the data, and effective validation (LJUNG, 2008).

As such, the identification procedure to build models using observed input-output data is characterized by three main components of a machine learning perspective, the information, the estimation method, and the validation. These three components will be detailed in this section exploiting machine learning concepts. It includes the core of a few fundamental classical system identification paradigms and statistical nature around the concepts of information, estimation (learning), and validation (generalization). Thus, we discuss these concepts and some challenges of system identification from a practical perspective, e.g., identification of nonlinear models, and robust estimation involving non-Gaussian noise.

## 2.1 On Extracting Information from Data

To extract information contained in data is a fundamental problem in data signal processing, because data hides, either in spatial redundancy or in time structure important clues to answer the questions of the information-processing in the identification procedure. Therefore, filtering information from data represents a paradigm in learning theory and adaptive systems (PRINCIPE, 2010). It is important to realize that this concerns both information provided by the observed data and prior information about the object to be modeled.

### *2.1.1 About the Data*

The input-output data that is used to select a model is the fundamental information source in the system identification procedure. Collecting data is the first step and consists of selecting the signals to be measured, and to decide how the input should be configured, in which the data collection with appropriate sampling procedures will have a major impact on the quality of the resulting model. It is important to realize that no model is a perfect representation of the true system under investigation. Any model will be an approximation of the phenomenon under investigation, and it will be affected by the aspects of the system that are excited during the experiment (SCHOUKENS; LJUNG, 2019).

### *2.1.2 On Data Modeling and Model Building*

Data modeling is one productive step involved in the information extraction from data. A model of the data essentially summarizes the process of its generation and allows the best design of data processing systems (PRINCIPE, 2010). To this task it is necessary to select the dynamic structure of the model, which includes the definition of the regressors $\boldsymbol{x}_t$, and to define a model class (a set, or collection, of models). It will generically be denoted by $\mathbb{M}$. As such, it could be a set that can be parameterized by a finite-dimensional parameter (LJUNG, 2008).

Recent research has shown that the model selection can be successfully tackled by different paradigms that lead to an interesting cross with the machine learning field (PILLONETTO; NICOLAO, 2010). Rather than postulating finite-dimensional hypothesis spaces, e.g., using autoregressive with exogenous inputs (ARX), autoregressive moving average with exogenous inputs (ARMAX), nonlinear ARX (NARX), nonlinear ARMAX (NARMAX), nonlinear output error (NOE), or nonlinear finite impulse response (NFIR), new learning paradigms have been introduced that postulate the problem as function estimation possibly in an infinite-dimensional space (AGUIRRE, 2015).

For the system identification task of interest to this thesis it is assumed the NARX structure, which can generally represent a wide class of discrete-time nonlinear systems (LJUNG, 1999):

$$\begin{aligned} y_t &= f(y_{t-1}, ..., y_{t-L_y}, u_{t-1}, ..., u_{t-L_u}) + \epsilon_t, \\ &= f(\boldsymbol{x}_t; \boldsymbol{\theta}) + \epsilon_t, \quad t = 1, ..., N, \end{aligned} \tag{2.1}$$

in which $L_u$ and $L_y$ denote the input and output memory orders, respectively. The target function $f(\cdot) \in \mathbb{M}$ is a static nonlinear mapping that aims at approximating the true mapping that generated the observations $y_t$ and $u_t$. Being that, $y_t$, $u_t$ and $\epsilon_t$ are, respectively, the system output, the system input, and the random noise[1] sample at time $t$. In this case, the input vector $\boldsymbol{x}_t$ is obtained by concatenating $L_y$ past observed outputs $y_t \in \mathbb{R}$ and $L_u$ past control inputs $u_t \in \mathbb{R}$ into a single regression vector:

$$\boldsymbol{x}_t = [y_{t-1}, ..., y_{t-L_y}, u_{t-1}, ..., u_{t-L_u}]^\top. \tag{2.2}$$

Thus, let us assume that we are given sequentially a stream of training samples:

$$\mathcal{D}_t = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_t, y_t)\}, \tag{2.3}$$

in which $(\boldsymbol{x}_t, y_t) \in \mathcal{X} \times \mathcal{Y}$ denotes the current input-output pair and $t \in N$. Here, the domain $\mathcal{X}$ is some nonempty set that the inputs (the regressor) $\boldsymbol{x}_t$ are taken from and the $y_t \in \mathcal{Y}$ are the targets.

In order to build a model from these data samples, we choose a particular version of the nonlinear model in Eq. (2.1), which now assumes the following linear-in-the-parameter form

$$\begin{aligned} y_t &= f(\boldsymbol{x}_t; \boldsymbol{\theta}) + \epsilon_t \\ &= f(\boldsymbol{x}_t; \boldsymbol{w}, b) + \epsilon_t \\ &= \boldsymbol{w}^\top \phi(\boldsymbol{x}_t) + b + \epsilon_t, \quad t = 1, ..., N, \end{aligned} \tag{2.4}$$

in which $\boldsymbol{\theta} = [\boldsymbol{w}^\top, b]^\top$ and the symbols $\boldsymbol{w}$ and $b$ denote the model's parameter vector and the bias term, respectively, with $\phi(\cdot)$ being a nonlinear mapping that project input vectors into a higher, possibly infinite, dimensional feature space.

The intrinsical ill-posedness of the parameter estimation problem is circumvented using regularization methods that also admit a Bayesian interpretation (RASMUSSEN, 2004). In particular, the impulse response is modeled as a zero-mean Gaussian process. In this way, prior information is introduced in the identification process by just assigning a covariance, named also kernel in the machine learning literature.

### 2.1.3 The Kernel Trick

Estimation and learning methods utilizing kernels have a denser mathematical and statistical development than earlier machine learning methods (e.g., neural networks). These methods formulate learning and estimation problems in a RKHS of functions defined on the data domain, expanded in terms of a kernel (HOFMANN *et al.*, 2008; HAYKIN, 2009).

[1] Usually assumed as additive white Gaussian noise (AWGN).

Real world dynamical modeling problems often require nonlinear methods to detect the kind of dependencies that allow successful prediction of properties of interest. By using a positive definite kernel it is possible to handle the nonlinearity required by the predictor mode. The kernel corresponds to a inner product in a (usually high-dimensional) feature space. In this space, our estimation methods are linear, but as long as we can formulate everything in terms of kernel evaluations, we never explicitly have to compute in the high-dimensional feature space.

In order to characterize a kernel we will suppose the empirical data stream as in Eq. (2.3) in which we have not made any assumptions on the domain $\mathcal{X}$ other than it being a set. In order to study the problem of learning, we need additional structure. In learning, we want to be able to generalize unseen data points. In a general form, we want to choose $y$ such that $(\boldsymbol{x}, y)$ is in some sense similar to the training examples. To this end, we need similarity measures in $\mathcal{X}$ and in $\mathcal{Y}$. For $\mathcal{Y}$ is easier, as two target values can only be identical or different. For $\mathcal{X}$, we require a function

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \quad (\boldsymbol{x}, \boldsymbol{x}') \longmapsto k(\boldsymbol{x}, \boldsymbol{x}'), \tag{2.5}$$

satisfying, for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$,

$$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle, \tag{2.6}$$

in which $\phi$ maps into some inner product space $\mathcal{H}$, sometimes called the feature space. The similarity measure $k$ is usually called a kernel, and $\phi$ is also called its feature map. The expansion of Eq. (2.5) for the symmetric kernel $k(\boldsymbol{x}, \boldsymbol{x}')$ is an important special case of Mercer's theorem that arises in functional analysis (MERCER; FORSYTH, 1909).

The key concept is we can learn nonlinear models while still using the original linear model formulation. Considering a data set $Z$ that is not linearly separable in $\mathbb{R}^N$ it may be linearly separable in a higher dimensional space $\mathbb{R}^M$ (in which $M > N$). Thus, if we have a transformation $\phi$ that lifts the data set $Z$ to a higher-dimensional $Z'$ such that $Z'$ is linearly separable, then we can train a linear model formulation (e.g. SVM or CKL) on $Z'$.

It turns out that the model has no need to explicitly work in the higher-dimensional space at training or testing time. During training, the optimization problem only uses the training examples to compute pair-wise inner products $\langle \boldsymbol{x}, \boldsymbol{x}' \rangle$. This is important because it shows the existence of functions that, given two vector in $\mathbb{R}^N$, implicitly computes the inner product between this vectors in a higher-dimensional $\mathbb{R}^M$ without explicitly transforming this vectors to $\mathbb{R}^M$.

This implies that we have required that a kernel satisfies Eq. (2.5), that is, it corresponds to a inner product in some inner product space. In the present section we show that the class of kernels that can be written in the form Eq. (2.5) coincides with the class of positive definite kernels. This coincidence has far-reaching consequences. There are examples of positive definite kernels which can be evaluated efficiently even though they correspond to inner products in infinite dimensional inner product spaces. In such cases, substituting $k(\boldsymbol{x}, \boldsymbol{x}')$ for $\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$ is crucial. This substitution is called the kernel trick.

The kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ obeys important mathematical properties and can compute a inner product between $\boldsymbol{x}, \boldsymbol{x}'$. The kernel function $k$ is also a covariance function as used in Gaussian processes, being the Gram matrix $\boldsymbol{K}$ (RASMUSSEN, 2004). A Gram matrix or also called kernel matrix can be defined as a $N \times N$ matrix $K := (k(\boldsymbol{x}_i, \boldsymbol{x}_j))_{ij}$ of $k$ with respect to $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ in which $i, j \in N$.

The advantage of using such a kernel as a similarity measure is that it allows us to construct algorithms in inner product spaces. However, it considers the computational consequences of increasing the dimensionality from $\mathbb{R}^N$ to $\mathbb{R}^M$. If $M$ grows very quickly with respect to $N$ (e.g. $M \in O(2^N)$), then learning models via data set transformations will incur serious computational and memory problems.

### *2.1.4 Model Complexity*

Model complexity is a concept related to the size and, hence, the flexibility[2] of a model class. A big challenge of dealing with non-parametric methods based on RKHS in system identification is the high model complexity. A characteristic of these techniques is that they deal with kernel expansions whose number of terms is equal to the number of input data $N$, making them unsuitable for online large-scale applications. The alternatives used on online applications generally are recursive and have two stages for performing system identification, first a model order control stage that limits the increase in the number of terms by including only valuable kernels into the so-called dictionary $\mathcal{D}^{sv}$ and then a parameter update stage to solve well large-scale data sets.

[2] Also to the number of free parameters, which provide degrees of freedom to fit the model to the observed data samples

In order to decrease the complexity of the model techniques that select items (basis or support vectors) to build a dictionary $\mathcal{D}^{sv}$, are called sparsification techniques. A variety of sparsification techniques have been proposed to address this issue by curbing the network growth (HONEINE, 2015) in which the sparsification can be used by including new elements into the dictionary (LI; PRÍNCIPE, 2018) or pruning obsolete elements (KOPPEL *et al.*, 2018).

## 2.2 On Estimation Process

This is the process of choosing a model driven by the information. The data used for choosing the model is called estimation data, or training data, and it will be denoted by $Z_{est}$. This process is also called learning.

All data sets contain both relevant and irrelevant information. In order not to get deceived by the irrelevant information it is necessary to meet the data with prior information of some sort.

Thus, with a given data set and model class, the identification task is to select the model ($\boldsymbol{\theta}$) that best describes the observed data. Most estimation methods rely on a criterion that quantifies the quality of the approximation based on the error between the observed output $y_t$ and the model output $\hat{y}_t$, which can conceptually be written as

$$J(Z_{est}, \boldsymbol{\theta}) = \sum_{t=1}^{N} (y_t - f(\boldsymbol{x}_t; \boldsymbol{\theta}) + \epsilon_t)^2 + \gamma ||\boldsymbol{\theta}||^2, \quad t = 1, ..., N, \tag{2.7}$$

and define the parameter estimation task as an optimization problem:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \left\{ J(Z_{est}, \boldsymbol{\theta}) \right\}, \tag{2.8}$$

in which the $\gamma > 0$ is the regularization constant that includes prior knowledge, or imposes a desired behavior like smoothness or exponential decay to the solution. If multiple solutions to the optimization problem exist, the Eq. (2.8) has to be interpreted as a set inclusion. We call this the prediction error method (PEM). It coincides with the maximum likelihood (ML) method if the noise source $\epsilon_t$ is Gaussian (LJUNG, 1999).

The learning process to estimate $\boldsymbol{\theta}$ has two important features barely mentioned in system identification literature, maybe to be more commonly explored in machine learning, they are the learning structure (estimation structure) and the learning criterion (cost function).

### 2.2.1 Learning Structure

From a machine learning perspective, the learning structure has a black-box setting. A black-box model is more difficult to access and to understand than physical models. The idea is to parameterize the function $f(\boldsymbol{x}_t;\boldsymbol{\theta})$ so that it can well approximate any feasible true functions $f_0(\boldsymbol{x})$ (SCHOUKENS; LJUNG, 2019). A typical choice is to use a function expansion

$$f(\boldsymbol{x}_t,\boldsymbol{\theta}) = \sum_{t=1}^{N} \alpha_t f_t(\boldsymbol{x}_t), \tag{2.9}$$

with some basis functions $f_t(\cdot)$. It turns out that a powerful choice of basis functions is to let them be generated from one and the same "mother function" $k(x)$ and scale and translate it according to

$$f_t(\boldsymbol{x}_t) = k(\boldsymbol{\beta}_k(\boldsymbol{x}_t - \boldsymbol{b}_k)). \tag{2.10}$$

The basis functions are thus characterized by the scale (dilation) parameter $\beta_k$ and the location (translation) parameters $\boldsymbol{b}_k$ (basis vector). Typical choices of $k(\cdot)$ are the sigmoid or the Gaussian function. As $\boldsymbol{x}_t$ is a vector, the argument can be interpreted as a scalar product with a vector $\boldsymbol{\beta_k}$ which then also determines suitable projections in the regressor space. Another possibility is to interpret the scaling as ellipsoidal symmetric. The resulting structure in Eq. (2.9) and (2.10) is very general and very much used. Particular options contain radial basis neural networks, one-hidden-layer sigmoidal neural networks, neuro-fuzzy models, wavelets, least-squares support vector machines, among other learning structures (LJUNG, 1999).

### 2.2.2 Optimality Criterion

The optimality criterion also called the cost function, objective function or loss function, has the goal of extracting information and reduce uncertainty in the model building process. Thus, the correct selection of the learning criterion ensures the ability to deal with non-Gaussian signal processing, determining the complexity of the model and approximation accuracy.

A common criterion used in the learning process is the MSE but it is not the only one, and using an appropriate criterion to deal with non-Gaussian signal processing is much better for the purposes of this thesis. MSE cost function performs very well when the error statistics are zero mean and the *pdf* of the noise is Gaussian or short-tailed. In many practical situations, these conditions are violated by outliers that can make the noise nonzero-mean and extend the tails of the distribution. There are many possible ways of mitigating outliers, either by removing them manually, using trimming methods or using other statistics that are more resilient to noise such as the rank statistics (for which the median is the best-known example) (SIDAK *et al.*, 1999). An alternative is the weighted least squares method (SUYKENS *et al.*, 2002), in which the second moment of the error is substituted by other less steeply increasing functions of the error that are customized for the application (linear increase, windowing, even saturating after some value as the bi-square proposed by Tukey) (HAMPEL *et al.*, 1985). The solution is less sensitive to outliers, therefore it is called robust.

A systematic way of handling outliers is achieved by introducing the concept of robustness in maximum likelihood (HUBER, 1981). Recall that the maximum likelihood estimator from a data observation $\boldsymbol{x}$ assuming the distribution $\rho(\boldsymbol{x}|\boldsymbol{\theta})$ is known except for the parameters $\boldsymbol{\theta}$, can be written as

$$\hat{\boldsymbol{\theta}}_{MLE} = \arg\max_{\theta} \rho(\boldsymbol{x}|\boldsymbol{\theta}). \tag{2.11}$$

If we assume i.i.d. observations we can rewrite them as

$$\begin{aligned} \hat{\boldsymbol{\theta}}_{MLE} &= \arg\max_{\boldsymbol{\theta}} \prod_{t=1}^{N} \rho(\boldsymbol{x}_t|\boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \prod_{t=1}^{N} \log \rho(\boldsymbol{x}_t|\boldsymbol{\theta}) \\ &= \arg\min_{\boldsymbol{\theta}} \prod_{t=1}^{N} (-\log \rho(\boldsymbol{x}_t|\boldsymbol{\theta})), \end{aligned} \tag{2.12}$$

due to the monotonicity of the logarithm. Huber defined the *M*-estimators as a generalization of maximum likelihood as

$$\min \sum_{t=1}^{N} \rho(\boldsymbol{x}_t) \text{ or } \sum_{t=1}^{N} \Psi(\boldsymbol{x}_t) = 0 \text{ with } \Psi(\boldsymbol{x}) = \frac{\partial \rho(\boldsymbol{x})}{\partial \boldsymbol{x}}, \tag{2.13}$$

in which $\rho(\boldsymbol{x})$ must obey the properties

$$\begin{cases} \rho(\boldsymbol{x}) \geq 0 \\ \rho(0) = 0 \\ \rho(\boldsymbol{x}) = \rho(-\boldsymbol{x}) \\ \rho(\boldsymbol{x}_t) \geq \rho(\boldsymbol{x}_j), |\boldsymbol{x}_t| > |\boldsymbol{x}_j| \end{cases} \tag{2.14}$$

This estimator can be readily applied to regression problems instead of the MSE criterion. Let us assume the linear model $y = \boldsymbol{w}^\top \boldsymbol{x} + \epsilon$. The robust cost function can be applied to the error

Table 1 – Comparison of Robust Least Square Estimators

| Method | Cost Function | Weighting Function |
|---|---|---|
| Least squares | $\rho_{LS}(e) = e^2$ | $\varpi_{LS}(e) = 1$ |
| Huber | $\rho_H(e) = \begin{cases} e^2/2, \lvert e\rvert < \alpha \\ \alpha\lvert e\rvert - \alpha^2/2, \lvert e\rvert > \alpha \end{cases}$ | $\varpi_H(e) = \begin{cases} 1, \lvert e\rvert < \alpha \\ \alpha/\lvert e\rvert, \lvert e\rvert > \alpha \end{cases}$ |
| Bi-square | $\rho_{Bi}(e) = \begin{cases} (\alpha^2/6)(1-(1-(e/\alpha)^2)^3), \lvert e\rvert < \alpha \\ \alpha^2/6, \lvert e\rvert > \alpha \end{cases}$ | $\rho_{Bi}(e) = \begin{cases} 1-(e/\alpha)^2)^2, \lvert e\rvert < \alpha \\ 0, \lvert e\rvert > \alpha \end{cases}$ |
| ECC | $\rho_{ECC}(e) = (1 - exp(-e^2/2\sigma^2))/\sqrt{2\pi}\sigma$ | $\rho_{ECC}(e) = exp(-e^2/2\sigma^2))/\sqrt{2\pi}\sigma^3$ |

Source: Principe (2010)

yielding

$$J(e) = \sum_{t=1}^{N} \rho(e_t) = \sum_{t=1}^{N} \rho(y_t - \boldsymbol{w}^\top \boldsymbol{x}_t). \tag{2.15}$$

Considering the derivative of the cost w.r.t. as the parameter vector $\boldsymbol{w}$ and equating it to zero produces a set of equations of the form,

$$\sum_{t=1}^{N} \Psi(y_t - \boldsymbol{w}^\top \boldsymbol{x}_t)\boldsymbol{x}_t^\top = 0. \tag{2.16}$$

Defining the weighting function $\varpi(e) = \Psi(e)/e$. Then the equivalent gradient becomes

$$\sum_{t=1}^{N} \varpi_t \boldsymbol{x}_t^\top e_t = 0, \tag{2.17}$$

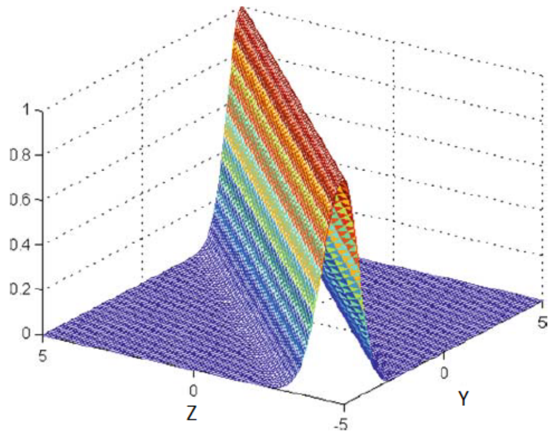
which corresponds to a weighted least squares cost function inasmuch as a method of iteratively re-weighted least squares (IRLS) can be used to solve optimization problems with objective functions as $\min_w \sum_{t=1}^{N} \varpi_t e_t^2$, since both the weighting function and the parameters depend upon the residues $e$ (FOX; MONETTE, 2002; BURRUS, 2012). This result establishes formally the link between *M*-estimation and weighted least-squares, which includes least-squares as a special case; that is, $\rho_{LS}(e) = e^2$ and $\varpi(e) = 1$ (see Table 1).

According to the ITL framework, a criterion successfully used is the error correntropy criterion (ECC) defined by a function of two arguments called cross-correntropy. Fig. 3 shows a plot of cross-correntropy for the Gaussian kernel with $\sigma = 1$. As one observes, cross-correntropy can also be used as a similarity measure in the joint space, but it differs appreciably from the MSE shown in Fig. 4. The cost emphasizes the behavior between $Z$ and $Y$ that exponentially attenuates contributions depending on the shape and parameter of the kernel utilized.

### *2.2.3 Correntropy: Definitions and Properties*

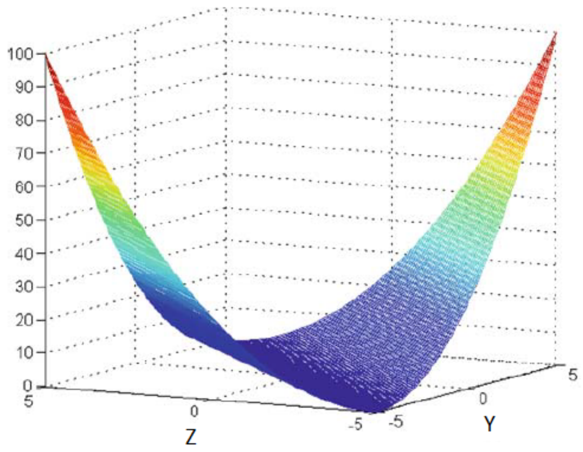
Correntropy is a generalized correlation measure among random signals. This measure is capable to extract statistical information of second- and higher-order from the signal

Figure 3 – Correntopy in the joint space.



Source: Principe (2010).

Figure 4 – The MSE cost function in the joint space.



Source: Principe (2010).

processing. Although by definition, correntropy is similar to correlation, recent studies have shown that it works better than the correlation when dealing with non-linear and non-Gaussian processes, without significant increase in computational cost (LIU *et al.*, 2007).

A more general form of cross-correntropy is defined between two arbitrary scalar random variables $Z$ and $Y$

$$V_\sigma(Z, Y) = E[k_\sigma(Z - Y)], \tag{2.18}$$

in which $E[\cdot]$ is the expected value operator and $k_\sigma$ is a symmetric positive definite function. Among the most common symmetric positive definite functions in machine learning are sigmoid, Gaussian, and polynomial (XU; PRINCIPE, 2008). In this thesis, it is chosen the Gaussian kernel defined by

$$k_\sigma(z - y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(z-y)^2}{2\sigma^2}\right\}, \tag{2.19}$$

in which $\sigma$ is kernel size or bandwidth. The kernel size or bandwidth is a free parameter that must be chosen by the user using concepts of density estimation, such as a maximum likelihood or Silverman's rule (SILVERMAN, 2018). The choice of kernel size, while important, is not as critical as in many other kernel methods and density estimation problems. For detection with the correntropy matched filter, it is trivial to choose a kernel size value using a quick check on a training set to select the best performance for the specific application (LIU *et al.*, 2008).

In practice, the joint *pdf* of Z and Y is unknown and only a finite number of data $(z_t, y_t)_{t=1}^{N}$ is available, leading to the following sample estimator of correntropy

$$\hat{V}_{N,\sigma}(Z,Y) = \frac{1}{N}\sum_{t=1}^{N} k_\sigma(z_t - y_t) = \frac{1}{N}\sum_{t=1}^{N} k_\sigma(e_t). \tag{2.20}$$

This equation converts the original cross-correntropy between two random variables to the correntropy of the difference (i.e., error) of these variables. Thus, a novel cost function for learning methods is introduced.

The ECC maximizes the error probability density at the origin and that yields the maximum correntropy criterion (MCC) algorithm (PRINCIPE, 2010) given by,

$$MCC = \arg\max_{\boldsymbol{w}} \hat{V}(E), \tag{2.21}$$

in which the parameters $\boldsymbol{w}$ control the *pdf* of the error $E = Z - Y$.

Important properties of the correntropy, assuming a stochastic and stationary process in discrete time are listed in the following (SANTAMARIA *et al.*, 2006):

- To any symmetric kernel, $V$ is a symmetric function;
- $V$ assumes the maximum value in origin;
- For the Gaussian kernel, the correntropy is positive and bounded: $0 < V(Z,Y) \leq 1/\sqrt{2\pi\sigma}$. It reaches its maximum if and only if $Z = Y$;
- The Toeplitz (GRAY, 2006) correntropy matrix is positive definite.

Comparing MSE with correntropy, we consider two random variables $Z$ and $Y$ with the error defined as $e = y - z$. Then, the $MSE(Z,Y)$ is defined as

$$MSE(Z,Y) = E[(Z-Y)^2] = \int_z\int_y (z-y)^2 f_{ZY}(z,y)dzdy = \int_e e^2 f(e)de, \tag{2.22}$$

and the $V(Z,Y)$ is defined as

$$V(Z,Y) = E[k(Z-Y)] = \int_z\int_y k(z-y) f_{ZY}(z,y)dzdy = \int_e k(e)f(e)de. \tag{2.23}$$

We can conclude that these two similarity measures are assessing similarity in rather different ways. Correntropy is local whereas MSE is global. By global, we mean that all the samples in the joint space will contribute to the value of the similarity measure while the locality of correntropy means that the value is primarily dictated by the kernel function along the line. Therefore, the MCC measure in Eq. 2.21 can be used as a new cost function for adaptive systems training. In summary, the MCC has the advantage that it is a local criterion of similarity and it should be very useful for cases when the measurement noise is nonzero mean, non-Gaussian, with outliers and there is a vast literature with variants techniques (LI *et al.*, 2014; CHEN *et al.*, 2018; LI *et al.*, 2021).

Furthermore, we can put MCC in a more general framework by showing that it bears a close relationship with M-estimation. M-estimation is a generalized maximum likelihood method proposed by Huber (1981) to estimate parameters $\theta$ under the cost function $\min_\theta \sum_{t=1}^{N} \rho(e_t|\theta)$, in which $\rho(\cdot)$ is a differentiable function satisfying the properties showed in Eq. (2.13).

In the case of adaptive systems, $\theta$ is a set of adjustable parameters and $e_t$ are errors produced by the system during supervised learning. This general estimation is also equivalent to a weighted least square problem as

$$\min_\theta \sum_{t=1}^{N} \varpi(e_t) e_t^2, \tag{2.24}$$

where the weight function $\varpi(e) = \rho'(e)/e$ in which $\rho'$ is the derivative of $\rho$. Defining $\rho(e) = (1 - \exp(-e^2/2\sigma^2))/\sqrt{2\pi}\sigma$ it is easy to see that $\rho$ satisfies all the conditions listed above. Moreover, it corresponds to the kernel of the error as can be easily shown as

$$\begin{aligned}\min_\theta \sum_{t=1}^{N} \rho(e_t) &= \min_\theta \sum_{t=1}^{N} (1 - \exp(-e_t^2/2\sigma^2))/\sqrt{2\pi}\sigma \\ &\Leftrightarrow \max_\theta \sum_{t=1}^{N} \exp(-e_t^2/2\sigma^2)/\sqrt{2\pi}\sigma \\ &= \max_\theta \sum_{t=1}^{N} k_\sigma(e_t).\end{aligned} \tag{2.25}$$

The weighting function in this case is

$$\varpi(e) = \exp(-e^2/2\sigma^2)/\sqrt{2\pi}\sigma^3. \tag{2.26}$$

In Liu *et al.* (2007), it was the first time a close relationship between M-estimation and methods of ITL was established. It is also interesting that there is no threshold in correntropy. The kernel size controls all the properties of the estimator. Moreover, this connection may provide one practical way to choose an appropriate kernel size for correntropy.

## 2.3 Model Validation

When a model has been estimated, some issues remain to be addressed: does it solve our problem satisfactorily? Is it in conflict with either the data or prior assumptions? These are in essence the issues behind the procedure of model validation.

This procedure ensures that the model is useful not only for the estimation data but also for future data. Data for this purpose are called validation data, to be denoted by $Z_v$. Another term for this process is generalization. Thus, the validation consists in verifying if predictions made with the trained model are acceptable, preferably using data similar but not equal to the records used in the section/building phases.

Often the decision is that the model is not good enough, so previous decisions on model building have to be revised. Usually, other model sets have to be tested or the conclusion might be that the data was not informative enough, so the experiment design must be reworked. This is the reason why system identification can be understood as an iterative problem with an identification loop illustrated in Fig. 5.

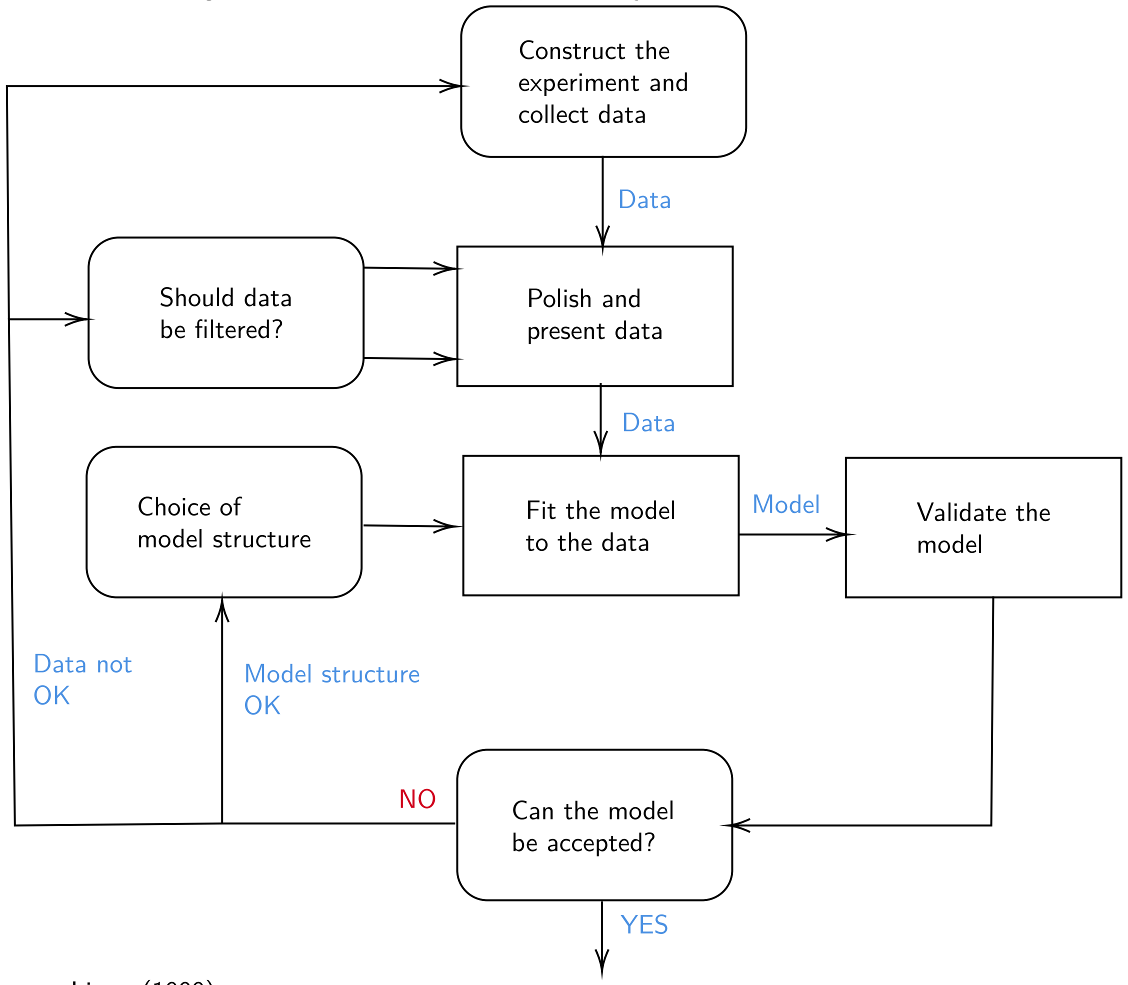
### *2.3.1 Prediction vs Simulation*

Once the model is properly obtained, it can be used to simulate the dynamical output of the identified system through iterative predictions. In this sense, there are two possible scenarios: *prediction* or *simulation*.

**Prediction model**: Roughly, a prediction model estimates the output of the system at time $t+1$, using the measured input up to time $t+1$, and the measured outputs up to time $t$. Prediction models are essential for modern control applications, in which essentially the predicted output is controlled. In this scenario we have one-step ahead (OSA) prediction, in which the next prediction is based on previous test inputs and test observed outputs until the current instant, in a feedforward fashion.

**Simulation model**: The alternative is that the measured outputs are not used at all, but the output is calculated from inputs only. This is called a simulation model, and it can be used to simulate the behavior of the system for new inputs. These models are useful to

Figure 5 – Identification cycle. Rectangles: the computer's main responsibility. Rounded rectangles: the user's main responsibility.


Construct the experiment and collect data
Data
Should data be filtered?
Polish and present data
Data
Choice of model structure
Fit the model to the data
Model
Validate the model
Data not OK
Model structure OK
NO
Can the model be accepted?
YES

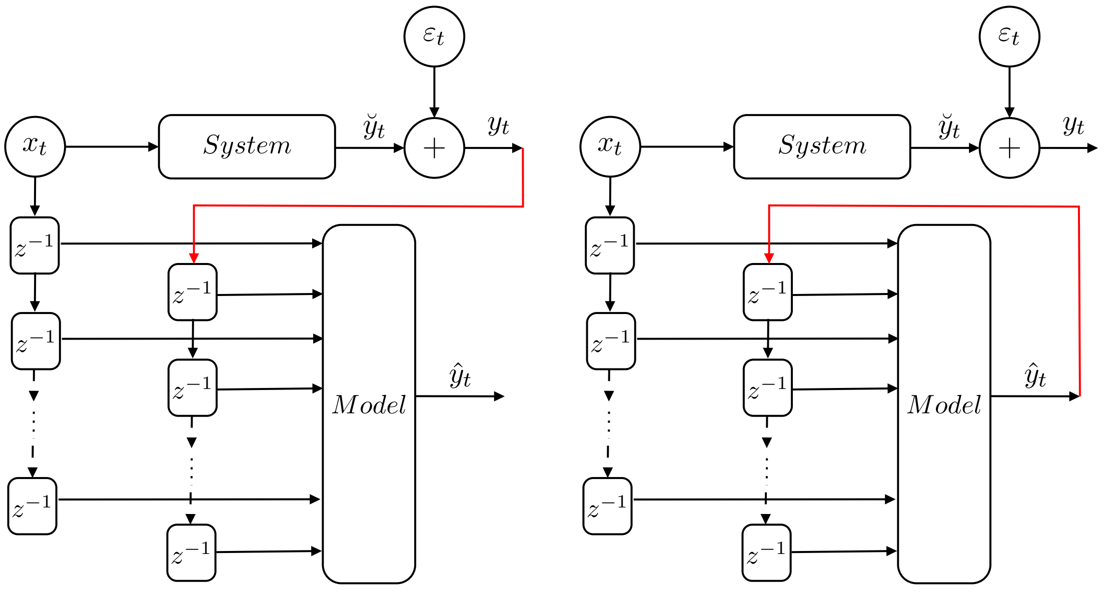
Source: Ljung (1999)

test what happens in new situations, to design systems and controllers, and to mimic physical systems. When predicted output values are fed back to the input regression vector we have a free simulation regime. This is a much more demanding task to tackle than the usual OSA prediction task. It is also called infinite-step ahead prediction, model predicted output or simply simulation.

It is harder to get a good nonlinear simulation than a prediction model. Free simulation models can become unstable, and it is harder to get small structural model errors than it is for OSA prediction models. Several models may have good OSA prediction accuracy and, however, perform poorly in free simulation, as they may not have correctly captured nonlinear dynamics useful in long-term behavior.

Fig. 6 illustrates the difference between the evaluation strategies described above, considering an autoregressive setup. The external input of the $t$-th iteration is denoted as $\boldsymbol{x}_t$, while the noiseless system output, noisy measurement (real system output corrupted by the

Figure 6 – Block diagrams of common evaluation methodologies for system identification. The $z^{-1}$ blocks indicate unit delays. The red lines highlight the difference between both approaches. Note that the left diagram presents a feedforward configuration, as opposed to the diagram in the right, which is recurrent.



(a) Prediction. (b) Simulation.

Source: Mattos (2017)

disturbance $\epsilon_t$), and model prediction are respectively $\breve{y}_t$, $y_t$, and $\hat{y}_t$. The $z^{-1}$ blocks indicate unit delays and the red lines highlight the difference between the two diagrams. Although OSA prediction can be useful for some tasks, it is argued by Billings (2013) that such validation methodology can be misleading, because even poor models can look good. Thus, throughout this thesis, it followed the more challenging free simulation approach.

### 2.3.2 Model Evaluation

One of the most common tools for model validation is *cross-validation*, which is to check how well the model performs for new data sets (validation data) that were not used to estimate the model. One way is to use the input of the validation data to simulate the model, to produce a simulated model output $\hat{y}_t$, and compare how well this model output reproduces the output $y_t$ of the validation data. The comparison could simply be a subjective, ocular inspection of the plots, to see if essential aspects of the system (for the intended application) are adequately reproduced. The comparison can also be done by computing numerical measures of the fit between the two signals. The root mean squared error (RMSE) between predictions and actual observations is a common choice to evaluate the goodness of fit of the model, i.e. it summarizes the discrepancy between the observed values and the predicted ones under the model in question

$$RMSE(f) = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_t - f(\boldsymbol{x}_t, \boldsymbol{\theta}))^2}, \qquad (2.27)$$

in which $N$ represents the number of test samples used for model validation. In principle, the smaller the RMSE, the better the reconstruction of the system dynamics. It is used the term out-of-sample error to refer to the RMSE calculated over test/validation samples, while in-sample error is the RMSE for training/estimation samples.

In active sequential learning, the interest is to obtain a model of low complexity and high predictive power. In modeling approaches which deal with building a support vectors dictionary, a score that evaluates the trade-off between the number of items in the dictionary (which must be as small as possible) and the predictive accuracy of the model must be defined. A feasible figure of merit for evaluating this goal is the geometric mean (GM) of the RMSE and the average size of the dictionary (#SVs).

$$GM = \sqrt{RMSE \times \#\text{SVs}}, \qquad (2.28)$$

whose choice is motivated by the fact that we want to minimize both indices, RMSE, and #SVs, simultaneously. Thus, the best model is one providing the minimum GM value.

## 2.4 Challenges in System Identification

System Identification is a mature area that has had a highly productive development. Much has been done, but many problems continue to exist. Ljung (2008) cites some challenges as identification of nonlinear models, convexification of optimization problems, model approximation with acceptable accuracy from data, model complexity reduction, non-Gaussian signal processing, and large databases processing. In the following sections, it is elaborated a bit more on each of these challenging issues.

### *2.4.1 Nonlinear Models*

A nonlinear dynamic model is one in which $\hat{y}_t = f(\boldsymbol{x}_t; \boldsymbol{\theta})$ is nonlinear in data set (but could be any function, including linear, of $\boldsymbol{\theta}$). Identification of nonlinear models is probably the most active area in system identification today (LJUNG; VICINO, 2005). Therefore, the central issues for dynamic systems are listed below (LJUNG, 2008):

- What are the useful parameterizations of $\hat{y}_t$ for nonlinear models of dynamic systems? Ideas and suggestions have proliferated and given rise to a rich collection of models.
- Stability of predictions and simulations: Consider the following simple example. Let a nonlinear model be given by $\hat{y}_t = f(y_{t-1}, u_{t-1}, \boldsymbol{\theta})$ in which the predictions involve two measured inputs and outputs. The simulation of this model for a given input is more tricky:

  $$\hat{y}_t = f(\hat{y}_{t-1}, u_{t-1}, \boldsymbol{\theta}), \quad t = 1, ..., N. \tag{2.29}$$

  This is a nonlinear dynamic system, and to establish for which values of $\boldsymbol{\theta}$ it is stable, is in general very difficult. Here, stability is achieved when the system dynamics is actually captured and predictions and simulations does not diverge from the real system. For control applications, it would be very helpful to find flexible classes of models, for which simulation stability could be tested with reasonable effort.
- How to identify a nonlinear system that operates in a closed-loop and is stabilized by an unknown regulator?
- Develop error models for linear or nonlinear models of nonlinear systems that can be used for robust control design.
- Find effective data-based non-linearity tests for dynamical systems.

### *2.4.2 Handling Non-Gaussian Noise*

A system identification problem involves estimating the coefficients of an unknown model, by studying its response to an input signal. The unknown process is modeled by the adaptive filter in such a way that for the same input, the difference between the outputs of the process and the filter is minimized.

This adaptation is complicated by the presence of measurement noise, erroneous, or unwanted data which gets added to the output of the process and which by themselves are not interesting to the system identification task. Classical approaches to deal with parameter estimation, use cost functions based on first- and second-order statistics, such as autocorrelation and the MSE (HAYKIN, 1996; HASSIBI, 2003; LIU *et al.*, 2010).

Models whose parameters are estimated using low-order statistics consider the noise in data as Gaussian and the effect of linear systems on low order statistics is well known (PAPOULIS, 1991). Under these Gaussianity assumptions, further supported by the central limit theorem, the MSE criterion, would be able to extract all possible information from a signal whose statistics are only defined by its mean and variance.

The Gaussian noise[3] having a *pdf* equal to that of the normal distribution. Thus, the probability density function $p$ of a Gaussian randoms with mean $\mu$ and variance $\sigma^2$ is given by

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(z-\mu)^2}{2\sigma^2}\right\}. \tag{2.30}$$

Although Gaussianity is a suitable assumption to most practical problems, it has become evident when dealing with non-Gaussian perturbations (PRINCIPE, 2010). Practical applications have perturbations with non-Gaussian distributions, e.g. Weibull distribution (WEIBULL, 1951), and Poisson distribution (LAFLEUR *et al.*, 1972), or large outliers that in the estimation data deteriorates considerably the resulting predictors performance.

Therefore, criteria that not only consider the second-order statistics, but that also take into account the higher-order statistical information of the systems and signals, are much desired.

### *2.4.3 Online Learning*

The term online means that the parameters of the model can be updated sample-by-sample, when a new input sample becomes available (LI *et al.*, 2013). Furthermore, the number of computations required per new sample must not increase (and preferably be small) as the number of samples increases (ENGEL *et al.*, 2004). One example of sequential learning application is in the online modeling of a process with a time-varying behavior (MATIAS *et al.*, 2015).

The online learning has also gained increasing attention in recent years from the machine learning community, due to its inherent ability to handle the demands of big data applications (HOI *et al.*, 2014) and also data nonstationarity (DITZLER *et al.*, 2015).

Strongly motivated by the success of the support vector regression (SVR) models in nonlinear regression, the application of the kernel principles (HAYKIN, 2009) to linear adaptive filters has produced powerful online nonlinear extensions of well-known signal processing algorithms (ROJO-ÁLVAREZ *et al.*, 2014), including the LMS (LIU *et al.*, 2008), the normalized LMS (SAIDE *et al.*, 2015), and the RLS (ENGEL *et al.*, 2004; SANTOS; BARRETO, 2017).

[3] A special case is white Gaussian noise (WGN), in which the values at any pair of times are statistically uncorrelated (and, hence, independent) and identically distributed.

These contributions have helped to establish another emerging field, that of KAF for nonlinear signal processing. In essence, the KAF algorithms are online kernel-based methods that aim to recover a signal of interest by adapting their parameters as new data become available.

### *2.4.4 Large-scale Data Sets*

The impact on identification and model building using large-scale data sets cannot be disregarded. More and more, it is possible to have data recording available for decades in a data base. That in itself constitutes an (implicit) model of the process, but it is a formidable task to process this data volume.

Learning algorithms based on LMS deal very well with large data sets but does not perform well for data sets with a small number of samples. To deal with small and large data sets in a one-pass learning regime an alternative is to use solutions based on RKHS learning theory (CORTES; VAPNIK, 1995) that can be very efficient.

Between the methods, the SVM learning paradigm are powerful frameworks, because kernel SVMs provide very powerful classification and regression models that often outperform the classical neural network models. However, at least in their customary Gaussian kernel formulation, their training and prediction costs may be too high for the big data problems currently dominating machine learning. This is so because the number of support vectors underlying any SVM model is usually linear concerning the sample size. This implies that for a size $N$ sample, the training procedure to solve the dual problem of kernel SVMs requires $N$ iterations each with $O(N)$ cost (FAN *et al.*, 2005). As a consequence, the training cost is, at least, $N^2$ (usually higher), even without counting kernel operations. Besides, predicting a single new pattern will also have an $O(N)$ cost and, thus, for big sample size problems prediction costs may be too high.

There is a large ongoing effort to adapt SVM-like models to big data settings. For instance, an important concept behind of SVMs is the nonlinear projection of the original patterns in a new space with a much larger dimension. However, when the original pattern dimension is large enough, kernels may not be needed and one can work with linear SVMs, either solving the primal or dual problems (YUAN *et al.*, 2012). Training becomes much faster and large samples can be more easily handled than in the kernel case, but the bias term has to be dropped to get rid of the equality constraint it imposes on the dual problem. In principle, dropping the offset should hamper the performance of a kernel SVM model, but this may not be always the case (STEINWART *et al.*, 2011). On the other hand, unless the pattern dimension is substantially high (at least in the thousands), the performance of Gaussian SVMs is usually better than the linear ones.

In any case, there is a huge literature on speeding up SVM training. For instance, early attempts to provide SVMs with online training are the well-known NORMA (KIVINEN *et al.*, 2004) and Pegasos (SHALEV-SHWARTZ *et al.*, 2011) procedures. Other recent proposals include decomposing large data sets in appropriate chunks (THOMANN *et al.*, 2016), applying multilevel techniques (SCHLAG *et al.*, 2019), approximating kernel operations using feature randomization (RAHIMI; RECHT, 2008), or applying budget constraints to dual training (QAADAN *et al.*, 2018).

Bearing in mind the system identification and signal processing theories, the next chapter will discuss the application of the correntropy in online kernel models dealing with the presence of outliers in estimation data.

# 3 A NOVEL SPARSE ONLINE CORRENTROPY KERNEL LEARNING MODEL

This chapter introduces a CKL model (LIU; CHEN, 2013) for online system identification in non-Gaussian noise scenarios. The CKL is built upon two modern model building paradigms, namely, the Vapnik's kernel learning (KL) framework (CORTES; VAPNIK, 1995) and the information-theoretic learning (ITL) (SANTAMARIA *et al.*, 2006). The KL framework provides the CKL model with the ability to deal with nonlinear systems via the kernel trick, which transforms data from the original input space to a possibly infinite-dimensional RKHS. The ITL paradigm, through correntropy, in its turn, aims at extracting more information from the data for system identification than 2nd order statistics, such as autocorrelation and the mean square error (MSE) (LIU *et al.*, 2007), and provides outlier-robustness to CKL.

The proposed extension of the CKL, hereafter named online sparse CKL (OS-CKL), improves the CKL in three important aspects. Firstly, the CKL model is modified to operate as a kernel adaptive filter, in which model building and parameter estimation are executed in an online fashion for each new incoming sample. Secondly, a sparsification procedure is used to build a parsimonious model using only a compact set of relevant input samples (the model's dictionary). Last but not least, the computational complexity of the proposed algorithm concerning CKL is reduced by estimating inverse matrices recursively instead of in batch mode.

It carries out a comprehensive evaluation of the proposed model using four benchmarking data sets, two of them are synthetic ones and the other two (CSTR and pH neutralization) are related to the process industry. Different levels of outlier contamination are considered in order to assess the robustness of the proposed model. The obtained results achieved by the proposed OS-CKL model highlight its ability to keep a high predictive power under an online learning regime with a reduced dictionary size in comparison to several state-of-the-art alternatives.

The remainder of this chapter is organized as follows. Section 3.1 briefly reviews the basics of KL and ITL paradigms and how they are combined to give rise to the CKL model. Section 3.2 introduces the online and sparse extension of CKL. Section 3.3 describes the data sets, reports the computer experiments, and discusses the results of the evaluated models. The chapter is concluded in Section 3.4.

## 3.1 Correntropy Kernel Learning

### *3.1.1 The Correntropy Framework*

The correntropy $V$ is a generalized similarity measure between two arbitrary scalar random variables $X$ and $Y$ defined by (LIU *et al.*, 2007) as $V(X, Y) = E[k(X, Y)]$ in which $E[\cdot]$ denotes the expected value and $k(\cdot, \cdot)$ is a positive definite kernel. In practice, the joint pdf of $(X, Y)$ is unknown and only a finite number of data $\{x_t, y_t\}_{t=1}^{N}$ is available, leading to the sample estimator of correntropy:

$$\hat{V}_{N,\sigma}(x, y) = \hat{V}_{N,\sigma}(x - y) = \frac{1}{N}\sum_{t=1}^{N} k_\sigma(x_t - y_t). \tag{3.1}$$

The Gaussian kernel defined in Eq. (2.19), as a symmetric and shift-invariant kernel, is of particular interest to the developments carried out along the current thesis.

As a measure of similarity, the correntropy can be used as an error-based cost function for adaptive systems training using the *maximum correntropy* criterion (MCC):

$$\hat{\boldsymbol{\theta}}_{mcc} = \text{arg max}_{\boldsymbol{\theta}}\sum_{t=1}^{N} k_\sigma(e_t|\boldsymbol{\theta}), \tag{3.2}$$

in which $e_t|\boldsymbol{\theta} = y_t - f(\boldsymbol{x}_t; \boldsymbol{\theta})$, $t = 1, ..., N$, is the prediction error (or residual) produced by an approximating model $f(\boldsymbol{x}_t; \boldsymbol{\theta})$ parameterized by the vector $\boldsymbol{\theta}$.

It can be shown that MCC bears a close relationship with $M$-estimation, as explained in SubSection 2.2.3, as an outlier-robust parameter estimation framework

$$\boldsymbol{\theta}_{mcc} = \text{arg min}_{\boldsymbol{\theta}}\sum_{t=1}^{N} \rho(e_t)e_t^2, \tag{3.3}$$

in which the weighting term $\rho(e_t)$ penalizes large errors, usually caused by outliers. If one chooses $\rho(e_t)$ as

$$\rho(e_t) = \frac{1}{\sqrt{2\pi}\sigma^3}\exp\left\{-\frac{e_t^2}{2\sigma^2}\right\}, \quad t = 1, ..., N, \tag{3.4}$$

the correntropy can be viewed as a robust cost function capable to handle outlying samples because it does not amplify their effects.

### *3.1.2 Kernel Learning Framework*

As shown in SubSection 2.1.2 in Eqs. (2.2)-(2.4) for the system identification task of interest it is assumed the NARX structure, which can generally represent a wide class of discrete-time nonlinear systems (LJUNG, 1999).

When the MCC and the regularized KL framework (CORTES; VAPNIK, 1995) are jointly applied to Eq. (2.4), we arrive at the following optimization problem:

$$\begin{cases} \text{min } J(\boldsymbol{w}, b, \rho) = \frac{\gamma}{2}\sum_{t=1}^{N}\rho(e_t)e_t^2 + \frac{1}{2}||\boldsymbol{w}||^2 \\ \text{s.t. } y_t - \boldsymbol{w}^\top\phi(\boldsymbol{x}_t) - b - e_t = 0, \quad t = 1, ..., N, \end{cases} \tag{3.5}$$

in which $\gamma$ $(\gamma > 0)$ is the regularization parameter determining the trade-off between model's complexity and approximation accuracy. The Lagrangian of the constrained optimization problem of Eq. (3.5) is written as

$$\begin{aligned} L = &\frac{1}{2}(||\boldsymbol{w}||^2 + \gamma\sum_{t=1}^{N}\rho(e_t)e_t^2) \\ &+ \sum_{t=1}^{N}\alpha_t[y_t - \boldsymbol{w}^\top\phi(\boldsymbol{x}_t) - b - e_t], \quad t = 1, ..., N, \end{aligned} \tag{3.6}$$

in which $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_N]^\top$ is the vector of Lagrange multipliers.

The above problem cannot be solved directly because the weighting terms $\rho(e_t)$ depend on the model coefficients $\boldsymbol{\theta} = [\boldsymbol{w}^\top \; b]^\top$. Hence, a two-stage iterative procedure has to be used. In the first stage, the terms $\rho(e_t)$ are fixed, which indicates that a weighted KL problem is formulated. In the second stage, the weighting terms $\rho(e_t)$ are updated using the estimated model coefficients $\boldsymbol{\theta} = [\boldsymbol{w}^\top \;\; b]^\top$ (LIU; CHEN, 2013).

Then, by applying the conditions for optimality to the Lagrangian in (3.6), we get

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{0} \rightarrow \boldsymbol{w} = \sum_{t=1}^{N}\alpha_t\phi(\boldsymbol{x}_t) \tag{3.7}$$

$$\frac{\partial L}{\partial b} = \boldsymbol{0} \rightarrow \sum_{t=1}^{N}\alpha_t = \boldsymbol{0} \tag{3.8}$$

$$\frac{\partial L}{\partial e_t} = \boldsymbol{0} \rightarrow \alpha_t = \gamma\rho(e_t)e_t \tag{3.9}$$

$$\frac{\partial L}{\partial \alpha_t} = \boldsymbol{0} \rightarrow y_t - \boldsymbol{w}^\top\phi(\boldsymbol{x}_t) - b - e_t = 0 \tag{3.10}$$

Inserting Eqs. (3.7)-(3.9) into Eq. (3.10), we can eliminate the variables $\boldsymbol{w}$ and $e_t$, and obtain the following linear system:

$$\begin{bmatrix} \boldsymbol{K} + \boldsymbol{\Omega} & \vdots & \boldsymbol{1} \\ \cdots\cdots & & \cdots \\ \boldsymbol{1}^\top & \vdots & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \\ b \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ \\ 0 \end{bmatrix}, \tag{3.11}$$

in which $\boldsymbol{y} = [y_1, ..., y_N]^\top$ and $\mathbf{1} \in \mathbb{R}^{N\times 1}$ is a vector of ones, $\boldsymbol{K} \in \mathbb{R}^{N\times N}$ is a kernel matrix, whose $(t, j)$ entry is $K_{tj} = \langle \phi(\boldsymbol{x}_t), \phi(\boldsymbol{x}_j)\rangle, \forall t, j = 1, ..., N$. $\boldsymbol{\Omega}$ is a diagonal matrix whose diagonal elements are computed as $\Omega_t = 1/(\gamma\rho(e_t)), t = 1, ..., N$.

Of particular interest to this thesis is the matrix $\boldsymbol{H} = \boldsymbol{K} + \boldsymbol{\Omega}$, whose inverse $\boldsymbol{P} = \boldsymbol{H}^{-1} = (\boldsymbol{K} + \boldsymbol{\Omega})^{-1}$ leads to the solution of the linear system in Eq. (3.11):

$$\begin{cases} \boldsymbol{\alpha} = \boldsymbol{P}\left[\boldsymbol{y} - \frac{\mathbf{1}\mathbf{1}^\top \boldsymbol{P}\boldsymbol{y}}{\mathbf{1}^\top \boldsymbol{y}\mathbf{1}}\right], \\ \\ b = \frac{\mathbf{1}^\top \boldsymbol{P}\boldsymbol{y}}{\mathbf{1}^\top \boldsymbol{y}\mathbf{1}}. \end{cases} \tag{3.12}$$

Finally, the prediction $\hat{y}_t$ for the sample $\boldsymbol{x}_t$ is computed as

$$\hat{y}_t = \sum_{i=1}^{N} \alpha_i k(\boldsymbol{x}_t, \boldsymbol{x}_i) + b = \boldsymbol{\alpha}^\top \boldsymbol{k}_t + b, \tag{3.13}$$

in which $\boldsymbol{k}_t = [k(\boldsymbol{x}_t, \boldsymbol{x}_1) \ \cdots \ k(\boldsymbol{x}_t, \boldsymbol{x}_N)]^\top$ is the kernel vector. From the exposed so far, the following issue is important to be addressed due to its practical implication.

**Remark 3.1**. The solution of the CKL model in (3.12) is a batch learning process since all the samples to time $t$ are used. The prediction for a new input vector also requires the storage of all $N$ training vectors. As a consequence, the dimensions of the matrix $\boldsymbol{\Omega}$ and the vector $\boldsymbol{y}$ increase for each new incoming pair $(\boldsymbol{x}_t, \ y_t)$, thus increasing the predictor complexity considerably as $t \to \infty$.

Since online system identification for nonlinear systems is the task of interest to this thesis, the parameters of the predictor must be modified following the arrival of each new sample. Furthermore, we require that the computational cost per sample does not increase with the size of the data set. In order to limit the increase in the model's complexity, we add a sparsification procedure to the formulation of the CKL model.

## 3.2 The Online Sparse CKL Model

This section is to develop a sparse variant of the CKL model for online system identification in non-Gaussian noise scenarios. The proposed approach, already named OS-CKL model, is based on the amalgamation of three methods: $(i)$ the correntropy criterion (SANTAMARIA *et al.*, 2006), $(ii)$ the kernel learning framework (SCHÖLKOPF; SMOLA, 2002), and $(iii)$ the approximate linear dependency (ALD) sparsification criterion (ENGEL *et al.*, 2004).

To be considered an online method in the sense of adaptive filtering, the dimension of the kernel matrix $\boldsymbol{K}$ cannot increase for each new incoming pair $(\boldsymbol{x}_t, y_t)$. It will increase from time to time, but only when the information in the incoming vector makes it eligible to be incorporated into the model's dictionary. If this occurs, the kernel matrix is also updated recursively.

The general idea is the following. Assuming that at time step $t$, after observing $t-1$ training samples $\{\boldsymbol{x}_i\}_{i=1}^{t-1}$, we have collected a dictionary of $m_t$ $(m_t \ll t)$ linearly independent basis vectors $\{\phi(\tilde{\boldsymbol{x}}_j)\}_{j=1}^{m_t-1}$ comprised of a subset of relevant input samples: $\mathcal{D}_{t-1}^{sv} = \{\tilde{\boldsymbol{x}}_j\}_{j=1}^{m_t-1}$.

For each new sample $\boldsymbol{x}_t$ we verify whether $\phi(\boldsymbol{x}_t)$ is linearly dependent on the current dictionary vectors. If not, we add it to the dictionary and increment $m_t$ by 1. The condition for an input vector $\boldsymbol{x}_t$ *not* to be added to the $dictionary$ is defined by the ALD criterion:

$$\delta_t \overset{\text{def}}{=} \text{min}_{\boldsymbol{a}} \left\| \sum_{m=1}^{m_{t-1}} a_m \phi(\tilde{\boldsymbol{x}}_m) - \phi(\boldsymbol{x}_t) \right\|^2 \leq \upsilon, \tag{3.14}$$

in which $\upsilon$ is the threshold defining the desired level of sparsification. The ALD test requires the estimation of a vector of coefficients $\boldsymbol{a} = (a_1, ..., a_{m_{t-1}})^\top$. For this purpose, we develop the expression of the squared norm $\| \sum_{m=1}^{m_{t-1}} a_m \phi(\tilde{\boldsymbol{x}}_m) - \phi(\boldsymbol{x}_t)\|^2$ and use the kernel trick $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)\rangle$, thus obtaining

$$\delta_t = \text{min}_{\boldsymbol{a}} \{\boldsymbol{a}^\top \tilde{\boldsymbol{K}}_{t-1} \boldsymbol{a} - 2\boldsymbol{a}^\top \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t) + k_{tt}\}, \tag{3.15}$$

in which $\tilde{\boldsymbol{K}}_{t-1} \in \mathbb{R}^{m_{t-1}} \times \mathbb{R}^{m_{t-1}}$ is the kernel matrix calculated with the dictionary samples, $\tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t) \in \mathbb{R}^{m_{t-1}}$ and $k_{tt} = k(\boldsymbol{x}_t, \boldsymbol{x}_t) \in \mathbb{R}$. The $(i,\ j)$ entry of $\tilde{\boldsymbol{K}}_{t-1}$ is computed as $[\tilde{\boldsymbol{K}}_{t-1}]_{i,j} = k(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j)$, while the $i$th component of $\tilde{\boldsymbol{k}}_{t-1}$ is computed as $(\tilde{\boldsymbol{k}}_{t-1})_i = k(\tilde{\boldsymbol{x}}_i, \boldsymbol{x}_t)$, for $i, j = 1, ..., m_{t-1}$.

Solving the expression in (3.15) we get $\boldsymbol{a}_t = \tilde{\boldsymbol{K}}_{t-1}^{-1} \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)$, which leads to the simplified expression of the ALD test:

$$\delta_t = k_{tt} - \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)^\top \boldsymbol{a}_t \leq \upsilon. \tag{3.16}$$

By defining the $t \times m_{t-1}$ matrix $\boldsymbol{A} = [\boldsymbol{a}_1 \,|\, \boldsymbol{a}_2 \,|\, \cdots \,|\, \boldsymbol{a}_t]^\top$ and taking the ALD expression in (3.14) as exact, we can express the full $N \times N$ training set kernel matrix $\boldsymbol{K}$ as $\boldsymbol{A}\tilde{\boldsymbol{K}}\boldsymbol{A}^\top$. For $\upsilon$ sufficiently small, $\boldsymbol{A}\tilde{\boldsymbol{K}}\boldsymbol{A}^\top$ is a good approximation of $\boldsymbol{K}$, and from this point onwards we assume that indeed $\boldsymbol{K} = \boldsymbol{A}\tilde{\boldsymbol{K}}\boldsymbol{A}^\top$ (ENGEL *et al.*, 2002).

As already mentioned, we need an algorithm whose time and memory requirements are independent on the number of training samples $N$ that, for the online case, equals the time index $t$. For the algorithm to be dependent only on the size of the dictionary $m_t$ we define a new set of variables: $\tilde{\boldsymbol{\alpha}} = \boldsymbol{A}^\top \boldsymbol{\alpha}$, in which it should be observed that $\tilde{\boldsymbol{\alpha}} \in \mathbb{R}^{m_t}$, while $\boldsymbol{\alpha} \in \mathbb{R}^N$, and typically $m_t \ll N$.

In the online case, $\boldsymbol{A}$ becomes $\boldsymbol{A}_t$ which is a growing $t \times m_t$ matrix to which the row vector $\boldsymbol{a}_t^\top$ is appended at each time step $t$. $\boldsymbol{y}_t$ is an increasingly large $t$-dimensional vector. Luckily, we need only to maintain and update its $m_t$-dimensional image under the transformation $\boldsymbol{A}_t^\top : \boldsymbol{A}_t^\top \boldsymbol{y}_t$.

From the exposed so far, it is possible to rewrite the linear system in Eq. (3.11) as

$$\begin{bmatrix} \tilde{\boldsymbol{K}} + \tilde{\boldsymbol{\Omega}} & \vdots & \boldsymbol{1} \\ \cdots\cdots & \cdots & \cdots \\ \boldsymbol{1}^\top & \vdots & 0 \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\alpha}} \\ \\ b \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_t^\top \boldsymbol{y}_t \\ \\ 0 \end{bmatrix}, \tag{3.17}$$

in which this system is equal to the one in Eq. (3.11) only when $\upsilon = 0$. Note that $\tilde{\boldsymbol{K}} + \tilde{\boldsymbol{\Omega}}$ is always full rank and is therefore invertible. We will assume $\tilde{\boldsymbol{H}} = \tilde{\boldsymbol{K}} + \tilde{\boldsymbol{\Omega}}$ and then its inverse can be computed as

$$\tilde{\boldsymbol{P}} = \tilde{\boldsymbol{H}}^{-1} = (\tilde{\boldsymbol{K}} + \tilde{\boldsymbol{\Omega}})^{-1}. \tag{3.18}$$

It should be noticed that now the dimensions of the components of Eq. (3.17) are given in terms of the dictionary size; thus, $\boldsymbol{1} \in \mathbb{R}^{m_t \times 1}$ is a vector of ones, $\tilde{\boldsymbol{K}} \in \mathbb{R}^{m_t \times m_t}$ is a kernel matrix whose entries are given by $\tilde{K}_{ij} = \langle \phi(\tilde{\boldsymbol{x}}_i), \phi(\tilde{\boldsymbol{x}}_j) \rangle, \forall\, i, j = 1, ..., m_t$ and $\tilde{\boldsymbol{\Omega}}$ is a diagonal matrix whose diagonal entries are computed as $\tilde{\Omega}_i = 1/(\gamma \rho(e_i)), i = 1, ..., m_t$.

If it is not necessary to update the dictionary we get $\delta < \upsilon$, i.e. the projection of the current input vector can be approximately written as a linear combination of the projections of the vectors in the dictionary. Thus, $\boldsymbol{x}_t$ is not added to the dictionary and the kernel matrix is not changed: $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv}$, and $\tilde{\boldsymbol{K}}_t = \tilde{\boldsymbol{K}}_{t-1}$. In this case, we compute only $\rho(e_i)$ for the sake of detecting outliers.

### 3.2.1 *Updating the Dictionary*

In this case, we get $\delta > \upsilon$, i.e., the projection of the current input vector cannot be written as a linear combination of the projections of the support vectors in the dictionary. This implies that $\boldsymbol{x}_t$ must be added to the dictionary, i.e., $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \bigcup \{\boldsymbol{x}_t\}$ and $m_t = m_{t-1} + 1$. As a consequence of this inclusion, the kernel matrix must be updated accordingly.

However, the computational load is $\mathcal{O}(m_t^3)$ for computing the inverse of the matrix, so we want to reduce this burden by computing it recursively. The challenge here is to compute $\tilde{\boldsymbol{K}}_t$ and $\tilde{\boldsymbol{P}}_t$ (and hence $\tilde{\boldsymbol{K}}_t^{-1}$) recursively using $\tilde{\boldsymbol{K}}_{t-1}$, $\tilde{\boldsymbol{P}}_{t-1}$ and the information provided by the new sample. For this purpose, we compute firstly the matrices $\tilde{\boldsymbol{K}}_t$ and $\tilde{\boldsymbol{K}}_t^{-1}$ as

$$\tilde{\boldsymbol{K}}_t = \begin{bmatrix} \tilde{\boldsymbol{K}}_{t-1} & \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t) \\ \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)^\top & k_{tt} \end{bmatrix} \tag{3.19}$$

and

$$\tilde{\boldsymbol{K}}_t^{-1} = \frac{1}{\delta_t} \begin{bmatrix} \delta_t \tilde{\boldsymbol{K}}_{t-1}^{-1} + \boldsymbol{a}_t \boldsymbol{a}_t^\top & -\boldsymbol{a}_t \\ -\boldsymbol{a}_t^\top & 1 \end{bmatrix} \tag{3.20}$$

If we took the instantaneous version of Eq. (3.18) as $\tilde{\boldsymbol{P}}_t = (\tilde{\boldsymbol{K}}_t + \tilde{\boldsymbol{\Omega}}_t)^{-1}$, we can compute the matrix $\tilde{\boldsymbol{P}}_t$ as

$$\tilde{\boldsymbol{P}}_t = \begin{bmatrix} \tilde{\boldsymbol{K}}_{t-1} + \tilde{\boldsymbol{\Omega}}_{t-1} & \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t) \\ \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)^\top & k_{tt} + \Omega_t \end{bmatrix}^{-1}. \tag{3.21}$$

It is somewhat easy to observe that

$$\tilde{\boldsymbol{P}}_t^{-1} = \begin{bmatrix} \tilde{\boldsymbol{P}}_{t-1}^{-1} & \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t) \\ \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)^\top & k_{tt} + \Omega_t \end{bmatrix}. \tag{3.22}$$

Using the block matrix inversion identity, we obtain the following expression:

$$\tilde{\boldsymbol{P}}_t = r_t^{-1} \begin{bmatrix} \tilde{\boldsymbol{P}}_{t-1} r_t + \boldsymbol{z}_t \boldsymbol{z}_t^\top & -\boldsymbol{z}_t \\ -\boldsymbol{z}_t^\top & 1 \end{bmatrix}, \tag{3.23}$$

in which $\boldsymbol{z}_t = \tilde{\boldsymbol{P}}_{t-1} \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)$ and

$$r_t = \Omega_t + k_{tt} - \boldsymbol{z}_t^\top \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t) \tag{3.24}$$

Using this approach the matrix $\tilde{\boldsymbol{P}}$ can be updated recursively and the solution of the model is given by

$$\begin{cases} \tilde{\boldsymbol{\alpha}}_t = \tilde{\boldsymbol{P}}_t \left[ \boldsymbol{A}_t^\top \boldsymbol{y}_t - \frac{\boldsymbol{1}\boldsymbol{1}^\top \tilde{\boldsymbol{P}}_t \boldsymbol{A}_t^\top \boldsymbol{y}_t}{\boldsymbol{1}^\top \boldsymbol{A}_t^\top \boldsymbol{y}_t \boldsymbol{1}} \right], \\ \\ b_t = \frac{\boldsymbol{1}^\top \tilde{\boldsymbol{P}}_t \boldsymbol{A}_t^\top \boldsymbol{y}_t}{\boldsymbol{1}^\top \boldsymbol{A}_t^\top \boldsymbol{y}_t \boldsymbol{1}}. \end{cases} \tag{3.25}$$

**Remark 3.2** - By comparing the solution of the CKL algorithm in Eq. (3.12) with the variant proposed in Eq. (3.25), one can easily notice that the parameters of the proposed OS-CKL are updated for each new sample (note the presence of the subscript $t$), while the computational cost per sample does not increase with the size of the data set.

In order to add robustness to outliers to OS-CKL, we follow an approach to iteratively reweigh and update the terms $\rho(e_t)$ using the estimated parameters $\tilde{\boldsymbol{\theta}}_t = [\tilde{\boldsymbol{\alpha}}_t^\top, b_t]^\top$ (LIU; CHEN, 2013). For that purpose, the terms $\rho(e_t)$ are firstly fixed with an initial value equal to one. Then, the parameters $\tilde{\boldsymbol{\theta}}_t = [\tilde{\boldsymbol{\alpha}}_t^\top, b_t]^\top$ are estimated using Eq. (3.25). The next step is to obtain the predicted values:

$$e_t = y_t - \hat{y}_t = y_t - \tilde{\boldsymbol{\alpha}}_t^\top \tilde{\boldsymbol{k}}_t - b_t, \tag{3.26}$$

in which $\tilde{\boldsymbol{k}}_t = [\tilde{k}_{t,1}, ..., \tilde{k}_{t,m_t}]^\top \in \mathbb{R}^{m_t \times 1}$ is a kernel vector with its element $\tilde{k}_{t,j} = \langle \phi(\boldsymbol{x}_t), \phi(\tilde{\boldsymbol{x}}_j) \rangle, \forall$ $j = 1, ..., m_t$.

Then, a new set of values for the weighted terms can be computed using Eq. (3.4), so that $\tilde{\boldsymbol{\Omega}}$ can update its diagonal elements $\tilde{\boldsymbol{\Omega}}_t = 1/(\gamma\rho(e_t))$, $t = 1, \ldots, m_t$. Finally, the new values of the parameters $\tilde{\boldsymbol{\theta}}_t = [\tilde{\boldsymbol{\alpha}}_t^\top, b_t]^\top$ are re-estimated using Eq. (3.25).

Since only the diagonal elements of $\tilde{\boldsymbol{\Omega}}$ are changed and the others elements in $\tilde{\boldsymbol{H}}$ and $\tilde{\boldsymbol{P}}$ are always kept unchanged, we use the Sherman-Morrison-Woodbury formula (GOLUB; LOAN, 2012) to update $\tilde{\boldsymbol{P}}$ via the following expression:

$$\tilde{\boldsymbol{P}}_j = \tilde{\boldsymbol{P}}_{j-1} - \frac{\tilde{\boldsymbol{P}}_{j-1} \boldsymbol{v}_j \boldsymbol{s}_j^\top \tilde{\boldsymbol{P}}_{j-1}}{1 + \boldsymbol{s}_j^\top \tilde{\boldsymbol{P}}_{j-1} \boldsymbol{v}_j}, \tag{3.27}$$

in which $\tilde{\boldsymbol{P}}_0 = \tilde{\boldsymbol{P}}_t$ and $j$ corresponds to new incoming sample $\boldsymbol{x}_j$ and $j = 1, ..., m_t$. The vectors $\boldsymbol{v}_j \in \mathbb{R}^{m_t \times 1}$ and $\boldsymbol{s}_j \in \mathbb{R}^{m_t \times 1}$ are given by

$$\boldsymbol{v}_j = \begin{bmatrix} 0 & \cdots & \frac{1}{\gamma} & \cdots & 0 \end{bmatrix}^\top, \tag{3.28}$$

and

$$\boldsymbol{s}_j = \begin{bmatrix} 0 & \cdots & \eta_{d_j} & \cdots & 0 \end{bmatrix}, \tag{3.29}$$

in which $\boldsymbol{d} = [d_1, ..., d_{m_t}]^\top \in \mathbb{R}^{m_t \times 1}$ is a vector with its elements $d_j = t$, $\forall j = 1, \ldots, m_t$ and $\eta_{d_j} = (1/\rho(e_{d_j})) - (1/\rho_{last})$, with $\rho_{last}$ defined as the previous $\rho(e_{d_j})$.

We need to determine if the difference between the last $\tilde{\boldsymbol{\Omega}}$ and the new provide new information for $\tilde{\boldsymbol{P}}$:

$$|(1/\rho(e_{d_j})) - (1/\rho_{last})| > \epsilon_{update}, \tag{3.30}$$

**Algorithm 1:** Pseudocode of the OS-CKL model.

```
Parameter: $\upsilon, \sigma, \gamma, \epsilon_{update}$;
Initialize: $\tilde{\boldsymbol{K}}_1 = k_{11}; \tilde{\boldsymbol{K}}_1^{-1} = 1/\tilde{\boldsymbol{K}}_1; \rho(e_1) = 1; \boldsymbol{A}_1 = 1; m_1 = 1; \boldsymbol{d}_1 = 1;$
$\tilde{\boldsymbol{\Omega}}_1 = 1/(\gamma\rho(e_1)); \tilde{\boldsymbol{H}}_1 = \tilde{\boldsymbol{K}}_1 + \tilde{\boldsymbol{\Omega}}_1; \tilde{\boldsymbol{P}}_1 = 1/\tilde{\boldsymbol{H}}_1;$
Compute $\tilde{\boldsymbol{\alpha}}_1$ and $b_1$ from Eq. (3.25);
Compute $\rho(e_1)$ from Eq. (3.4);
for $t = 2 : N$,
  Get new sample: $(\boldsymbol{x}_t, y_t)$
  Compute $\tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)$
  ALD test:
  $\boldsymbol{a}_t = \tilde{\boldsymbol{K}}_{t-1}^{-1}\tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)$;
  $\delta_t = k_{tt} - \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)^\top \boldsymbol{a}_t$;
  $e_t = y_t - \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} - b_{t-1}$;
  if $\delta_t > \upsilon$
    $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \bigcup \{\boldsymbol{x}_t\}$; % add $\boldsymbol{x}_t$ to dictionary
    $\boldsymbol{z}_t = \tilde{\boldsymbol{P}}_{t-1}\tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)$;
    $\rho(e_t) = 1$;
    $\tilde{\boldsymbol{\Omega}}_t = 1/(\gamma\rho(e_t))$;
    $r_t = \Omega_t + k_{tt} - \boldsymbol{z}_t^\top \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)$;
    Compute $\tilde{\boldsymbol{K}}_t^{-1}$ from Eq. (3.20);
    Compute $\tilde{\boldsymbol{P}}_t$ from Eq. (3.23);
    Compute $\tilde{\boldsymbol{\alpha}}_t$ and $b_t$ from Eq. (3.25);
    $m_t = m_{t-1} + 1; \boldsymbol{d}_{m_t} = t$;
    Iteratively re-weighting:
    for $j = 1 : m_t$,
      $\rho_{last} = \rho(e_{d_j})$;
      $e_{d_j} = y_{d_j} - \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_{d_j})^\top \tilde{\boldsymbol{\alpha}}_t - b_t$;
      Update $\rho(e_{d_j})$ from Eq. (3.4);
      if $|((1/\rho(e_{d_j})) - (1/\rho_{last})| > \epsilon_{update}$
        Update $\tilde{\boldsymbol{P}}_j$ from Eq. (3.27);
      end if
    end for
    Update $\tilde{\boldsymbol{\alpha}}_t$ and $b_t$ from Eq. (3.25);
  else
    $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv}$; % unchanged dictionary
    $e_t = y_t - \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} - b_{t-1}$;
    Compute $\rho(e_t)$ from Eq. (3.4);
  end if
end for
Output: $\tilde{\boldsymbol{\alpha}}_t, b_t, \rho(e_t), \mathcal{D}_t^{sv}$.
```

in which $\epsilon_{update} > 0$ is a user-defined small value. It is not necessary to update $\tilde{\boldsymbol{P}}$ if the difference is too small. In Algorithm 1 we provide the pseudocode of the proposed OS-CKL model. In this table we use ";" to denote vertical concatenation.

Table 2 – Summary of the data sets.

| data set | Important features of the selected data sets | | | |
|---|---|---|---|---|
| | Estimation | Test | $L_u$ | $L_y$ |
| Artificial 1 | 200 | 200 | 3 | 2 |
| Artificial 2 | 300 | 100 | 1 | 1 |
| CSTR | 400 | 600 | 5 | 5 |
| pH data | 400 | 600 | 5 | 5 |

## 3.3 Results and Discussion

In this section, the results of comprehensive computer experiments comparing the proposed OS-CKL with the original CKL are reported. Also, are included in the comparison, other state-of-art kernel adaptive filtering methods, namely, the KRLS (ENGEL *et al.*, 2004), the robust KRLS (ROB-KRLS) (SANTOS; BARRETO, 2017) and the KRMC (WU *et al.*, 2015) models. The experiments involve benchmarking data sets. Two of them are synthetic ones, while the other two are related to the process industry, being available from the DaISy repository (MOOR, 2019). The main features and corresponding characteristics of the data sets are summarized in Table 2.

The 10-fold cross-validation strategy is used to set the hyperparameters, e.g., $v$ (used in the ALD test criterion by OS-CKL, KRLS, and ROB-KRLS), and $\delta_1$, $\delta_2$ (used by the sparsification method of the KRMC model). Others hyperparameters are $\sigma$ (used by the Gaussian kernel by OS-CKL, CKL, and KRMC), $\epsilon_{update}$ (used in the matrix update criterion by OS-CKL and CKL model), $\sigma_2$ (used by KRMC models to compute the error weights), and the regularization factor $\gamma$ (for the OS-CKL, CKL, KRLS, ROB-KRLS, and KRMC model) that are determined by grid searching for their optimal values. The figure of merit for evaluating the numerical performance of the OS-CKL is the root mean square error (RMSE) and the degree of sparsity computed for test samples over 40 independent runs. During the test phase, all models are required to predict out values under the *free simulation* regime, in which predicted output values are fed back to the input regression vector. This is a much more demanding task to tackle than the usual OSA prediction task.

### 3.3.1 Results on Synthetic Data Sets

The input-output time series for the *Artificial 1* data set is generated according to the following dynamical discrete-time system (NARENDRA; PARTHASARATHY, 1990):

$$\begin{cases} y_{k+1} = \frac{y_k y_{k-1} y_{k-2}(y_{k-2}-1)u_{k-1}+u_k}{1+y_{k-2}^2+y_{k-1}^2} \\ u_k = \begin{cases} \sin(\pi k/125), k \leq 203 \\ 0.8\sin(\pi k/125) + 0.2\sin(2\pi k/25), k \geq 203 \end{cases} \end{cases} \tag{3.31}$$

The outliers were generated as impulsive Gaussian noise with *pdf* given by the mixture model $p_z(z) = (1-q) \times \mathcal{N}(0, 0.02) + q \times \mathcal{N}(0.2, \sigma(y))$ in which $q$ is determined according to the percentage of outliers in the scenario and $\sigma(y)$ is the standard deviation of the original training data.

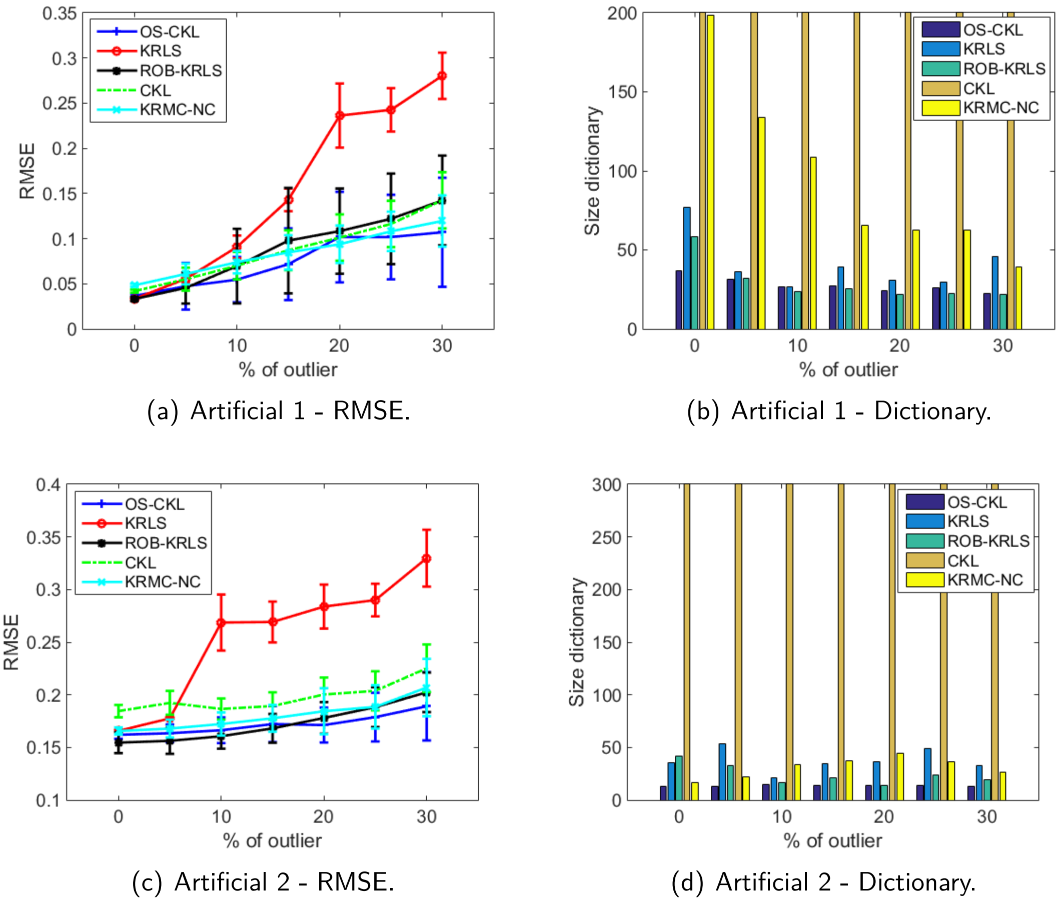
The *Artificial 2* data set (NARENDRA; PARTHASARATHY, 1990) is described as

$$\begin{cases} y_k = \frac{y_{k-1}}{1+y_{k-1}^2} + u_k^3 \\ u_k = \begin{cases} U(-2,2), \text{for training data} \\ \sin(2\pi k/25) + \sin(2\pi k/10), \text{for test data} \end{cases} \end{cases} \tag{3.32}$$

in which $U(-2,2)$ denotes a uniformly distributed random number in the specified range. The outliers were generated as impulsive Gaussian noise with pdf $p_z(z) = (1-q) \times \mathcal{N}(0, 0.08) + q \times \mathcal{N}(0.5, \sigma(y))$ in which $q$ is determined according to the percentage of outliers in the scenario.

It is presented in Figs. 7(a) and 7(c) the averaged RMSE curves and corresponding error bars ($\pm$ one standard deviation) for the synthetic data sets for increasing levels of outlier contamination. Most of the evaluated models achieved equivalent performances, with a slight advantage for the proposed OS-CKL model. The exception is the KRLS model which presented the poorest performance for levels of contamination higher than 10%. This was somewhat expected since the KRLS, despite being a sparse model, is not an outlier-robust one. Fig. 7(c) shows the RMSE curves produced by the evaluated models for the Artificial 2 data set, from which equivalent conclusions can be drawn. In Figs. 7(b) and 7(d) we show the bar plots of the resulting dictionary sizes averaged for the 40 independent runs for increasing levels of outlier contamination. The dictionary size produced by the evaluated models for the Artificial 1 data set is shown in Fig. 7(b) in which we observe an equivalence in performance for the OS-CKL, KRLS, and ROB-KRLS models. Since the original CKL is not an adaptive method it makes use of all the training data. For both figures, the models using the ALD sparsification

Figure 7 – *(a) and (c) are line charts for the RMSE values related to the free simulation on test data with different levels of contamination by outliers. (b) and (d) are the bar plots of the average size of the dictionary.*



Source: prepared by the author (2021).

method (OS-CKL, KRLS, and ROB-KRLS) have achieved much smaller dictionary sizes when compared to the KRMC, which uses *novelty detection* criterion. The proposed OS-CKL and the ROB-KRLS achieved the smallest dictionary sizes among all the evaluated models.

### 3.3.2 Results on Process Industry Related Data Sets

The final experiments involve simulated data of a pH neutralization process in a constant volume stirring tank and a continuous stirring tank reactor (CSTR). An outlier-free scenario and a 30%-level outlier-contaminated scenario were considered. Outliers were sampled randomly as impulsive noise from $p_z(z) = \mathcal{N}(0.1, \sigma(y))$ and $p_z(z) = \mathcal{N}(0.2, \sigma(y))$ to pH and CSTR, respectively. No Gaussian noise was added to the data.

Tables 3 and 4 show the results of the free simulation task on test data for the two scenarios. The tables contain the values of the performance indices RMSE and the average

Table 3 – Test results for the CSTR data set.

| Method | Without outliers | | With 30% outliers | |
|---|---|---|---|---|
| | RMSE | #SV | RMSE | #SV |
| OS-CKL | **4.82E-2±4.90E-5** | **155** | **6.75E-2±1.24E-4** | **108** |
| ROB-KRLS | 5.91E-2±1.21E-5 | 400 | 1.11E-1±1.60E-3 | 153 |
| KRMC | 4.80E-2±1.20E-5 | 385 | 8.14E-2±1.07E-4 | 231 |

Table 4 – Test results for the pH data set.

| Method | Without outliers | | With 30% outliers | |
|---|---|---|---|---|
| | RMSE | #SV | RMSE | #SV |
| OS-CKL | **1.97E-1±7.11E-6** | 104 | **2.77E-1±7.50E-3** | 104 |
| ROB-KRLS | 5.72E-1±3.16E-6 | 91 | 5.50E-1±1.12E-2 | **95** |
| KRMC | 2.90E-1±5.05E-6 | **90** | 2.79E-1±5.70E-3 | 155 |

dictionary size (#SV). An analysis of the results reveals that the best performance in terms of RMSE and degree of sparseness is achieved for both scenarios by the proposed OS-CKL model. This is a quite remarkable result, since we were able to improve the CKL model by reducing its complexity and widening its applicability to online prediction tasks while keeping its predictive power.

## 3.4 Conclusions

This chapter addressed the task of online nonlinear system identification in the presence of outliers by introducing the OS-CKL model, which is motivated by recent advances in ITL (e.g. correntropy) and kernel adaptive filtering methods. The OS-CKL improves the original CKL in important dimensions since it can now be efficiently used for online learning and its model complexity is considerably reduced by the introduction of a sparsification procedure. It is evaluated the proposed model using four benchmarking data sets, two synthetic and two related to process industry (CSTR and pH neutralization), for different levels of outlier contamination. The obtained results revealed that the OS-CKL keeps presenting a high predictive power even under an online learning regime with a reduced dictionary size in comparison to several state-of-the-art alternatives, such as the KRLS and KRMC models.

The next chapter will introduce a variant of the OS-CKL model designed to have a fully adaptive dictionary that grows and shrinks only when it is necessary, according to the dynamical evolution of the system.

# 4 FULLY ADAPTIVE DICTIONARY USING PROXIMAL METHODS

The kernel principles applied to linear adaptive filters have produced powerful nonlinear extensions of these algorithms, which include the KLMS (LIU *et al.*, 2008), KRLS (ENGEL *et al.*, 2004), extended KRLS (EX-KRLS) (FAÇANHA *et al.*, 2020; LIU *et al.*, 2009b), and kernel affine projection algorithms (KAPA) (LIU; PRINCIPE, 2008). However, the resulting adaptive filters are not online algorithms in the strict sense of adaptive filtering theory; that is, a model that updates its parameters sample-by-sample without increasing the complexity of the model. In an online algorithm, the model-building is a continuous process and executed on the fly for each new incoming sample, and the adaptive update mechanism can dynamically track the real-time weight, allowing the modeling of non-stationary signals (ZHONG *et al.*, 2020).

In order to tackle this issue, kernel adaptive algorithms use sparsification mechanisms (HONEINE, 2015), to avoid storing all the input samples and rebuild the predictor every time a new incoming sample appears. Just a few relevant ones need to be stored, comprising a dictionary of support vectors. The model is then built using only the support vectors of the dictionary, thus resulting in a sparse model as time passes by. There are several sparsity-inducing criteria and methods available in the related literature, such as the approximate linear dependency (ALD) (ENGEL *et al.*, 2004), the novelty criterion (PLATT, 1991), the coherence criterion (RICHARD *et al.*, 2009), the Hessian matrix (FAN; SONG, 2013), instantaneous mutual information (FAN *et al.*, 2014), the Nyström approximation (WANG *et al.*, 2019), the vector quantization method (SHEN *et al.*, 2020), and the surprise criterion (LIU *et al.*, 2009a), which can be used either alone or in combination (LI; PRÍNCIPE, 2018). Despite their algorithmic differences, a common feature of them relies on the fact that they always grow a dictionary; that is, items are always added to the pool of relevant samples, never discarded. This is an issue that will be tackled in this chapter by the introduction of a novel online technique based on proximal methods to remove items from the dictionary. This feature of the proposed algorithm aims at building a fully adaptive dictionary, capable of dealing more efficiently with nonstationary scenarios and of keeping the size of the dictionary under strict control by discarding items that are not necessary anymore.

More recently, strategies to delete items from the dictionary have been proposed to avoid excessive growth of the dictionary size, an inherent feature of online kernel-based models (PRINCIPE, 2010; GAO *et al.*, 2013; CHEN *et al.*, 2016). The proposed methods use a combined strategy of growing and pruning the dictionary aiming at better tracking changes in system dynamics over time. In general, these pruning methods discard either the oldest items in every time instant or the least significant ones according to a more elaborated decision mechanism. The main issue here is to correctly rebuild the kernel matrix and update the parameters recursively without compromising the accuracy of the predictor. While this is relatively simple for KLMS-based models (GAO *et al.*, 2013; CHEN *et al.*, 2016), it is not for KRLS-based ones (PRINCIPE, 2010).

Another issue with enormous implications for real-world applications of adaptive filters is the robustness to outliers. Traditionally, the use of mean square error (MSE) criterion in adaptive filtering is optimal under the Gaussian assumption. However, when the signals are non-Gaussian, the performance will deteriorate seriously, especially when the system is disturbed by a heavy-tailed impulsive (CHEN *et al.*, 2017). Despite the importance of this topic, just a few strategies for designing outlier-robust kernel adaptive filters have emerged, in recent years (LIU *et al.*, 2009a; LIU; CHEN, 2013; LIU; CHEN, 2014; WU *et al.*, 2015; SANTOS; BARRETO, 2017; SANTOS; BARRETO, 2018; GUIMARÃES *et al.*, 2018). In this regard, the correntropy function has offered a suitable alternative for the development of outlier-robust kernel adaptive filters. As mentioned in the previous chapters, correntropy is built upon the framework of information-theoretic learning (PRINCIPE, 2010) and aims at replacing conventional 2nd-order cost functions, such as the famed mean squared error function, to capture higher-order statistics from stochastic dynamic systems. Correntropy-based kernel models offer significant performance improvements, especially when data contain large outliers or are disturbed by impulsive noise (SHEN *et al.*, 2020; WANG *et al.*, 2019; GUIMARÃES *et al.*, 2018; CHEN *et al.*, 2017).

From the exposed, this chapter introduces an outlier-robust correntropy-based kernel adaptive filter for online learning. The proposed model is designed to have a fully adaptive dictionary that grows and shrinks only when it is necessary, according to the dynamical evolution of the system. The pruning approach is based on proximal optimization methods, and a novel strategy to recompute the inverse of the kernel matrix recursively after pruning is developed. The chosen application is the dynamic system identification using two large-scale benchmarking data sets (Silverbox and Wiener-Hammerstein). Through comprehensive performance evaluation, we can highlight the more advantageous performance of the model over powerful alternatives.

The remainder of this chapter is organized as follows. In Section 4.1, it is introduced a sparse online variant of it. In Section 4.2, we present the theory behind the proposed model, discussing in details the pruning strategy based on proximal methods. Section 4.3, describes the data sets, reports the computer experiments and discusses the results of the evaluated models. The chapter is concluded in Section 4.4.

## 4.1 The Online Sparse CKL Model

For the task of interest, we assume again that the nonlinear system that produces the data has an underlying dynamical structure given by the already described NARX model (LJUNG, 1999) as described in Eq. (2.1).

For this task, the input vector $\boldsymbol{x}_t$ is obtained by concatenating $L_y$ past observed outputs $y_t \in \mathbb{R}$ and $L_u$ past inputs $u_t \in \mathbb{R}$ into a single regression vector $\boldsymbol{x}_t = [y_{t-1}, \ldots, y_{t-L_y}, u_{t-1}, \ldots, u_{t-L_u}]^\top$. Thus, let us assume that data samples come sequentially as a stream of training examples

$$\mathcal{D}_t = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_t, y_t), \ldots, (\boldsymbol{x}_N, y_N)\}, \tag{4.1}$$

in which $(\boldsymbol{x}_t, y_t) \in \mathbb{R}^{L_u+L_y} \times \mathbb{R}$ denotes the current input-output pair.

The online nature of the system identification task of interest demands a learning model whose complexity does not increase for each new input-output pair $(\boldsymbol{x}_t, y_t)$. In other words, the storage of each new input sample is not necessary, and parameter updating is carried out incrementally only when it is required. Thus, we do not need to solve the linear system in Eq. (3.11) for every new input-output pair $(\boldsymbol{x}_t, y_t)$, but rather to adjust the parameters incrementally using the previously estimated parameters in a recursive way only when relevant input samples are met. Such relevant data samples are the ones to be stored, comprising a dictionary of support vectors. The model is then built using only the support vectors of the dictionary, thus resulting in a sparse model as time passes by.

As mentioned in the introduction, there are many sparsity-promoting criteria, which can be roughly divided into two groups (BARTO *et al.*, 2013). One group is comprised of those based on a *surprising* mechanism (e.g., ALD and surprise), while the other group uses a *novelty detection* mechanism (e.g., coherence and novelty). Despite some differences in their definitions, all these sparsity-inducing criteria have essentially the same goal, that of sequentially building a dictionary for the model by detecting how relevant an incoming sample is. The performance of the proposed algorithm for different sparsity-promoting criteria is evaluated and the corresponding results are discussed in depth in SubSection 4.3.2. For now, it suffices to say that they can be equally used to build a growing dictionary.

For that purpose, at time step $t$ $(2 \leq t \leq N)$, after having observed $t-1$ training samples $\{\boldsymbol{x}_i\}_{i=1}^{t-1}$, a dictionary of support vectors should contain a subset of $m_t - 1$ relevant input samples, $\mathcal{D}_{t-1}^{sv} = \{\widetilde{\boldsymbol{x}}_j\}_{j=1}^{m_t-1}$, with $m_t \ll N$. The symbol $\sim$ is used to mark the vectors in the dictionary.

For each new sample $\boldsymbol{x}_t$, we should verify whether $\phi(\boldsymbol{x}_t)$ satisfies the chosen sparsity-inducing criterion using the current dictionary vectors. If the criterion is met, we add it to the dictionary and increment $m_t$ by 1. The first vector in the dictionary is always the first vector in the sequence.

The immediate practical consequence of the use of sparsity-inducing mechanisms is that the full-sized kernel matrix $\boldsymbol{K} \in \mathbb{R}^N \times \mathbb{R}^N$ is replaced with the one built using the $m_t$ SVs currently in the dictionary, i.e. $\widetilde{\boldsymbol{K}}_t \in \mathbb{R}^{m_t} \times \mathbb{R}^{m_t}$. Thus, the linear system in (3.11) can be rewritten as

$$\begin{bmatrix} \widetilde{\boldsymbol{K}}_t + \widetilde{\boldsymbol{\Omega}}_t & \vdots & \mathbf{1} \\ \cdots\cdots & \cdots & \cdots \\ \mathbf{1}^\top & \vdots & 0 \end{bmatrix} \begin{bmatrix} \widetilde{\boldsymbol{\alpha}}_t \\ \\ b_t \end{bmatrix} = \begin{bmatrix} \boldsymbol{y_d} \\ \\ 0 \end{bmatrix}, \tag{4.2}$$

in which $\mathbf{1} \in \mathbb{R}^{m_t \times 1}$ is a vector of ones, $\widetilde{\boldsymbol{K}}_t \in \mathbb{R}^{m_t \times m_t}$ is a kernel matrix whose entry $(t, j)$, $\widetilde{K}_{tj} = \phi^\top(\widetilde{\boldsymbol{x}}_t)\phi(\widetilde{\boldsymbol{x}}_j) = k(\widetilde{\boldsymbol{x}}_t, \widetilde{\boldsymbol{x}}_j)$, $\forall t, j = 1, \ldots, m_t$ is computed using a suitable kernel function, and $\widetilde{\boldsymbol{\Omega}}$ is a diagonal matrix whose diagonal element $\widetilde{\Omega}_i = 1/(\gamma\rho(e_{d_j})), j = 1, \ldots, m_t$. The outputs vector[1] $\boldsymbol{y_d}$ is comprised of $m_t$ outputs occurring at time instants indicated by the index vector $\boldsymbol{d} = [d_1 \ \cdots \ d_{m_t}]^\top \in \mathbb{R}^{m_t \times 1}$ whose elements are $d_j = t$, $\forall j = 1, \ldots, m_t$.

Since $\widetilde{\boldsymbol{H}}_t = \widetilde{\boldsymbol{K}}_t + \widetilde{\boldsymbol{\Omega}}_t$ is always full rank, its inverse can be computed as

$$\widetilde{\boldsymbol{P}}_t = \widetilde{\boldsymbol{H}}_t^{-1} = (\widetilde{\boldsymbol{K}}_t + \widetilde{\boldsymbol{\Omega}}_t)^{-1}, \tag{4.3}$$

and the solution of the linear system in (4.2) is given by

$$\begin{cases} \widetilde{\boldsymbol{\alpha}}_t = \widetilde{\boldsymbol{P}}_t \left[ \boldsymbol{y_d} - \dfrac{\mathbf{1}\mathbf{1}^\top \widetilde{\boldsymbol{P}}_t \boldsymbol{y_d}}{\mathbf{1}^\top \boldsymbol{y_d} \mathbf{1}} \right], \\ \\ b_t = \dfrac{\mathbf{1}^\top \widetilde{\boldsymbol{P}}_t \boldsymbol{y_d}}{\mathbf{1}^\top \boldsymbol{y_d} \mathbf{1}}. \end{cases} \tag{4.4}$$

[1] The operation result of $\boldsymbol{A}_t^\top \boldsymbol{y}_t$ in Eq. (3.25) corresponds to $\boldsymbol{y_d}$.

For this sparse version of the CKL model, the prediction $\hat{y}_t$ for the sample $\boldsymbol{x}_t$ is computed as

$$\hat{y}_t = \sum_{i=1}^{m_t} \widetilde{\alpha}_i k(\boldsymbol{x}_t, \widetilde{\boldsymbol{x}}_i) + b_t = \widetilde{\boldsymbol{\alpha}}_t^\top \widetilde{\boldsymbol{k}}_t + b_t, \tag{4.5}$$

in which $\widetilde{\boldsymbol{k}}_t = [k(\boldsymbol{x}_t, \widetilde{\boldsymbol{x}}_1) \ \ldots \ k(\boldsymbol{x}_t, \widetilde{\boldsymbol{x}}_{m_t})]^\top$ is the kernel vector computed over the SVs currently in the dictionary.

**Remark 4.1**. No matter the sparsity-inducing criterion used, it should be applied for every incoming sample $\boldsymbol{x}_t$. If the inclusion of this sample in the dictionary is rejected, i.e. if it is not considered relevant by the chosen sparsity-inducing criterion, then $\boldsymbol{x}_t$ is not added to the dictionary and the kernel matrix does not increase in size. Hence, $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv}$, $m_t = m_{t-1}$, and the matrix $\widetilde{\boldsymbol{P}}_t$ in Eq. (4.3) is not updated, i.e. $\widetilde{\boldsymbol{P}}_t = \widetilde{\boldsymbol{P}}_{t-1}$.

**Remark 4.2**. However, if the incoming sample $\boldsymbol{x}_t$ is considered relevant by the chosen sparsity-inducing criterion, then it is added to the dictionary, i.e., $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \bigcup \{\boldsymbol{x}_t\}$ and $m_t = m_{t-1} + 1$. As a consequence of this inclusion, the kernel matrix must be updated accordingly. In the next section, we show how to do it incrementally.

### *4.1.1 An Online Variant for the Sparse CKL*

As mentioned, once a new input vector is added to the dictionary, the kernel matrix $\widetilde{\boldsymbol{K}}_t$ needs to be recomputed, increasing its size by one. As a consequence, the parameter vector $\widetilde{\boldsymbol{\alpha}}_t$ needs to be re-estimated. It should be noticed that for the type of identification task we are interested in, parameter estimation is to be carried out in an online, incremental way. This implies that the matrix $\widetilde{\boldsymbol{P}}_t$ in Eq. (4.3) has to be updated recursively in order to avoid its recalculation using all the samples in the dictionary, a procedure whose computational cost is $\mathcal{O}(m_t^3)$.

In Chapter 3 is introduced a technique for the recursive updating of the matrix $\widetilde{\boldsymbol{P}}_t$, which is described next for the sake of completeness. Let us assume that the incoming sample $\boldsymbol{x}_t$ is to be added to the dictionary, i.e., $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \bigcup \{\boldsymbol{x}_t\}$ and $m_t = m_{t-1} + 1$. Thus, one can compute the matrix $\widetilde{\boldsymbol{P}}_t$ as

$$\widetilde{\boldsymbol{P}}_t = \begin{bmatrix} \widetilde{\boldsymbol{K}}_{t-1} + \widetilde{\boldsymbol{\Omega}}_{t-1} & \widetilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t) \\ \widetilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)^\top & k_{tt} + \Omega_t \end{bmatrix}^{-1}, \tag{4.6}$$

in which $\widetilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t) = [k(\boldsymbol{x}_t, \widetilde{\boldsymbol{x}}_1) \ \ldots \ k(\boldsymbol{x}_t, \widetilde{\boldsymbol{x}}_{m_{t-1}})]^\top$ and $k_{tt} = k(\boldsymbol{x}_t, \boldsymbol{x}_t)$.

Based on the matrix inversion lemma, we obtain the following expression for $\widetilde{\boldsymbol{P}}_t^{-1}$:

$$\widetilde{\boldsymbol{P}}_t^{-1} = \begin{bmatrix} \widetilde{\boldsymbol{P}}_{t-1}^{-1} & \widetilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t) \\ \widetilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)^\top & k_{tt} + \Omega_t \end{bmatrix}, \tag{4.7}$$

in which $\widetilde{\boldsymbol{P}}_{t-1}^{-1} = \widetilde{\boldsymbol{K}}_{t-1} + \widetilde{\boldsymbol{\Omega}}_{t-1}$. Since this matrix is partitioned into four blocks, it can be inverted blockwise using the block matrix inversion identity (JUAREZ-RUIZ *et al.*, 2016), leading to the following expression:

$$\widetilde{\boldsymbol{P}}_t = r_t^{-1} \begin{bmatrix} \widetilde{\boldsymbol{P}}_{t-1} r_t + \boldsymbol{z}_t \boldsymbol{z}_t^\top & -\boldsymbol{z}_t \\ -\boldsymbol{z}_t^\top & 1 \end{bmatrix}, \tag{4.8}$$

in which $\boldsymbol{z}_t = \widetilde{\boldsymbol{P}}_{t-1} \widetilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)$ and

$$r_t = \Omega_t + k_{tt} - \boldsymbol{z}_t^\top \widetilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t). \tag{4.9}$$

The *online sparse CKL* approach (OS-CKL) differs from the original CKL (LIU; CHEN, 2013; LIU; CHEN, 2014) by the recursive updating of the matrix $\widetilde{\boldsymbol{P}}_t$. Hence, the parameter vector $\widetilde{\boldsymbol{\alpha}}_t$ is updated recursively only for each sample inserted in the dictionary, instead of using all the previous $N$ samples as required by the original CKL, reducing the computational cost per sample of the model drastically.

## 4.2 The FADOS-CKL Model

For some applications, the dynamics of the system may change over time, and the identification model has to be updated accordingly. As pointed out so far, it is important to have a dictionary that grows with time by the inclusion of new relevant samples when necessary. However, samples that once were considered relevant may become irrelevant later because the dynamics of the system changed to another operating point.

Thus, a mechanism to discard some of the samples in the dictionary becomes a desirable feature of an identification model. The big issue here is how to add this feature to the OS-CKL model without compromising its online learning nature. Bearing this in mind, a novel discarding scheme is developed in this chapter with the following two features:

**Feature 1** - The process of discarding samples of the dictionary is to be executed only when a new sample is included in it.

**Feature 2** - Many samples can be discarded at once, as we will show later on in this chapter. We could have limited the number of discarded samples to one per time, but the identification model became more efficient when more than one sample is allowed to be eliminated from the dictionary when necessary.

The resulting identification model is a powerful extension of the OS-CKL model with a fully adaptive dictionary (FADOS-CKL). By a fully adaptive dictionary, we mean one that can grow and shrink in size *on the fly* according to dynamic changes observed in the system that is being modeled.

### 4.2.1 Dual Formulation of the Problem

The theory which is underlying the development of the FADOS-CKL model begins with the dual formulation of the primal problem shown in Eq. (3.5) whose development is introduced next.

Based on Eq. (3.7), the parameter vector of the primal optimization problem is given by

$$\boldsymbol{w} = \sum_{t=1}^{N} \alpha_t \boldsymbol{\phi}(\boldsymbol{x}_t) = \boldsymbol{\Phi}\boldsymbol{\alpha}, \tag{4.10}$$

in which $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\boldsymbol{x}_1), \cdots, \boldsymbol{\phi}(\boldsymbol{x}_N)]$ is a matrix with the $N$ higher feature space projected input vectors arranged along with the columns, and $\boldsymbol{\alpha} = [\alpha_1 \ \cdots \ \alpha_N]^\top$ is the vector of Lagrange multipliers.

Then, using Eq. (4.10), we can rewrite the Lagrangian in Eq. (3.6) as a function of $\boldsymbol{\alpha}_t$ as follows:

$$\begin{aligned} L(\boldsymbol{\alpha}_t, b) &= \underbrace{\left(\boldsymbol{e}^T\boldsymbol{V}\boldsymbol{e} + \boldsymbol{\alpha}^\top\boldsymbol{K}\boldsymbol{\alpha}\right)/2}_{\text{1st term}} \\ &+ \underbrace{\left(\boldsymbol{\alpha}^\top[\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{1}^\top b - \boldsymbol{e}]\right)}_{\text{2nd term}} \\ &= \underbrace{\left((\boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{y})^T\boldsymbol{V}(\boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{y}) + \boldsymbol{\alpha}^\top\boldsymbol{K}\boldsymbol{\alpha}\right)/2}_{\text{1st term}} \\ &+ \underbrace{\left(\boldsymbol{\alpha}^\top[\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{1}^\top b - (\boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{y})]\right)}_{\text{2nd term}}, \end{aligned} \tag{4.11}$$

in which $\boldsymbol{K} = \boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ (due to kernel trick), $\boldsymbol{y} = [y_1 \ \cdots \ y_N]^\top$, $\boldsymbol{e} = [e_1 \ \cdots \ e_N]^\top$, and $\boldsymbol{V}$ is a diagonal matrix whose diagonal elements are given by

$$V_t = \gamma\rho(e_t), \quad t = 1, \ldots, N. \tag{4.12}$$

The minimization of the Lagrangian in Eq. (4.11) leads to the linear system shown in Eq. (3.11). An equivalent Lagrangian, $L(\widetilde{\boldsymbol{\alpha}}_t, b)$, can be easily derived for the sparse CKL model, whose solution leads to the linear system shown in Eq. (4.2).

The function defined in Eq. (4.11) is a general form of the results on Regularization Networks, RKHS, Gaussian process, kringing, kernel ridge regression, least squares support vector machine (LS-SVM), weighted least squares support vector machine (WLS-SVM), KRLS, KRMC, and ROB-KRLS (SCHÖLKOPF; SMOLA, 2002; SANTOS, 2017), in which by elimination of restriction of the problem defined in Eq. (3.5) and consequently the *2th Term* of Eq. (4.11) we can obtain the solution proposed in algorithms as, i.e, Regularization Networks, KRLS, KRMC and ROB-KRLS. Already using the complete formulation defined in 4.11 we can obtained the solution proposed in algorithms as, i.e, LS-SVM, WLS-SVM and CKL.

### *4.2.2 Pruning the Dictionary via Proximal Methods*

For eliminating items from the dictionary, we consider the following optimization problem with a sparsity-inducing regularization function $\Lambda(\cdot)$ over the parameter vector $\widetilde{\boldsymbol{\alpha}}_t$:

$$\widetilde{\boldsymbol{\alpha}}_t^* = \underset{\widetilde{\boldsymbol{\alpha}}_t \in \mathbb{R}^{m_t}, b \; fixed}{\arg\min} \left\{ J(\widetilde{\boldsymbol{\alpha}}_t) = L(\widetilde{\boldsymbol{\alpha}}_t, b) + \lambda \Lambda(\widetilde{\boldsymbol{\alpha}}_t) \right\}, \tag{4.13}$$

in which $J(\cdot)$ is a generic differentiable loss function with Lipschitz-continuous gradient[2], $\Lambda(\cdot)$ is a convex, continuous, but not necessarily differentiable regularization function and $\lambda$ is a regularization parameter. This composite problem has been extensively studied in the literature, e.g. see Bach *et al.* (2011).

**Remark 4.3**. - It should be noticed that the minimization problem in Eq. (4.13) is written as a function of the parameter vector $\widetilde{\boldsymbol{\alpha}}_t$ computed using only the vectors currently present in the dictionary, and it is assumed that the bias term is kept constant during the pruning procedure.

**Remark 4.4**. - Adding the convex penalty term $\lambda\Lambda(\widetilde{\boldsymbol{\alpha}}_t)$ is essentially equivalent to adding another constraint to the Lagrangian shown in Eq. (4.11) (BORWEIN; LEWIS, 2006). This non-smooth component of the function $J(\widetilde{\boldsymbol{\alpha}}_t)$ is often used for inducing sparsity in regression models and can be treated separately by means of an associated proximal operator (GAO *et al.*, 2013), henceforth denoted as $\widetilde{\boldsymbol{\alpha}}_t^* = Prox_{\lambda\Lambda}[\widetilde{\boldsymbol{\alpha}}_t]$.

[2] A differentiable function $f$ is said to have an $L$-Lipschitz continuous gradient if, for some $L > 0$, we get $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$, $\forall x, y$.

Thus, to set $\Lambda(\widetilde{\boldsymbol{\alpha}}_t)$ in Eq. (4.13) as the $\ell_1$-norm regularization function $\Lambda(\widetilde{\boldsymbol{\alpha}}_t) = \sum_j \beta_j |\widetilde{\alpha}_t(j)|$ (ZOU, 2006), in which $\beta_j$'s are weights to be dynamically adjusted, is equivalent to define the following proximity operator:

$$\begin{aligned} \widetilde{\boldsymbol{\alpha}}_t^*(j) &= Prox_{\lambda\Lambda}[\widetilde{\boldsymbol{\alpha}}_t](j), \\ &= \text{sign}\{\widetilde{\alpha}_t(j)\} \max\{|\widetilde{\alpha}_t(j)| - \lambda\beta(j), 0\}, \end{aligned} \tag{4.14}$$

so that sign$(\cdot)$ is the sign function, and

$$\beta(j) = \frac{1}{|\widetilde{\alpha}_{t-1}(j)| + \xi}, \tag{4.15}$$

with $\xi$ denoting a small positive constant to avoid division by zero. Thus, by the operation in Eq. (4.14), we can eliminate from the current dictionary $\mathcal{D}_t^{sv}$ those SVs for which $\widetilde{\boldsymbol{\alpha}}_t^*(j) = 0$.

In sum, the sparsity-promoting convex regularization function shown in Eq. (4.13) with the proximal operator defined in Eq. (4.14) allows the systematic discard of irrelevant SVs from the dictionary $\mathcal{D}_t^{sv}$.

It should be noted, however, that when the dictionary is altered either by the inclusion of a new item or by the elimination of some, the matrix $\widetilde{\boldsymbol{P}}_t^{-1}$ must be updated accordingly. The updating strategy due to the inclusion of an item is carried out by Eq. (4.8). In the next section, we develop an updating technique for situations of deletion of items from the dictionary.

## 4.2.3 *Updating the Matrix After Pruning the Dictionary*

As mentioned, once some SVs are discarded from the dictionary, we need to recompute the inverse matrix $\widetilde{\boldsymbol{P}}_t$ using an updating scheme suitable for online learning purposes. Bearing this demand in mind, we approach the problem by a recursive methodology, which makes the use of the relationship between the inverse of a matrix and a submatrix (JUAREZ-RUIZ *et al.*, 2016).

First, recall that $\widetilde{\boldsymbol{P}}_t = \widetilde{\boldsymbol{H}}_t^{-1}$. Then, let us consider a submatrix $\widetilde{\boldsymbol{H}}_{\bar{p};\bar{q}}$ obtained from $\widetilde{\boldsymbol{H}}_t$ by eliminating the $p$-th row and $q$-th column. This problem is directly related to how to calculate the inverse of a perturbed matrix $(\widetilde{\boldsymbol{H}}_t + \boldsymbol{Z})^{-1}$, in which $\boldsymbol{Z}$ is a perturbation matrix of $\widetilde{\boldsymbol{H}}_t$. The inverse can be derived from the well-known Sherman-Morrison-Woodbury formula:

$$(\widetilde{\boldsymbol{H}}_t - \boldsymbol{r}\boldsymbol{u}^\top)^{-1} = \widetilde{\boldsymbol{H}}_t^{-1} + \frac{(\widetilde{\boldsymbol{H}}_t^{-1}\boldsymbol{r})(\boldsymbol{u}^\top\widetilde{\boldsymbol{H}}_t^{-1})}{1 - \boldsymbol{u}^\top\widetilde{\boldsymbol{H}}_t^{-1}\boldsymbol{r}}, \tag{4.16}$$

in which $\boldsymbol{r}, \boldsymbol{u} \in \mathbb{R}^{m_t}$ are auxiliary column vectors. Thus, let us define $\boldsymbol{r} = \widetilde{\boldsymbol{h}}_q - \boldsymbol{c}_p$, in which $\widetilde{\boldsymbol{h}}_q$ is the $q$-th column vector of $\widetilde{\boldsymbol{H}}_t$, $\boldsymbol{c}_p \in \mathbb{R}^{m_t}$ is the $p$-th canonical column vector, and $\boldsymbol{u} = \boldsymbol{c}_q$ is the $q$-th canonical column vector. With these definitions, $\widetilde{\boldsymbol{H}}_t - \boldsymbol{r}\boldsymbol{u}^\top$ is equal to $\widetilde{\boldsymbol{H}}_t$ except in its $q$-th column, which is equal to $\boldsymbol{c}_p$. By applying the Sherman-Morrison formula shown in Eq. (4.16) to calculate $(\widetilde{\boldsymbol{H}}_t - \boldsymbol{r}\boldsymbol{u}^\top)^{-1}$, then $(\widetilde{\boldsymbol{H}}_{\bar{\boldsymbol{p}};\bar{\boldsymbol{q}}})^{-1}$ is obtained by eliminating the $q$-th row and $p$-th column of $(\widetilde{\boldsymbol{H}}_{\bar{\boldsymbol{p}};\bar{\boldsymbol{q}}})^{-1}$ obtaining the updating of $\widetilde{\boldsymbol{P}}_t$ when items are eliminated from the dictionary.

**Remark 4.5**. - The canonical basis of $\widetilde{\boldsymbol{H}}_t$ is formed by vectors that have '1' in a coordinate and '0' in the others. Moreover, the $q$-th row and $p$-th column are equal, and then it is enough to determine one row of $\widetilde{\boldsymbol{H}}$ and to transform this in its canonical form.

### *4.2.4 Updating the Weights for Improved Estimation*

After testing for the inclusion and deletion of items in and from the dictionary at time step $t$, we have seen that the matrix $\widetilde{\boldsymbol{P}}_t$ must be updated accordingly. However, this has been done by considering the terms $\rho(e_{d_j})$ fixed. For an accurate estimation of the parameters $\widetilde{\boldsymbol{\alpha}}_t$ and $b$ at time step $t$, we have to recompute the weighting terms $\rho(e_{d_j})$ after adaptation of the dictionary.

A new set of values for the weighting terms can be computed using Eq. (3.4), so that the diagonal elements of the matrix $\widetilde{\boldsymbol{\Omega}}$ are updated as $\widetilde{\boldsymbol{\Omega}}_{d_j} = 1/(\gamma\rho(e_{d_j}))$ and, in sequence, the new values of the parameters $\widetilde{\boldsymbol{\theta}}_t = [\widetilde{\boldsymbol{\alpha}}_t^\top, b_t]^\top$ are re-estimated using Eq. (4.4).

Since only the diagonal elements of $\widetilde{\boldsymbol{\Omega}}$ are changed and the others elements in $\widetilde{\boldsymbol{H}}$ and $\widetilde{\boldsymbol{P}}$ are always kept unchanged, we use the Sherman-Morrison-Woodbury formula (GOLUB; LOAN, 2012) to update $\widetilde{\boldsymbol{P}}$ via the Eq. (3.27).

After updating the weighting values, the prediction $\hat{y}_t$ for the sample $\boldsymbol{x}_t$ can be finally computed as

$$\hat{y}_t = \sum_{j=1}^{m_t} \widetilde{\alpha}_j k(\boldsymbol{x}_t, \boldsymbol{x}_{d_j}) + b = \widetilde{\boldsymbol{\alpha}}^\top \widetilde{\boldsymbol{k}}_t + b, \tag{4.17}$$

in which $\widetilde{\boldsymbol{k}}_t = [k(\boldsymbol{x}_t, \boldsymbol{x}_{d_1}) \ \dots \ k(\boldsymbol{x}_t, \boldsymbol{x}_{d_{m_t}})]^\top$ is the kernel vector.

All the steps involved in the implementation of the proposed FADOS-CKL model are summarized in the pseudocode showed in Algorithm 2.

**Algorithm 2:** Pseudocode of FADOS-CKL model.

```
Requirements
The hyperparameters of the chosen sparsity criterion, of the
proximity operator, and of the kernel function.
Initialization
$\widetilde{\boldsymbol{K}}_1 = k_{11}; \tilde{\boldsymbol{\Omega}}_1 = 1; \widetilde{\boldsymbol{H}}_1 = \widetilde{\boldsymbol{K}}_1 + \widetilde{\boldsymbol{\Omega}}_1; \widetilde{\boldsymbol{P}}_1 = 1/\tilde{\boldsymbol{H}}_1;$
Compute $\widetilde{\boldsymbol{\alpha}}_1$ and $b_1$ using Eq. (4.4);
for $t = 2 : N$,
  Get incoming pair: $(\boldsymbol{x}_t, y_t)$;
  Update the sparsity with a sparsity criterion (e.g. ALD) to verify if perform the
  dictionary update;
  if an item is included in the dictionary,
    Add $\boldsymbol{x}_t$ to the dictionary $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \bigcup \{\boldsymbol{x}_t\}$;
    Compute $\widetilde{\boldsymbol{P}}_t$ using Eq. (4.8);
    Compute $\widetilde{\boldsymbol{\alpha}}_t$ and $b_t$ using Eq. (4.4);
    Pruning the dictionary:
      Apply the proximity operator in Eq. (4.14) to $\widetilde{\boldsymbol{\alpha}}_t$;
      Remove $k(\cdot, \boldsymbol{u}_{d_j})$ from the dictionary if $\widetilde{\boldsymbol{\alpha}}_t^*(d_j) = 0$;
      Update $\widetilde{\boldsymbol{P}}_t$ using Eq. (4.16);
    Updating the weighting terms:
      Compute $\rho(e_{d_j})$ using Eq. (3.4);
      Update $\widetilde{\boldsymbol{P}}_t$ using Eq. (3.27);
      Update $\widetilde{\boldsymbol{\alpha}}_t$ and $b_t$ using Eq. (4.4);
  else dictionary is unchanged
    $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv}$;
  end if
end for
Output $\widetilde{\boldsymbol{\alpha}}_t, b_t, \mathcal{D}_t^{sv}$.
```

## 4.3 Results and Discussion

In this section, we report the results of computer experiments designed for evaluating the proposed FADOS-CKL model. Firstly, we carry out a performance evaluation of different sparsity-inducing criteria using the OS-CKL model. Roughly speaking, the OS-CKL is the FADOS-CKL without the pruning of items from the dictionary. The goal of this experiment is to select the best criterion for dictionary construction for the dynamical system identification task. Then, we evaluate the performance of the proposed FADOS-CKL in its ability to dynamically adjust the dictionary size by promoting an efficient balance between the number of items going in and out of the dictionary.

The experiments involve two real-world large-scale benchmarking data sets for nonlinear system identification, namely, the Silverbox (WIGREN; SCHOUKENS, 2013) and the

Wiener-Hammerstein (WH) (SCHOUKENS *et al.*, 2009). We consider two settings: outlier-free and outlier-contaminated scenarios. We perform the search for optimal values of the hyperparameters of the sparsity and pruning criteria trying to keep approximately the same level of sparsity for the different outlier-contamination scenarios.

Independently of the data set and of the contamination scenarios, all the kernel filters evaluated in this chapter require an adequate choice of the kernel function and, hence, the specification of the corresponding hyperparameter. The choice of the kernel function, however, is data-dependent. An issue that demands from the user the experimentation with portions of the data. In order to reduce this burden, we constrained our choice to two widely used kernel functions, namely, the Gaussian (a.k.a., radial basis functions) and the polynomial.

The figure of merit for evaluating the numerical performance is the geometric mean (GM) of the root mean square error (RMSE) and the average size of the dictionary (#SVs), computed for testing samples over 20 independent runs. We use the GM by the fact that we want to minimize both indices, RMSE, and #SVs, simultaneously. Thus, the best model is one that provides the minimum GM value.

Model evaluation during the testing phase corresponds to the *free simulation* task, in which the predicted output value is fed back to the regressor input. This is a much more adequate task to evaluate nonlinear models engaged in system identification than the more common one-step-ahead prediction task since we can evaluate how well the models capture the actual dynamics of the underlying systems and how they deal with the propagation errors.

### 4.3.1 Large-scale Data Sets

The Silverbox data set (WIGREN; SCHOUKENS, 2013) corresponds to an electric circuit simulating a mass-spring-damper system and contains a total of 131,072 samples for the input-output voltage pairs $(\{u_n\}, \{y_n\})$. The first 40,000 samples of $(\{u_n\}$ and $\{y_n\}$ are used for model evaluation and the remaining 91,072 ones for model building. From the model-building samples, we split the first 45,000 samples for parameter estimation and the remaining 46,072 ones for model validation. For scenarios with outlier contamination, we considered a non-Gaussian noise model whose samples are generated from a Student's *t*-distribution with zero mean and 2 degrees of freedom. The non-Gaussian noise samples are spread randomly along the model building sequence. For this data set, we selected the polynomial kernel function, and the memory parameters for the regressor input are set to $L_u = L_y = 10$.

Table 5 – Test results of the OS-CKL model for different sparsity-inducing criteria.

| data set | Criterion | Without outliers | | | With outliers | | |
|---|---|---|---|---|---|---|---|
| | | RMSE | #SVs | GM | RMSE | #SVs | GM |
| Silverbox | ALD | 3.28E-3 ± 0.00E0 | 317 ± 0 | 1.02 | 8.26E-3 ± 2.01E-4 | 317 ± 0 | 1.62 |
| | Novelty | **2.95E-3 ± 0.00E0** | **243 ± 0** | **0.85** | **6.75E-3 ± 3.48E-4** | **348 ± 9** | **1.53** |
| | Coherence | 2.04E-2 ± 0.00E0 | 362 ± 0 | 2.72 | 1.96E-2 ± 6.04E-4 | 362 ± 2 | 2.66 |
| | Surprise | 3.41-3 ± 0.00E0 | 330 ± 0 | 1.06 | 7.92E-3 ± 3.10E-4 | 348 ± 0 | 1.66 |
| WH | ALD | 7.95E-2 ± 0.00E0 | 328 ± 0 | 5.11 | 7.96E-2 ± 3.60E-3 | 328 ± 0 | 5.11 |
| | Novelty | **4.66E-2 ± 0.00E0** | **356 ± 0** | **4.07** | **5.42E-2 ± 2.44E-3** | **380 ± 63** | **4.54** |
| | Coherence | 1.02E-1 ± 0.00E0 | 362 ± 0 | 6.08 | 1.04E-1 ± 2.69E-3 | 362 ± 1 | 6.14 |
| | Surprise | 5.84E-2 ± 0.00E0 | 343 ± 0 | 4.48 | 6.15E-2 ± 2.21E-3 | 353 ± 0 | 4.66 |

The Wiener-Hammerstein data set (SCHOUKENS *et al.*, 2009) contains high-quality measurements of a nonlinear electric circuit with a total of 188,000 samples for the input ($\{u_n\}$) and output ($\{y_n\}$). Firstly, we discard the first and last 10,000 samples of the input $\{u_n\}$ and output $\{y_n\}$ sequences. From the 10,001-th sample on, the first 50,000 input-output pairs ($\{u_n\}$, $\{y_n\}$) are used for model evaluation and the remaining 118,000 for model building. From the model-building samples, we separate the first 70,000 for estimation of the model and the remaining 48,000 samples for validating the parameters of models. For scenarios with outlier contamination, we considered a non-Gaussian noise model whose samples are generated from a Poisson distribution with a mean value of 0.05. For this data set, we selected the RBF kernel function, and the memory parameters for the regressor input are also set to $L_u = L_y = 10$.

As a final remark, for the scenarios with the inclusion of outliers in the training (i.e., parameter estimation) sequence, we consider a contamination level of 5%. This means that 5% of the samples of the output variable are randomly contaminated with the outliers generated as explained in the previous paragraph.

### 4.3.2 Selecting the Best Sparsity-Promoting Criteria

In this set of experiments, we aim at evaluating the following sparsity-promoting criteria: ALD, surprise, coherence, and novelty. In Table 5, we report the results of the OS-CKL model using the aforementioned criteria for the test sequences after being trained without and with outliers. The table contains the values of the average and standard deviation RMSE, the average and standard deviation dictionary size (#SVs), and the average geometric mean. All values are averaged after 20 independent runs.

A closer look at the reported results reveals that in terms of the smallest GM values, the novelty criterion achieved the best performance among all evaluated criteria for both data sets. This means that the best compromise between the smallest possible value of the RMSE value and the smallest possible value of the dictionary size is obtained by the OS-CKL model by using the novelty criterion. Thus, we select this criterion to continue with the next set of experiments.

For the sake of completeness, we report in Table 6 the results of a performance comparison involving the OS-CKL model using the novelty criteria (OS-CKL-NC) and the state of the art in kernel adaptive filtering for dynamic system identification, such as the kernel least mean square (KLMS), the kernel recursive least square (KRLS), and the kernel recursive maximum correntropy (KRMC) for the Silverbox data set. The OS-CKL-NC model achieved better results than the other three models.

**Remark 4.6**. - In addition to being the best sparsity-inducing criteria in our experiments, the novelty criterion introduced by (PLATT, 1991) is also by far the lightest one in terms of computational complexity. This method computes distances in the original Euclidean space of the SVs, not in the feature (i.e. RKHS) space. More specifically, the smallest distance from the current input vector $dist_t^* = \min_{\widetilde{\boldsymbol{x}}_j \in \mathcal{D}_{t-1}^{sv}} \|\boldsymbol{x}_t - \widetilde{\boldsymbol{x}}_j\|$ to the items currently present in the dictionary $\mathcal{D}_{t-1}^{sv} = \{\widetilde{\boldsymbol{x}}_j\}_{j=1}^{m_t-1}$ is determined. If $dist_t^* < \delta_1$, then $\boldsymbol{x}_t$ will not be inserted into the dictionary, with $\delta_1$ denoting a preset threshold. All the other sparsity-promoting methods evaluated in this chapter require heavier kernel computations.

**Remark 4.7**. - What could be thought initially as a positive aspect, the computation of distances in the original Euclidean space makes the novelty criterion more sensitive to outliers than the other kernel-based sparsity-promoting criteria evaluated in this chapter, such as the ALD, Coherence, and Surprise. This sensitivity is reduced in these criteria because the kernel computations constrain the bad influence of outliers. This is the case for the Gaussian kernel function, higher distances between the current sample $\boldsymbol{x}_t$ and samples in the dictionary $\widetilde{\boldsymbol{x}}_j$ produce small values of the kernel function $k(\boldsymbol{x}_t, \widetilde{\boldsymbol{x}}_j) = k(\|\boldsymbol{x}_t - \widetilde{\boldsymbol{x}}_j\|)$. As a consequence of its higher sensitivity to outliers, the Novelty criterion will present a larger dispersion for the size of the dictionary in outlier-contaminated scenarios. This characteristic, however, will not affect the overall better performance of the FADOS-CKL model, as will be shown in the experiments carried out in the following section.

### *4.3.3 Results for the FADOS-CKL with the Novelty Criterion*

In the second set of experiments, we analyze the effect of the proposed method for pruning items from the dictionary, a feature that gave rise to the FADOS-CKL model.

Table 6 – Comparison of the OS-CKL-NC model with other kernel models (Silverbox data set, no outliers).

| Model | Obtained results | | |
|---|---|---|---|
| | RMSE | #SVs | GM |
| OS-CKL-NC | **2.95E-3 ±0.00E0** | **243** | **0.85** |
| KLMS-NC | 2.90E-2 ± 0.00E0 | 418 | 3.47 |
| KRMC-NC | 4.90E-3 ± 0.00E0 | 350 | 1.31 |
| KRLS-ALD | 3.50E-3 ± 0.00E0 | 302 | 1.03 |

Table 7 – Results of the FADOS-CKL model using the novelty criterion for scenarios with and without outliers.

| data set | Outliers | $\lambda$ | RMSE | #SVs | GM | SV-Red (%) |
|---|---|---|---|---|---|---|
| Silverbox | NO | baseline | 2.95E-3 ± 0.00E0 | 243 ± 0 | 0.84 | - |
| | | 0.001 | 2.92E-3 ± 0.00E0 | 189 ± 0 | 0.74 | 22.22 |
| | | 0.002 | 3.36E-3 ± 0.00E0 | 113 ± 0 | 0.61 | 53.49 |
| | | 0.005 | **5.60E-3 ± 0.00E0** | **61 ± 0** | **0.58** | **74.89** |
| | YES | baseline | 6.75E-3 ± 3.48E-4 | 348 ± 9 | 1.53 | - |
| | | 0.001 | 7.37E-3 ± 2.30E-3 | 290 ± 19 | 1.46 | 16.66 |
| | | 0.002 | 8.31E-3 ± 2.50E-3 | 227 ± 18 | 1.37 | 34.77 |
| | | 0.005 | **9.86E-3 ± 2.40E-3** | **103 ± 15** | **1.00** | **70.40** |
| WH | NO | baseline | 4.66E-2 ± 0.00E0 | 356 ± 0 | 4.07 | - |
| | | 0.001 | 5.12E-2 ± 0.00E0 | 292 ± 0 | 3.87 | 17.97 |
| | | 0.002 | 4.88E-2 ± 0.00E0 | 290 ± 0 | 3.76 | 18.53 |
| | | 0.005 | **4.66E-2 ± 0.00E0** | **273 ± 0** | **3.57** | **23.31** |
| | YES | baseline | 5.53E-2 ± 2.44E-3 | 380 ± 63 | 4.58 | - |
| | | 0.001 | 5.53E-2 ± 1.58E-3 | 269 ± 10 | 3.86 | 29.21 |
| | | 0.002 | 5.66E-2 ± 2.13E-3 | 250 ± 9 | 3.76 | 34.21 |
| | | 0.005 | **5.58E-2 ± 2.35E-3** | **243 ± 10** | **3.68** | **36.05** |

The numerical results are shown in Table 7, for different values of the parameter $\lambda$. In this table, in addition to the values of the average RMSE and the average dictionary size, we report the relative decrease in the number of SVs in the dictionary as another figure of merit. The value SV-Red is in percentage, taken with respect to the number of items in the dictionary of the OS-CKL-NC model. As before, the results are gathered for the free simulation task on the test sequences. We searched for values of the hyperparameter $\lambda$ so that the average dictionary size is smaller than or equal to those reported in Table 6.

An analysis of the results shown in Table 7 reveals that the proposed FADOS-CKL model using the novelty criterion tends to minimize the size of the dictionary while keeping a high predictive power as inferred from the reported values of the GM index. The best FADOS-CKL models in terms of GM values are those with $\lambda = 0.005$, for both data sets.

In Figures 8 and 9, we show typical results produced by the FADOS-CKL model for both data sets. For generating the sequences of predicted output values the following values of the hyperparameters were used:

**Silverbox data set** - Polynomial kernel of the order of 3; correntropy bandwidth set to 0.2 without outliers and 0.4 with outliers; $\lambda = 0.005$.

**Wiener-Hammerstein data set** - Gaussian kernel bandwidth set to 0.01; correntropy bandwidth set to 0.2 in both scenarios; $\lambda = 0.005$.

In Figure 8(a), we report the predicted (in red) and the actual values (in blue) of the output time series for the Silverbox data set. Zoomed portions of the predicted output time series for outlier-free and outlier-contaminated scenarios are also shown on the right-hand side of the figure. For both scenarios, the FADOS-CKL model was able to accurately capture the underlying dynamics of the system of interest. In Figure 8(b), we report the corresponding results for the WH data set. For this data set, the FADOS-CKL model also performed accurately, capturing the correct dynamics of the underlying system.

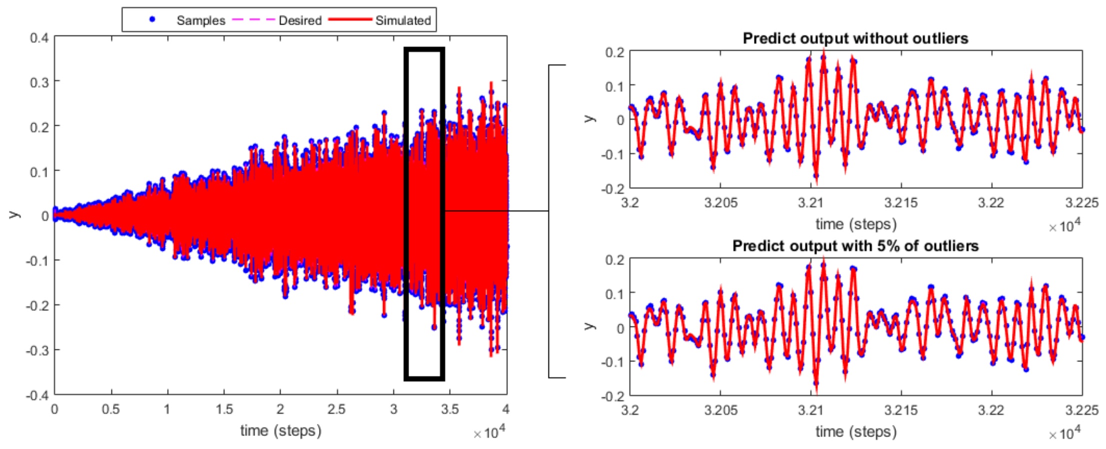
The curves in Figure 9 are of particular interest since they corroborate the main ideas put forward in this chapter. They show the numbers of items inserted in the dictionary by the novelty criterion (red curve) per time step, the number of items pruned from the dictionary by the proposed proximal method (orange curve), and the difference between these numbers (blue curve). The figure axes are in the log scale.

In Figure 9(a) it is possible to visualize the evolution of the three curves (insertion of items, deletion of items, difference curve) for the Silverbox data set. Similar curves are shown in Figure 9(b) for the WH data set. If one takes the red curve alone, it corresponds to models without a dictionary pruning mechanism, such as those showed in Table 6.

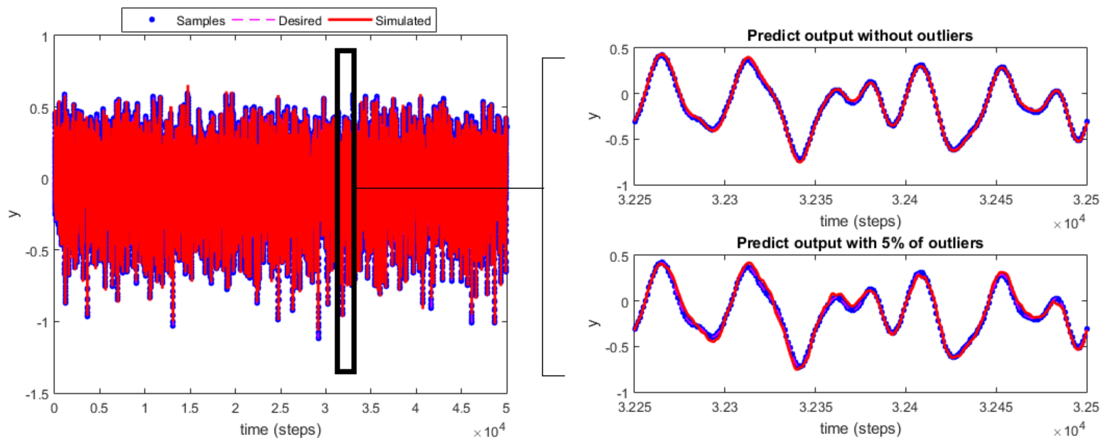
In both figures, one can clearly observe a tendency of the blue curve to decrease its slope as time passes by. This means that the rate of increase of the dictionary (and, hence, of the complexity of the predictive model) is reduced significantly in comparison to models without dictionary pruning. In other words, the proposed approach for deleting items from the dictionary acts to stabilize the flow of items coming in and out of the dictionary.

In conclusion, from the results of the comprehensive experiments reported in this chapter, we are confident in stating that the proposed FADOS-CKL model consistently outperforms the OS-CKL model in the task of interest. The FADOS-CKL model was able to provide accurate results using a fully adaptive dictionary, giving rise to a powerful compact representation of the dynamic systems has been modeled.

Figure 8 – Predicted output sequences for test data using the FADOS-CKL model with novelty criterion. Featuring: free simulation task, without outlier and with 5%-outlier contamination, and $\lambda$=0.005.



(a) Silverbox. Novelty threshold of 0.215, and using a polynomial kernel of order 3.



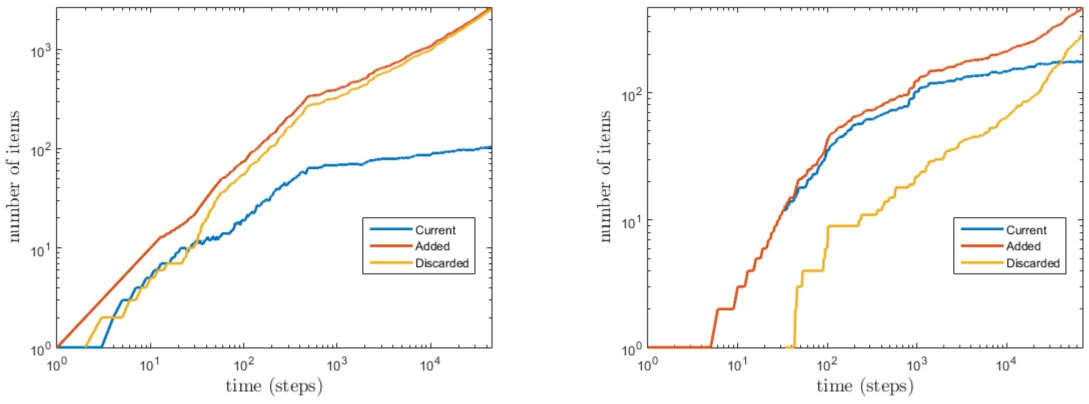
(b) Wiener-Hammerstein. Novelty threshold of 1.2, and using Gaussian kernel with a bandwidth of 0.1.

Source: prepared by the author (2021).

## 4.4 Conclusions

In this chapter, we started with the problem of designing kernel-based dynamic models for system identification. In this regard, several sparsity-inducing criteria have been proposed in the literature to handle complex issues related to the size of the kernel matrix. By means of these techniques, the size of the kernel matrix is substantially reduced. However, most of the sparsity-inducing mechanisms only include items in the dictionary. As time passes by, the system may change its dynamics, and many of the dictionary items may become irrelevant for predicting the current output. Thus, adaptive strategies for discarding items from the dictionary are highly desirable. Such strategies also help to keep the dictionary at suitable sizes, avoiding limitless growth.

Figure 9 – Evolution of the dictionary size during training of the FADOS-CKL model with novelty criterion. Featuring: 5%-outlier contamination, $\lambda$=0.005.



(a) Silverbox. (b) Wiener-Hammerstein.

Source: prepared by the author (2021).

Bearing these issues in mind, we developed a pruning strategy based on proximal optimization methods for a kernel adaptive filter to be capable of discarding irrelevant items from the dictionary. The resulting model was named FADOS-CKL. This is the main contribution of the current chapter, i.e., a simple efficient mechanism for pruning items from the current dictionary on the fly. As a consequence of the proposed pruning mechanism, a novel rule for updating the parameters of the predictive model in case of deletion of items from the dictionary had to be devised. In this regard, the proposed FADOS-CKL model has a fully adaptive dictionary in the sense that items can be inserted when considered relevant and can also be removed from the dictionary if they become obsolete. This novel feature, to the best of our knowledge, has not been observed in alternative kernel adaptive filters currently available in the literature. Another positive aspect of the proposed pruning mechanism is the general formulation that allows it to be used in conjunction with any sparsity-inducing criteria available and by any sparse kernel adaptive filter. A complete mathematical development leading to the proposed model was described in the chapter and a pseudo-code is provided in order to facilitate implementation by interested readers.

For the first set of reported experiments, we carried out a comprehensive investigation of the state of the art in sparsity-inducing criteria aiming at selecting the best performing one for the task of interest. For these experiments, we used the OS-CKL model and selected the novelty criterion for the subsequent experiments. In the second set of experiments, we evaluated the performance of the proposed FADOS-CKL model in two benchmarking data sets of nonlinear dynamic system identification. Scenarios with and without outliers were investigated. In this regard, we were able to show that the proposed FADOS-CKL model was able to capture the underlying dynamics correctly from the available data sets, delivering a model which is more accurate (in terms of predictive power) and also more compact (in terms of the size of the dictionary) than its predecessor, the OS-CKL model.

The next chapter will introduce a primal solution of the OS-CKL model using random Fourier features (RFF) as an equivalent positive-definite kernel that induces a new finite-dimensional RKHS.

# 5 A PRIMAL SOLUTION TO THE CKL MODEL

As we verified in the previous chapters, the CKL is indeed a powerful and versatile kernel method to solve nonlinear problems in signal processing and machine learning. Learning in CKL relies on a cost function and equality constraints instead of the quadratic programming problem as on SVR-type models, which demands high computational resources for its implementation. A globally optimal solution for the CKL problem is simpler to obtain since it requires only a set of linear equations.

However, despite this computationally attractive feature, the CKL model has some potential limitations, especially for large-scale data sets. For example, the lack of sparsity in the solution vector, implying that all training examples will be used as support vectors in order to make new predictions. The standard approach relies on the kernel trick to perform pairwise evaluations of a kernel function, which leads to scalability issues for large data sets due to its linear growth with respect to the training data.

## 5.1 Back to the Reality

As already mentioned, non-sparsity is a limiting factor to use the CKL model in some modeling applications, for example, those requiring the processing of large amounts of data. Therefore in chapters 3 and 4 we proposed attractive alternatives to solve the CKL model promoting solutions online, sparse and parsimonious. However, these solutions occur in dual space and the model size depends on sparsity criteria. The approach in this chapter provides a solution for the CKL problem in primal space, which is inherently sparse since it relies on a selected sub-sample of support vectors.

A popular approach to tackle this problem, known as RFF, samples from distribution to obtain the data-independent basis of a higher finite-dimensional feature space, in which the inner product approximates the kernel function (RAHIMI; RECHT, 2007). In this chapter, the inner product operations involving explicit mappings is not viewed as an approximation, but as an equivalent positive-definite kernel that induces a new finite-dimensional RKHS (LI; PRINCIPE, 2019). Therefore, here the main goal is to develop an online and robust solution of the CKL model built upon primal space capable of operating with an adaptive filter using RFF. Here, it is adopted the classical RFF, but other possibilities are the RFF with complex exponentials (RFF2) (RAHIMI; RECHT, 2007) or deterministic frameworks that provide exact polynomial solutions such as using Gaussian quadrature (GQ) (DAO *et al.*, 2017) and Taylor series (TS) expansion (COTTER *et al.*, 2011), which is related to the fast Gauss transformation in kernel density estimation (GREENGARD; STRAIN, 1991).

### 5.1.1 Estimation on Primal Space

Let us assume the estimation data set $\{(\boldsymbol{x}_t, y_t)\}_{t=1}^{N}$ with the input vector $\boldsymbol{x}_t \in \mathbb{R}^{N_x}$ and corresponding output $y_t \in \mathbb{R}$. In a regression problem, the goal is to search a function $f(\cdot)$ that approximates sufficiently well the responses $y_t$ for all instances of the available data. For the nonlinear case, $f$ usually takes the following form

$$f(\boldsymbol{x}_t) = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_t) + b, \tag{5.1}$$

in which $\boldsymbol{\phi}(\cdot) : \mathbb{R}^{N_x} \to \mathbb{R}^d$ is a nonlinear map into a higher dimensional feature space (whose dimensionality $d$ might be infinite), $\boldsymbol{w} \in \mathbb{R}^d$ is a parameter vector and $b \in \mathbb{R}$ is a bias term.

Thus, we can construct a feature matrix for all the training instances as

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi_1(\boldsymbol{x}_1) & \dots & \phi_d(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{x}_N) & \dots & \phi_d(\boldsymbol{x}_N) \end{bmatrix}_{N\times d} = \begin{bmatrix} \boldsymbol{\phi}^\top(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{\phi}^\top(\boldsymbol{x}_N) \end{bmatrix}_{N\times d}. \tag{5.2}$$

In order to search for a suitable approximation to the function $f(\cdot)$, it is formulated a primal optimization problem integrating the SVR framework with the MCC that leads to minimize the following functional

$$\begin{cases} \text{min } J(\boldsymbol{w}, \rho) = \frac{\gamma}{2}\sum_{t=1}^{N} \rho(e_t)e_t^2 + \frac{1}{2}||\boldsymbol{w}||^2 \\ \text{s.t. } y_t - \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_t) - b - e_t = 0, \quad t = 1, ..., N, \end{cases} \tag{5.3}$$

in which $\gamma$ $(\gamma > 0)$ is the regularization parameter.

It is important to recall that the MCC bears close relationship with $M$-estimation, which is a generalized maximum likelihood method proposed by Huber (1981) as an outlier-robust parameter estimation framework:

$$\boldsymbol{\theta}_{mcc} = \text{arg min}_{\boldsymbol{\theta}} \sum_{t=1}^{N} \rho(e_t)e_t^2, \tag{5.4}$$

in which the weighting term $\rho(e_t)$ penalizes large errors, usually caused by outliers. If one chooses $\rho(e_t)$ as

$$\rho(e_t) = \frac{1}{\sqrt{2\pi}\sigma^3} \exp\left\{-\frac{e_t^2}{2\sigma^2}\right\}, \quad t = 1, ..., N, \tag{5.5}$$

the correntropy can be viewed as a robust cost function capable to handle outlying samples because it does not amplify their effects.

In order to solve (5.3), one must first write it as the following unconstrained optimization problem:

$$J(\boldsymbol{w}, b) = \frac{\gamma}{2}\sum_{t=1}^{N}\rho(e_t)(y_t - \boldsymbol{w}^\top\boldsymbol{\phi}(\boldsymbol{x}_t) - b)^2 + \frac{1}{2}\boldsymbol{w}^\top\boldsymbol{w}, \tag{5.6}$$

and search for its solution by computing the gradients with respect to each parameter and setting them to zero. Then, we get

$$\frac{\partial J}{\partial \boldsymbol{w}} = \boldsymbol{w} - \gamma\sum_{t=1}^{N}\rho(e_t)[y_t - (\boldsymbol{w}^\top\boldsymbol{\phi}(\boldsymbol{x}_t) + b)]\boldsymbol{\phi}(\boldsymbol{x}_t) = \mathbf{0}_d, \tag{5.7}$$

and

$$\frac{\partial J}{\partial b} = -\gamma\sum_{t=1}^{N}\rho(e_t)[y_t - (\boldsymbol{w}^\top\boldsymbol{\phi}(\boldsymbol{x}_t) + b)] = 0, \tag{5.8}$$

in which $\mathbf{0}_d \in \mathbb{R}^d$ is a vector of zeros. The expressions in (5.7) and (5.8) can be arranged in corresponding matrix forms respectively, as

$$\begin{aligned}&\frac{1}{\gamma}\boldsymbol{w} + \boldsymbol{\Phi}^\top\boldsymbol{Q}\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{\Phi}^\top\boldsymbol{Q}\boldsymbol{y} + \boldsymbol{\Phi}^\top\boldsymbol{Q}\mathbf{1}_N b = \mathbf{0}_d,\\ &\quad(\boldsymbol{\Phi}^\top\boldsymbol{Q}\boldsymbol{\Phi} + \frac{1}{\gamma}\boldsymbol{I}_d)\boldsymbol{w} + \boldsymbol{\Phi}^\top\boldsymbol{Q}\mathbf{1}_N b = \boldsymbol{\Phi}^\top\boldsymbol{Q}\boldsymbol{y},\end{aligned} \tag{5.9}$$

and

$$\begin{aligned}&-\mathbf{1}_N^\top\boldsymbol{Q}\boldsymbol{y} + \mathbf{1}_N^\top\boldsymbol{Q}\boldsymbol{\Phi}\boldsymbol{w} + \mathbf{1}_N^\top\boldsymbol{Q}\mathbf{1}_N b = 0\\ &\quad\mathbf{1}_N^\top\boldsymbol{Q}\boldsymbol{\Phi}\boldsymbol{w} + \mathbf{1}_N^\top\boldsymbol{Q}\mathbf{1}_N b = \mathbf{1}_N^\top\boldsymbol{Q}\boldsymbol{y},\end{aligned} \tag{5.10}$$

with $\boldsymbol{Q} = \text{diag}\{\rho(e_t)\}_{t=1}^N \in \mathbb{R}^N \times \mathbb{R}^N$ being a diagonal matrix whose diagonal elements are the correntropy values $\rho(e_t)$. For instance, $\boldsymbol{Q} = \boldsymbol{I}_N$ for the unweighted model.

Finally, to determine the batch model solution, we can arrange the expressions (5.9) and (5.10) into a single linear system of $d + 1$ equations, which is given by

$$\underbrace{\begin{bmatrix}\boldsymbol{\Phi}^\top\boldsymbol{Q}\boldsymbol{\Phi} + \frac{1}{\gamma}\boldsymbol{I}_d & \vdots & \boldsymbol{\Phi}^\top\boldsymbol{Q}\mathbf{1}_N\\ \cdots\cdots & \cdots & \cdots\cdots\\ \mathbf{1}_N^\top\boldsymbol{Q}\boldsymbol{\Phi} & \vdots & \mathbf{1}_N^\top\boldsymbol{Q}\mathbf{1}_N\end{bmatrix}}_{(d+1)\times(d+1)}\underbrace{\begin{bmatrix}\boldsymbol{w}\\ \\ b\end{bmatrix}}_{(d+1)\times 1} = \underbrace{\begin{bmatrix}\boldsymbol{\Phi}^\top\boldsymbol{Q}\boldsymbol{y}\\ \\ \mathbf{1}_N^\top\boldsymbol{Q}\boldsymbol{y}\end{bmatrix}}_{(d+1)\times 1}, \tag{5.11}$$

**Remark 5.1**. The solution of the model in (5.11) is a batch learning process since all the samples to time $t$ are used. The prediction for a new input vector also requires the storage of all $N$ training vectors. As a consequence, the dimensions of the matrix $\mathbf{\Phi}$ and the vector $\boldsymbol{y}$ increase for each new incoming pair $(\boldsymbol{x}_t,\, y_t)$, thus increasing the complexity of the predictor considerably as $t \to \infty$.

**Remark 5.2**. Furthermore, the solution of the model in (5.11) in the primal space involves explicit knowledge of the nonlinear feature mapping $\boldsymbol{\phi}$; however, this is usually not the case. For the dual problem, the lack of knowledge about this mapping is cleverly solved by the *kernel trick*, in which the inner product in feature space $\boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}')$ can be computed by means of kernel function evaluations.

Since we are interested in online adaptive filtering for nonlinear system identification solved upon primal space, the parameters of the predictor must be modified following the arrival of each new sample, and the kernel trick though cannot be used. Furthermore, we require that the computational cost per sample does not increase with the size of the data set. In order to solve it, we use an explicit feature-map reproducing kernel, add a sparsification procedure to select support vectors, and update the parameters to each new sample in an adaptive filtering mode.

### *5.1.2 Explicit Feature-map Reproducing Kernel*

For this task, we use the RFFs and take up samples from distribution to obtain the data-independent basis of a higher finite-dimensional feature space, in which its inner product approximates the kernel function. Here, the inner product of these explicit mappings not as an approximation, but as an equivalent positive-definite kernel that induces a new finite-dimensional RKHS.

In order to avoid the computation of a full $N \times N$ pairwise kernel matrix to solve learning problems on $N$ input samples, the continuous shift-invariant kernel can be approximated as $k(\boldsymbol{x}, \boldsymbol{x}') : \mathbb{X} \times \mathbb{X} \to \mathbb{R} \approx z(\boldsymbol{x})^\top z(\boldsymbol{x}')$, in which $z : \mathbb{X} \to \mathbb{R}^D$, and $D$ is the embedding dimension.

Based on the Fourier transform $p(\omega)$ of the kernel $k$

$$z(\boldsymbol{x}) := \sqrt{\frac{2}{D}} \begin{bmatrix} sin(\omega_1^\top \boldsymbol{x}) \\ cos(\omega_1^\top \boldsymbol{x}) \\ \vdots \\ sin(\omega_{D/2}^\top \boldsymbol{x}) \\ cos(\omega_{D/2}^\top \boldsymbol{x}) \end{bmatrix}, \omega_i \sim p(\omega) \qquad (5.12)$$

in which $\omega$ is drawn from $p(\omega)$. In this case $z(.)$ is a new explicit expression for $\phi$ or an approximation to the feature map $z(\cdot) = \hat{\boldsymbol{\phi}}(\cdot) : \mathbb{R}^{N_x} \to \mathbb{R}^D$, in which $D \ll d$. Thus we can rewrite (5.2) as

$$\hat{\boldsymbol{\Phi}} = \begin{bmatrix} \hat{\phi}_1(\boldsymbol{x}_1) & \dots & \hat{\phi}_D(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \hat{\phi}_1(\boldsymbol{x}_N) & \dots & \hat{\phi}_D(\boldsymbol{x}_N) \end{bmatrix}_{N\times D} = \begin{bmatrix} \hat{\boldsymbol{\phi}}^\top(\boldsymbol{x}_1) \\ \vdots \\ \hat{\boldsymbol{\phi}}^\top(\boldsymbol{x}_N) \end{bmatrix}_{N\times D}, \tag{5.13}$$

and consequently, rewrite (5.11) as

$$\underbrace{\begin{bmatrix} \hat{\boldsymbol{\Phi}}^\top \boldsymbol{Q}\hat{\boldsymbol{\Phi}} + \frac{1}{\gamma}\boldsymbol{I}_D & \vdots & \hat{\boldsymbol{\Phi}}^\top \boldsymbol{Q}\mathbf{1}_N \\ \dots\dots\dots & & \dots\dots \\ \mathbf{1}_N^\top \boldsymbol{Q}\hat{\boldsymbol{\Phi}} & \vdots & \mathbf{1}_N^\top \boldsymbol{Q}\mathbf{1}_N \end{bmatrix}}_{(D+1)\times(D+1)} \underbrace{\begin{bmatrix} \hat{\boldsymbol{w}} \\ \\ \hat{b} \end{bmatrix}}_{(D+1)\times 1} = \underbrace{\begin{bmatrix} \hat{\boldsymbol{\Phi}}^\top \boldsymbol{Q}\boldsymbol{y} \\ \\ \mathbf{1}_N^\top \boldsymbol{Q}\boldsymbol{y} \end{bmatrix}}_{(D+1)\times 1}, \tag{5.14}$$

### *5.1.3 Selecting Support Vectors*

Instead of using all samples in the estimation of $\hat{\boldsymbol{w}}$ and $\hat{b}$, an approximation is carried out using a subset of $M$ ($M \ll N$) support vectors (SV) extracted from the whole training data set.

In this step, the *novelty criterion* is chosen to determine whether a new $\tilde{\boldsymbol{\phi}}(\mathrm{x}_t)$ unit is allocated for this training sample. If $\min_M \|\tilde{\boldsymbol{\phi}}(\mathrm{x}_t) - \tilde{\boldsymbol{\Phi}}_M\|$ (i.e., the minimum distance to the nearest center) is larger than a preset constant threshold $\delta$, the augmented input $\tilde{\boldsymbol{\phi}}(\mathrm{x}_t)$ will be added to the dictionary, i.e., $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \bigcup \{\tilde{\boldsymbol{\phi}}(\mathrm{x}_t)\}$ and $m_t = m_t + 1$.

Thus we can rewrite (5.13) as

$$\tilde{\boldsymbol{\Phi}} = \begin{bmatrix} \tilde{\phi}_1(\boldsymbol{x}_1) & \dots & \tilde{\phi}_D(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \tilde{\phi}_1(\boldsymbol{x}_{m_t}) & \dots & \tilde{\phi}_D(\boldsymbol{x}_{m_t}) \end{bmatrix}_{m_t\times D} = \begin{bmatrix} \tilde{\boldsymbol{\phi}}^\top(\boldsymbol{x}_1) \\ \vdots \\ \tilde{\boldsymbol{\phi}}^\top(\boldsymbol{x}_{m_t}) \end{bmatrix}_{m_t\times D}, \tag{5.15}$$

and consequently rewrite (5.14) as

$$\underbrace{\begin{bmatrix} \tilde{\boldsymbol{\Phi}}^\top \tilde{\boldsymbol{Q}}\tilde{\boldsymbol{\Phi}} + \frac{1}{\gamma}\boldsymbol{I}_D & \vdots & \tilde{\boldsymbol{\Phi}}^\top \tilde{\boldsymbol{Q}}\mathbf{1}_{m_t} \\ \dots\dots\dots & & \dots\dots \\ \mathbf{1}_{m_t}^\top \tilde{\boldsymbol{Q}}\tilde{\boldsymbol{\Phi}} & \vdots & \mathbf{1}_{m_t}^\top \tilde{\boldsymbol{Q}}\mathbf{1}_{m_t} \end{bmatrix}}_{(D+1)\times(D+1)} \underbrace{\begin{bmatrix} \tilde{\boldsymbol{w}} \\ \\ \tilde{b} \end{bmatrix}}_{(D+1)\times 1} = \underbrace{\begin{bmatrix} \tilde{\boldsymbol{\Phi}}^\top \tilde{\boldsymbol{Q}}\tilde{\boldsymbol{y}} \\ \\ \mathbf{1}_{m_t}^\top \tilde{\boldsymbol{Q}}\tilde{\boldsymbol{y}} \end{bmatrix}}_{(D+1)\times 1}, \tag{5.16}$$

### 5.1.4 *Adaptive Filtering Mode*

For this model to work as an adaptive filter the parameters model are updated for each new sample. Thus, we observe that the estimation involves a matrix $\tilde{\boldsymbol{H}} = \tilde{\boldsymbol{\Phi}}^\top \tilde{\boldsymbol{Q}} \tilde{\boldsymbol{\Phi}} + \frac{1}{\gamma}\boldsymbol{I}_D$, whose inverse $\tilde{\boldsymbol{P}} = \tilde{\boldsymbol{H}}^{-1} = (\tilde{\boldsymbol{\Phi}}^\top \tilde{\boldsymbol{Q}} \tilde{\boldsymbol{\Phi}} + \frac{1}{\gamma}\boldsymbol{I}_D)^{-1}$ leads to the solution of the linear system in Eq. (5.16). The dimension of the matrix $\tilde{\boldsymbol{P}}$ is constant at each time step $(D \times D)$ so we can resort to the Sherman-Morrison-Woodbury formula to update $\tilde{\boldsymbol{P}}$ via the following expression

$$\tilde{\boldsymbol{P}}_t = \tilde{\boldsymbol{P}}_{t-1} - \frac{\rho(e_t)\tilde{\boldsymbol{P}}_{t-1}\tilde{\boldsymbol{\phi}}(\boldsymbol{x}_t)\tilde{\boldsymbol{\phi}}(\boldsymbol{x}_t)^\top \tilde{\boldsymbol{P}}_{t-1}}{1 + \rho(e_t)\tilde{\boldsymbol{\phi}}(\boldsymbol{x}_t)^\top \tilde{\boldsymbol{P}}_{t-1}\tilde{\boldsymbol{\phi}}(\boldsymbol{x}_t)}, \tag{5.17}$$

in which $t$ corresponds to the instant of observation of a new sample $\tilde{\boldsymbol{\phi}}(\boldsymbol{x}_t)$. Finally, we rewrite (5.16) as a primal solution in adaptive filtering mode for robust and online learning

$$\underbrace{\begin{bmatrix} \tilde{\boldsymbol{\Phi}}_t^\top \tilde{\boldsymbol{Q}}_t \tilde{\boldsymbol{\Phi}}_t + \frac{1}{\gamma}\boldsymbol{I}_D & \vdots & \tilde{\boldsymbol{\Phi}}_t^\top \tilde{\boldsymbol{Q}}_t \mathbf{1}_{m_t} \\ \cdots\cdots\cdots & \cdots & \cdots\cdots\cdots \\ \mathbf{1}_{m_t}^\top \tilde{\boldsymbol{Q}}_t \tilde{\boldsymbol{\Phi}}_t & \vdots & \mathbf{1}_{m_t}^\top \tilde{\boldsymbol{Q}}_t \mathbf{1}_{m_t} \end{bmatrix}}_{(D+1)\times(D+1)} \underbrace{\begin{bmatrix} \tilde{\boldsymbol{w}}_t \\ \\ \tilde{b}_t \end{bmatrix}}_{(D+1)\times 1} = \underbrace{\begin{bmatrix} \tilde{\boldsymbol{\Phi}}_t^\top \tilde{\boldsymbol{Q}}_t \tilde{\boldsymbol{y}}_t \\ \\ \mathbf{1}_{m_t}^\top \tilde{\boldsymbol{Q}}_t \tilde{\boldsymbol{y}}_t \end{bmatrix}}_{(D+1)\times 1}, \tag{5.18}$$

The solution of the model is then given by solving the linear system in (5.18) as

$$\begin{cases} \tilde{\boldsymbol{w}}_t = \tilde{\boldsymbol{P}}_t \left[ \tilde{\boldsymbol{V}}_t - \frac{\tilde{\boldsymbol{\Phi}}_t^\top \tilde{\boldsymbol{Q}}_t \mathbf{1}_{m_t} \tilde{\boldsymbol{P}}_t \tilde{\boldsymbol{V}}_t}{\tilde{\boldsymbol{V}}_t} \right], \\ \\ \tilde{b}_t = \frac{\tilde{\boldsymbol{P}}_t \tilde{\boldsymbol{V}}_t}{\tilde{\boldsymbol{V}}_t}. \end{cases} \tag{5.19}$$

in which $\tilde{\boldsymbol{V}}_t = \tilde{\boldsymbol{\Phi}}_t^\top \tilde{\boldsymbol{Q}}_t \tilde{\boldsymbol{y}}_t$.

Thus, the predictions for a new incoming vector $\boldsymbol{x}^*$ can be computed as

$$\tilde{f}(\boldsymbol{x}^*) = \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{\phi}}(\boldsymbol{x}^*) + \tilde{b}. \tag{5.20}$$

**Remark 5.3** - By comparing the Eq. (5.11) with the variant proposed in Eq. (5.18), one can note that the parameters of the proposed approach are updated for each new sample. Hence, the computational cost per sample does not increase with the size of the data set. The model complexity also does not increase, since the dimension of $\tilde{\boldsymbol{w}}$ remains constant and equal to $D$. In Algorithm 3 we provide the pseudocode of the proposed model, referred to henceforth as the *primal* OS-CKL (PRIMOS-CKL) model.

**Algorithm 3:** Pseudocode of PRIMOS-CKL model.

```
**Parameters**:
$\delta$: sparse threshold;
$\sigma$: bandwith correntropy;
$\gamma$: initial value to seed the inverse matrix P;
**Initialize**:
$\tilde{\boldsymbol{\phi}}(\cdot) : \mathbb{X} \rightarrow \mathbb{R}^D$: feature map;
$\tilde{\boldsymbol{\phi}}(\boldsymbol{x}_0) = \mathbf{0}$: in which $\mathbf{0}$ is $D \times 1$;
$\mathcal{D}_0^{sv} = \{\tilde{\boldsymbol{\phi}}(\boldsymbol{x}_0)\}$: add $\tilde{\boldsymbol{\phi}}(\boldsymbol{x}_0)$ to dictionary;
$\tilde{\boldsymbol{P}}_0 = \gamma \boldsymbol{I}$: in which $\boldsymbol{I}$ is the $D \times D$ identity matrix;
$\tilde{\boldsymbol{w}}_0 = \mathbf{0}$ and $\tilde{b}_0 = 0$
for $t = 1 : N$,
  Get new sample: $(\boldsymbol{x}_t, y_n)$
  Compute $\tilde{\boldsymbol{\phi}}(\boldsymbol{x}_t)$ from Eq. (5.12);
  if $\min ||\tilde{\boldsymbol{\phi}}(\boldsymbol{x}_t) - \tilde{\boldsymbol{\Phi}}_{m_t}||^2 > \delta$
    $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \bigcup \{(\boldsymbol{x}_t, y_n)\}$: changed dictionary;
    Compute $\rho(e_t)$ from Eq. (5.5);
    Compute $\tilde{\boldsymbol{P}}_t$ from Eq. (5.17);
    Compute $\tilde{\boldsymbol{w}}_t$ and $\tilde{b}_t$ from Eq. (5.18);
    $m_t = m_{t-1} + 1$;
  else
    $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv}$: unchanged dictionary;
end for
Output $\tilde{\boldsymbol{w}}_t, \tilde{b}_t, \tilde{\boldsymbol{\phi}}(\cdot)$.
```

## 5.2 Results and Discussion

This section, reports the results of a comprehensive set of numerical experiments comparing the proposed PRIMOS-CKL with the OS-CKL and other state-of-art methods. The experiments involve the Silverbox data set and Mackey-Glass (MG) (MACKEY; GLASS, 1977) chaotic time series.

### *5.2.1 Dynamical System Identification*

The first experiment involves the Silverbox in the task of system identification in which the results are evaluated in free simulation. For this task it is assumed the same NARX structure for discrete nonlinear model described in previous chapters of this thesis. The main features and corresponding parameters of data set are summarized in Table 8.

We compare the proposed PRIMOS-CKL with the OS-CKL and KRMC, all using the novelty criterion. Hence, we perform the search for optimal values of the hyperparameters of the sparsity trying to keep approximately the same level of sparsity and RMSE for the

Table 8 – Summary of the large-scale data set.

| data set | Important features of the large scale data set | | | |
|---|---|---|---|---|
| | Estimation | Test | $L_u$ | $L_y$ |
| Silverbox | 40,000 | 45,000 | 10 | 10 |

different scenarios.

In the experiment, the feature space or RKHS dimension to PRIMOS-CKL model is $D = 200$, it is not immediately obvious which estimate is preferable so it is determined by grid searching for their optimal value.

Two scenarios are considered: Gaussian noise and non-Gaussian noise (outlier-contaminated scenarios). For scenarios with outlier contamination the outliers were generated as impulsive Gaussian noise with *pdf* given by the mixture model $p_z(z) = (1-q) \times \mathcal{N}(0, 0.005) + q \times \mathcal{N}(0.05, 0.05)$ in which $q$ is determined according to the percentage of outliers in the scenario. We consider 10 independent runs for increasing levels of outlier contamination.

It is presented in Fig. 10(a) a RMSE boxplot for the Silverbox for increasing levels of outlier contamination. Fig. 10(b) showed the bar plots of the resulting dictionary sizes averaged. It is presented in Fig. 11 the predicted output in free simulation regime to PRIMOS-CKL.

A closer look at the results reveals that the proposed PRIMOS-CKL model ends up with a dictionary with more support vectors than the alternative models. However, the dimensionality of the regression vectors is $D = 200$, so that the resulting model is much less complex that the KRMC and OS-CKL models, both with a model size of 400.

### 5.2.2 Convergence for Time Series Prediction

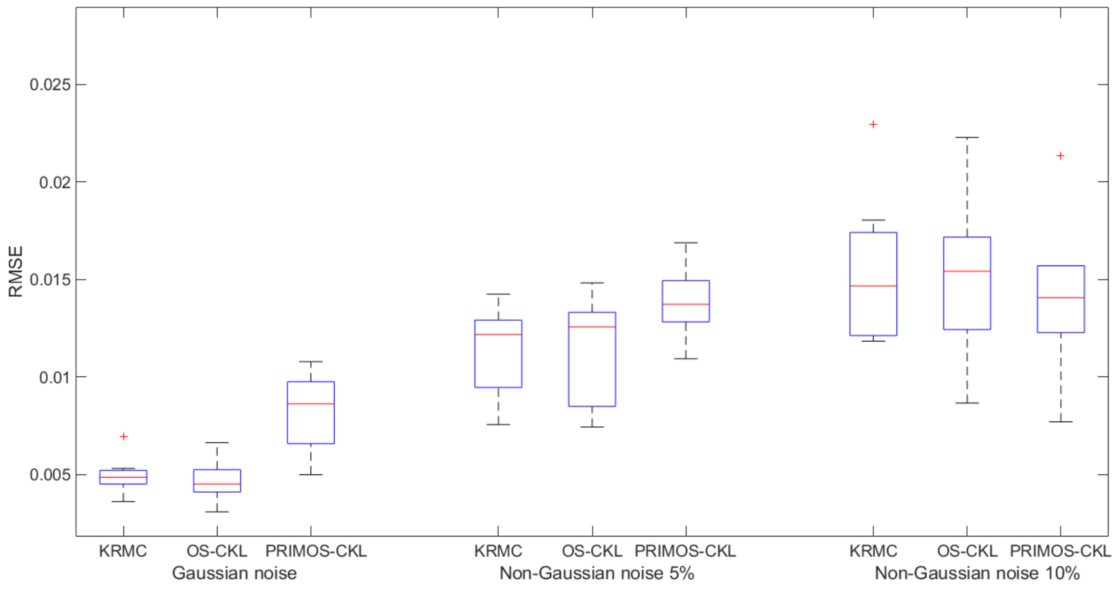
The second experiment involves the chaotic MG time series calculated from the following time-delay differential system:

$$\frac{dx}{dt} = \beta x_t + \frac{\alpha x(t-\tau)}{1 + x(t-\tau)^{10}}, \tag{5.21}$$

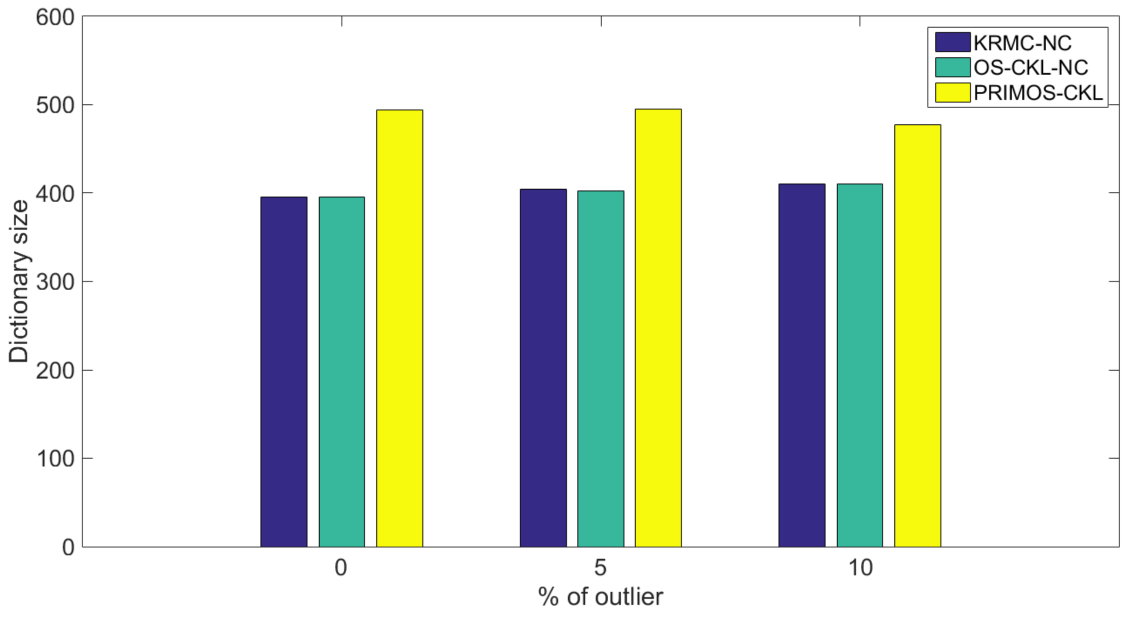
in which $x$ is the time series $t$ and $\tau$ the delay time. We assume the MG30 in which $\tau = 30$.

The setting of this experiment scenario is based on Li and Príncipe (2019) and the goal is to assess the convergence of the model during the training phase. Data samples are standardized, then followed by diving the resulting maximum absolute value to guarantee the sample values are within the range of $[-1, +1]$. The results are averaged over 200 independent

Figure 10 – *(a) Boxplot of the RMSE values related to the free simulation on test data with different levels of contamination by outliers. (b) Bar plots of the average size of the dictionary.*



(a)



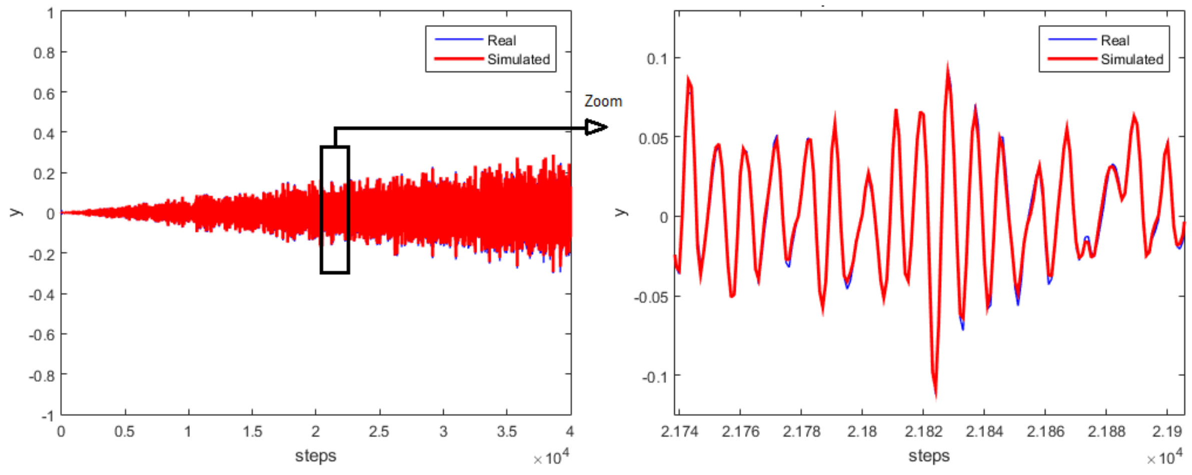
(b)

Source: prepared by the author (2021).

trials. In each trial, 2000 consecutive samples with a random starting point in the time series are used for training, and the subsequent 200 samples are used for testing the predictive performances of the evaluated models.

We compare the proposed PRIMOS-CKL with alternative models. To PRIMOS-CKL, LMS-RFF1, LMS-RFF2, LMS-GQ and LMS-TS the feature space is set at $D = 330$,

Figure 11 – Typical predicted time series for test data under free simulation regime with $10\%$ of outlier contamination using the proposed PRIMOS-CKL model.

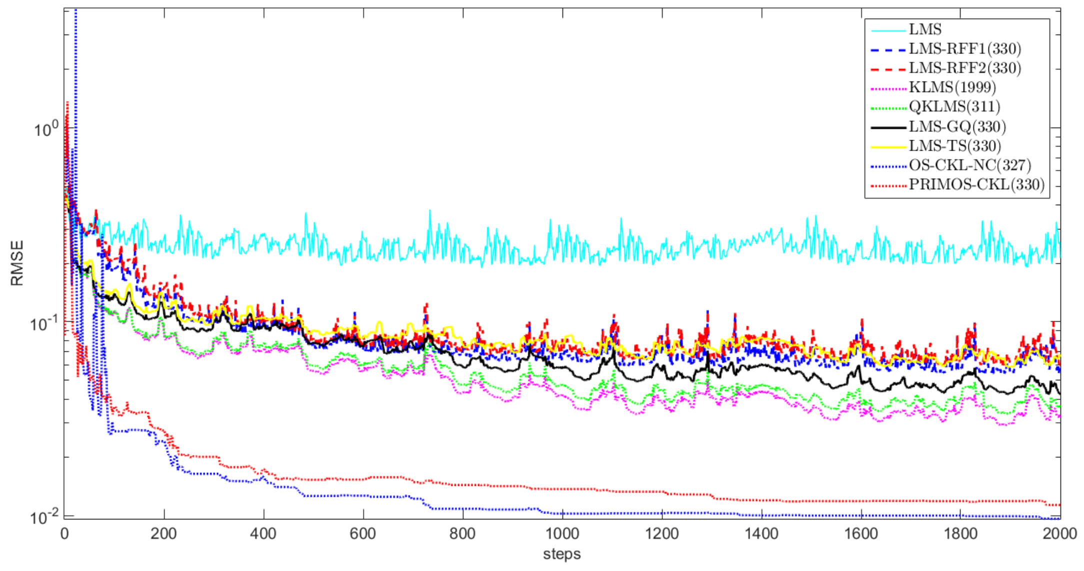


Source: prepared by the author (2021).

corresponding to Taylor polynomials of degrees up to 4, for input dimension $d = 7$. The GQ mapping is fixed across all trials, while RFFs are randomly generated in each trial. Taylor series expansion mappings are completely deterministic and fixed for all trials. We select an 8-th degree quadrature rule with sub-sampling to generate the explicit feature mapping. The learning rate for the LMS-type models is $= 0.4$, with the linear LMS as a baseline. The KLMS use is not sparse and use all samples as support vectors. The dictionary size of the QKLMS and OS-CKL is impossible to fix a priori across different trials, we fixed the vector quantization parameter to QKLMS ($q_{\text{factor}} = 0.07$) and the novelty criterion to OS-CKL (distance $= 0.26$ and error $= 0.0001$) such that the average value (311 and 327, respectively) of the dictionary size is close to the dimension of the finite-dimensional RKHS (330).

The learning curve is presented in Fig. 12 for the RMSE values comparing the PRIMOS-CKL with other adaptive filter methods such as the linear LMS model, kernel LMS models built on the primal (using RFFs, GQ, and TS), and kernel adaptive filters built on the dual (KLMS, QKLMS, and OS-CKL). It is evident that the PRIMOS-CKL performs better than all LMS-type models and it achieves almost the same testing RMSE as the OS-CKL but with model size fixed.

Figure 12 – Convergence curves for all the evaluated models.



Source: prepared by the author (2021).

## 5.3 Conclusions

This chapter introduced the PRIMOS-CKL model, which is motivated by recent advances in ITL (e.g. correntropy) and kernel adaptive filtering methods. The PRIMOS-CKL solves the CKL framework in primal space and it can now be efficiently used for online learning. It used random Fourier features of a different perspective by viewing these nonlinear features as an equivalent reproducing kernel in a new finite-dimensional reproducing kernel Hilbert space with universal approximation property. The obtained results revealed that the PRIMOS-CKL keeps presenting a high predictive power even under online learning in tasks of system identifications and times series prediction.

The next chapter will discuss the application of the Tikhonov's regularization theory, RKHS and correntropy for recursive online learning through the construction and sequential updating of matrices and their inverses.

# 6 A SPARSE AND ROBUST SOLUTION FOR REGULARIZED NETWORKS

Tikhonov's regularization theory (TIKHONOV; ARSENIN, 1977) and RKHS (ARONSZAJN, 1950) have been explored in several machine learning algorithms, such as regularized networks (RN) (GIROSI *et al.*, 1995), GP (MATTOS *et al.*, 2016) and SVM (SCHÖLKOPF; SMOLA, 2002) to improve their performances in the task of learning from noisy data samples (CHIUSO; PILLONETTO, 2019). These techniques known as kernel methods have a variety of applications, such as noise cancellation, system identification, and channel equalization, and are gaining momentum due to their abilities to handle nonlinear signal processing tasks.

However, techniques based on kernel methods lose accuracy when dealing with large-scale problems, in which the amount of data is huge, or when recursive estimation is needed, such as problems with adaptive control or fault detection. This is due to the increase of the size of the kernel matrix whose dimension is equal to the number of input data. As mentioned in previous chapters, to circumvent this problem, several sparsification procedures have been proposed (WU *et al.*, 2015; ENGEL *et al.*, 2002; SAIDE *et al.*, 2013; ZHAO *et al.*, 2018). However, these procedures are not suitable in a non-stationary scenario in which, in order to track changes in the operational state of the system, the model has to discard stored information that is not relevant anymore (VAN VAERENBERGH *et al.*, 2012; GAO *et al.*, 2013).

In addition to the problems related to the increase of the kernel matrix, the parameter estimation technique has to tackle non-Gaussian noise present in real data. The alternative has been to replace the cost functions based on second-order statistics, such as the MSE, with performance metrics developed from the ITL framework, such as the correntropy (PRINCIPE, 2010; SANTAMARIA *et al.*, 2006; LIU *et al.*, 2007).

In this chapter, it is proposed a variant of the regularized batch network for recursive online learning through the construction and sequential updating of matrices and their inverses. In other words, our goal is to extend the standard batch RN to deal with large-scale, time-varying, and non-Gaussian data processing. This adaptation is based on the following learning strategies.

1. Adaptive filtering, so that the model is built continuously for each new sample of data making the network suitable for recursive online learning;
2. ITL, to allow extracting more information from the data than second-order statistics and providing robustness to non-Gaussian noise;

3. A criterion for adding support vectors, to make the model parsimonious, in which we apply the novelty criterion (PLATT, 1991) to select the support vectors;
4. Explore a new criterion for discarding support vectors (VAN VAERENBERGH *et al.*, 2012), to make the network adaptive and capable of handling time-varying data.

The proposal, henceforth called correntropy-based regularized online network with sparsity (CRONOS) model improves the standard batch by the ability to deal with large-scale data, variations in time and contamination with non-Gaussian noise through a sparsification procedure, that selects, adds and removes support vectors.

In order to evaluate the performance of the CRONOS model computational experiments are carried out as a proof of concept using three data sets in the task of identifying nonlinear systems (two small scales and one large scale). Different levels of contamination by outliers are considered to assess the robustness of the proposed model and a free simulation regime (i.e. with predicted output feedback) is established to assess the generalization capacity.

The rest of this chapter is organized as follows. In Sections 6.1, 6.2 and 6.3 we review the methods used to tackle the challenges faced in this work. Section 6.4 provides the details of the development of the CRONOS model and discusses how to perform the addition and removal of support vectors with low computational cost. Section 6.5 reports the analysis of the computational experiments and discusses the achieved results. The conclusions are presented in Section 6.6.

## 6.1 Regularization Networks on RKHS

Regularized networks (RN) are interpreted as a neural network with a hidden layer, in which it is assumed that the set of input and output data $g = \{(\boldsymbol{x}_t, y_t) \in \mathbb{R}^d \times \mathbb{R}\}$ is obtained by a random sampling of the function $f(\cdot)$, that belongs to the same space as the function $\phi$ defined in $\mathbb{R}^d$, in the presence of noise. For recovering the function $f(\cdot)$, or an approximation, a common choice for solving this problem is to minimize the following function (GIROSI *et al.*, 1995):

$$J(f) = \sum_{t=1}^{N}(f(\boldsymbol{x}_t) - y_t)^2 + \lambda||f||^2_{\mathcal{H}}, \tag{6.1}$$

in which $\lambda$ is a positive number called regularization parameter and $||\cdot||_{\mathcal{H}}$ denotes the norm of a smoothness function in Hilbert's space $\mathcal{H}$. The first term of Eq. (6.1) imposes proximity between the data and the second smoothness, while the regularization parameter controls the fit between these two terms.

The coupling of this regularization theory with that of RKHS allows the definition of the smoothing function in Hilbert space $\mathcal{H}$ in kernel form $\boldsymbol{K}$, as a symmetric and positive definite function (CHIUSO; PILLONETTO, 2019). In this case, the smoothing function can be defined, for example, as $||f||^2_{\mathcal{H}} = \int d\boldsymbol{x} f^2(\boldsymbol{x})/\boldsymbol{K}(\boldsymbol{x})$. Thus, the function that minimizes the Eq. (6.1) has the following form:

$$f(\boldsymbol{x}) = \sum_{t=1}^{N} \alpha_t \boldsymbol{K}(\boldsymbol{x} - \boldsymbol{x}_t) + \sum_{j=1}^{k} d_j \psi_j(\boldsymbol{x}), \tag{6.2}$$

in which the set $\{\psi_j\}_{j=1}^k$ forms a $k$-dimensional basis in the null space of the smoothness function, which in many cases is a polynomial term. The coefficients $d_j$ and $\alpha_t$ depend on the data in such a way that they satisfy the following linear system:

$$(\boldsymbol{K} + \lambda \boldsymbol{I})\boldsymbol{\alpha} + \Psi^\top \boldsymbol{d} = \boldsymbol{y}, \tag{6.3}$$

in which $\boldsymbol{I}$ is the identity matrix and the coefficient $\boldsymbol{\alpha}$ can be determined (neglecting the polynomial term by simplification) as

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}. \tag{6.4}$$

**Remark 6.1**. The RN estimation function in the RKHS defined in this section is based on the MSE which is not robust to outliers. In addition, it is observed in Eq. (6.4) a batch learning process in which to estimate the coefficient $\boldsymbol{\alpha}$ it is necessary to use all the input data to assemble the kernel matrix (Gram Matrix) and a matrix inversion operation with computational cost $\mathcal{O}(N)^3$.

## 6.2 A Criterion for Adding Support Vectors

During the process of learning all the inputs $\boldsymbol{x}$ are collected as support vectors for calculating the kernel matrix $\boldsymbol{K}$. In other words, it grows exponentially and has dimension $N \times N$, according to which $N$ is the total amount of input samples so far. However, in recursive applications, the amount of input data grows to infinity and, hence, the cost for inverting a $N \times N$ kernel matrix $\mathbf{K}$ that grows for each new input-output pair $(\mathbf{x}_t, y_t)$ becomes prohibitive. Several criteria have been used to select the support vectors that will comprise a dictionary $\mathcal{D}^{sv}$ of support vectors (PLATT, 1991; ENGEL *et al.*, 2004; ZHAO *et al.*, 2018; LI; PRÍNCIPE, 2018). Using these criteria, the kernel matrix $\boldsymbol{K}$ grows from time to time by the selective addition of support vectors.

The general idea is to assume that at each instant of time $t$, after $t-1$ observation of training samples, we will assemble a dictionary of $m_t$ ($m_t \ll t$) support vectors comprised of a subset of samples of relevant entries $\mathcal{D}^{sv}_{t-1} = \{\tilde{\boldsymbol{x}}_j\}_{j=1}^{m_t-1}$.

In this chapter, the novelty criterion (PLATT, 1991) is adopted because of its low demand for memory use and simplicity of implementation, desirable characteristics in problems that deal with data sequentially and over time. The novelty criterion can basically be defined as a measure of distance between the current input sample and past samples or dictionary, in which without loss of generality, we simply use the Euclidean distance (LI; PRÍNCIPE, 2018). For each input sample $\boldsymbol{x}_t$ we compute the quadratic distance to its closest neighbor $\tilde{\boldsymbol{x}}$ in the dictionary $\mathcal{D}_t$ its closest neighbors ($\tilde{\boldsymbol{x}}$ in $\mathcal{D}_t$) with a vector quantization threshold $\delta \geq 0$. The incoming sample is considered novel enough to be added to the dictionary if $\min ||\boldsymbol{x}_t - \tilde{\boldsymbol{x}}||^2 > \delta$. Raising this threshold reduces the size of the final dictionary. In the practical applications, a criterion for checking the magnitude of the prediction error is also required and computed by $||\hat{y}_t - y_t|| > \varepsilon$ in which $\hat{y}_t$ is the output predicted by the model. For the incoming sample, $\boldsymbol{x}_t$ to be added to the dictionary the two criteria (novelty and error) must be met simultaneously.

## 6.3 A Criterion for Removing Support Vectors

In a sequential and time-varying learning regime, the input samples $\boldsymbol{x}$ may become less relevant to the model over time. Therefore, removing the least relevant support vectors after adding a new one in the dictionary is important to induce efficient sparsity in the model as explored in previously chapters.

A criterion used to remove support vectors is to select the support vector $i$ that minimizes the Kullback-Leibler (KL) divergence between the exact prediction and the posterior approximation considering the information lost with the support vector removal following the standard setup of GP regression (VAN VAERENBERGH *et al.*, 2012)

$$KL(p(f_t|\mathcal{D}_t)||p([f_t]_t|[f_t]_{-i})p([f_t]_{-i}|\mathcal{D}_t). \tag{6.5}$$

This cost function tries to approximate the full posterior $p(f_t|\mathcal{D}_t)$ with the product of some distribution $p([f_t]_t|[f_t]_{-i})$ in which the $i$-th basis has been removed times $p([f_t]_{-i}|\mathcal{D}_t)$ which does not depend on any data.

This cost function can be evaluated for each SV in the dictionary to determine which basis should be removed. It is possible to compute this KL divergence analytically for every basis in the dictionary, but it is computationally too expensive. Instead, to decrease the computational cost, here we will consider minimizing the square of the error that the approximation induces in the mean of the posterior. This approach produces equally effective results in pruning the least relevant SVs.

Without lack of generality, we first consider the case in which we remove the most recently added SV. This implies approximating the posterior distribution $p(f_t|\mathcal{D}_t)$ with $p([f_t]_t|[f_t]_{-i})p([f_t]_{-i}|\mathcal{D}_t)$ from a probabilistic perspective. For the exact distribution the outputs of $f_t$ known to be $\hat{\boldsymbol{y}}_t$, can be based in Eq. (6.4) in an online learning regime expressed as

$$\hat{\boldsymbol{y}}_t = \boldsymbol{K}_t \boldsymbol{P}_t \boldsymbol{y}_t, \tag{6.6}$$

in which $\boldsymbol{P}_t = (\boldsymbol{K}_t + \lambda I)^{-1}$.

For the approximate distribution, the $\tilde{\boldsymbol{y}}_t$ is

$$\tilde{\boldsymbol{y}}_t = \boldsymbol{K}_t \begin{bmatrix} \boldsymbol{P}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} \boldsymbol{y}_t. \tag{6.7}$$

Subtracting both and simplifying, we can compute the error that the approximation introduced in each element of the posterior output,

$$\hat{\boldsymbol{y}}_t - \tilde{\boldsymbol{y}}_t = \begin{bmatrix} \mathbf{0} \\ [\boldsymbol{P}_t \boldsymbol{y}_t]_t \,/\, [\boldsymbol{P_t}]_{t,t} \end{bmatrix}, \tag{6.8}$$

in which $[\cdot]_t$ and $[\cdot]_{t,t}$ indicate that the last component have been removed (CSATÓ; OPPER, 2000; DE KRUIF; DE VRIES, 2003). It is observed that only the outputs of the removed SV, is distorted. The outputs of the remaining SVs are left unaffected. With this result, we can compute the squared error that would be introduced if we removed each of the SVs, and thus flag for removal the SV that incurs in the least error, using the following criteria for pruning the dictionary:

$$\text{arg min}_i \left( \frac{[\boldsymbol{P}_t \boldsymbol{y}_t]_i}{[\boldsymbol{P_t}]_{i,i}} \right)^2 > \gamma. \tag{6.9}$$

in which $\gamma$ is the threshold that determines whether the support vector that produces the least error will be removed. The notation $[\cdot]_i$ refers to a vector in which the $i$-th row has been removed, and $[\cdot]_{i,i}$ to matrix in which the $i$-th row and column have been removed. Note that the matrix $\boldsymbol{P}_t$ should be built sequentially for each time step $t$. This issue will be addressed in the next section.

## 6.4 Proposed Approach

The main objective in this chapter is to present a variant of the original RN, which is sparse, adaptive and robust to outliers, in order to be capable of modeling dynamic systems from sequential data. The proposed CRONOS model is then based on the merging of the methods discussed in Section 6.1: $(i)$ regularized network (GIROSI *et al.*, 1995), $(ii)$ correntropy (SANTAMARIA *et al.*, 2006), $(iii)$ novelty criterion (PLATT, 1991) and $(iv)$ quadratic error criterion (VAN VAERENBERGH *et al.*, 2012).

The first step is to replace the cost function based on MSE and defined in Eq. (6.1) by the MCC:

$$J(f) = \sum_{t=1}^{N} k_\sigma(f(\boldsymbol{x}_t) - y_t) + \lambda ||f||^2_{\mathcal{H}}, \tag{6.10}$$

in which instead of finding the function $f(\boldsymbol{x}_t)$ that minimizes the Eq. (6.1) it is now desired to find the function $f(\boldsymbol{x}_t)$that maximizes the Eq. (6.10).

Calculating the gradient with respect to $||f||$ using the same methodology described previously (GIROSI *et al.*, 1995) the function that maximizes the Eq. (6.10) will have the coefficient $\tilde{\boldsymbol{\alpha}}$ given by

$$\tilde{\boldsymbol{\alpha}}_t = (\tilde{\boldsymbol{K}}_t \tilde{\boldsymbol{B}}_t + \lambda \sigma_1^2 \tilde{\boldsymbol{I}})^{-1} \tilde{\boldsymbol{B}}_t \tilde{\boldsymbol{y}}_t, \tag{6.11}$$

in which using the matrix inversion property this can be rewritten as

$$\tilde{\boldsymbol{\alpha}}_t = (\tilde{\boldsymbol{K}}_t + \lambda \sigma_1^2 \tilde{\boldsymbol{B}}_t^{-1})^{-1} \tilde{\boldsymbol{y}}_t, \tag{6.12}$$

Now, using the sparsification procedure, the dimension of the elements in Eq. (6.12) is $m_t$ and they are calculated with the sample subset of the dictionary, in which $\tilde{\boldsymbol{K}} \in \mathbb{R}^{m_t \times m_t}$ is an array of kernels whose entries are given by $\tilde{K}_{tj} = \langle \phi(\tilde{\boldsymbol{x}}_t), \phi(\tilde{\boldsymbol{x}}_j) \rangle, \forall\ t, j = 1, \ldots, m_t$. $\tilde{\boldsymbol{B}}_t$ is a diagonal matrix whose diagonal elements are computed as $[\tilde{\boldsymbol{B}}_t]_{j,j} = \rho(e_j), \quad j = 1, \ldots, m_t$.

For making this proposal suitable for recursive learning (note the presence of the subscript '$t$' in the Eq. (6.11) and avoid the high computational cost $\mathcal{O}(m_t^3)$ of matrix inversion we will recursively compute the elements of the Eq. (6.11) to which the updating of the matrices will depend on the sparsification criteria applied.

### *6.4.1 Adding Support Vectors*

If the selection criterion for addition to the dictionary is met, the sample $\boldsymbol{x}_t$ should be added to the dictionary $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \bigcup \{\boldsymbol{x}_t\}$ and $m_t = m_{t-1} + 1$. As a consequence the kernel matrix $\boldsymbol{K}$ should be updated accordingly.

Making $\tilde{\boldsymbol{P}}_t = \boldsymbol{H}_t^{-1} = (\tilde{\boldsymbol{K}}_t + \lambda\sigma_1^2\tilde{\boldsymbol{B}}_t^{-1})^{-1}$, we have

$$\boldsymbol{P}_t = \begin{bmatrix} \tilde{\boldsymbol{K}}_{t-1} + \lambda\sigma_1^2\tilde{\boldsymbol{B}}_{t-1}^{-1} & \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t) \\ \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)^\top & \tilde{k}_{tt} + \lambda\sigma_1^2\tilde{b}_t \end{bmatrix}^{-1}, \tag{6.13}$$

in which

$$\tilde{b}_t = \frac{1}{\rho(e_t)}, \tag{6.14}$$

with

$$e_t = \hat{y}_t - y_t, \tag{6.15}$$

$$\hat{y}_t = \sum_{j=1}^{m_t} \tilde{\alpha}_j k(\boldsymbol{x}_t, \tilde{\boldsymbol{x}}_j) = \tilde{\boldsymbol{\alpha}}^\top \tilde{\boldsymbol{k}}_t, \tag{6.16}$$

$$\tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t) = [k(\boldsymbol{x}_t, \tilde{\boldsymbol{x}}_1) \ \cdots \ k(\boldsymbol{x}_t, \tilde{\boldsymbol{x}}_{m_t})]^\top \tag{6.17}$$

and

$$\tilde{k}_{tt} = k(\tilde{\boldsymbol{x}}_t, \tilde{\boldsymbol{x}}_t). \tag{6.18}$$

It can be observed that

$$\tilde{\boldsymbol{P}}_t^{-1} = \begin{bmatrix} \tilde{\boldsymbol{P}}_{t-1}^{-1} & \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t) \\ \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)^\top & \tilde{k}_{tt} + \lambda\sigma_1^2\tilde{b}_t \end{bmatrix}. \tag{6.19}$$

Using the block matrix inversion identity we get the following expression:

$$\tilde{\boldsymbol{P}}_t = r_t^{-1} \begin{bmatrix} \tilde{\boldsymbol{P}}_{t-1} r_t + \boldsymbol{z}_t\boldsymbol{z}_t^\top & -\boldsymbol{z}_t \\ -\boldsymbol{z}_t^\top & 1 \end{bmatrix}, \tag{6.20}$$

in which $\boldsymbol{z}_t = \tilde{\boldsymbol{P}}_{t-1}\tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)$ and

$$r_t = \lambda\sigma_1^2\tilde{b}_t + \tilde{k}_{tt} - \boldsymbol{z}_t^\top\tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t). \tag{6.21}$$

### *6.4.2 Removing Support Vectors*

Every time a new support vector is added to the dictionary $\mathcal{D}$ we apply the criterion defined in Section 6.3 to determine which support vectors have become of little relevance and if they can be removed from the dictionary $\mathcal{D}_t^{sv} = \mathcal{D}_t^{sv} - \{\boldsymbol{x}_t\}$ and $m_t = m_{t-1} - 1$.

Firstly, we compute the quadratic error for each candidate support vector to be removed using the Eq. (6.9) and then we remove the support vector that produces the smallest error by updating the matrix $\tilde{\boldsymbol{P}}_t$ and eliminating the corresponding input and output pair $(\boldsymbol{x}_t, y_t)$. For updating the matrix $\tilde{\boldsymbol{P}}_t$ with a low computational cost we use a methodology based on the relationship between the inverse of a matrix and its submatrices (JUAREZ-RUIZ *et al.*, 2016).

Thus, we will have a submatrix $\tilde{\boldsymbol{H}}_{\bar{p};\bar{q}}$ obtained from $\tilde{\boldsymbol{H}}_t$ by eliminating the line $p$ and the column $q$. This problem is directly related to the calculation of a disturbed inverse matrix $(\tilde{\boldsymbol{H}}_t + \boldsymbol{Z})^{-1}$, in which $\boldsymbol{Z}$ is the disturbance of a matrix $\tilde{\boldsymbol{H}}_t$. This inverse matrix can be calculated using the Sherman-Morrison-Woodbury formula:

$$\tilde{\boldsymbol{P}}_t = (\tilde{\boldsymbol{H}}_t - \boldsymbol{v}\boldsymbol{u}^\top)^{-1} = \tilde{\boldsymbol{H}}_t^{-1} + \frac{(\tilde{\boldsymbol{H}}_t^{-1}\boldsymbol{v})(\boldsymbol{u}^\top \tilde{\boldsymbol{H}}_t^{-1})}{1 - \boldsymbol{u}^\top \tilde{\boldsymbol{H}}_t^{-1}\boldsymbol{v}}, \tag{6.22}$$

in which $\boldsymbol{v}, \boldsymbol{u} \in \mathbb{R}^{m_t}$ are column vectors of the Sherman-Morrison-Woodbury formula. Therefore, $\tilde{\boldsymbol{H}}_t - \boldsymbol{v}\boldsymbol{u}^\top$ is defined by $\boldsymbol{v} = \tilde{\boldsymbol{H}}_q - \mathbf{c}_p$, in which $\tilde{\boldsymbol{H}}_q$ is the column vector $q$ of $\tilde{\boldsymbol{H}}_t$, $\mathbf{c}_p \in \mathbb{R}^{m_t}$ is the canonical vector of the column $p$, and $\boldsymbol{u} = \mathbf{c}_q$ is the canonical vector of the column $q$. With these settings, $\tilde{\boldsymbol{H}}_t - \boldsymbol{v}\boldsymbol{u}^\top$ is equal to $\tilde{\boldsymbol{H}}_t$ except in the column $q$, that is equal to $\mathbf{c}_p$. By applying the Sherman-Morrison-Woodbury formula in Eq. (6.22) to calculate $(\tilde{\boldsymbol{H}}_t - \boldsymbol{v}\boldsymbol{u}^\top)^{-1}$, $(\tilde{\boldsymbol{H}}_{\bar{p};\bar{q}})^{-1}$ is obtained by eliminating the row $q$ and the column $p$ of $(\tilde{\boldsymbol{H}}_{\bar{p};\bar{q}})^{-1}$ obtaining the update of $\tilde{\boldsymbol{P}}_t$.

The canonical base of the matrix $\tilde{\boldsymbol{H}}$ is formed by vectors that have 1 in one coordinate and 0 in the others, in addition, the row $q$ and the column $p$ are equal, so it is sufficient to determine from a row of $\tilde{\boldsymbol{H}}$ its canonical form.

**Remark 6.2**. Since our proposal uses in the sparsification procedure a criterion to select support vectors that will be removed, we can remove them from the criterion to add support vectors to the error-based condition, keeping only the pure novelty criterion.

Algorithm 4 provides the pseudocode of the algorithm proposed in this chapter.

**Algorithm 4:** Pseudocode CRONOS model.

```
**Hyperparameters**: $\lambda, \sigma_1, \sigma_2, \delta, \gamma$;
**Initialization**: $\tilde{\boldsymbol{P}}_1 = 1/\tilde{k}_{11}; \quad b_1 = 1; \quad m_1 = 1$;
Compute $\tilde{\boldsymbol{\alpha}}_1$ from Eq. (6.12);
Compute $\tilde{b}_1 = 1/\rho(e_1)$ from Eq. (3.4);
**for** $t = 2 : N$,
    **Take an incoming sample**: $(\boldsymbol{x}_t, y_t)$
    **Test to add support vectors**:
    **if** $\min ||\boldsymbol{x}_t - \tilde{\boldsymbol{x}}||^2 > \delta$
        $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \bigcup \{\boldsymbol{x}_t\}$;
        $\boldsymbol{z}_t = \tilde{\boldsymbol{P}}_{t-1}\tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)$;
        $\tilde{b}_t = 1/\rho(e_t)$ from Eq. (3.4);
        $r_t = \lambda\sigma_1^2\tilde{b}_t + \tilde{k}_{tt} - \boldsymbol{z}_t^\top \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)$;
        Compute $\tilde{\boldsymbol{P}}_t$ from Eq. (6.20);
        Compute $\tilde{\boldsymbol{\alpha}}_t$ from Eq. (6.12);
        $m_t = m_{t-1} + 1$;
        **Test to remove vector-support**:
        **if** $\min([\tilde{\boldsymbol{P}}_t\tilde{\boldsymbol{y}}_t]_t/[\tilde{\boldsymbol{P}}]_{i,i})^2 > \gamma$
            $\mathcal{D}_t^{sv} = \mathcal{D}_t^{sv} - \{\boldsymbol{x}_t\}$;
            Update $\tilde{\boldsymbol{P}}_t$ from Eq. (6.22);
            Update $\tilde{\boldsymbol{\alpha}}_t$ from Eq. (6.12);
            $m_t = m_{t-1} - 1$;
        **end if**
    **else**
        $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv}$;
    **end if**
**end for**
**Output** $\tilde{\boldsymbol{\alpha}}_t, \mathcal{D}_t^{sv}$.
```

## 6.5 Results and Discussion

In this section, it is reported the results of a set of computer simulations of the nonlinear system identification task. For this task, it is assumed the same NARX structure for discrete nonlinear model described in previous chapters of this thesis. In this case, the input vector $\boldsymbol{x}_t$ is obtained by concatenating $L_y$ observable past output $y_t \in \mathbb{R}$ and $L_u$ past controlled inputs $u_t \in \mathbb{R}$ within a single regression vector:

$$\boldsymbol{x}_t = [y_{t-1}, \dots, y_{t-L_y}, u_{t-1}, \dots, u_{t-L_u}]^\top. \tag{6.23}$$

Thus, its assumed that the training samples are sequentially made available:

$$\mathcal{D}_t = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_t, y_t)\}, \tag{6.24}$$

in which $(\boldsymbol{x}_t, y_t) \in \mathbb{R}^{L_u+L_y} \times \mathbb{R}$ denotes the input and output pair of the system.

Table 9 – Summary of data sets.

| data sets | Number of samples | | NARX order | | Kernel | |
|---|---|---|---|---|---|---|
| | Train | Test | $L_u$ | $L_y$ | Type | Width ($\sigma_2$) |
| Artificial 1 | 200 | 200 | 3 | 2 | Gaussian | 0.1 |
| Artificial 2 | 300 | 100 | 1 | 1 | Gaussian | 0.1 |
| Silverbox | 45,000 | 40,000 | 10 | 10 | Polynomial | 1 |

The experiments involve two sets of artificial data of small scale and one of large scale, in which two scenarios are considered, without and with contamination by outliers. The contamination by outliers allows evaluating the performance of the models in the presence of non-Gaussian noise. The figures of merit explored in this work are the root mean square error (RMSE) and the degree of sparsity (dictionary size). All evaluations of the model during the test phase correspond to the free simulation regime, with predicted output feedback, i.e., the values of the predicted output are returned to the regression vector. This evaluation regime is much more demanding than the usual one-step-ahead prediction. In these experiments, all data are rescaled to the interval $[-1, 1]$.

It is assessed the performance of the CRONOS proposal in comparison with the standard regularized network (GIROSI *et al.*, 1995) and with KRMC (WU *et al.*, 2015). For the sake of fairness, the evaluated KRMC model also uses the novelty criterion for inducing sparsity.

### *6.5.1 Small-scale Data Sets*

In this first experiment, two small-scale artificial data sets are used (NARENDRA; PARTHASARATHY, 1990). The data sets *Artificial 1* and *Artificial 2* are generated as presented in Subsection 3.3.1.

This experiment evaluated the results for both scenarios, one free of contamination from outliers and other with this kind of contamination. In the contaminated scenario, the output samples $y$ of the training are randomly contaminated with the percentages of 10%, 20%, and 30%. The outliers are generated as an impulsive noise with *pdf* from a Gaussian mixture given by $p_z(z) = (1-q)\times\mathcal{N}(0, 0, 065) + q\times\mathcal{N}(0.1, 2\sigma_1(y))$, in which $q$ is determined according to the percentage of outliers in the scenario and $\sigma_1(y)$ is the standard deviation of the original training data. In addition to that, 10 independent training/testing runs are executed, each run with a different distribution of contamination, and the hyperparameters are adjusted by grid searching. Table 9 contains the number of training and test samples, the NARX order of the model, and the kernel type and width adjusted according to the state of the art (NARENDRA; PARTHASARATHY, 1990).

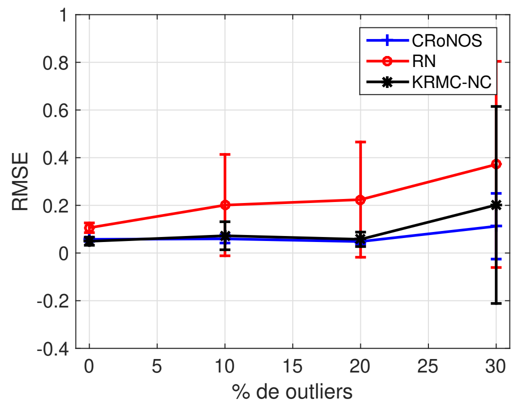
Figures 13(a) and 13(c) report the RMSE results, in which we show the average values and standard deviation for each data set in the proposed scenarios. These figures reveal that the RN based on the MSE cost function is not able to maintain a good predictive capacity of the model as the contamination by outliers increases. On the other hand, comparing the performance of the proposed CRONOS model with the KRMC-NC, we verified similar numerical results, but in the scenario of maximum contamination (30%) the former presents superior performance than the latter. The CRONOS model, however, was able to build smaller dictionaries than the KRMC-NC model, while keeping similar numerical performances. As can be seen in Figures 13(b) and 13(d) in which the average number of support vectors selected by each model is presented.

### 6.5.2 Large-scale Data Set

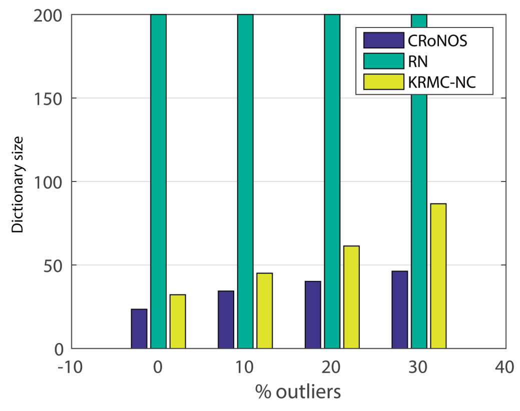
This experiment used the Silverbox data set, already described in Chapter 4. (WIGREN; SCHOUKENS, 2013). This data set represents an electrical circuit simulating a mass-spring damping system and contains a total of 131,072 samples. We used the first 40,000 samples for model evaluation and 45,000 of the rest for model training. Two scenarios are evaluated, one without contamination by outliers and the other with contamination by outliers of 5% using a non-Gaussian noise by sampling a $t$-distribution with zero mean and 2 degrees of freedom. For this experiment, 5 independent runs are performed, in which each round has a different distribution of contamination, and the hyperparameters are also adjusted by grid searching. Table 9 contains the number of training and test samples, the NARX order of the model, and the kernel type and width adjusted according to the state of the art (WIGREN; SCHOUKENS, 2013).

The results of this experiment are presented in Figures 14(a) and 14(b), in which it is presented, respectively, through a boxplot of the RMSE values, and through the bar graph the average size of the dictionary for free simulation prediction of the 10 independent runs. In this experiment, we compare the performance of the proposed CRONOS model only with the KRMC-NC model because for problems with large amounts of data the standard (i.e. batch) RN model grows excessively due to the use of all pairs of input and output samples. Comparing

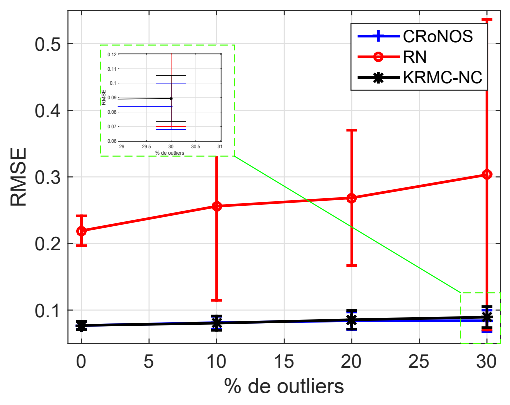
Figure 13 – (a) and (c) Plots of the RMSE values related to the test in free simulation with the data containing different levels of contamination by outliers. (b) and (d) Plots of the average size of the dictionary.



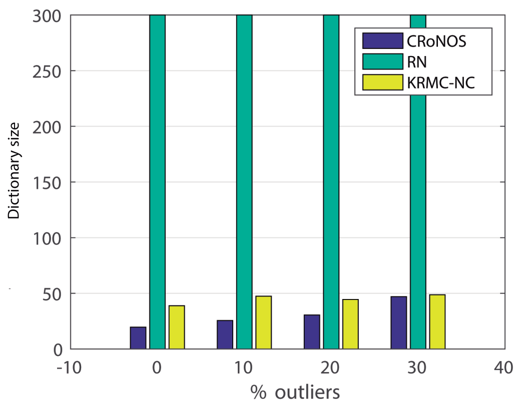
(a) Artificial 1 - RMSE.



(b) Artificial 1 - Dictionary.



(c) Artificial 2 - RMSE.



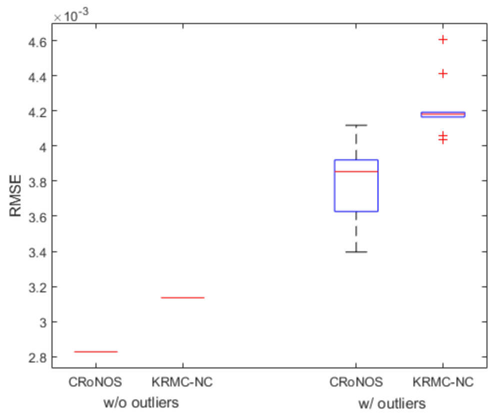
(d) Artificial 2 - Dictionary.

Source: prepared by the author (2021).

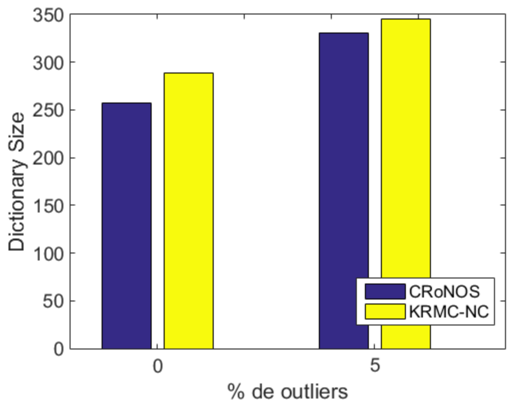
the performances of CRONOS and KRMC-NC, it can be observed that the CRONOS model outperforms the KRMC-NC in terms of predictive capacity (smaller RMSE values) and also in terms of building smaller dictionaries.

A dictionary of smaller size leads to a smaller norm of the coefficient vector $\boldsymbol{\alpha}$ estimated in the construction of the model, in which we obtain an average norm of $1.6$ for CRONOS and $4.8$ for KRMC-NC. The smaller norm of the coefficient vector of the CRONOS model causes the model to propagate smaller prediction errors in the free simulation regime, as can be seen in Figures 15(a) and 15(b), in which it is possible to confirm the best fit of the prediction curve produced by CRONOS.

Figure 14 – (a) Boxplot of the RMSE values related to the free simulation test with data containing different levels of contamination by outliers. (b) is the graph of the average size of the support vector dictionary.



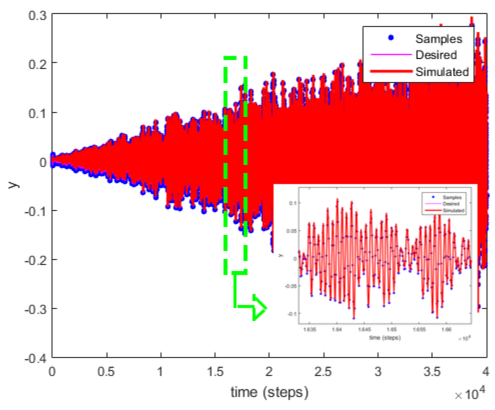
(a) Silverbox - RMSE.



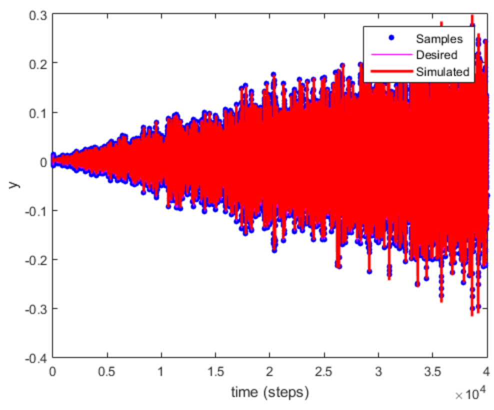
(b) Silverbox - Dictionary.

Source: prepared by the author (2021).

Figure 15 – Predicted output for the free simulation test with outliers using CRONOS (a) and KRNC-NC (b).



(a) CRONOS.



(b) KRMC-NC.

Source: prepared by the author (2021).

## 6.6 Conclusions

This chapter presented a new proposal for a class of neural network models known as regularized networks. The proposal, called CRONOS, has characteristics to be recursive, sparse, robust to outliers, and adaptive.

The CRONOS learning model integrates the regularized network architecture in the RKHS, from a perspective of kernel adaptive filtering methods, with the MCC. This proposal improves the classic RN in important dimensions: $(i)$ it can be used efficiently for recursive learning online; $(ii)$ its computational complexity is considerably reduced by the introduction of a sparsification procedure; $(iii)$ this sparsification procedure can in addition to selecting the support vectors that will be part of the dictionary can also discard kernel functions that are not relevant to the model.

The CRONOS model was evaluated through the task of identifying nonlinear systems using two small-scale artificial data sets and one large-scale data set (Silverbox), in a free simulation regime with training samples contaminated with outliers. Our experiments as a proof of concept clearly showed the following benefits:

1. Reduction of computational complexity through the recursive strategy of building and updating matrices;
2. Increased robustness to outliers of the model by replacing the MSE cost function with the MCC;
3. Reduced memory usage due to a hybrid sparsification procedure that allows a dictionary of relevant data samples to grow and shrink as time passes.

In summary, the obtained results have shown that the proposed model is capable of recursive learning and maintaining a high predictive power in a free simulation regime with a reduced dictionary size.

Inspired by the spatiotemporal mapping ability of the reservoir computing and the robustness of MCC, we will introduce in the next chapter a robust kernel adaptive algorithm that incorporates the advantages of the online and sparse KRMC algorithm with a dynamic reservoir of echo states.

# 7 KERNEL ADAPTIVE FILTERING AND RESERVOIR COMPUTING

It has been shown in the previous chapters that *kernel adaptive filtering* (KUNG, 2014) and *information-theoretic learning* (PRINCIPE, 2010) have been successfully applied to address the issues of nonlinear and non-Gaussian signal processing, especially in system identification. In particular, the *maximum correntropy criterion* finds successful applications when used for developing robust adaptive filters, such as the kernel maximum correntropy (KMC) (ZHAO *et al.*, 2011), the correntropy kernel learning (CKL) (LIU; CHEN, 2013) and the kernel recursive maximum correntropy (KRMC) (WU *et al.*, 2015).

It should be recalled that the KAF framework under the ITL paradigm provides the model with the ability to deal with nonlinear systems via the kernel trick, which transforms data from the original data space to a possibly infinite-dimensional RKHS. The ITL paradigm allows extracting more information from the data for system identification than second-order statistics as correlation and the MSE (LIU; CHEN, 2013) in which the MCC, more specifically, provides outlier-robustness to adaptive filter algorithms.

As we have seen, a characteristic of KAF algorithms is that the dimensions of the kernel matrix and, hence, the number of terms in the prediction model is equal to the number of input samples, making the resulting kernel filters unsuitable for continual online learning applications[1]. Therefore, a typical alternative used in online applications is to divide model building in two stages, a stage to control the size of the model and a subsequent recursive parameter estimation stage (DUARTE; BARRETO, 2019).

In terms of architecture, KAF models can be understood as a time-delay single-layer feedforward neural network (TDFNN). As such, they are incapable of capturing long term temporal dependencies of nonlinear dynamic systems (KREMER, 2001). A TDFNN learns a static mapping in which the outputs of the network depend solely on the current short-term window comprised of time-delayed inputs, thereby neglecting the dependency between samples separated in time by wider lags. For handling long temporal dependencies, it is possible to incorporate a reservoir computing (RC) framework into the KAF algorithm (ZHOU *et al.*, 2018; SHI; HAN, 2007). The reservoir is used typically in echo state networks (ESN) (LUKOSEVICIUS; JAEGER, 2009) and may provide the KAF algorithm with the ability to perform a spatiotemporal mapping that transforms the past inputs into a high-dimensional reservoir state space. Furthermore, as the reservoir is randomly generated and stays fixed

[1] For short signals, such kernel adaptive filters are suitable.

during all learning processes, the dynamic reservoir has a fixed recurrent topology and thus only the output weights need to be estimated.

Thus, in this chapter we propose a robust kernel adaptive algorithm that incorporates the advantages of the online and sparse KRMC algorithm with a dynamic reservoir of states, generating a novel echo state kernel recursive maximum correntropy (ES-KRMC) algorithm. The proposed approach uses an ESN network that allows the KRMC to deal with dynamical systems in which the output depends not only on the history of the inputs but also on the output feedback. For this purpose, it is used a correntropy-based feedback criterion (CFC) to promote a stable dynamical reservoir.

It is carried out a comprehensive evaluation of the proposed algorithm using four real-world benchmarking data sets for nonlinear system identification. Including, three smaller-scale data sets and a large-scale one. Different levels of outlier contamination are considered in order to assess the robustness of the proposed algorithm. The results achieved by the proposed ES-KRMC algorithm highlight its ability to keep a high predictive power under an online learning regime in comparison to several state-of-the-art alternatives.

The remainder of this chapter is organized as follows. In Section 7.2 we introduce online echo state networks with kernel adaptive filtering. In Section 7.3 we describe the KRMC algorithm for nonlinear system identification applications. In Section 7.4 we present the outlier-robust formulation of the proposed ES-KRMC model. In Section 7.5 we report the computer experiments and discuss the results of the evaluated models. The chapter is concluded in Section 7.6.

## 7.1 Echo State Network

The ESN is a recurrent neural network inserted under the context of reservoir computing. This dynamic learning paradigm is based on an internal reservoir with random and sparse connectivity in order to extract long lasting temporal information which may be useful for dynamical system modeling. ESN is typically comprised of randomly connected hidden neurons with sigmoid-like activation function. In practice, the network is used only as a reservoir of dynamics, and only the readout layer is effectively trained (JAEGER, 2001). Figure 16 illustrates the basic topology of ESN, in which the internal reservoir state at each time step is updated by

$$\boldsymbol{x}_t = h(\boldsymbol{W}_{in}\boldsymbol{u}_t + \boldsymbol{W}\boldsymbol{x}_{t-1}), \quad t = 1, \ldots, N, \tag{7.1}$$

in which $\boldsymbol{x}_t \in \mathbb{R}^{N_x}$ is a vector of activations of the reservoir neurons at time step $t$, $\boldsymbol{u}_t \in \mathbb{R}^{N_u}$ is the input vector fed to the network, $h(\cdot)$ is the neuron activation function, in which in this chapter we use the symmetric $tanh(\cdot)$, applied element-wise, $\boldsymbol{W}_{in} \in \mathbb{R}^{N_x \times N_u}$ is the input weight matrix and $\boldsymbol{W} \in \mathbb{R}^{N_x \times N_x}$ is a weight matrix of internal reservoir connections. The network is usually started with the initial state $\boldsymbol{x}_0 = \boldsymbol{0}$. *Bias values are omitted in Eq. (7.1).* In the readout (i.e., output) layer $\hat{y}_t$ is implemented with the following form

$$\hat{y}_t = \boldsymbol{W}_{out}\boldsymbol{\psi}_t, \quad t = 1, \ldots, N, \tag{7.2}$$

in which $\boldsymbol{W}_{out} \in \mathbb{R}^{1 \times N_\psi}$ is the vector of output weights. Typically, $N_x \gg N_\psi$ and we will often consider $\boldsymbol{\psi}_t = [\boldsymbol{u}_t; \boldsymbol{x}_t] \in \mathbb{R}^{N_u + N_x}$ the concatenation of the input vector and the internal state.

In many nonlinear dynamic systems, the function to be learned depends not only on the history of the input but also on the history of the output (JAEGER, 2001). In this case the state equation shown in Eq. (7.1) can be extended to

$$\boldsymbol{x}_t = h(\boldsymbol{W}_{in}\boldsymbol{u}_t + \boldsymbol{W}\boldsymbol{x}_{t-1} + \boldsymbol{W}_{back}\hat{y}_{t-1}), \quad t = 1, \ldots, N, \tag{7.3}$$

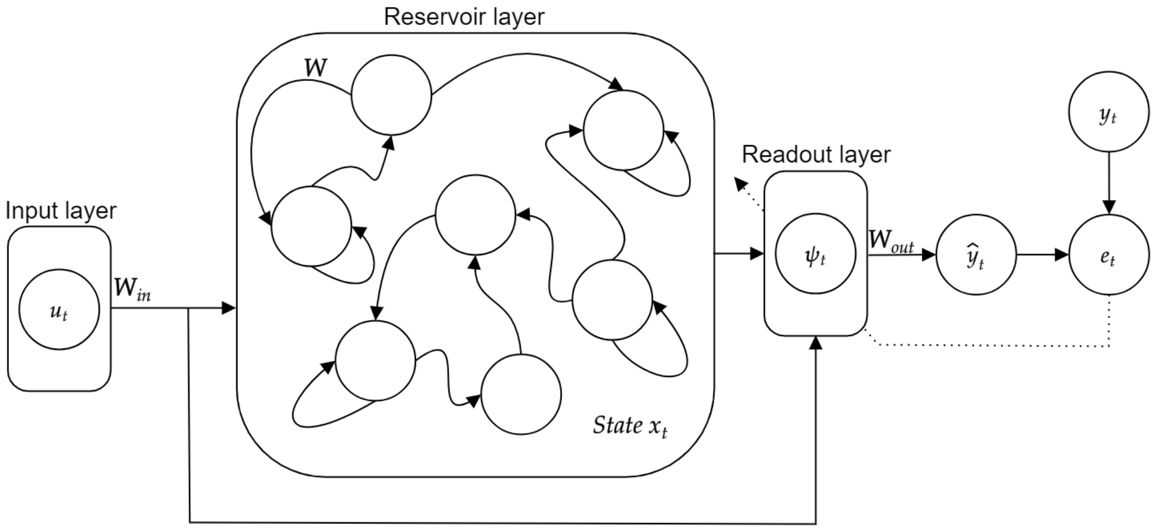
in which $\boldsymbol{W}_{back} \in \mathbb{R}^{N_x \times 1}$ is an output feedback weight vector. The augmented vector $\boldsymbol{\psi}_t$ is redefined as $\boldsymbol{\psi}_t = [\boldsymbol{u}_t; \boldsymbol{x}_t; \hat{y}_{t-1}] \in \mathbb{R}^{N_u + N_x + 1}$ concatenating also the previous predicted output $\hat{y}_{t-1}$. The ESN can exhibit unstable behavior due to the amplification of errors due to the output feedback loop. This problem can be avoided using a teacher forcing learning strategy replacing the previous predicted output $\hat{y}_{t-1}$ with its actual observed value $y_{t-1}$. (JAEGER; HAAS, 2004).

The original and most popular batch training method to compute $\boldsymbol{W}_{out}$ is the OLS estimation method which may suffer from ill-posed and over-fitting problems (HAYKIN, 2009). Lukoševičius (2012) presents practical techniques and recommendations for successfully applying ESNs, as well as some more advanced application-specific modifications. A computationally cheaper online training alternative is discussed in the next section.

## 7.2 Online Learning for ESN Models

Many applications require methods for tracking time-varying dynamics. In these applications, new data are available sequentially and are used typically online estimation algorithms with recursive strategies to estimate the parameters and states of a model. In ESN

Figure 16 – Basic network topology of ESN at iteration $t$.



Source: prepared by the author (2021).

context $\boldsymbol{W}_{out}$ acts as an adaptive linear combiner in which the simplest way to train $\boldsymbol{W}_{out}$ would be to use LMS algorithm that incrementally adjusts the output weights (HASSIBI, 2003). However, the convergence performance of LMS is severely impaired by large eigenvalue spreads of the correlation matrix of the reservoir state (LUKOSEVICIUS; JAEGER, 2009).

A well-known alternative to LMS is the RLS algorithm that is insensitive to the detrimental effects of eigenvalue spread and boosts a much faster convergence because it takes into account information extracted from second-order statistics (i.e., correlations). In its simplest form, the RLS algorithm minimizes the sum of squared estimation errors, at each time step $t$, yielding the cost function

$$J(\boldsymbol{W}_{out}) = \sum_{j=1}^{t}(y_j - \boldsymbol{W}_{out}\boldsymbol{\psi}_j)^2. \tag{7.4}$$

The RLS algorithm usually leads to acceptable performance of the ESN model. However, the accuracy is impaired by an inversely proportional relationship between the nonlinearity of the mapping and the memory capability with respect to the reservoir.The larger the nonlinearity of the reservoir, the lesser its memory capacity. (VERSTRAETEN *et al.*, 2010). A solution proposed in Zhou *et al.* (2018) incorporates a dynamic reservoir into the KRLS algorithm. Thus, the readout layer is constructed in RKHS, with the reservoir tuned to perform a temporal mapping, while a kernel-induced mapping focuses on the nonlinear and static transformation of the reservoir state.

### 7.2.1 The Readout Layer in RKHS

It must be considered the problem of learning a nonlinear mapping $f : \mathcal{U} \rightarrow \mathbb{R}$ based on a sequence of input-output pair $\{\boldsymbol{u}_j, y_j\}_{j=1}^{t}$, in which $\mathcal{U}$ is a compact input domain in $\mathbb{R}^d$, $\boldsymbol{u}_j \in \mathcal{U}$ is the input vector and $y_j \in \mathbb{R}$ is the corresponding target output. The hypothesis space for learning is assumed to be an RKHS associated with a positive definite kernel. Thus, $\kappa_{\sigma_2}(\cdot)$ denotes the kernel function for RKHS, with $\sigma_2$ being the kernel size. Kernel function can be, for example, a Gaussian or polynomial kernel function. According to properties of positive definite kernels (HAYKIN, 2009), any positive definite kernel induces a mapping $\boldsymbol{\phi}$ from input space to a high and possibly infinite-dimensional feature space (RKHS). The inner products in the feature space can be calculated by the well-known kernel trick, that is $\kappa_{\sigma_2}(\boldsymbol{u}_t, \boldsymbol{u}_j) = \kappa_{\sigma_2}(\boldsymbol{u}_t - \boldsymbol{u}_j) = \boldsymbol{\phi}_t^{\top}\boldsymbol{\phi}_j$, in which $\boldsymbol{\phi}_t$ stands for $\boldsymbol{\phi}(\boldsymbol{u}_t)$. To guarantee the existence of the inverse of the data autocorrelation matrix especially during the initial update stages, a norm penalizing term is introduced. The learning problem in feature space is then to find a high-dimensional weight vector $\boldsymbol{\Omega}$ by minimizing

$$\min_{\boldsymbol{\Omega}_t} J = \sum_{j=1}^{t}(y_j - \boldsymbol{\Omega}_t^{\top}\phi(\boldsymbol{u}_j))^2 + \gamma_1\|\boldsymbol{\Omega}_t\|^2, \tag{7.5}$$

in which $\gamma_1$ is a positive number, called the regularization factor.

The computational separation between the reservoir and the readout layer makes it possible to incorporate a reservoir into the KRLS algorithm. Therefore, the readout layer is reformulated as a linear model in RKHS with the following form

$$y_t = g_t(\boldsymbol{x}_t, \boldsymbol{u}_t) = g_t(\boldsymbol{\psi}_t) = \boldsymbol{\Omega}_t^{\top}\phi(\boldsymbol{\psi}_t). \tag{7.6}$$

Thus, the Eq. (7.5) can be reformulated as

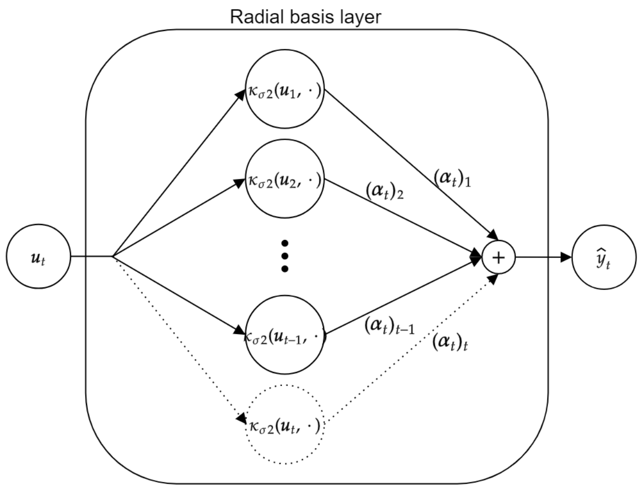
$$\min_{\boldsymbol{\Omega}_t} J = \sum_{j=1}^{t}(y_j - \boldsymbol{\Omega}_t^{\top}\phi(\boldsymbol{\psi}_j))^2 + \gamma_1\|\boldsymbol{\Omega}_t\|^2. \tag{7.7}$$

The solution to this problem can be derived as

$$\boldsymbol{\Omega}_t = (\boldsymbol{\Phi}_t\boldsymbol{\Phi}_t^{\top} + \gamma_1\boldsymbol{I})^{-1}\boldsymbol{\Phi}_t\boldsymbol{y}_t, \tag{7.8}$$

in which $\boldsymbol{\Phi}_t = [\phi(\boldsymbol{\psi}_1), \phi(\boldsymbol{\psi}_2), \ldots, \phi(\boldsymbol{\psi}_t)], t = 1, ..., N$, $\mathbf{I} \in \mathbb{R}^{t\times t}$ and $\boldsymbol{y}_t = [y_1, y_2, \cdots, y_t]^{\top}$. By the *Representer Theorem* (MICCHELLI *et al.*, 2006), the KRLS produces an RBF type network, which is a linear combination of the kernel functions as can be seen in Fig. 17 in which $\boldsymbol{\alpha}_t$ denotes the coefficient vector of the network at iteration $t$, $(\boldsymbol{\alpha}_t)_j$ denotes the $j$-th

Figure 17 – Basic network topology of KRLS or KRMC at iteration $t$.


Radial basis layer
$u_t$
$\kappa_{\sigma 2}(u_1,\cdot)$
$\kappa_{\sigma 2}(u_2,\cdot)$
$\kappa_{\sigma 2}(u_{t-1},\cdot)$
$\kappa_{\sigma 2}(u_t,\cdot)$
$(\alpha_t)_1$
$(\alpha_t)_2$
$(\alpha_t)_{t-1}$
$(\alpha_t)_t$
+
$\hat{y}_t$

Source: prepared by the author (2021).

component of $\boldsymbol{\alpha}_t$. The update equation of the combination coefficients can be derived as follows (LIU *et al.*, 2010)

$$\begin{aligned}\boldsymbol{\alpha}_t = \boldsymbol{P}_t\boldsymbol{y}_t = r_t^{-1}\begin{bmatrix}\boldsymbol{P}_{t-1}+\boldsymbol{z}_t\boldsymbol{z}_t^\top & -\boldsymbol{z}_t\\ -\boldsymbol{z}_t^\top & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{y}_{t-1}\\ y_t\end{bmatrix}\\ = r_t^{-1}\begin{bmatrix}\boldsymbol{\alpha}_{t-1}-\boldsymbol{z}_t e_t\\ e_t\end{bmatrix},\end{aligned} \tag{7.9}$$

in which $e_t = y_t - \phi(\boldsymbol{\psi}_t)^\top\boldsymbol{\Phi}_{t-1}\boldsymbol{\alpha}_{t-1}$ is the prediction error, $\boldsymbol{P}_t = [\boldsymbol{\Phi}_t^\top\boldsymbol{\Phi}_t + \gamma_1\boldsymbol{t}]^{-1}$, $\boldsymbol{z}_t = \boldsymbol{P}_{t-1}(\boldsymbol{\Phi}_{t-1}^\top\phi(\boldsymbol{\psi}_t))$, $\boldsymbol{y}_{t-1} = [y_1, y_2, \cdots, y_{t-1}]^\top$, $\boldsymbol{r}_t = \gamma_1 + \phi(\boldsymbol{\psi}_t)^\top\phi(\boldsymbol{\psi}_t) - \boldsymbol{z}_t^\top\boldsymbol{h}_t$ and $\boldsymbol{h}_t = [\kappa_{\sigma_2}(\boldsymbol{\psi}_t - \boldsymbol{\psi}_1), \cdots, \kappa_{\sigma_2}(\boldsymbol{\psi}_t - \boldsymbol{\psi}_{t-1})]^\top$.

From the exposed so far, the following issues are important to be mentioned due to their practical implications.

**Remark 7.1**. The solution of the KRLS-type algorithm in Eq. (7.8) is based on a second-order statistics of the error measure. Hence, this solution is not robust to non-Gaussian noise processing, especially in impulsive noise environments.

**Remark 7.2**. By comparing the Eq. (7.1) with the Eq. (7.3), the latter is more interesting for nonlinear system identification than the former because it models the input-output dynamics.

**Remark 7.3**. As pointed out in Jaeger e Haas (2004), the output feedback can destabilize the reservoir making it difficult for the algorithm to converge.

## 7.3 Kernel Recursive Maximum Correntropy

The KRMC model employs the MCC to derive a KRLS-type algorithm, in which the KRMC inherits the advantages of both, hence it amalgamates a high predictive power for nonlinear systems with outlier-robustness. The KRMC model and the KRLS produce an RBF type network, as can be seen in Fig. 17. Proposed in Wu *et al.* (2015) it uses an exponentially-weighted mechanism assigning more emphasis on recent data and de-emphasizing data from the remote past. Therefore, Eq. (7.5) can be reformulated as following cost function

$$\max_{\boldsymbol{\Omega}_t} J = \sum_{j=1}^{t} \beta^{t-j} G_{\sigma_1}(y_j - \boldsymbol{\Omega}_t^\top \phi(\boldsymbol{x}_j)) - \frac{1}{2}\beta^t \gamma_2 \|\boldsymbol{\Omega}_t\|^2, \tag{7.10}$$

in which $\beta$ is the forgetting factor, usually $0 \ll \beta \leq 1$, $\gamma_2$ is the regularization factor and

$$G_{\sigma_1}(y_j - \boldsymbol{\Omega}_t^\top \phi(\boldsymbol{x}_j)) = \exp\left(-\frac{(y_j - \boldsymbol{\Omega}_t^\top \phi(\boldsymbol{x}_j))^2}{2\sigma_1^2}\right). \tag{7.11}$$

Setting its gradient with respect to $\boldsymbol{\Omega}_t$ equals to zero, one can obtain the solution

$$\boldsymbol{\Omega}_t = (\boldsymbol{\Phi}_t \boldsymbol{B}_t \boldsymbol{\Phi}_t^\top + \gamma_2 \beta^t \sigma_1^2 \boldsymbol{I})^{-1} \boldsymbol{\Phi}_t \boldsymbol{B}_t \boldsymbol{y}_t, \tag{7.12}$$

in which $\boldsymbol{B}_t \in \mathbb{R}^N \times \mathbb{R}^N$ is a diagonal matrix whose diagonal elements are $B_{t,t} = \beta^{t-1} \exp(-((y_t - \boldsymbol{\Omega}^\top \phi(\boldsymbol{x}_j))^2)/2\sigma_1^2), t = 1, ..., N$. Using the matrix inversion lemma with the following associations $\gamma_2 \beta^t \sigma_1^3 \boldsymbol{I} \to A$, $\boldsymbol{\Phi}_t \to B$, $\boldsymbol{\Phi}_t^\top \to D$, we arrive at

$$(\boldsymbol{\Phi}_t \boldsymbol{B}_t \boldsymbol{\Phi}_t^\top + \gamma_2 \beta^t \sigma_1^2 \boldsymbol{I})^{-1} \boldsymbol{\Phi}_t \boldsymbol{B}_t = \boldsymbol{\Phi}_t (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \gamma_2 \beta^t \sigma_1^2 \boldsymbol{B}^{-1})^{-1}. \tag{7.13}$$

Substituting the above result into Eq. (7.12) yields

$$\boldsymbol{\Omega}_t = \boldsymbol{\Phi}_t (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \gamma_2 \beta^t \sigma_1^2 \boldsymbol{B}^{-1})^{-1} \boldsymbol{y}_t. \tag{7.14}$$

The weight vector can be expressed explicitly as a linear combination of the transformed data, that is, $\boldsymbol{\Omega}_t = \boldsymbol{\Phi}_t \boldsymbol{\alpha}_t$, in which the coefficients vector $\boldsymbol{\alpha}_t = (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \gamma_2 \beta^t \sigma_1^2 \boldsymbol{B}^{-1})^{-1} \boldsymbol{y}_t$ can be updated using the kernel trick.

By denoting $\boldsymbol{P}_t = (\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \gamma_2 \beta^t \sigma_1^2 \boldsymbol{B}^{-1})^{-1}$, we have

$$\boldsymbol{P}_t = \begin{bmatrix} \boldsymbol{\Phi}_{t-1}^\top \boldsymbol{\Phi}_{t-1} + \gamma_2 \beta^t \sigma_1^2 \boldsymbol{B}_{t-1}^{-1} & \boldsymbol{\Phi}_{t-1}^\top \phi(\boldsymbol{x}_t) \\ \phi(\boldsymbol{x}_t)^\top \boldsymbol{\Phi}_{t-1} & \phi(\boldsymbol{x}_t)^\top \phi(\boldsymbol{x}_t) + \gamma_2 \beta^t \sigma_1^2 \theta_t \end{bmatrix}^{-1}. \tag{7.15}$$

If we take $V_t = (\exp(-(e_t^2/2\sigma_1^2)))^{-1}$ and $e_t = y_t - \boldsymbol{\Omega}^\top \phi(\boldsymbol{x}_t)$, one can observe that

$$\boldsymbol{P}_t^{-1} = \begin{bmatrix} \boldsymbol{P}_{t-1}^{-1} & \boldsymbol{h}_t \\ \boldsymbol{h}_t^\top & \gamma_2 \beta^t \sigma_1^2 V_t + \phi(\boldsymbol{x}_t)^\top \phi(\boldsymbol{x}_t) \end{bmatrix}, \tag{7.16}$$

in which $\boldsymbol{h}_t = \boldsymbol{\Phi}_{t-1}^\top \phi(\boldsymbol{x}_t)$. Using the block matrix inversion identity, we obtain the following expression:

$$\boldsymbol{P}_t = r_t^{-1} \begin{bmatrix} \boldsymbol{P}_{t-1} r_t + \boldsymbol{z}_t \boldsymbol{z}_t^\top & -\boldsymbol{z}_t \\ -\boldsymbol{z}_t^\top & 1 \end{bmatrix}, \tag{7.17}$$

in which $\boldsymbol{z}_t = \boldsymbol{P}_{t-1} \boldsymbol{h}_t$ and

$$r_t = \gamma_2 \beta^t \sigma_1^2 V_t + \phi(\boldsymbol{x}_t)^\top \phi(\boldsymbol{x}_t) - \boldsymbol{z}_t^\top \boldsymbol{h}_t. \tag{7.18}$$

Then we obtain the KRMC algorithm for nonlinear system identification, in which the coefficients update follows the same form as in Eq. (7.9), except that now $r_t$ is computed as in Eq. (7.18). Furthermore, the prediction $\hat{y}_t$ for the sample $\boldsymbol{x}_t$ based on kernel trick is computed as

$$\hat{y}_t = \sum_{j=1}^{N} \alpha_j \kappa(\boldsymbol{x}_t, \boldsymbol{x}_j) = \boldsymbol{\alpha}^\top \boldsymbol{\kappa}_t, \tag{7.19}$$

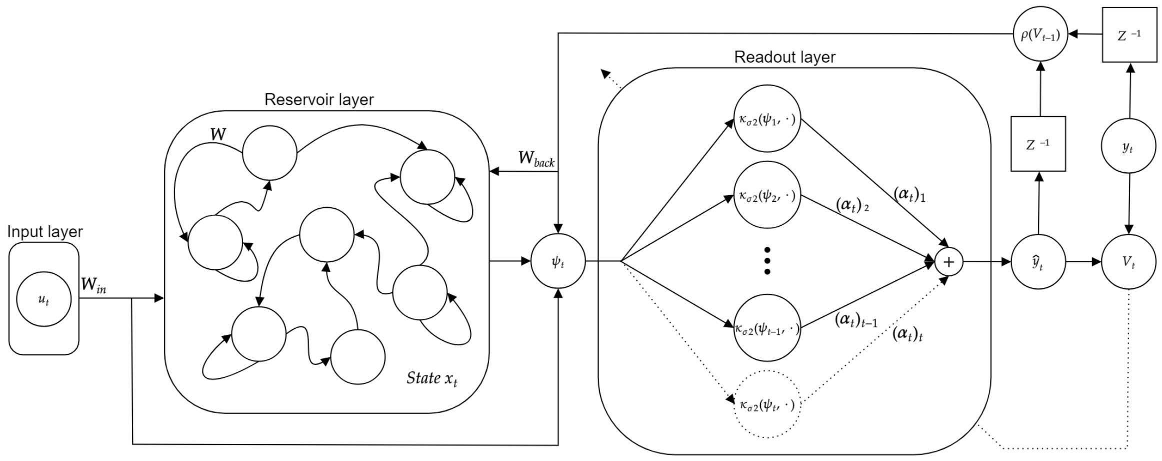
in which $\boldsymbol{\kappa}_t = [\kappa(\boldsymbol{x}_t, \boldsymbol{x}_1) \ \cdots \ \kappa(\boldsymbol{x}_t, \boldsymbol{x}_N)]^\top$ is the kernel vector.

## 7.4 An Outlier-robust Formulation of the Online ESN Model

As exposed in Section 7.2 and 7.3 in order to guarantee an efficient outlier-robust algorithm for nonlinear system identification in online learning regime one can put together the best features of the KRMC model and the ESN architecture. The proposed approach, already named *echo state kernel recursive maximum correntropy* (ES-KRMC) algorithm, can be seen in Fig. 18, is the result of four frameworks: (i) the correntropy criterion (LIU *et al.*, 2007), (ii) the kernel recursive maximum correntropy (WU *et al.*, 2015), (iii) the echo state network (JAEGER, 2001), and (iv) the novelty sparsification criterion (PLATT, 1991).

In order to be considered an online method, in the sense of adaptive filtering, the dimension of the kernel matrix $\boldsymbol{\kappa}$ cannot increase for each new incoming readout vector $\boldsymbol{\psi}_t$. It will increase from time to time, but only when the information in the incoming vector makes it eligible to be incorporated into the model's dictionary $\mathcal{D}^{sv}$. If this occurs, the kernel matrix is also updated recursively. There are several sparsification techniques that can be applied to

Figure 18 – Network topology of ES-KRMC at iteration $t$.



Source: prepared by the author (2021).

KRMC (WANG *et al.*, 2019; WU *et al.*, 2015; ZHAO *et al.*, 2018); however, in this thesis we have been using preferably, either the ALD or the novelty criteria. In this particular chapter, we decided for the use of the novelty criterion. We call $\mathcal{D}^{sv} = \{\cdot, \boldsymbol{\psi}_{\omega_t}\}$ the dictionary, which has to be built from input data, and $M$ is the number of items in the dictionary (the dictionary size, for short). The subscript $\omega_t$ allows us to clearly distinguish dictionary elements $\boldsymbol{\psi}_{\omega_1}, \ldots, \boldsymbol{\psi}_{\omega_M}$ from readout data $\boldsymbol{\psi}_{\omega_t}$.

As exposed in Eq. (7.6) the readout layer can be reformulated in RKHS, then the cost function in Eq. (7.10) can be also reformulated to receive the readout vector of the reservoir $\boldsymbol{\psi}_j = [\boldsymbol{x}_j; \boldsymbol{u}_j; \bar{y}_{j-1}]$ (in which $\bar{y}_{j-1}$ is given by the CFC criterion to be described soon) as

$$\max_{\boldsymbol{\Omega}_t} J = \sum_{j=1}^{t} \beta^{t-j} G_{\sigma_1}(y_j - \boldsymbol{\Omega}_t^\top \phi(\boldsymbol{\psi}_j)) - \frac{1}{2}\beta^t \gamma_2 \|\boldsymbol{\Omega}_t\|^2, \tag{7.20}$$

in which $\boldsymbol{\Omega}_t = \boldsymbol{\Phi}_t \boldsymbol{\alpha}_t$ and $\boldsymbol{\Phi}_t = [\phi(\boldsymbol{\psi}_{\omega_1}), \ldots, \phi(\boldsymbol{\psi}_{\omega_M})]$. Using the kernel trick Eq. (7.20) can be further written as

$$\max_{\tilde{\boldsymbol{\alpha}}_t} J = \sum_{j=1}^{t} \beta^{t-j} G_{\sigma_1}(y_j - \sum_{\omega_t=\omega_1}^{\omega_M} \tilde{\alpha}_t^{\omega_t} \kappa_{\sigma_2}(\boldsymbol{\psi}_{\omega_t}, \boldsymbol{\psi}_j)) - \frac{1}{2}\beta^t \gamma_2 \tilde{\boldsymbol{\alpha}}_t^\top \boldsymbol{\kappa}_{B,t} \tilde{\boldsymbol{\alpha}}_t, \tag{7.21}$$

in which the matrix $\boldsymbol{\kappa}_{B,t}$ is given by

$$\boldsymbol{\kappa}_{B,t} = \begin{bmatrix} \kappa_{\sigma_2}(\boldsymbol{\psi}_{\omega_1}, \boldsymbol{\psi}_{\omega_1}) & \cdots & \kappa_{\sigma_2}(\boldsymbol{\psi}_{\omega_1}, \boldsymbol{\psi}_{\omega_M}) \\ \kappa_{\sigma_2}(\boldsymbol{\psi}_{\omega_2}, \boldsymbol{\psi}_{\omega_1}) & \cdots & \kappa_{\sigma_2}(\boldsymbol{\psi}_{\omega_2}, \boldsymbol{\psi}_{\omega_M}) \\ \vdots & \ddots & \vdots \\ \kappa_{\sigma_2}(\boldsymbol{\psi}_{\omega_M}, \boldsymbol{\psi}_{\omega_1}) & \cdots & \kappa_{\sigma_2}(\boldsymbol{\psi}_{\omega_M}, \boldsymbol{\psi}_{\omega_M}) \end{bmatrix}. \tag{7.22}$$

Setting its gradient with respect to $\tilde{\boldsymbol{\alpha}}$ equals to zero, one can obtain the solution

$$\tilde{\boldsymbol{\alpha}}_t = (\boldsymbol{\kappa}_{P,t}\boldsymbol{B}_t\boldsymbol{\kappa}_{P,t}^\top + \gamma_2\beta^t\sigma_1^2\boldsymbol{\kappa}_{B,t})^{-1}\boldsymbol{\kappa}_{P,t}\boldsymbol{B}_t\tilde{\boldsymbol{y}}_t, \tag{7.23}$$

in which

$$\boldsymbol{\kappa}_{P,t} = \begin{bmatrix} \kappa_{\sigma_2}(\boldsymbol{\psi}_{\omega_1}, \boldsymbol{\psi}_1) & \cdots & \kappa_{\sigma_2}(\boldsymbol{\psi}_{\omega_1}, \boldsymbol{\psi}_t) \\ \kappa_{\sigma_2}(\boldsymbol{\psi}_{\omega_2}, \boldsymbol{\psi}_1) & \cdots & \kappa_{\sigma_2}(\boldsymbol{\psi}_{\omega_2}, \boldsymbol{\psi}_t) \\ \vdots & \ddots & \vdots \\ \kappa_{\sigma_2}(\boldsymbol{\psi}_{\omega_M}, \boldsymbol{\psi}_1) & \cdots & \kappa_{\sigma_2}(\boldsymbol{\psi}_{\omega_M}, \boldsymbol{\psi}_t) \end{bmatrix}. \tag{7.24}$$

Using the matrix inversion lemma we arrive at

$$\begin{aligned} \tilde{\boldsymbol{\alpha}}_t &= (\boldsymbol{\kappa}_{P,t}\boldsymbol{B}_t\boldsymbol{\kappa}_{P,t}^\top + \gamma_2\beta^t\sigma_1^2\boldsymbol{\kappa}_{B,t})^{-1}\boldsymbol{\kappa}_{P,t}\boldsymbol{B}_t\tilde{\boldsymbol{y}}_t \\ &= \boldsymbol{\kappa}_{P,t}(\boldsymbol{\kappa}_{P,t}^\top\boldsymbol{\kappa}_{P,t} + \gamma_2\beta^t\sigma_1^2\boldsymbol{\kappa}_{B,t}\boldsymbol{B}^{-1})^{-1}\tilde{\boldsymbol{y}}_t. \end{aligned} \tag{7.25}$$

Denoting $\tilde{\boldsymbol{P}}_t = (\boldsymbol{\kappa}_{P,t}^\top\boldsymbol{\kappa}_{P,t} + \gamma_2\beta^t\sigma_1^2\boldsymbol{\kappa}_{B,t}\boldsymbol{B}^{-1})^{-1}$, and substituting the above result into Eq. (7.23) yields

$$\tilde{\boldsymbol{\alpha}}_t = \tilde{\boldsymbol{P}}_t\boldsymbol{\kappa}_{P,t}\tilde{\boldsymbol{y}}_t. \tag{7.26}$$

In order to derive the recursive updating, two dictionary conditions shall be considered.

### *7.4.1 Unchanged Dictionary*

During the first stage of the sparsification procedure, it is not necessary to update the dictionary. Thus, $\boldsymbol{\psi}_t$ is not added to the dictionary $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv}$ and the kernel matrix is not changed. Using the matrix inversion lemma (GOLUB; LOAN, 2012) $\tilde{\boldsymbol{P}}_t = (\tilde{\boldsymbol{P}}_{t-1}^{-1} + \boldsymbol{B}_{t-1}\boldsymbol{\kappa}_{tB}\boldsymbol{\kappa}_{tB}^\top)^{-1}$ can be updated by

$$\tilde{\boldsymbol{P}}_t = \tilde{\boldsymbol{P}}_{t-1} - \frac{\tilde{\boldsymbol{P}}_{t-1}B_t\boldsymbol{\kappa}_{tB}\boldsymbol{\kappa}_{tB}^\top\tilde{\boldsymbol{P}}_{t-1}}{1 + B_t\boldsymbol{\kappa}_{tB}^\top\tilde{\boldsymbol{P}}_{t-1}\boldsymbol{\kappa}_{tB}}, \tag{7.27}$$

in which $\boldsymbol{\kappa}_{tB} = \kappa_{\sigma_2}(\boldsymbol{\psi}_{\omega_1}, \boldsymbol{\psi}_t, \ldots, \kappa_{\sigma_2}(\boldsymbol{\psi}_{\omega_M}, \boldsymbol{\psi}_t))^\top$. Hence, the coefficient vector is recalculated as

$$\tilde{\boldsymbol{\alpha}}_t = \tilde{\boldsymbol{P}}_t\bar{\boldsymbol{\kappa}}_{P,t}\boldsymbol{y}_t. \tag{7.28}$$

in which $\bar{\boldsymbol{\kappa}}_{P,t} = [\boldsymbol{\kappa}_{P,t}, \boldsymbol{\kappa}_{tB}]$, and $\boldsymbol{y}_t = [\boldsymbol{y}_{t-1}, y_t]^\top$. In practice applications, this stage is not necessary because it does not have a significant impact on performance.

### 7.4.2 Updating the Dictionary

At the following step, the NC procedure is adopted to determine whether a new sample is inserted into the dictionary. If $dist_t = \min_M \|\boldsymbol{\psi}_{\omega_M} - \boldsymbol{\psi}_t\|$ (i.e., the distance to distance to nearest center) and $e_t$ (current error) are larger than the preset constant thresholds $\delta_1$ and $\delta_2$, respectively, the augmented input $\boldsymbol{\psi}_t$ will be added to the dictionary, i.e., $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \bigcup \{\boldsymbol{\psi}_t\}$ and $m_t = m_t + 1$. Hence, the inverse matrix $\tilde{\boldsymbol{P}}_t$ is augmented as

$$\tilde{\boldsymbol{P}}_t = \tilde{r}_t^{-1} \begin{bmatrix} \tilde{\boldsymbol{P}}_{t-1}\tilde{r}_t + \tilde{\boldsymbol{z}}_t\tilde{\boldsymbol{z}}_t^\top & -\tilde{\boldsymbol{z}}_t \\ -\tilde{\boldsymbol{z}}_t^\top & 1 \end{bmatrix}, \tag{7.29}$$

in which $\tilde{\boldsymbol{z}}_t = \tilde{\boldsymbol{P}}_{t-1}\tilde{\boldsymbol{h}}_t$, and

$$\tilde{\boldsymbol{h}}_t = [\kappa_{\sigma_2}(\boldsymbol{\psi}_t, \boldsymbol{\psi}_{\omega_1}), \ldots, \kappa_{\sigma_2}(\boldsymbol{\psi}_t, \boldsymbol{\psi}_{\omega_M})]^\top, \tag{7.30}$$

and

$$\tilde{r}_t = \gamma_2 \beta^t \sigma_1^2 V_t + \kappa_{\sigma_2}(\boldsymbol{\psi}_t, \boldsymbol{\psi}_t) - \tilde{\boldsymbol{z}}_t^\top \tilde{\boldsymbol{h}}_t, \tag{7.31}$$

recalling that

$$V_t = (\exp(-(e_t^2/2\sigma_1^2)))^{-1}, \tag{7.32}$$

and $e_t = y_t - \tilde{\boldsymbol{\alpha}}^\top \tilde{\boldsymbol{h}}_t$.

Hence, the coefficient vector is re-estimated as

$$\tilde{\boldsymbol{\alpha}}_t = \tilde{\boldsymbol{P}}_t \boldsymbol{\kappa}_{P,t} \boldsymbol{y}_t = \tilde{r}_t^{-1} \begin{bmatrix} \tilde{\boldsymbol{\alpha}}_{t-1} - \tilde{\boldsymbol{z}}_t e_t \\ e_t \end{bmatrix}. \tag{7.33}$$

### 7.4.3 Correntropy Feedback Criterion

As mentioned in Section 7.2 the output feedback can destabilize the reservoir. In order to alleviate this problem we propose to use a criterion that performs the outlier-detection in a continuous process and is executed on the fly for each new incoming sample.

The general idea is as follows. Assuming that at a time step $t$, after having observed $t-1$ output training samples, it is verified if $y_{t-1}$ should be concatenated or not to the readout vector. For this purpose, the augmented input vector $\boldsymbol{\psi}_t$ must satisfy the following condition:

$$\rho(V_{t-1}) = (\exp(-((y_{t-1} - \tilde{\boldsymbol{\alpha}}^\top \tilde{\boldsymbol{h}}_{t-1})^2/2\sigma_1^2)))^{-1} \leq \vartheta \tag{7.34}$$

**Algorithm 5:** Pseudocode ES-KRMC model.

```
Require:
    Select the hyperparameters;
Initialize:
    Generate a random reservoir $\boldsymbol{W}, \boldsymbol{W}_{in}, \boldsymbol{W}_{back}$;
    Get first sample: $(\boldsymbol{u}_1, y_1)$;
    Update the reservoir state based on $\boldsymbol{x}_1 = h(\boldsymbol{W}_{in}\boldsymbol{u}_1)$;
    Concatenate $\boldsymbol{\psi}_1 = [\boldsymbol{x}_1; \boldsymbol{u}_1; 0]$;
    Let $\mathcal{D}^{sv} = \mathcal{D}^{sv} \bigcup \{\boldsymbol{\psi}_1\}$ and $M = 1$;
    Compute $\tilde{\boldsymbol{P}}_1 = (\gamma_2 \beta^1 \sigma_1^2 + \kappa_{\sigma_2}(\boldsymbol{\psi}_1, \boldsymbol{\psi}_1))^{-1}$;
    Compute $\tilde{\boldsymbol{\alpha}}_1 = \tilde{\boldsymbol{P}}_1 y_1$;
    Compute $\rho(\theta_1)$ from Eq. (7.34);
Compute:
    for $t = 2 : N$ do
        Get new sample: $(\boldsymbol{u}_t, y_t)$;
        if $\rho(V_{t-1}) < \vartheta$
            $\bar{y}_{t-1} = \hat{y}_{t-1}$;
        else
            $\bar{y}_{t-1} = y_{t-1}$;
        end if
        Update the reservoir state based on $\boldsymbol{x}_t = h(\boldsymbol{W}_{in}\boldsymbol{u}_t + \boldsymbol{W}\boldsymbol{x}_{t-1} + \boldsymbol{W}_{back}\bar{y}_{t-1})$;
        Concatenate $\boldsymbol{\psi}_t = [\boldsymbol{x}_t; \boldsymbol{u}_t; \bar{y}_{t-1}]$;
        Update the sparsity hyperparameter with a sparsity criterion (e.g. NC) to verify
        whether to update the dictionary or not;
        if Updating the dictionary (e.g. $dist_t > \delta_1$ and $e_t > \delta_2$)
            Add $\boldsymbol{\psi}_t$ to dictionary $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \bigcup \{\boldsymbol{\psi}_t\}$;
            $m_t = m_t + 1$;
            Compute $V_t$ from Eq. (7.32);
            Compute $\tilde{\boldsymbol{P}}_t$ and $\tilde{\boldsymbol{\alpha}}_t$ using Eqs. (7.29) and (7.33), respectively;
        else if Unchanged dictionary
            $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv}$;
            Compute $\tilde{\boldsymbol{P}}_t$ and $\tilde{\boldsymbol{\alpha}}_t$ using Eqs. (7.27) and (7.28), respectively;
        end if
        Compute $\rho(V_t)$ from Eq. (7.34);
    end for
Output: $\tilde{\boldsymbol{\alpha}}_t, \mathcal{D}_t^{sv}, \boldsymbol{\psi}_t, \boldsymbol{W}, \boldsymbol{W}_{in}, \boldsymbol{W}_{back}$.
```

in which $\vartheta$ is the threshold for outlier-detection. In case the last output system $y_{t-1}$ to be an outlier, it would be substituted by the last system output $\hat{y}_{t-1}$, determining thus $\bar{y}_{t-1}$ in the readout vector of the reservoir.

Algorithm 5 summarizes the general application of the proposed ES-KRMC model for nonlinear system identification.

**Remark 7.4**. The output feedback mechanism of the proposed ES-KRMC model relies on a test that is based on a higher-order statistics provided by the correntropy function.

Hence, this model is robust to non-Gaussian noise.

**Remark 7.5** The incorporation of the proposed correntropy-mediated feedback mechanism into the ESN model provides the desired stability to the ES-KRMC model, allowing a safe use of this model in nonlinear system identification tasks.

## 7.5 Results and Discussion

In this section we evaluate the proposed ES-KRMC model in nonlinear system identification tasks. We evaluate the proposed models on four real-world benchmarking data sets, including a large-scale data set. More specifically, we use the following small-scale data sets: *hair dryer*, *flexible robot arm*, and *heat exchanger* from the DaISy repository (MOOR, 2019). Next, we use the *Wiener-Hammerstein* large-scale data set (SCHOUKENS *et al.*, 2009). Besides the general performance evaluation of the ES-KRMC model in comparison to alternative models, we also assess the effects of reservoir size and correntropy feedback criterion in the efficiency of the proposed algorithm. Table 10 presents the basic characteristics of the above-mentioned data sets.

The reported results correspond to comprehensive computer experiments comparing the proposed ES-KRMC with the original KRMC (WU *et al.*, 2015). For the latter, a NARX structure of order $L_y = L_u = 10$. For the methods that incorporate a dynamic reservoir as echo state kernel recursive least square (ES-KRLS) (ZHOU *et al.*, 2018) and the robust echo state recursive M-estimate (ES-RLM) (BESSA; BARRETO, 2019), it is used only the current input on the input layer, equivalent to $L_y = 0$ and $L_u = 1$.

All experiments involve two scenarios, outlier-free and outlier-contaminated. For the three smaller data sets in contaminated scenarios, percentages of 5%, 10%, 15%, and 20% of the output, training samples are randomly chosen to be contaminated by outliers. For the smaller data sets, the outliers were generated as impulsive Gaussian noise with pdf given by the mixture model $p_z(z) = (1-q)\times\mathcal{N}(0, 0.01) + q\times\mathcal{N}(0.5, 1)$ in which $q$ is determined according to the percentage of outliers in the scenario. For large-scale data sets only the percentage of 5% is considered in a contaminated scenario and the outliers were generated by sampling from a Student's t-distribution with zero mean and 2 degrees of freedom. All data sets were normalized to the $[-1, 1]$ interval.

The strategy to set the hyperparameters' optimal values is to perform 10-fold cross-validation. In order to perform balanced experiments the reservoir parameters of the

Table 10 – Important features of the evaluated data sets for the computational experiments.

| Parameter | Hair Dryer | Robot Arm | Heat Exchanger | Wiener-Hammerstein |
|---|---|---|---|---|
| Training data size | 500 | 512 | 300 | 95,000 |
| Test data size | 500 | 512 | 800 | 84,000 |
| Spectral radius | .98 | .98 | .98 | .98 |
| Reservoir size | 250 | 500 | 500 | 500 |
| Kernel type | Poly | Poly | Poly | Gaussian |
| Kernel size | 1 | 0.5 | 0.5 | 0.01 |

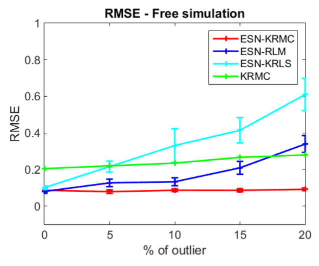
ESN-based models are tuned likewise, including the spectral radius ($\varrho(W)$) and the reservoir size ($N_x$), according to Table 10. The entries of the internal weight matrix $\boldsymbol{W}$, the input weight matrix $\boldsymbol{W}_{in}$, and the weight matrix that project back from the output to the internal units $\boldsymbol{W}_{back}$ are sampled from a uniform distribution over $[-1, 1]$. For the ES-KRMC, ES-KRLS, and KRMC models, the kernel parameters (type and width) also are described in Table 10. The other hyperparameters are optimized for each method, including the thresholds $\upsilon$ (used in the ALD test criterion by ES-KRLS), $\delta_1$ and $\delta_2$ (used by the sparsification method of the ES-KRMC and KRMC models), $\sigma_1$ (used by ES-KRMC and KRMC models to compute the correntropy), the $\varepsilon$ (used by ES-RLM models to compute the error weights), the forgetting factor and the regularization factor (for all models).

The figure of merit for evaluating the numerical performance of the ES-KRMC is the RMSE, the test errors, and the degree of sparsity of dictionary, computed for test samples over 20 and 10 runs in the smaller and large data sets, respectively. Evaluation of the models during the test phase corresponds to the *free simulation* prediction regime, also known as recursive prediction, in which the predicted output value is fed back to the regressor input.

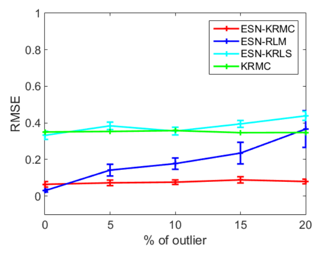
### 7.5.1 Experiments on Smaller-scale Data Sets

In Fig. 19 it is shown the results of the free simulation task on test data for all scenarios (outlier-free and outlier-contaminated) for all the evaluated methods. Figs. 19(a), 19(b), and 19(c) are line charts for the RMSE mean and standard deviation for the *Hair dryer*, *Robot arm*, and *Heat exchanger* data sets, respectively. Figs. 19(d), 19(e), and 19(f) contain absolute test errors of the 20 simulation runs on scenarios with 20% outliers also for *Hair dryer*, *Robot arm*, and *Heat exchanger* data sets, respectively. Finally, the Figs. 19(g), 19(h) and 19(i) are bar plots of the average size of the dictionary for the kernel-based methods for *Hair dryer*, *Robot arm*, and *Heat exchanger* data sets, respectively.

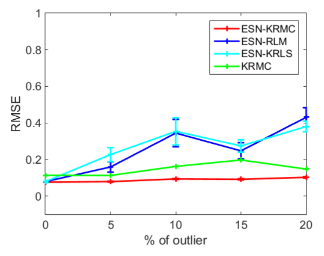
Figure 19 – (a), (b), and (c) line charts for the RMSE values related to the free simulation on test data with different levels of contamination by outliers. (d), (e) and (f) points for comparison between the absolute test errors with 20% outliers. (g), (h) and (i) bar plots of the average size of the dictionary.



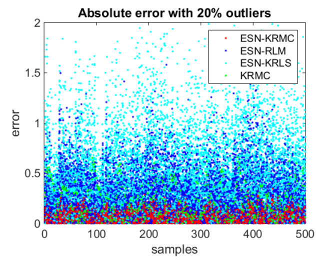
(a) Hair dryer.



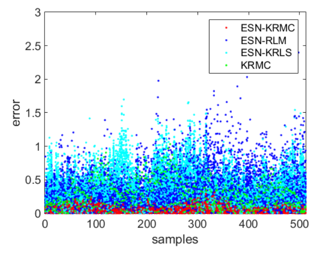
(b) Robot arm.



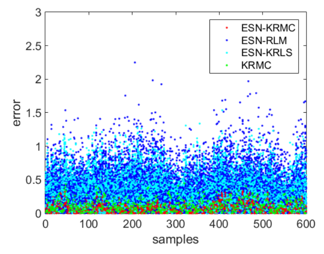
(c) Heat exchanger.



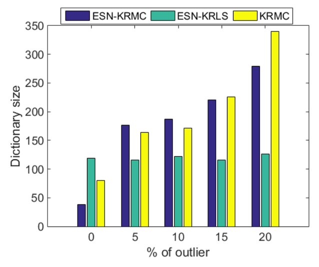
(d) Hair dryer.



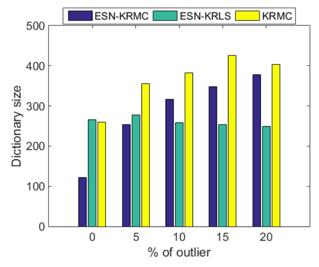
(e) Robot arm.



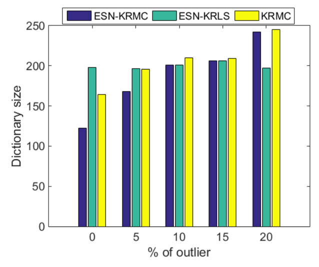
(f) Heat exchanger.



(g) Hair dryer.



(h) Robot arm.



(i) Heat exchanger.

Source: prepared by the author (2021).

#### *7.5.1.1 Results: Hair Dryer Data Set*

The hair dryer is described in Ljung (1999). It is obtained from a laboratory setup acting like a hair dryer in which the air is fanned through a tube and heated at the inlet. The air temperature is measured by a thermocouple at the output. The input is the voltage over the heating device (a mesh of resistor wires). It is comprised of 1000 samples in which the input is the voltage of the heating device and the output is the air temperature. As we show in Table 10, the first 500 samples for model building (training) and the next 500 samples for model testing.

An analysis of Fig. 19(a) reveals that in the outlier-free scenario, the ES-KRMC, ES-KRLS, and ES-RLM models achieved similar and smaller RMSE values compared to the KRMC models. However, the ES-KRMC requires a significantly smaller amount of SVs into the dictionary in comparison to the ES-KRLS and KRMC models as we can see in Fig. 19(g). One can note yet in Fig. 19(a) that the performances of the models have been degraded with the insertion of outliers. However, the ES-KRMC model was barely affected, corroborating the strong outlier-robustness of the proposed model.

The better performance of the ES-KRMC model is verified by analyzing the Figs. 19(d) and 19(g). In Fig. 19(d) we have the absolute test errors in the scenario of maximum outlier contamination. It is verified that, as it happens in the free simulation, the error is not propagated at each step ahead. In Fig. 19(g), the proposed ES-KRMC keeps a significant smaller amount of SVs into the dictionary, although a performance degradation has been observed due to the outliers. It is directly linked to sparsification criterion of the ES-KRMC model, since other criteria such as the ALD used in ES-KRLS did not degrade the model performance with the presence of outliers.

#### *7.5.1.2 Results: Robot Arm Data Set*

For this evaluation, the data are obtained from a *flexible robot arm* (MOOR, 2019). The arm is installed on an electrical motor and modeled from the measured reaction torque of the structure on the ground to the acceleration of the flexible arm. The applied input is a periodic sine sweep. It is comprised of 1024 samples in which the input is the reaction torque of the structure and the output is the acceleration of the flexible arm. The first half samples for model training and the remaining samples for model testing, as detailed in Table 10.

Fig. 19(b) illustrates that in the outlier-free scenario, only the ES-KRMC and ES-RLM models achieved smaller RMSE values. It occurs because the KRMC has no recurrences and the ES-KRLS uses a basic network topology of ESN (ZHOU *et al.*, 2018) not being capable of learning the system dynamics as revealed in the results of the free simulation prediction regime. Furthermore, the ES-KRMC ended up with a significantly smaller amount of SVs into the dictionary in comparison to the ES-KRLS and KRMC models, as shown in Fig. 19(h). An analysis of Fig. 19(b) also reveals that the performance of the ES-RLM model was degraded with the insertion of outliers. By its turn, the influence of the outliers in the performance of the proposed ES-KRMC model is almost imperceptible.

In Fig. 19(e) we have the absolute test errors in the scenario of maximum outlier contamination. It is verified that, as it happens in the free simulation, the error is not propagated at each step ahead. In Fig. 19(h), the proposed ES-KRMC keeps a significant smaller amount of SVs into the dictionary, although a performance degradation has been observed due to the outliers.

#### *7.5.1.3 Results: Heat Exchanger Data Set*

The heat exchanger data set is described in Bittanti and Piroddi (1997). The process is a *liquid-satured steam heat exchanger*, in which the water is heated by pressurized saturated steam through a copper tube. The output variable is the outlet liquid temperature. The input variable is the liquid flow rate. It is comprised of 4000 samples from which we discard the first 100 samples. Starting from sample 101 the first 300 samples are used for model training and the subsequent 800 for model testing, as detailed in Table 10.

One can see in Fig. 19(c) that in the outlier-free scenario, all models achieved similar and smaller RMSE values, with the ES-KRMC achieving again a significantly smaller amount of SVs into the dictionary in comparison with the ES-KRLS and KRMC models as shown in Fig. 19(i). Moreover, Fig. 19(c) also reveals that the performances of the models were degraded with the insertion of outliers with the ES-KRMC and KRMC models being much less affected. An analysis of Fig. 19(f) that compares the absolute test errors in the scenario of maximum outlier contamination confirms that as it happens in the free simulation, the error is not propagated at each step ahead by the ES-KRMC model. However, Fig. 19(i) illustrates that the proposed ES-KRMC degraded the dictionary size with the insertion of outliers.

All experiments in this section provided evidences that the KRMC is also outlier-robust but it does not have a high predictive power in the free simulation regime. The outlier-robust of the KRMC is integrally aggregated to the ES-KRMC model that keeps a high predictive power same in much outliers presence for many dynamical systems.

### ***7.5.2 Performance on a Large-scale Data Set***

In this subsection, the results achieved in the experiments involving a large-scale data set are assessed. The *Wiener-Hammerstein* data set contains samples from an electronic nonlinear system with high-quality measurements of a cascade of a linear dynamical block, a static nonlinear block, and a final linear dynamical block. It presents a very high signal-to-noise ratio (SNR) of 70 dB and contains a total of 188,000 samples (SCHOUKENS *et al.*, 2009). According to the state of the art we discard the first 5,000 and the last 4,000 samples. Starting from the sample 5,001, the first 95,000 samples are used for model training and the remaining 84,000 for model testing, as one can see in Table 10. In order to evaluate the approaches, outlier-contaminated scenarios are considered. For the latter, a proportion of 5% of contamination is adopted, with the outliers being sampled from a $t$-distribution with zero mean and 2 degrees of freedom.

#### *7.5.2.1 Comparison Evaluation*

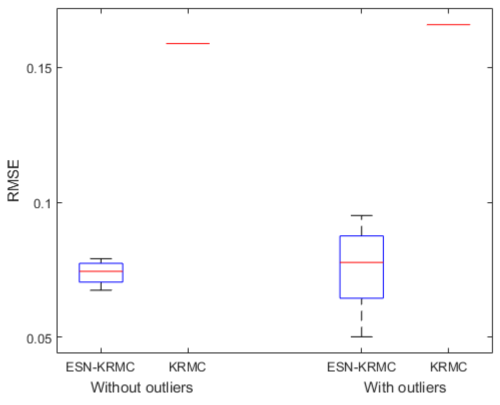
Since the ES-KRMC model outperformed most of the models evaluated in the previous subsection for small-scale data sets, only the KRMC and ES-KRMC models will be evaluated for the Wiener-Hammerstein data set. The results are illustrated in Fig. 20 and, once more, it is shown clearly that the ES-KRMC model is the best performing in terms of RMSE and dictionary size. A closer look at the results shown in Fig. 20(a) reveals that the ES-KRMC keeps presenting a high predictive power even in the outlier-contaminated scenario. The result is obtained when the hyperparameters of the ES-KRMC and KRMC models are set as in Table 10. Fig. 20(b) illustrates the bar plots of the average dictionary size in which the results reveal that the ES-KRMC is capable of obtaining a reduced dictionary size in comparison to the alternative models. Figs. 20(c) and 20(d) show the prediction error distributions of the best simulation result without and with outliers, respectively. In order to obtain this result we use the histogram command of Matlab and specify the 'normalization' property to be 'probability' with a bin width of 0.025. In both evaluated scenarios the ES-KRMC owns a low concentration of the prediction errors around the mean what confirms the good obtained results.

#### *7.5.2.2 Complexity Evaluation*

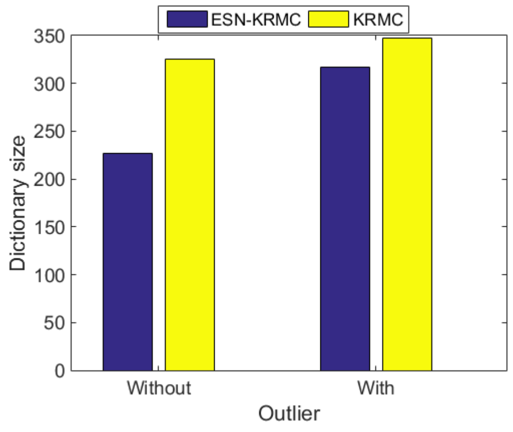
In the next experiments, we show how the reservoir size and the dictionary size impact the performance of the ES-KRMC model. Fig. 21 shows the boxplots for the RMSE values and dictionary size for the reservoir size set to 100, 300, 500, 700, and 900.

The results obtained in Figs. 21(a) and 21(b) reveal that the increase of reservoir size improves the performance of the ES-KRMC in terms of the RMSE. In other words, it increases the predictive power in the free simulation regime on test data. However, the bigger the reservoir and dictionary size are, the bigger the running times of the ES-KRMC are. An analysis of Fig. 22 indicates that increasing the reservoir size, the running times of the model increases accordingly.

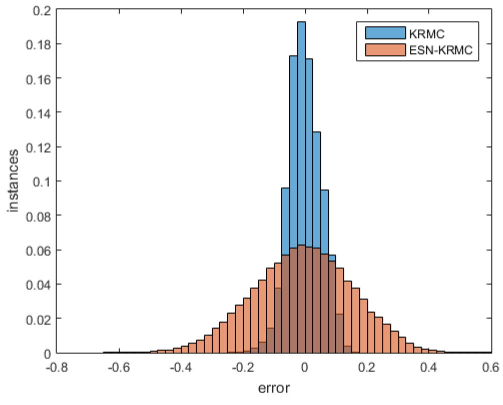
Figure 20 – (a) boxplot for the RMSE values related to the free simulation on test data with and without outliers. (b) bar plots of the average size of the dictionary. (c) and (d) histograms with the prediction error distributions without and with outliers, respectively.



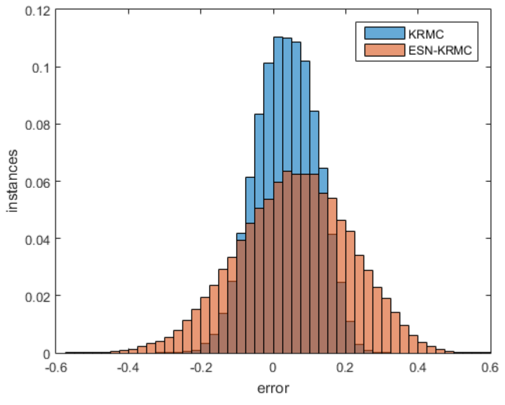
(a)



(b)



(c)



(d)

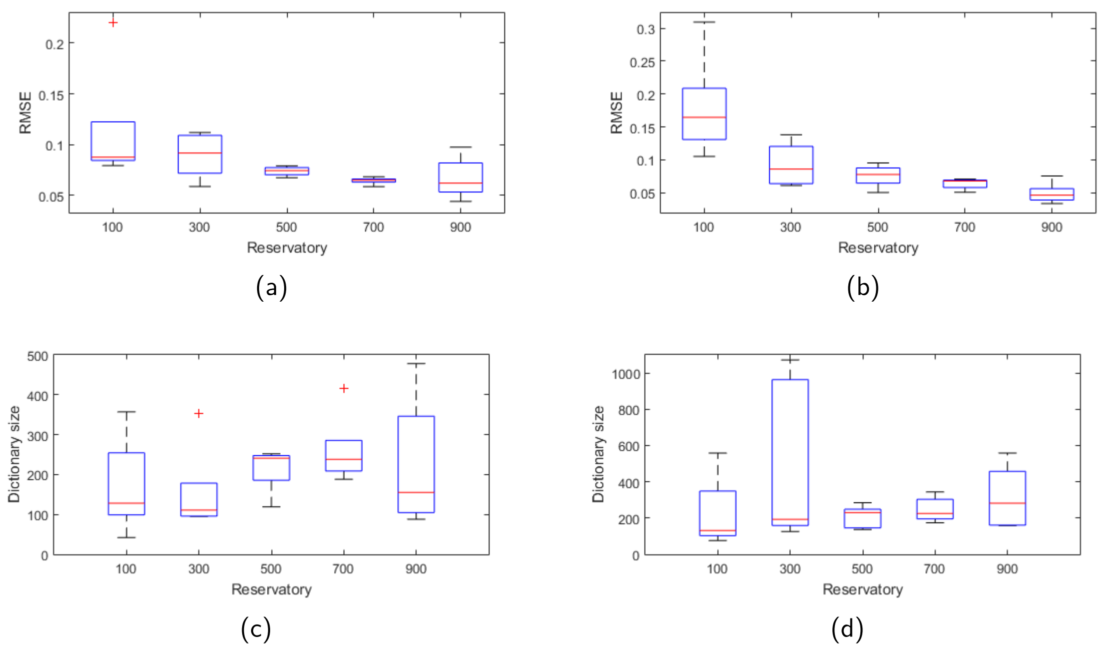
Source: prepared by the author (2021).

### *7.5.2.3 Correntropy Feedback Criterion Evaluation*

In this final set of experiments, we briefly evaluate the effects of the correntropy feedback criterion over the performance of the approach proposed in this work. For this purpose, we executed the training only in outlier-free scenario and evaluated the RMSE values and dictionary size. The hyperparameters of the ES-KRMC are set as shown in Table 10 and the others hyperparameters are optimized using a grid searching.

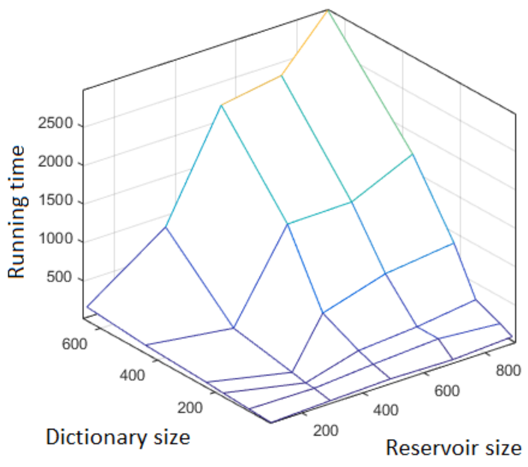
An analysis of Fig. 23(a) reveals that the use of the correntropy feedback criterion improves the performance of the ES-KRMC considerably in terms of RMSE. This occurs because the proposed feedback strategy avoids that outliers are introduced into the reservoir and degrade its performance. One can observe in Fig. 23(b) that the absence of the correntropy

Figure 21 – Boxplots of the RMSE values of the ES-KRMC model for different reservoir sizes without (a), (c), and with (b), (d) outliers.



Source: prepared by the author (2021).

Figure 22 – Relation between reservoir size, dictionary size, and running time of ES-KRMC.



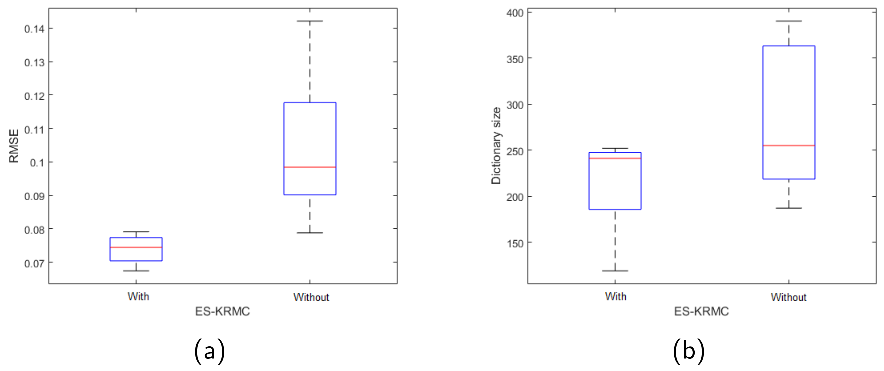
Source: prepared by the author (2021).

feedback criterion increases the dictionary size and, hence, the running times of the proposed model.

## 7.6 Conclusions

It was introduced in this chapter, a novel and efficient algorithm which results from the incorporation of dynamic reservoirs into an outlier-robust kernel adaptive algorithm that integrates the kernel space under the maximum correntropy criterion (MCC). Named

Figure 23 – Boxplot for the RMSE values (a) and dictionary size (b) of the ES-KRMC model with and without the proposed correntropy-based feedback criterion.



Source: prepared by the author (2021).

ES-KRMC, the proposed approach has presented a high predictive power even under an online learning regime in the presence of outliers in the estimation samples.

The output of many dynamical systems depends not only on the past inputs but also on the past outputs. Therefore, the ES-KRMC uses a fully connection reservoir with a feedback of the input and output which is capable of apprehending different dynamics of nonlinear systems and dealing with long-term temporal dependencies. Our experiments show that the presence of outliers in the estimation samples can influence the dynamical reservoir. Then, without resorting to sophisticated mechanism, the outlier samples can be detected using a correntropy-based feedback criterion during the online learning. This is achieved by replacing the observed output contaminated by its predicted version causing less damage to the reservoir. Thus, the use of the correntropy feedback criterion in the training procedure ensures the stability of the solution and yields excellent performance in the simulations.

The ES-KRMC model was successfully evaluated in nonlinear dynamical system identification tasks using four real-world data sets, including a large-scale data set, in the presence of outliers. For all experiments, the prediction regime during the test phase is the free simulation task, also known as recursive prediction, in which the predicted output value is fed back to the regressor input. This is a much more demanding task to tackle than the usual one-step-ahead prediction task and truly evaluates the generalization capability of the model.

For every data set, the ES-KRMC presented better performance than the original KRMC model and alternative ESN-based approaches for online estimation, in terms of RMSE values with a reduced dictionary size of support vectors. This superiority of the proposed model became even more evident as the number of outliers was increased. With the large-scale data set using free simulation, which is also a very challenging scenario, we have analyzed the effects of the reservoir size and of the correntropy feedback criterion on the performance of the ES-KRMC model, and it was also observed the superiority of the ES-KRMC algorithm in terms of accuracy and complexity

In the next chapter the correntropy is inserted into a recurrent neural network for prediction of chaotic dynamics using a backpropagation technique.

# 8 RECURRENT NEURAL NETWORK TO CHAOTIC DYNAMICS LEARNING

Nonlinear dynamic systems can be found in our lives in different contexts and applications (BOEING, 2016; FAN *et al.*, 2020) such as physiological changes, climate changes, financial market movements, mobile communications, and modeling of chaotic systems. These systems arise from innumerable iterations between the constituent parts that make it intrinsically complex.

Among the numerous complex and dynamic systems some present a special type of instability, called sensitivity to the initial conditions, which makes them not foreseeable in the long term. These systems are known as chaotic (SHABESTARI *et al.*, 2018). The consequence of this instability in the prediction results is that such deterministic systems have a great sensitivity to small perturbations. Thus, it is necessary that the modeling techniques are robust and capable of modeling the dynamics of non-linear systems. In recent years, this topic became more and more studied, given the development of lots of machine learning techniques in the area of time series analysis and prediction (SANGIORGIO; DERCOLE, 2020).

Bearing in mind this challenging task, in this chapter a novel use of recurrent neural networks (RNNs) in a nontraditional scheme is developed. Recurrent neurons demonstrated to be extremely efficient in the prediction of chaotic dynamics and have been used for dynamic modeling of systems by several researchers using techniques such as backpropagation through time, real-time recurrent learning, Kalman filtering, reservoir computation, and learning/reconstruction of phase space (ALY, 2020; JUNGLING *et al.*, 2019; HAN *et al.*, 2004; TRISCHLER; D'ELEUTERIO, 2016). Generally, second-order statistics are extracted by means of the mean squared error (MSE) in which the results depend heavily on assumptions of Gaussianity and linearity. The replacement of this learning criterion by ITL-based criteria allows preserving the non-parametric nature of learning by correlation and MSE and extracting more information from the data, obtaining more accuracy in non-Gaussian and non-linear signal processing (LIU *et al.*, 2008; DUARTE; BARRETO, 2019). In this regard, we develop a correntropy-based Elman recurrent network (CYBERNET) built upon two pillars. Firstly, a correntropy-based backpropagation algorithm is developed in order to robustify learning in the presence of outliers. Secondly, a correntropy propagation parameter is used to ensure learning of nonlinear correlations. Therefore, in the present paper, we revisit the work of Bakker *et al.* (2000) by extending it in two aforementioned aspects and also by including a test to monitor the attractor during training and stopping it when the prediction error is considered small enough for the model to pass this test.

The model that we are going to build must represent the dynamics of the system in an autonomous way so that recursive prediction will be used in order to evaluate two scenarios, namely

1. Short-term prediction: The prediction of the time series $\hat{x}_t$ must track the original time series $x_t$ for a period of time as close as possible to the prediction horizon;
2. Long-term prediction: The nonlinear invariant descriptors of the predicted time series $\hat{x}_t$ should correspond to the original time series $x_t$.

In summary, in this chapter the task of interest is the robust dynamic modeling of chaotic time series by means of a novel correntropy-based recurrent neural network with Elman architecture (KREMER, 2001).

The remainder of this chapter is organized as follows. Section 8.1 introduces the model structure used to perform the modeling of chaotic systems. Section 8.2 describes the proposed correntropy-based robust dynamic modeling. Section 8.3 describes the data sets and reports the computer experiments and discusses the results of the evaluated models. The chapter is concluded in Section 8.4.

## 8.1 Model Structure

In many real-world applications, it is common to measure only the current value of a variable of interest, even if the system dynamics depend on several others (BAKKER *et al.*, 2000; GRIFFITH *et al.*, 2019). That said, in this chapter, we are interested in discrete-time deterministic nonlinear dynamic systems, whose temporal sequence can be expressed by the following nonlinear map:

$$x_{t+1} = F(\boldsymbol{x}_t), \tag{8.1}$$

in which $F : \mathbb{R}^\tau \to \mathbb{R}$ is a (usually unknown) nonlinear function, $x_{t+1} \in \mathbb{R}$ is the next value of the sequence, $\boldsymbol{x}_t \in \mathbb{R}^\tau$ is the system state at time $t$. From the function approximation viewpoint, we want to accurately emulate or learn the dynamics of the system in Eq. (8.1) having only a finite-length sequence of observed values of the variable $x_t$.

For this purpose, it is possible to replace the vector of *true* state variables by a regression vector defining a sliding window of delayed samples of the measured variable (GRASSBERGER *et al.*, 1991).

In this case, the dynamics of the system in Eq: (8.1) is approximated by the following model:

$$\hat{x}_{t+1} = f(\tilde{\boldsymbol{x}}_t|\boldsymbol{\theta}) + e_t, \tag{8.2}$$

in which $f$ denotes the approximating model (e.g., a recurrent neural network) with parameters $\boldsymbol{\theta}$, $\hat{x}_{t+1}$ is the predicted value, and $e_t$ is a random variable synthesizing inaccuracy in modeling assumptions and/or measurement errors.

In chaotic dynamical system modeling, the regression vector $\tilde{\boldsymbol{x}}_t$ is usually built using Takens' embedding method as

$$\tilde{\boldsymbol{x}}_t = [x_t \;\; x_{t-\tau} \;\; x_{t-2\tau} \;\; \cdots \;\; x_{t-(m-1)\tau}]^T, \tag{8.3}$$

in which $x_t$ is the measured scalar variable, $\tau$ is a positive integer parameter called the *embedding delay*, and the positive parameter $m$ is the *embedding dimension*. The delays method (TAKENS, 1981) takes the Eq. (8.1) if we choose $m \geq 2\Delta + 1$, in which $\Delta$ is the dimension of the attractor (which can be fractional).

It is important during the tuning process to observe that the data length affects the choice of time delay and embedding dimension when reconstructing the phase space. Moreover, the embedding dimension needs a minimum of data, depending on the preprocessing method used, to determine the steady-state value of the embedding dimension (HUSSAIN *et al.*, 2020).

A common approach to learning the dynamics of the system in Eq. (8.1) uses neural network models (BAKKER *et al.*, 2000; MENEZES; BARRETO, 2008). In this regard, the regressor input at iteration $t$ is built as in Eq. (8.3) and presented as an input vector to the network. The network is then used to predict the scalar $\hat{x}_{t+1}$ as an output value. Then, the neural network is used to approximate the unknown function $F(\cdot)$ as a black box model.

If the chosen neural network architecture is the standard multilayer perceptron (MLP), the only source of temporal information comes from the input regressor. For this reason, the resulting model is called a focused time-delay neural network (FTDNN) (MENEZES; BARRETO, 2008). As one can argue, this approach has limitations, since it captures only short-term information about the dynamics of the chaotic system of interest.

Furthermore, this approach is highly dependent on the selection of appropriate values for the embedding delay $\tau$ and dimension $m$ (BAKKER *et al.*, 2000). Usually, the embedding delay $\tau$ is chosen as the value for which the first minimum of the mutual information function occurs (KUGIUMTZIS *et al.*, 1994). For the embedding dimension $m$, Cao's method (CAO, 1997) is often used.

An alternative to the FTDNN relies on using recurrent neural networks (MENEZES; BARRETO, 2008) to learn longer temporal dependencies typical of chaotic systems. This type of neural architecture has feedback pathways between and/or within layers so that temporal information can circulate and be sustained for longer periods of time allowing the learning process of the network to suitably model long-term dependencies.

However, learning long-term temporal dependencies by means of recurrent neural networks are also dependent on the learning algorithm used to estimate the weights and biases at each layer. The MSE function is a common loss function. A gradient descent learning approach over the MSE loss function gives rise to the well-known backpropagation algorithm. Second-order learning algorithms based on the Hessian matrix (or approximation of it) of the MSE loss function are also usual. It is important to observe that MSE has limitations in capturing long-term dependencies of chaotic systems, despite some successful results reported in the literature (MENEZES; BARRETO, 2008)

## 8.2 Dynamic Modeling

After configuring the model structure, we need to find a set of parameters for the model, so that it has an attractor as close as possible to that of the real system. For this purpose, we have to train the neural network to learn the behavior of the system in the long term and minimize short-term prediction error, since according to Principe *et al.* (1992) not every prediction model presents the appropriate behavior of the system in the long term.

If the model is properly obtained, it can be used to simulate the output dynamics of the identified system through iterative predictions (SANTOS; BARRETO, 2017). As our interest is to evaluate the behavior in the short and long terms it will be used the free simulation, in which the predictions are based on previous predicted outputs and, consequently, the prediction errors tend to propagate through non-linear transformations. This hinders the modeling and identification phase and requires additional care to ensure the stability of the model. Even in this hostile scenario, free simulation is, unlike one-step-ahead (OSA) or open-loop predictions, an adequate way of testing whether the model can explain measured observations and short-term and long-term behavior of the system. If the free simulation has a predetermined prediction horizon, this is called multi-step-ahead (MSA) or closed loop prediction.

In order to train a predictive model, we minimize the squared difference between the predicted and measured future values. The prediction is computed as exposed in Eq. (8.2). As a consequence, the OSA prediction will be highly correlated with the most recent measurements and eventually may be dominated by the noise. This will result in poor MSA predictions during test phase. Alternative solutions are discussed in the following works. In Su *et al.* (1992) it is proposed to train the network with an output error method, in which the state is updated with previously predicted outputs instead of measured outputs. Yet Kuo and Príncipe (1994) and Jaeger and Kantz (1996) proposed to train the network with MSA predictions. This training procedure is known as parallel mode, in which it is used MSA prediction in training and test phases. If the training procedure uses OSA prediction and in test phase a MSA prediction, this procedure is known as series-parallel mode. These modes of training an identification model were first discussed in the article by Narendra and Parthasarathy (1990).

One of the disadvantages of the parallel mode is the instability of the learning process, that is, the neural network convergence is impaired during training, as quoted by Bakker *et al.* (2000), in which they use the compromise method (SAMAD, 1997) to mitigate this problem. We adapt the compromise method to reduce the disadvantages of the MSA prediction, stabilizing the learning process when necessary in which instead of propagating the error we propagate a correntropy-based measure to prevent the dissemination of large outlier-induced errors through the feedback loops.

### *8.2.1 The Correntropy Framework*

As a measure of similarity, the correntropy can be used as an error-based cost function for adaptive systems training by means of the MCC and we use it as a learning criterion to training the RNN:

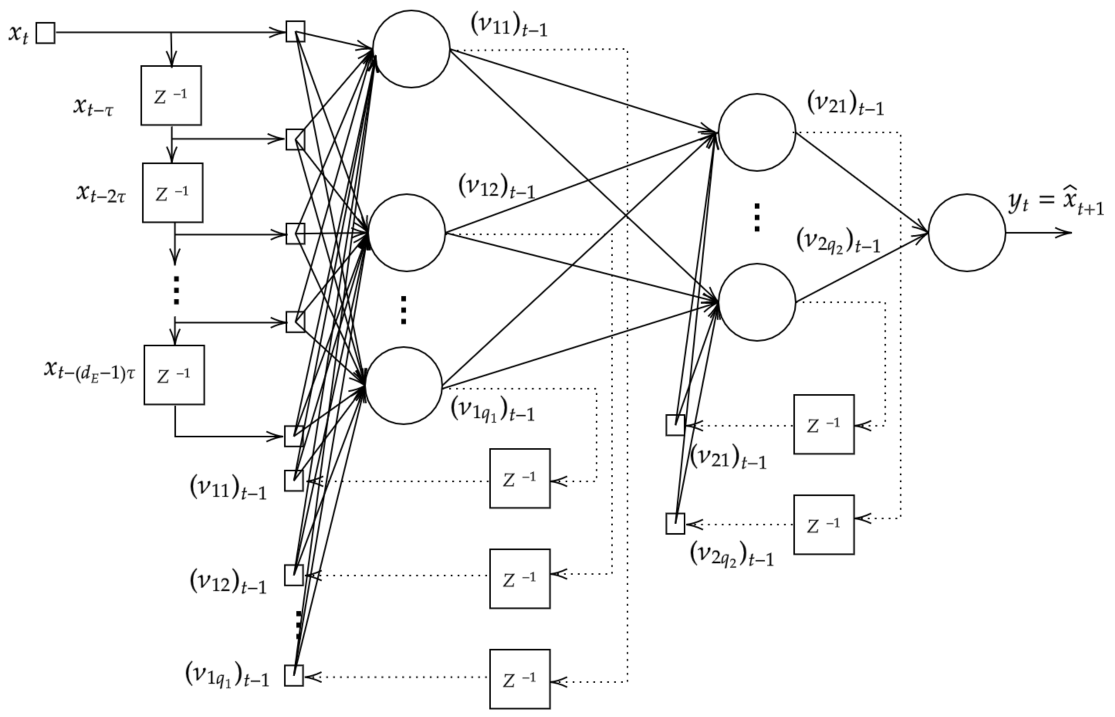
$$\boldsymbol{\theta}_{mcc} = \text{arg max}_{\boldsymbol{\theta}} \sum_{t=1}^{N} k_{\sigma}(e_t|\boldsymbol{\theta}), \tag{8.4}$$

in which $e_t|\boldsymbol{\theta} = y_t - f(\boldsymbol{x}_t;\boldsymbol{\theta})$, $t = 1, ..., N$, is the prediction error produced by an approximating model $f(\boldsymbol{x}_t;\boldsymbol{\theta})$ parameterized by the vector $\boldsymbol{\theta}$.

Then, Eq. (8.4) is differentiated with respect to $\boldsymbol{\theta} = [\theta_1, ..., \theta_N]^{\top}$. The derivatives are set to zero, and a system of $N$ equations can be obtained

$$\sum_{i=1}^{N} e_t \rho(e_t) \frac{\partial e_t}{\partial \theta_j}, \quad j = 1, \ldots, N, \tag{8.5}$$

Figure 24 – Architecture of the proposed recurrent neural network with two hidden layers.



Source: prepared by the author (2021).

in which the weighting term $\rho(e_t)$ penalizes large errors, usually caused by outliers. If one chooses $\rho(e_t)$ as

$$\rho(e_t) = \frac{1}{\sqrt{2\pi}\sigma^3} \exp\left\{-\frac{e_t^2}{2\sigma^2}\right\}, \quad t = 1, ..., N, \tag{8.6}$$

the correntropy can be viewed as a robust cost function capable to handle outlying samples because it does not amplify their effects. The kernel width $\sigma$ plays an important role in the smoothing process and some approaches can be utilized to determinate the kernel width $\sigma$ for $\rho(e_t)$, e.g., the common Silverman's rule that can be computed as (MUNOZ; CHEN, 2012)

$$\sigma = \frac{\max |e_t|}{2\sqrt{2}}. \tag{8.7}$$

### *8.2.2 Training Algorithm*

The training algorithm to be proposed will be used by an Elman-like recurrent neural network architecture (KREMER, 2001), as illustrated in Fig. 24. The underlying learning algorithm is the gradient descent based backpropagation with momentum term. The network has two hidden layers with feedback of the hidden outputs to the previous layer, known as context inputs. The size of the MLP network is determined by grid searching.

The forward pass of the correntropy-based backpropagation algorithm requires the calculation of the activation and outputs of all the neurons of the hidden layers and of all the neurons of the output layer. Thus, the information flow travels from the input neurons to the output neurons, passing through hidden layers. After the presentation of an input vector $\boldsymbol{x}_t$, in the iteration $t$, the first step is to calculate the activation of the neurons of the first hidden layer of the Elman network. We introduce a correntropy-based compromise method during the training phase that mixes the predicted and measured values inspired by strategy to train a neural network proposed in Bakker *et al.* (2000). That said, we formulate the activation of the neurons of the first hidden layer to be calculated by

$$\begin{aligned} u_{i,t} = &\sum_{j=0}^{p} w_{ij,t}((1-\eta\rho(x_{j,t}-\hat{x}_{j,t}))x_{j,t}+\eta\rho(x_{j,t}-\hat{x}_{j,t})\hat{x}_{j,t}) \\ &+\sum_{l=1}^{q_1} w^c_{il,t}\nu_{i,t}, \quad i=1,\ldots,q_1, \end{aligned} \tag{8.8}$$

or, equivalently,

$$u_{i,t} = \sum_{j=0}^{p} w_{ij,t}(x_{j,t}+\eta\rho(e_{t-1})e_{t-1}) + \sum_{l=1}^{q_1} w^c_{il,t}\nu_{i,t-1}, \quad i=1,\ldots,q_1, \tag{8.9}$$

where the error term $e_{t-1}$ is defined $e_{t-1} = x_{j,t} - \hat{x}_{j,t}$ partially propagating prediction errors to the next time step, $p$ denotes the effective number of input variables used in the problem, $q_1$ indicates the number of neurons in the first hidden layer, $w_{ij}$ and $w^c_{il}$ are the input and context synaptic weights of the first hidden layer, respectively. $\eta\rho(e_{t-1})e_{t-1}$ is the correntropy-weighted error propagation term: previous correntropy-based measure is propagated to the next time step, depending on the parameter $\eta$ $(0 < \eta < 1)$. The output of the $i$-th hidden neuron is given by

$$\nu_{i,t} = \phi[u_{i,t}], \tag{8.10}$$

which for this thesis $\phi(\cdot)$ is the hyperbolic tangent function.

The activation of the neurons of the second hidden layer are calculated by

$$u_{s,t} = \sum_{i=0}^{q_1} b_{si,t}v_{i,t} + \sum_{l=1}^{q_2} b^c_{sl,t}\nu_{s,t-1}, \quad s=1,\ldots,q_2, \tag{8.11}$$

in which $q_2$ indicates the number of neurons in the second hidden layer, $b_{si}$ and $b^c_{sl}$ are the input and context synaptic weights of the second hidden layer, respectively. The output of the $s$-th hidden neuron is given by

$$\nu_{s,t} = \phi[u_{s,t}]. \tag{8.12}$$

The activation and outputs of the neurons of the last layer are calculated as

$$u_{k,t} = \sum_{s=0}^{q_2} m_{ks,t}\nu_{s,t}, \quad k = 1, \dots, m, \tag{8.13}$$

in which $m \geq 1$ is the number of output neurons. Then, the outputs of the neurons of the last layer are calculated by the following equation:

$$\hat{y}_{k,t} = \phi[u_{k,t}] = \phi\left[\sum_{s=0}^{q_2} m_{ks,t}v_{s,t}\right], \quad k = 1, \dots, m. \tag{8.14}$$

Finally, in the reverse pass the backpropagation algorithm requires the computation of the local gradients and adjusting the weights of all neurons of the hidden layer and the output layer. The first step of the reverse pass consists of calculating the local gradients $\delta_{k,t}$ of the output neurons using the correntropy as learning criterion in Eq. (8.4) is given by

$$\delta_{k,t} = \rho(e_t)e_t\phi'[u_{k,t}], \tag{8.15}$$

the derivative $\phi'[u_{k,t}]$ of the hyperbolic tangent function is given by

$$\phi'[u_{k,t}] = \frac{1}{2}[1 - \hat{y}_{k,t}^2]. \tag{8.16}$$

In sequel the local gradients $\delta_{s,t}$ of the second hidden layer neurons are calculated as

$$\delta_{s,t} = \phi'[u_{s,t}]\sum_{k=1}^{t} m_{ks,t}\delta_{k,t}, \quad s = 1, \dots, q_2, \tag{8.17}$$

in which the derivative $\phi'[u_{s,t}]$ is calculated as in Eq. (8.16).

Then, we calculate the local gradients $\delta_{i,t}$ of the first hidden layer neurons as

$$\delta_{i,t} = \phi'[u_{i,t}]\sum_{s=1}^{t} b_{si,t}\delta_{s,t}, \quad i = 1, \dots, q_1, \tag{8.18}$$

in which the derivative $\phi'[u_{i,t}]$ is calculated as in Eq. (8.16).

Therefore, the weight updating rule for $w_{ij}$, which corresponds to the weights of the first hidden layer, represented by the vector $\boldsymbol{w}$, is given by

$$w_{ij,t+1} = w_{ij,t} + \alpha\delta_{i,t}x_{j,t} + \gamma(w_{ij,t} - w_{ij,t-1}), \tag{8.19}$$

and for the weights of context units of the first hidden layer, $w_{il}^c$, represented by the vector $\boldsymbol{w}^c$, we have that the update rule is given by

$$w_{il,t+1}^c = w_{il,t}^c + \alpha\delta_{i,t}v_{i,t} + \gamma(w_{il,t}^c - w_{il,t-1}^c). \tag{8.20}$$

**Algorithm 6:** Pseudocode CYBERNET model.

```
Require:
    Select time delay, embedding dimension and propagation term;
Initialize:
    Generate random weights $\boldsymbol{w}, \boldsymbol{w}^c, \boldsymbol{b}, \boldsymbol{b}^c$, and $\boldsymbol{m}$;
Compute:
    for $i = 1 : Epochs$ do
        for $t = 1 : N$ do
            Build input vector: $\boldsymbol{x}_t$;
            Compute neurons activation of first hidden layer $\nu_{i,t}$ using Eq. (8.10);
            Compute neurons activation of second hidden layer $\nu_{s,t}$ using Eq. (8.12);
            Compute outputs $y_{k,t}$ using Eq. (8.14);
            Compute $\rho(e_t)$ using Eq. (8.6);
            Compute local gradients: $\delta_{k,t}$, $\delta_{s,t}$ and $(\delta_{i,t}$ using Eqs. (8.15), (8.17), and
            (8.18), respectively;
            Update the weights $\boldsymbol{w}, \boldsymbol{w}^c, \boldsymbol{b}, \boldsymbol{b}^c$, and $\boldsymbol{m}$ using Eqs. (8.19), (8.20), (8.21),
            (8.22), and, (8.23), respectively;
        end for
    end for
Output: $\boldsymbol{w}, \boldsymbol{w}^c, \boldsymbol{b}, \boldsymbol{b}^c$, and $\boldsymbol{m}$
```

The weight updating rule for $b_{si}$ which corresponds to the weights of the second hidden layer, represented by the vector $\boldsymbol{b}$, is given by

$$b_{si,t+1} = b_{si,t} + \alpha\delta_{s,t}v_{i,t} + \gamma(b_{si,t} - b_{si,t-1}), \tag{8.21}$$

and for the weights of context units of the second hidden layer, $b^c_{sl}$, represented by the vector $\boldsymbol{b}^c$, we have that the update rule is given by

$$b^c_{sl,t+1} = b^c_{sl,t} + \alpha\delta_{s,t}v_{s,t} + \gamma(b^c_{sl,t} - b^c_{sl,t-1}). \tag{8.22}$$

Finally, considering the weights of the output layer, $m_{ks}$, represented by the vector $\boldsymbol{m}$, we have that the update rule is given by

$$m_{ks,t+1} = m_{ks,t} + \alpha\delta_{k,t}y_{s,t} + \gamma(m_{ks,t} - m_{ks,t-1}), \tag{8.23}$$

in which $0 < \alpha \leq 1$ is the learning rate and $0 < \gamma \leq 1$ is the moment term. Algorithm 6 summarizes CYBERNET model.

### *8.2.3 Monitoring Attractor Reconstruction*

In order to decide when to stop training we use *Diks test monitoring* (DIKS *et al.*, 1996) by selecting the number of iterations that produces an acceptable Diks test value and prediction error. If during the training the neural network is having difficulty learning the attractor, we adjust the parameters of the CYBERNET to stabilize it or bring it to the learning of the attractor aiming at maximizing the prediction horizon. We perform the Diks test every 20 iterations and consider the attractor acceptable if the test result stays below the threshold at least for 100 time steps. The prediction error depends on the time series and it is calculated as a function of the prediction horizon in the MSA.

This test evaluates the differences between the chaotic attractors during the training and to establish a stopping criterion for the training when the short-term prediction error is small and there are no differences between the attractors. Many authors use dynamic invariant such as the Lyapunov exponent, entropy-based measures, and correlation dimension to compare attractors (BAKKER *et al.*, 2000). These invariants are certainly valid, but they do not accurately identify the chaotic system, and their confidence limits are difficult to estimate. Using the Diks test, if two attractors are equal, the test result will be zero for a sufficient number of samples. This method can be used instead of the visual comparison of attractors plotted in phase space.

For this purpose, the following null hypothesis is tested: the two sets of delay vectors (model generated, measured) are drawn from the same multidimensional probability distribution. The test uses an estimate for $Q$ and its variance, $\mathrm{V_c}$, in which $Q$ is the distance measure between distributions $\rho_1, \rho_2$. It contains a parameter $\sigma$ that may be identified as bandwidth and it represents the length scale at which the two sets are compared. A small value of $\sigma$ will take into account differences between the distributions at small scales.

A unbiased estimator $\hat{Q}$ for $Q$ is given, and the variance $V_c(\hat{Q})$ of $Q$ under the null hypothesis is calculated conditionally on the observed vectors given their independence (DIKS *et al.*, 1996). The distribution of $S = \hat{Q}/\sqrt{V_c}$ is examined numerically for a fixed value of the bandwidth $\sigma$. Two issues need special attention: the test assumes independence between all vectors, and Gaussian kernels use a bandwidth parameter $\sigma$ that determines the smoothing length scale.

### *8.2.4 Prediction Horizon*

A deterministic (i.e., a noise-free non-chaotic) signal can be predicted for any time horizon. On the other hand, a chaotic signal can not be predicted in the long-term since long-term because different trajectories diverge in the long run due to the positive Lyapunov exponent. The prediction horizon will be calculated through random estimates of the signal itself and this measure is related to the Lyapunov's exponent. Thus, we can adopt the view of Nehmzow (2006) who developed a method to determine the prediction horizon. This method consists of splitting the data into two equal halves of length $T$ each, from which we will use the first half as a model of the second. In order to predict future data points $D(t_2)$, $t_2 = T+1, \ldots, 2T$ of the second half, we construct a three-dimensional embedding $D(t_2)$ as given by

$$D(t_2) = [D(t_2), D(t_2 - \tau), D(t_2 - 2\tau)], \tag{8.24}$$

in which $\tau$ is the embedding delay. We search through the first half of the data for the vector $D(t_1)$, $1 \leq t_1 \leq T$ that is closer to $D(t_2)$.

We then predict the next $\kappa$ data points $D_m(t_2+1) \ldots D_m(t_2+\kappa)$ as given by

$$D_m(t_2+i) = D(t_1+i), \quad i = 1, \ldots, \kappa, \tag{8.25}$$

for $1 < t_1 < T$ and $T < t_2 < 2T$.

This prediction is compared with our baseline, which states that we select a point $D_B(t_r), 1 < t_r < T$ at random, and predict $D_B(t_2+i)$ as given

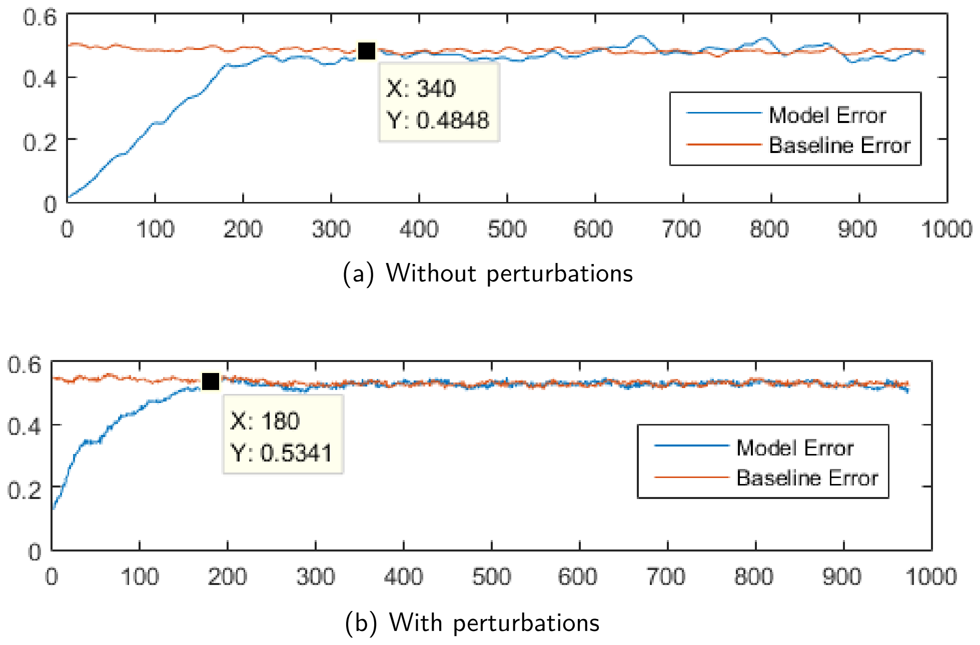
$$\begin{aligned} D_B(t_2+i) &= D_B(t_r+i), i = 1, \ldots, \kappa, \\ &1 < t_r < T, T < t_2 < 2T \end{aligned} \tag{8.26}$$

The point at which the average model error $|D_M - D|$ is the same as the average baseline error $|D_B - D|$ is the prediction horizon. In Figs. 25(a) and 25(b), we have the horizon prediction calculated for the Lorenz time series without and with perturbation, respectively, in which we can observe that the horizon decreases in the presence of perturbations.

## 8.3 Results and Discussion

In this section, we report and discuss the results obtained by the proposed architecture for three benchmarking time series: Mackey-Glass, Lorenz, and *Sunspot*. The results will first be evaluated for the short-term behavior by analyzing the error and the OSA and MSA prediction schemes. Results for the free simulation prediction will be presented subsequently. The results will be evaluated for long-term behavior by analyzing the attractor reconstructed in phase space and the result of the largest exponent of Lyapunov.

Figure 25 – Estimation of the prediction horizon for the Lorenz time series. (a) Without perturbations. (b) With perturbations.



(a) Without perturbations

(b) With perturbations

Source: prepared by the author (2021).

**Remark 8.1**. As the objective of the experiments is to analyze the ability of CYBERNET to capture dynamics of chaotic time series in outlier-contaminated scenarios, the network is configured to train with the same data sequence for several epochs. Thus, despite the recursive nature of this proposal, it will not be trained in one pass of the data. As the proposed approach is recursive, the CYBERNET can indeed be trained in one pass for system identification tasks, simply adjusting the number of training epochs to 1.

**Remark 8.2**. The number of samples in a time series depends on the sampling rate. On systems with high rates, we will have more samples per second. This can allow the model training in one pass of the data in time series prediction task.

### *8.3.1 Time Series Data Sets*

The Mackey-Glass synthetic time series are obtained from the time-delay differential system (MACKEY; GLASS, 1977):

$$\frac{dx_t}{dt} = \beta x_t + \frac{\alpha x_{t-\tau}}{1 + x_{t-\tau}^{10}}, \tag{8.27}$$

in which $x_t$ is the variable at time $t$ and $\tau$ is the time delay[1]. When we assume $\alpha = 0.2$, $\beta = 0.1$ and $\tau = 17$ for $\tau \geq 17$, the time series presents a chaotic behavior. In the experiments, it

[1] This delay is different from the embedding delay in Takens' embedding.

Table 11 – Hyperparameters for the experiments with the CYBERNET model.

| **Series** | **MG-SR** | **MG-CR** | **L-SR** | **L-CR** | ***Sunspot*** |
|---|---|---|---|---|---|
| Epochs | 1000 | 1000 | 4000 | 4000 | 1000 |
| Train | 500 | 500 | 5000 | 5000 | 2500 |
| Test | 1000 | 500 | 1000 | 1000 | 536 |
| $q_1$ | 12 | 14 | 32 | 36 | 60 |
| $q_2$ | 6 | 7 | 24 | 24 | 16 |
| $m$ | 1 | 3 | 1 | 5 | 5 |
| $\tau$ | 26 | 12 | 37 | 12 | 39 |

evaluated the Mackey-Glass series without noise (MG-SR) and with noise and outliers (MG-CR).

The Lorenz synthetic time series is generated from a system of three 1st-order coupled differential equations (SPARROW, 1985):

$$\begin{aligned} \frac{dx_t}{dt} &= \sigma(y_t - x_t), \\ \frac{dy_t}{dt} &= x_t(\rho - z_t) - y_t, \\ \frac{dz_t}{dt} &= x_t y_t - \beta z_t. \end{aligned} \tag{8.28}$$

For $\sigma = 10$, $\beta = 8/3$ and $\rho \geq 28$, the time series presents a chaotic behavior. In the experiments, we use the noise-free Lorenz series (L-SR) and a different version of it contaminated with noise and outliers (L-CR).

In both, Mackey-Glass and Lorenz series, the noise and outliers are modeled as an impulsive Gaussian blend, in which its *pdf* is given by

$$p_z(z) = 0.99N(0, 0.05) + 0.01N(3, 0.05) \tag{8.29}$$

Finally, we use the well-known Sunspot time series (SILSO World Data Center, 1749-2018) in which the data correspond to the monthly average of the total number of sunspots obtained from the SILSO World Data Center, containing 3233 samples from 1749 to 2018.

Table 11 shows the CYBERNET hyperparameters used for processing each time series, in which the 3rd and 4th rows represent how many samples were used for training and testing, respectively. In the 5th and 6th rows, we show how many neurons were used in the two hidden layers ($q_1$ and $q_2$). The two last rows show the embedding dimension $m$, embedding delay $\tau$.

Table 12 – Prediction horizons for the prediction task for each of the time series used in the experiments.

| **Series** | **MG-SR** | **MG-CR** | **L-SR** | **L-CR** | ***Sunspot*** |
|---|---|---|---|---|---|
| Steps | 772 | 470 | 340 | 180 | 154 |

Table 13 – Quantity of steps forward that the model generated can track the observed time series in MSA prediction.

| **Series** | **MG-SR** | **MG-CR** | **L-SR** | **L-CR** | ***Sunspot*** |
|---|---|---|---|---|---|
| Steps | 250 | 150 | 200 | 45 | 280 |

It is important to note that in the time series prediction task with perturbations (MG-CR, L-CR, and *Sunspot*) we do not use small values for $\tau$ to avoid samples with high serial correlation.

### *8.3.2 Results: Short-term Behavior*

In order to analyze the short-term behavior in the free simulation process after modeling the dynamics, it is necessary to determine first the prediction horizon (HOP), i.e., how many steps forward can be predicted without the forecast being considered as a hunch (NEHMZOW, 2006). After estimating the prediction horizon, neural network training is performed. For the time series of the experiments, the prediction horizons are presented in Table 12.

Thus, we first evaluate how many steps forward the model generated can track the observed time series as close as possible to the observed samples in the MSA prediction for each time series through the evolution of the error. The results are show in Table 13. Comparing the correntropy criterion with the MSE, the former performs better than the latter in learning the system dynamics in the presence of perturbations.

In order to evaluate the prediction error, we use the normalized mean squared error (NMSE)

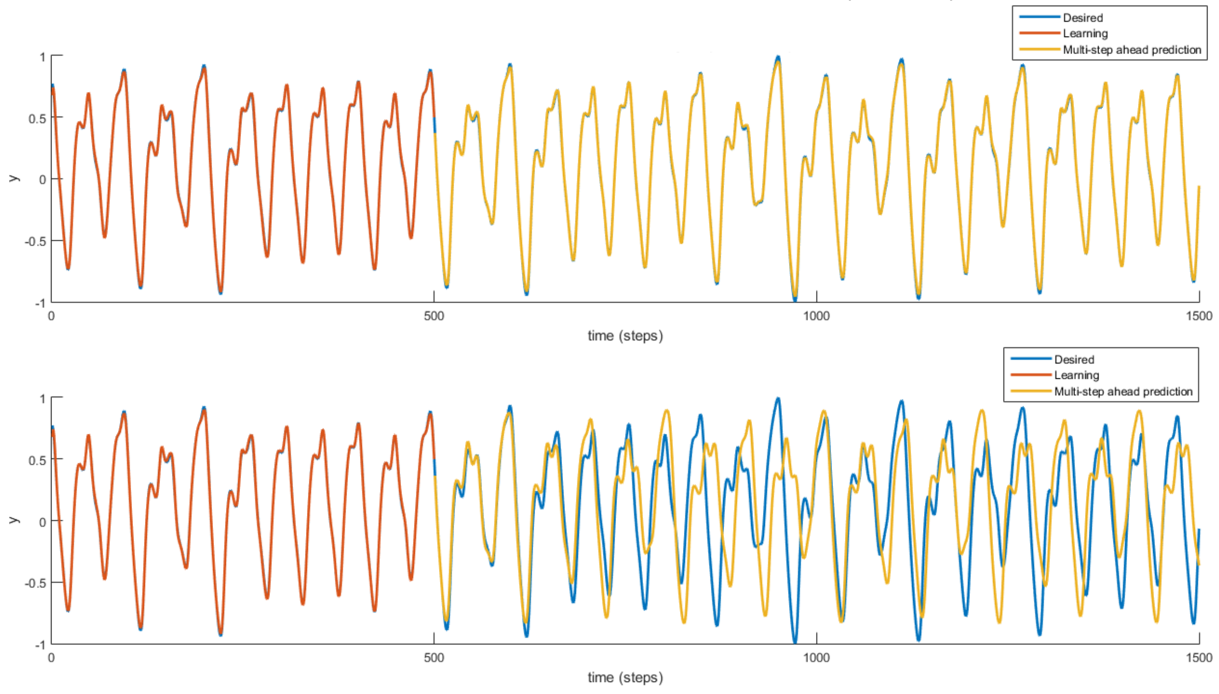
$$NMSE = \frac{\frac{1}{N}\sum_{t=1}^{N}(y_t - f(\tilde{\boldsymbol{x}}_t|\boldsymbol{\theta}))^2}{\sigma_x^2}, \tag{8.30}$$

in which $\sigma_x^2$ is the sample variance of the time series used to create the model. Table 14 shows the NMSE for the OSA and MSA prediction tasks calculated as a function of the HOP over 10 independent runs. It observed that due to the chaotic nature of the system it is not possible to track the original time series for a period of time equal to the prediction horizon.

Table 14 – NMSE values for the short-term prediction task for each of the considered time series.

| **Series** | **OSA** | **MSA** |
|---|---|---|
| MG-SR | 5.73E-7 ± 1.29E-7 | 7.54E-4 ± 3.04E-4 |
| MG-CR | 4.68E-6 ± 8.20E-7 | 9.07E-4 ± 6.30E-4 |
| L-SR | 4.16E-7 ± 2.29E-8 | 2.50E-3 ± 1.34E-3 |
| L-CR | 1.05E-5 ± 5.70E-6 | 1.43E-2 ± 2.20E-3 |
| *Sunspot* | 2.76E-2 ± 4.60E-3 | 4.27E-2 ± 2.05E-2 |

Figure 26 – Examples of OSA (Upper figure) and MSA (Lower figure) prediction results for the Mackey-Glass time series without perturbations (MG-SR).



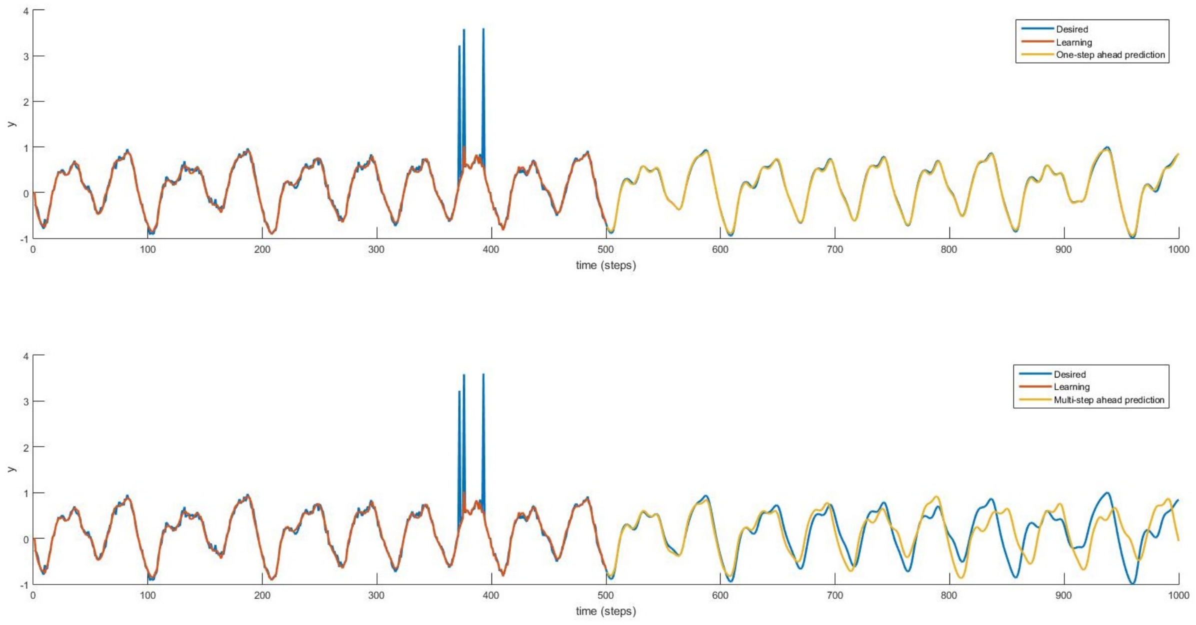
Source: prepared by the author (2021).

Typical instances of the time series predicted by the proposed CYBERNET model under the free simulation regime are presented in Figs. 26, 27, 28, 29, and 30 for the outlier-free and outlier-contaminated scenarios, respectively.

### *8.3.3 Results: Long-term Behavior*

In order to evaluate if the dynamic modeling of the system was performed correctly, it is necessary to proceed with the free simulation for $k$ steps ahead in order to perform the attractor reconstruction in the phase space. In addition to the visual analysis, we also use the result of the Diks test monitoring and the largest Lyapunov exponent. The amount $k$ steps ahead used for each time series is 1000 and 500 to Mackey-Glass series without and with

Figure 27 – Examples of OSA (Upper figure) and MSA (Lower figure) prediction results for the Mackey-Glass time series with perturbation (MG-CR).



Source: prepared by the author (2021).

Table 15 – Largest Lyapunov exponent values estimated from the training and predicted time series.

| **Series** | **MG-SR** | **MG-CR** | **L-SR** | **L-CR** | ***Sunspot*** |
|---|---|---|---|---|---|
| Training | 0.55 | 0.64 | 1.25 | 0.70 | 0.07 |
| Predicted | 0.53 | 0.65 | 1.20 | 0.90 | 0.12 |

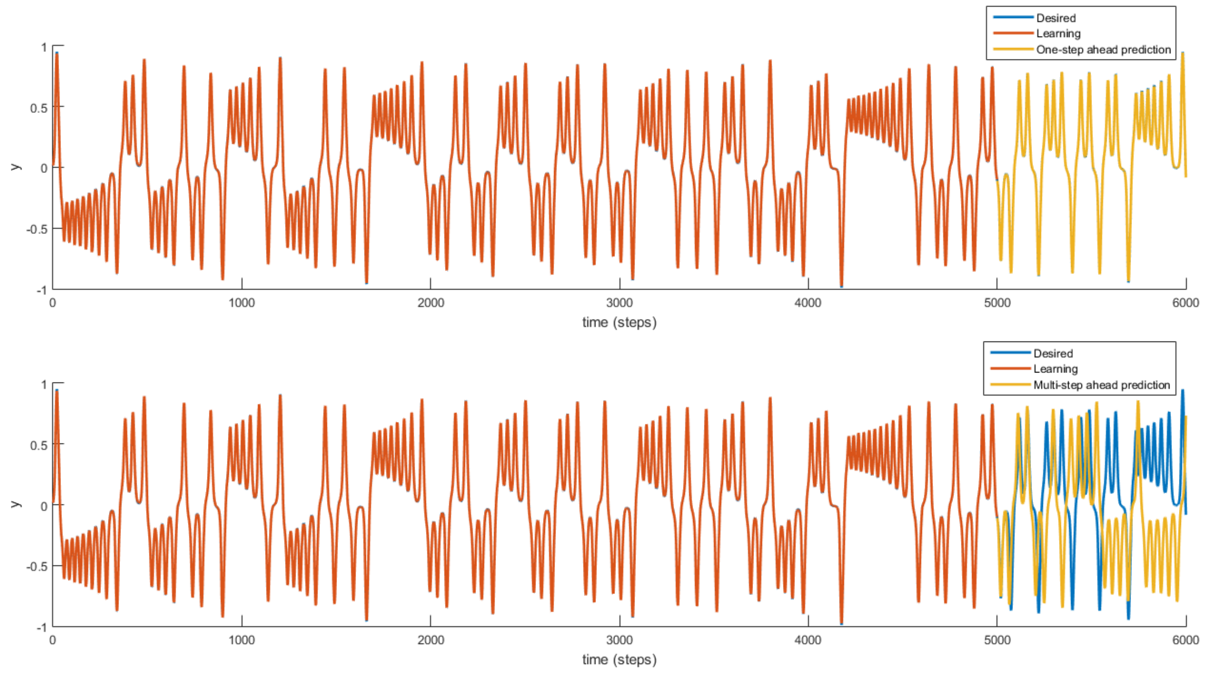
outliers, respectively; 1000 to series Lorenz in both scenarios; 536 to *Sunspot* series.

Table 15 shows the largest Lyapunov exponents of the observed training samples and of the MSA-predicted samples. The values were estimated using the algorithm of Rosenstein *et al.* (1993) in which one can see that the resulting values are approximately equal to those obtained for the training time series.

It is performed the reconstruction of the attractor time series and we observed that the CYBERNET is able to model the chaotic dynamics even in the presence of perturbations. For the MG-SR time series, the reconstructed attractor is shown in Fig. 31, while the reconstructed attractor for the MG-CR time series is shown in Fig. 32. For both figures, it was used $\tau = 12$.

For the L-SR times series, the reconstructed attractor is shown in Fig. 33, while for the reconstructed attractor for the L-CR time series is shown in Fig. 34. For both time series, we used $\tau = 6$. In the *Sunspot* time series we obtain the attractor of Fig. 35, for the reconstruction of the attractor we use $\tau = 6$.

Figure 28 – Examples of OSA (Upper figure) and MSA (Lower figure) prediction results for the Lorenz time series without perturbation (L-SR).



Source: prepared by the author (2021).

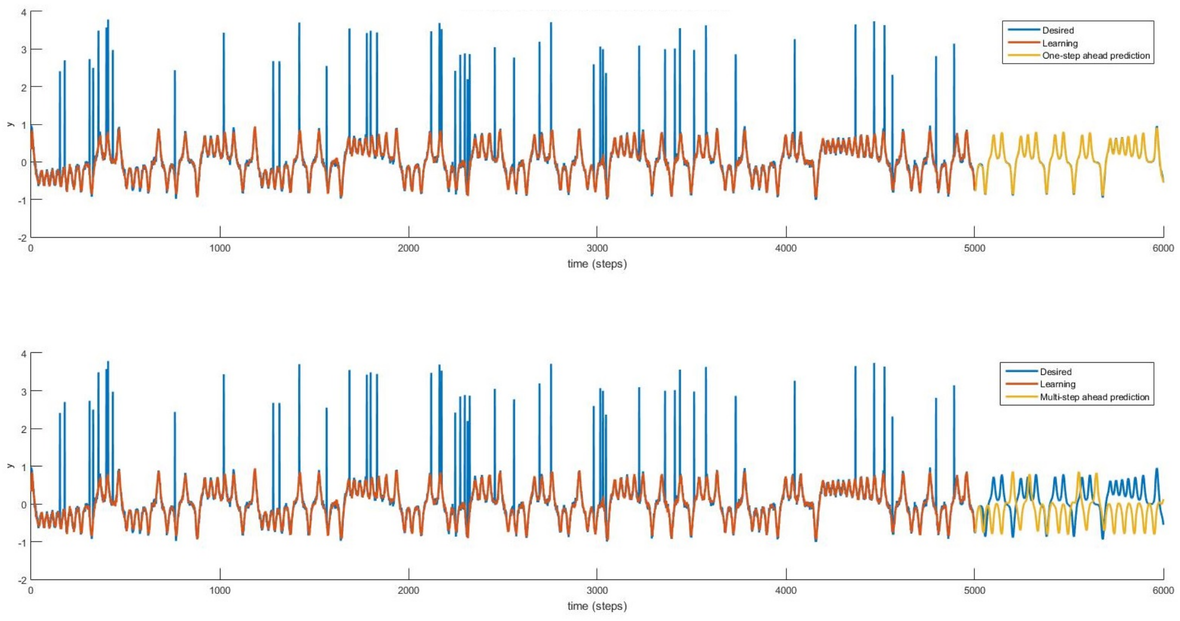
The reconstructed attractor is similar to the Rossler system, expressed by its $z$-variable. Letellier *et al.* (2006) provides strong evidence that the underlying dynamics of sunspot cycles may be similar to the Rossler system.

In the three time series we find that the reconstructed attractors are similar to those of the test set and that the the corresponding largest Lyapunov exponents are also close to those of the test time series. Diks test monitoring during training converged to a value less than 3, which according to Diks *et al.* (1996), this means that the two series are drawn from the same multidimensional probability distribution. However, the resolution of this test may have been compromised by the presence of disturbances in the time series.

## 8.4 Conclusions

This chapter addressed the task of prediction and modeling of chaotic time series in the presence of outliers by introducing a novel correntropy-enhanced variant of the Elman recurrent network, named CYBERNET. This is motivated by a decade of advances in ITL methods. Although the free simulation does not closely follow the test time series, it does not mean that the model is incorrect because, as we could verify, in the evaluation of the

Figure 29 – Examples of OSA (Upper figure) and MSA (Lower figure) prediction results for the Lorenz time series with perturbations (L-CR).



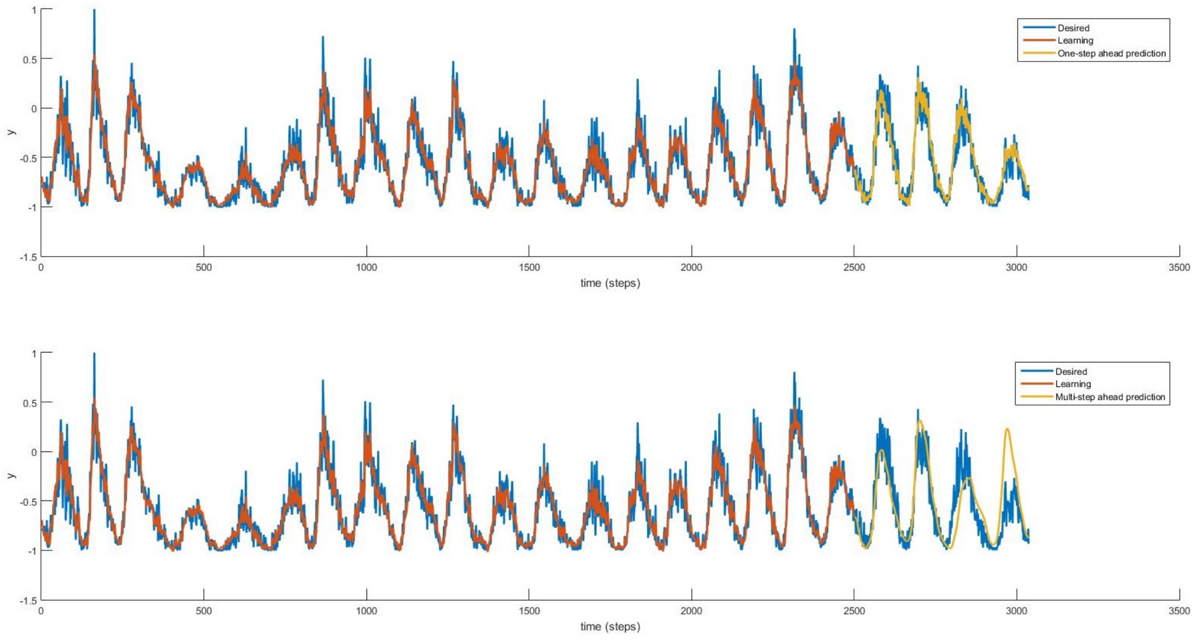
Source: prepared by the author (2021).

long-term behavior. The dynamic modeling of the system was correctly performed, once it gives the characteristics of the reconstructed attractors, of Diks test and the estimated values of the Lyapunov exponents from the predicted time series.

Some of the difficulties encountered were that training the proposed RNN required a large number of samples. Another difficulty is that the large number of parameters adjustable by the RNN makes the learning process slow, and the length of the time window influences the short-term prediction. Thus, with the results of the experiments, we were able to show that the proposed CYBERNET architecture allows efficient modeling of the underlying chaotic system, apprehending the short and long term behaviors even in the presence of perturbations when using the correntropy as a weighting mechanism in the learning equations of the backpropagation algorithm.

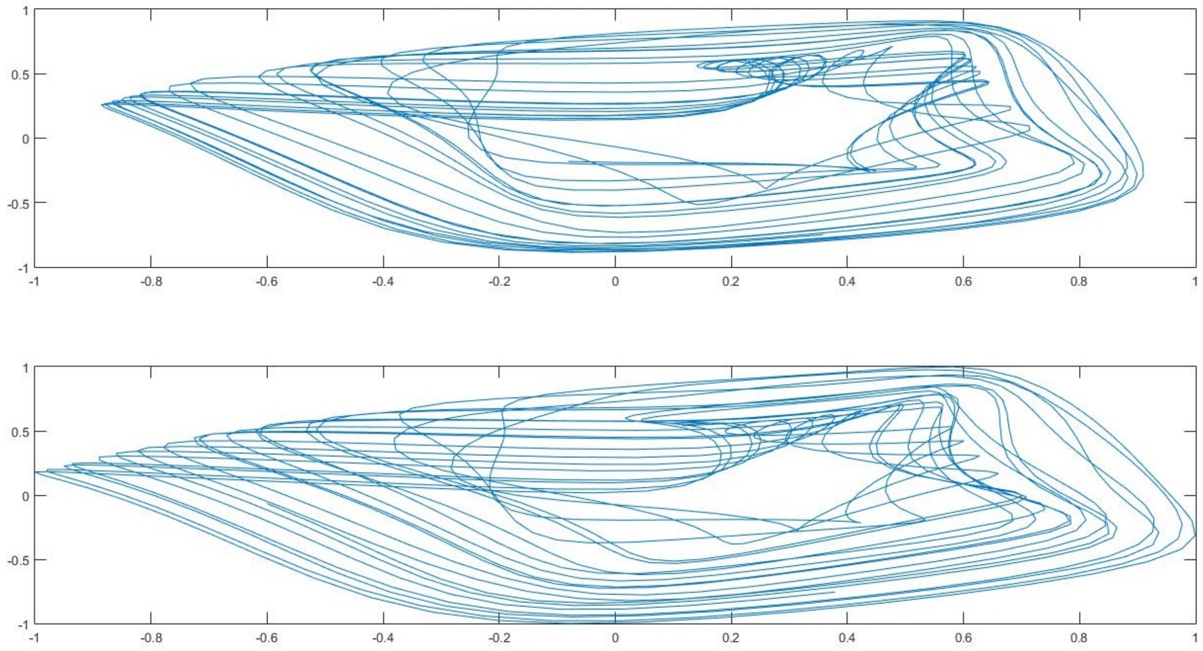
The next chapter conclude this thesis with a discussion about the contributions and directions for future research on related topics.

Figure 30 – Examples of OSA (Upper figure) and MSA (Lower figure) prediction results for the Sunspot time series.



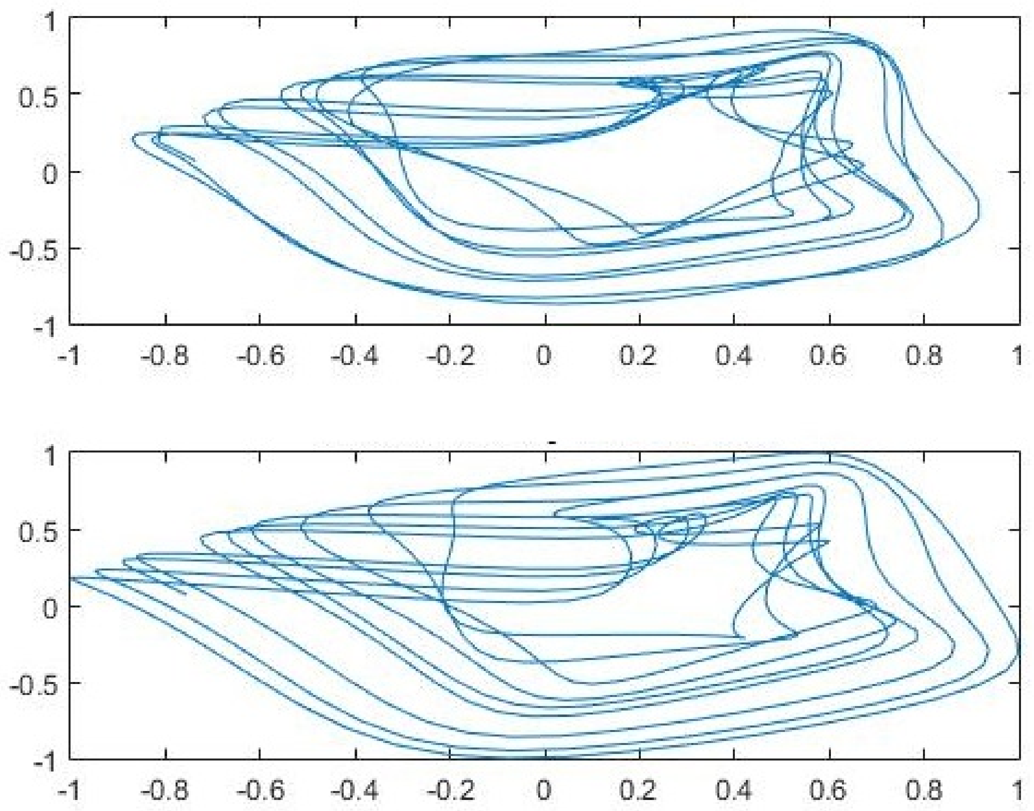
Source: prepared by the author (2021).

Figure 31 – (Upper figure) Reconstructed attractor for the MG-SR time series. (Lower figure) Original attractor for this series.



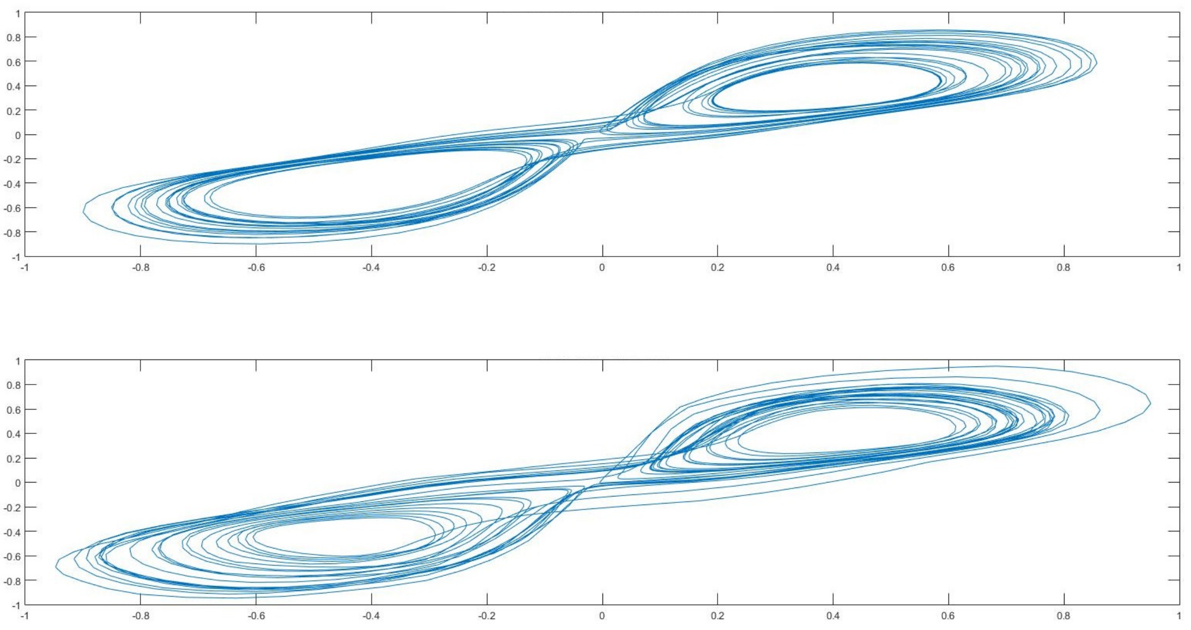
Source: prepared by the author (2021).

Figure 32 – (Upper figure) Reconstructed attractor for the MG-CR time series. (Lower figure) Original attractor for this series.



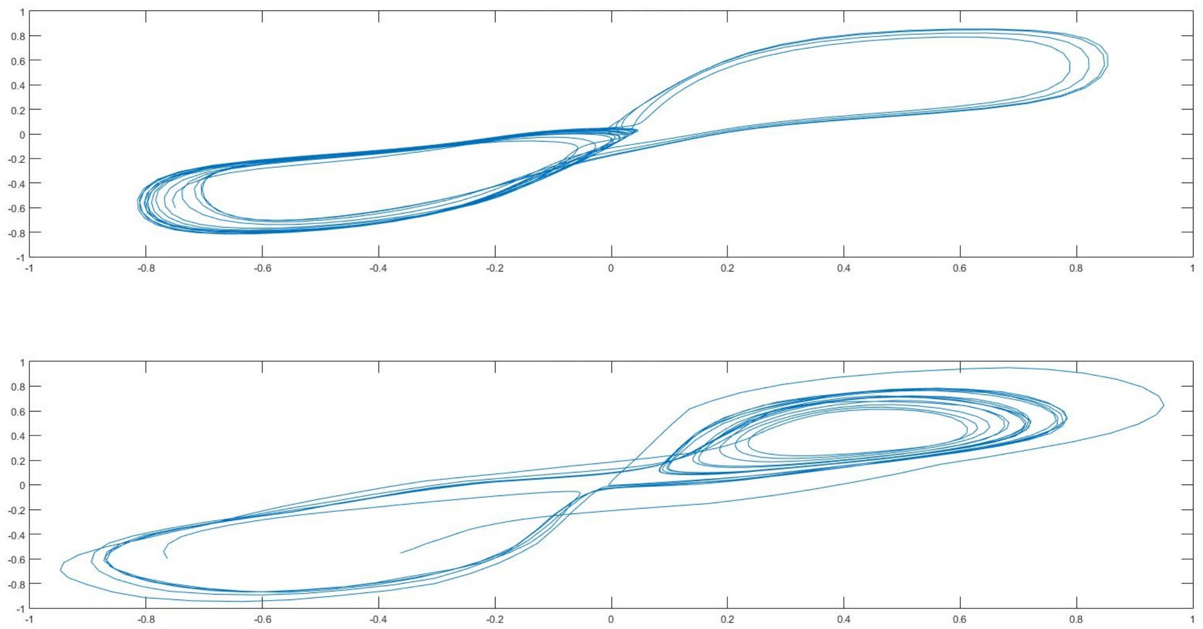
Source: prepared by the author (2021).

Figure 33 – (Upper figure) Reconstructed attractor for the L-SR time series. (Lower figure) Original attractor for this series.



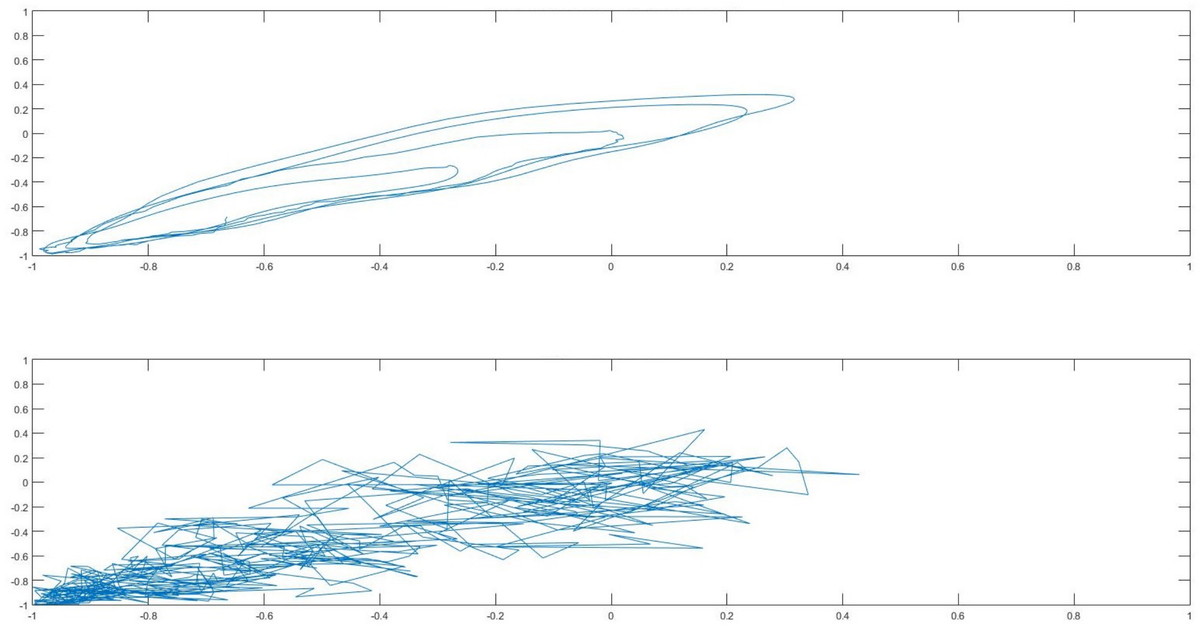
Source: prepared by the author (2021).

Figure 34 – (Upper figure) Reconstructed attractor for the L-CR time series. (Lower figure) Original attractor for this series.



Source: prepared by the author (2021).

Figure 35 – (Upper figure) Reconstructed attractor for the sunspot time series. (Lower figure) Original attractor for this series.



Source: prepared by the author (2021).

# 9 CONCLUSIONS AND FUTURE DIRECTIONS

This chapter presents the conclusions and points out possible directions for further research.

## 9.1 General Conclusions

Dynamical system modeling in system identification or times series prediction involves challenges as nonlinear relationship among variables, model approximation with acceptable accuracy from data, model complexity reduction, non-Gaussian noise, and large databases processing. All these challenges are even more pronounced when the learning regime is online. In order to deal with them, the machine learning perspective has gained momentum, offering suitable approaches to each of the aforementioned challenges. Therefore, this thesis investigated different machine learning paradigms to model nonlinear dynamical systems only from their inputs and outputs, especially those related to system identification tasks, as such as kernel adaptive filtering and reservoir computing.

The state-of-art of those methods include learning criteria based on 2nd order statistics, such as autocorrelation and the MSE, producing learning rules which are not robust to non-Gaussian perturbations. During the model building, these perturbations can manifest themselves as large outliers or impulsive noises in the estimation data and, hence, deteriorates considerably the resulting predictor performance.

The approach chosen in this research to deal with non-Gaussian noise in all proposed methods was to substitute 2nd order statistics by a learning criterion based on ITL, namely the MCC, that has found successful applications when used in the development of robust adaptive filters.

Additionally, other techniques, such as forward-backward splitting, support vector selection criteria, and random Fourier features were used to model complexity reduction. Recursively matrix computation was used to create an on-the-fly learning regime for the proposed outlier-robust kernel filters. Random projections and correntropy feedback were used to promote efficient spatiotemporal mapping.

That said, the first learning method proposed in this thesis was a sparse and online CKL model, named OS-CKL, through the ALD sparsification criterion and by computing matrices recursively. This method used concepts from correntropy and kernel adaptive filtering to introduce a robust model that can be used for online learning with model complexity considerably reduced by the introduction of a sparsification procedure. The proposed model is evaluated using four benchmarking data sets, two synthetic and two related to the process industry, for different levels of outlier contamination. The obtained results revealed that the OS-CKL keeps presenting a high predictive power even under an online learning regime with a reduced dictionary size in comparison to several state-of-the-art alternatives, such as the KRLS and KRMC models.

The second contribution was the formulation of a fully adaptive dictionary of support vectors for online and sparse learning models through proximal methods, and also computing matrices and submatrices recursively. The fully adaptive dictionary is applied to OS-CKL model, leading to the FADOS-CKL model. This method introduces a simple efficient mechanism for pruning items from the current dictionary on the fly. As a consequence of the proposed pruning mechanism, a novel rule for updating the parameters of the predictive model in case of deletion of items from the dictionary had to be devised. The proposed model is evaluated by a set of experiments. Firstly it was carried out a comprehensive investigation of the state of the art in sparsity-inducing criteria aiming at selecting the best performing one for the task of interest. For these experiments, it was used the OS-CKL model with the novelty criterion. Secondly, it was carried out an investigation of the performance of the proposed FADOS-CKL model in two benchmarking data sets of nonlinear dynamic system identification. Scenarios with and without outliers were investigated. In this regard, the proposed FADOS-CKL model was able to capture the underlying dynamics correctly from the available data sets, delivering a model which is more accurate (in terms of predictive power) and also more compact (in terms of the size of the dictionary) than its predecessor, the OS-CKL model.

The third contribution was a new method to develop an online SVR-type model in primal space using random Fourier features as an explicit nonlinear mapping to RKHS. The resulting model solves the SVR framework in primal space and it can be efficiently used for online learning with reduced model complexity. The obtained results revealed that the PRIMOS-CKL achieved a high predictive power even under online learning and free simulation regime.

The fourth contribution was the proposal of an alternative method to formulate an adaptive dictionary of support vectors in online and sparse learning models through Kullback–Leibler divergence, and by computing matrices and submatrices recursively. The proposed learning model integrates the regularized network architecture in the RKHS, from a perspective of kernel adaptive filtering methods, with the MCC. The proposed model was evaluated in the task of identifying nonlinear systems using two small-scale artificial data sets and one large-scale data set, in a free simulation regime and including outlier-contamination scenarios. The results demonstrated that the proposed method is capable of performing recursive learning and maintaining a high predictive power in a free simulation regime with a reduced dictionary size.

The fifth contribution was an outlier-robust learning model based on the KRMC algorithm, reservoir computation in the ESN sense, and correntropy information feedback. Named ES-KRMC, this proposed approach achieved high predictive power even under an online learning regime in the presence of outliers in the estimation samples. The ES-KRMC uses a fully connected reservoir with feedback of the input and output able to apprehend different dynamics and to deal with long temporal dependencies. The experiments have shown that the presence of outliers in the estimation samples can influence the dynamical reservoir. Then, we devised a feedback strategy in which the outlying samples are detected using a correntropy feedback criterion during the online learning. This procedure substitutes the measured output contaminated by the predicted output. Thus, the correntropy feedback criterion ensured the stability of the solution and led to excellent performance in the simulations.

The last contribution involved an Elman-like recurrent network embodied with the MCC criterion for the purpose of robust learning and it used the correntropy information feedback to ensure stable learning of nonlinear correlations. The CYBERNET was evaluated in the task of chaotic time series prediction and with the results of the experiments, it was capable of apprehending the short and long term behaviors even in the presence of perturbations when the correntropy is used as learning criterion.

In summary, in this thesis we developed new sequential learning models capable of dealing with nonlinear, time-varying, non-Gaussian data signals. All proposed models have high predictive power, are outliers-robust, and work in an online learning regime. The investigation developed in this research has shown that selecting correct basis vectors increases the prediction power, that fully adaptive dictionaries lead to better performance than growing ones, and that correntropy-mediated feedback stabilizes learning of recurrent networks in outlier-contaminated scenarios.

The Table 16 summarizes the main characteristics of each kernel-based model

Table 16 – Summary comparative among the proposed methods ($D \ll m_t \ll N$).

| **Methods vs Goals** | LS-SVR | OS-CKL | FADOS-CKL | CRONOS | ES-KRMC | PRIMOS-CKL |
|---|---|---|---|---|---|---|
| Online learning | No | Yes | Yes | Yes | Yes | Yes |
| Nonlinear processing | Yes | Yes | Yes | Yes | Yes | Yes |
| Non-Gaussian processing | No | Yes | Yes | Yes | Yes | Yes |
| Non-Stationary tracking | No | No | Yes | Yes | No | No |
| Model complexity | N | $m_t$ | $m_t$ | $m_t$ | $m_t$ | D |
| Model size | Grows | Grows | Grows/Shrinks | Grows/Shrinks | Grows | Constant |
| Computational cost | $\mathbb{O}(N^3)$ | $\mathbb{O}(m_t^2)$ | $\mathbb{O}(m_t^2)$ | $\mathbb{O}(m_t^2)$ | $\mathbb{O}(Nm_t^2)$ | $\mathbb{O}(m_t^2)$ |
| Memory storage | $\mathbb{O}(N^2)$ | $\mathbb{O}(m_t^2)$ | $\mathbb{O}(m_t^2)$ | $\mathbb{O}(m_t^2)$ | $\mathbb{O}(Nm_t)$ | $\mathbb{O}(m_tD)$ |

proposed in this thesis. We can use this table to support the decision-making process as to decide which of these methods must be used in dynamical system modeling. The decision can consider the data information and the hardware available. For data in which non-stationary operating states are present, the FADOS-CKL and CRONOS models are the more suitable ones, not to mention the FADOS-CKL model presents a differential compared to the CRONOS model for pruning the dictionary more powerfully. For data in which it is necessary to capture long-term temporal dependencies, the ES-KRMC model is presented as more suitable due to the presence of the recurrence layer. For problems in which the hardware does not allow a dynamic size model, the PRIMOS-CKL model is more suitable. In other cases, the OS-CKL model appears as a good choice due to its simpler structure and fewer hyperparameters for adjustment compared to the others thesis proposes.

## 9.2 Future Directions

The comprehensive framework considered in this thesis allows further theoretical investigations and practical applications.

The FADOS-CKL and CRONOS models are capable of growing and shrinking through the ability to add new SVs and remove obsolete SVs. Therefore, an interested reader can carry out experiments with these models in a non-stationary scenario and evaluate other sparsity criteria to select and discard support vectors.

Another research direction that can be followed is the development of variants of the FADOS-CKL, CRONOS, and ES-KRMC models aiming at real-world implementations in software/hardware embedded systems. In this regard, the highest computational demand of the FADOS-CKL, CRONOS, and ES-KRMC models comes from kernel computations. For small-scale data sets, this is not a big deal, but the scalability of kernel machines is indeed a big issue for large-scale data sets due to the storage and computation required by large kernel matrices. This issue has been addressed in the signal processing literature, with many papers recommending tackling this problem by means of low-rank approximations of the kernel matrix. In particular, the memory-efficient kernel approximation (MEKA) method introduced by Si *et al.* (2017) seems to be a viable technique.

The models proposed in this thesis were applied to single-input single-output (SISO) systems. However, many practical applications of system modeling and identification, especially in industrial environments, are designed for multiple-input multiple-output (MIMO) systems. Bearing this in mind, an interesting research direction is to explore MIMO systems.

Finally, other interesting directions for additional research may be to explore the versatility of the PRIMOS-CKL model proposed in this work. For instance, the evaluation of different criteria for dictionary building (e.g., ALD, coherence, and surprise), the evaluation of different explicit nonlinear maps for RKHS, such as Gaussian quadrature and Taylor series, and the incorporation of a pruning mechanism to the dictionary building procedure.

# BIBLIOGRAPHY

AGUIRRE, L. **Introdução à Identificação de Sistemas**. Belo Horizonte: UFMG, 2015. ISBN 978-85-423-0079-6.

AKAIKE, H. A new look at the statistical model identification. **IEEE Transactions on Automatic Control**, v. 19, n. 6, p. 716–723, 1974.

ALY, H. H. An intelligent hybrid model of neuro wavelet, time series and recurrent kalman filter for wind speed forecasting. **Sustainable Energy Technologies and Assessments**, v. 41, p. 100802, 2020. ISSN 2213-1388.

ARONSZAJN, N. Theory of reproducing kernels. **Transactions of the American Mathematical Society**, American Mathematical Society, v. 68, n. 3, p. 337–404, 1950. ISSN 00029947.

BACH, F.; JENATTON, R.; MAIRAL, J. **Optimization with Sparsity-Inducing Penalties (Foundations and Trends(R) in Machine Learning)**. Hanover, MA, USA: Now Publishers Inc., 2011. ISBN 160198510X, 9781601985101.

BAKKER, R.; SCHOUTEN, J. C.; GILES, C. L.; TAKENS, F.; BLEEK, C. M. van den. Learning chaotic attractors by neural networks. **Neural Computation**, v. 12, p. 2355–2383, 2000.

BARTO, A.; MIROLLI, M.; BALDASSARRE, G. Novelty or surprise? **Frontiers in Psychology**, Frontiers Media SA, v. 4, 2013.

BESSA, R.; BARRETO, G. A. A robust echo state network for recursive system identification. **15th International Work-Conference on Artificial Neural Networks (IWANN 2019)**, June 2019.

BILLINGS, S. Nonlinear system identification: Narmax methods in the time, frequency, and spatio-temporal domains. **Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains**, 08 2013.

BISHOP, C. M. **Pattern Recognition and Machine Learning (Information Science and Statistics)**. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 0387310738.

BITTANTI, S.; PIRODDI, L. Nonlinear identification and control of a heat exchanger: A neural network approach. **Journal of the Franklin Institute**, v. 334, n. 1, p. 135 – 153, 1997. ISSN 0016-0032.

BOEING, G. Visual analysis of nonlinear dynamical systems: Chaos, fractals, self-similarity and the limits of prediction. **Systems**, v. 4, n. 4, 2016. ISSN 2079-8954.

BORWEIN, J. M.; LEWIS, A. S. **Convex Analysis and Nonlinear Optimization: Theory and Examples**. New York, NY, MA: Springer-Verlag, 2006. ISBN 978-0-387-31256-9.

BURRUS, C. S. Iterative reweighted least squares. **OpenStax CNX**, p. 1–14, 2012.

CAO, L. Practical method for determining the minimum embedding dimension of a scalar time series. **Physica D: Nonlinear Phenomena**, v. 110, n. 1, p. 43 – 50, 1997. ISSN 0167-2789.

CHEN, B.; LIANG, J.; ZHENG, N.; PRINCIPE, J. C. Kernel least mean square with adaptive kernel size. **Neurocomputing**, v. 191, p. 95–106, 2016.

CHEN, B.; LIU, X.; ZHAO, H.; PRINCIPE, J. C. Maximum correntropy Kalman filter. **Automatica**, v. 76, p. 70–77, 2017.

CHEN, B.; WANG, X.; LU, N.; WANG, S.; CAO, J.; QIN, J. Mixture correntropy for robust learning. **Pattern Recognition**, v. 79, p. 318–327, 2018. ISSN 0031-3203.

CHIUSO, A.; PILLONETTO, G. System identification: A machine learning perspective. **Annual Review of Control, Robotics, and Autonomous Systems**, v. 2, n. 1, p. 281–304, 2019.

CORTES, C.; VAPNIK, V. Support-vector networks. **Mach. Learn.**, Kluwer Academic Publishers, USA, v. 20, n. 3, p. 273–297, set. 1995. ISSN 0885-6125.

COTTER, A.; KESHET, J.; SREBRO, N. Explicit approximations of the gaussian kernel. **CoRR**, abs/1109.4603, 2011.

CSATÓ, L.; OPPER, M. Sparse representation for gaussian process models. In: **Proceedings of the 13th International Conference on Neural Information Processing Systems**. Cambridge, MA, USA: MIT Press, 2000. (NIPS'00).

DAO, T.; SA, C. D.; RÉ, C. Gaussian quadrature for kernel features. In: **Proceedings of the 31st International Conference on Neural Information Processing Systems**. Red Hook, NY, USA: Curran Associates Inc., 2017. (NIPS'17), p. 6109–6119. ISBN 9781510860964.

DE KRUIF, B.; DE VRIES, T. Pruning error minimization in least squares support vector machines. **IEEE transactions on neural networks and learning systems**, IEEE Computational Intelligence Society, v. 14, n. 3, p. 696–702, 2003. ISSN 2162-237X.

DIKS, C.; VAN ZWET, W. R.; TAKENS, F.; DEGOEDE, J. Detecting differences between delay vector distributions. **Phys. Rev. E**, American Physical Society, v. 53, p. 2169–2176, Mar 1996.

DITZLER, G.; ROVERI, M.; ALIPPI, C.; POLIKAR, R. Learning in nonstationary environments: A survey. **IEEE Computational Intelligence Magazine**, v. 10, n. 4, p. 12–25, 2015.

DUARTE, M. S.; BARRETO, G. A. Online sparse correntropy kernel learning for outlier-robust system identification. **IFAC Proceedings Volumes**, v. 1, n. 1, p. 424 – 429, 2019. ISSN 1474-6670. 12th IFAC International Symposium on Dynamics and Control of Process Systems.

ENGEL, Y.; MANNOR, S.; MEIR, R. Sparse online greedy support vector regression. **Machine Learning: ECML 2002**, 2002.

ENGEL, Y.; MANNOR, S.; MEIR, R. The kernel recursive least-squares algorithm. **IEEE Transactions on Signal Processing**, v. 52, n. 8, p. 2275–2285, Aug 2004. ISSN 1053-587X.

FAN, H.; JIANG, J.; ZHANG, C.; WANG, X.; LAI, Y.-C. Long-term prediction of chaotic systems with machine learning. **Phys. Rev. Research**, American Physical Society, v. 2, p. 012080, Mar 2020.

FAN, H.; SONG, Q. A sparse kernel algorithm for online time series data prediction. **Expert Systems with Applications**, v. 40, n. 6, p. 2174–2181, 2013.

FAN, H.; SONG, Q.; XU, Z. An information theoretic sparse kernel algorithm for online learning. **Expert Systems with Applications**, v. 41, n. 9, p. 4349–4359, 2014.

FAN, R.-E.; CHEN, P.-H.; LIN, C.-J. Working set selection using second order information for training support vector machines. **Journal of machine learning research**, v. 6, 12 2005.

FAÇANHA, T. S.; BARRETO, G. A.; COSTA FILHO, J. T. A novel Kalman filter formulation for improving tracking performance of the extended kernel RLS. **Circuits, Systems, and Signal Processing**, 2020.

FOX, J.; MONETTE, G. **R and S-Plus Companion to Applied Regression**. USA: Sage Publications, Inc., 2002. ISBN 0761922792.

GAO, W.; CHEN, J.; RICHARD, C.; HUANG, J.; FLAMARY, R. Kernel lms algorithm with forward-backward splitting for dictionary learning. p. 5735–5739, May 2013. ISSN 1520-6149.

GIROSI, F.; JONES, M.; POGGIO, T. Regularization theory and neural networks architectures. **Neural Computation**, v. 7, n. 2, p. 219–269, 1995.

GOLUB, G. H.; LOAN, C. F. V. Matrix computations. The Johns Hopkins University Press, 2012.

GRASSBERGER, P.; SCHREIBER, T.; SCHAFFRATH, C. Nonlinear time sequence analysis. **International Journal of Bifurcation and Chaos**, v. 01, n. 03, p. 521–547, 1991.

GRAY, R. M. Toeplitz and circulant matrices: A review. **Foundations and Trends® in Communications and Information Theory**, v. 2, n. 3, p. 155–239, 2006. ISSN 1567-2190.

GREENGARD, L.; STRAIN, J. The fast gauss transform. **SIAM Journal on Scientific and Statistical Computing**, Society for Industrial & Applied Mathematics (SIAM), v. 12, n. 1, p. 79–94, jan. 1991.

GRIFFITH, A.; POMERANCE, A.; GAUTHIER, D. J. Forecasting chaotic systems with very low connectivity reservoir computers. **Chaos: An Interdisciplinary Journal of Nonlinear Science**, v. 29, n. 12, p. 123108, 2019.

GUIMARÃES, J. P.; FONTES, A. I.; REGO, J. B.; MARTINS, A. d. M.; PRINCIPE, J. C. Complex correntropy function: Properties, and application to a channel equalization problem. **Expert Systems with Applications**, v. 107, p. 173–181, 2018.

HAMPEL, F. R.; RONCHETTI, E. M.; J., R. P. Robust statistics: The approach based on influence functions. Wiley, New York, 1985.

HAN, M.; XI, J.; XU, S.; YIN, F.-L. Prediction of chaotic time series based on the recurrent predictor neural network. **IEEE Transactions on Signal Processing**, v. 52, n. 12, p. 3409–3416, 2004.

HASSIBI, B. On the robustness of lms filters. In: _____. **Least-Mean-Square Adaptive Filters**. New York: John Wiley  Sons, Ltd, 2003. cap. 4, p. 105–144. ISBN 9780471461289.

HAYKIN, S. **Adaptive Filter Theory (3rd Ed.)**. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996. ISBN 0-13-322760-X.

HAYKIN, S. S. **Neural networks and learning machines**. Third. Upper Saddle River, NJ: Pearson Education, 2009.

HOFMANN, T.; SCHOLKOPF, B.; SMOLA, A. J. Kernel methods in machine learning. **The Annals of Statistics**, Institute of Mathematical Statistics, v. 36, n. 3, p. 1171 – 1220, 2008.

HOI, S. C.; WANG, J.; ZHAO, P. Libol: A library for online learning algorithms. **Journal of Machine Learning Research**, v. 15, n. 15, p. 495–499, 2014.

HONEINE, P. Analyzing sparse dictionaries for online learning with kernels. **IEEE Transactions on Signal Processing**, v. 63, n. 23, p. 6343–6353, Dec 2015. ISSN 1053-587X.

HUBER, P. Robust statistics. 1981. New York: Wiley.

HUSSAIN, V. S.; SPANO, M. L.; LOCKHART, T. E. Effect of data length on time delay and embedding dimension for calculating the lyapunov exponent in walking. **Journal of The Royal Society Interface**, v. 17, n. 168, p. 20200311, 2020.

JAEGER, H. The" echo state" approach to analysing and training recurrent neural networks-with an erratum note'. **Bonn, Germany: German National Research Center for Information Technology GMD Technical Report**, v. 148, 01 2001.

JAEGER, H.; HAAS, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. **Science**, American Association for the Advancement of Science, v. 304, n. 5667, p. 78–80, 2004. ISSN 0036-8075.

JAEGER, L.; KANTZ, H. Unbiased reconstruction of the dynamics underlying a noisy chaotic time series. **Chaos**, v. 6 3, p. 440–450, 1996.

JUAREZ-RUIZ, E.; CORTES-MALDONADO, R.; PEREZ-RODRIGUEZ, F. Relationship between the inverses of a matrix and a submatrix. **Computacion y Sistemas**, scielomx, v. 20, p. 251 – 262, 06 2016. ISSN 1405-5546.

JUNGLING, T.; LYMBURN, T.; STEMLER, T.; CORRÊA, D.; WALKER, D.; SMALL, M. Reconstruction of complex dynamical systems from time series using reservoir computing. In: **2019 IEEE International Symposium on Circuits and Systems (ISCAS)**. Sapporo, Japan: IEEE, 2019. p. 1–5. ISSN 2158-1525.

KIVINEN, J.; SMOLA, A.; WILLIAMSON, R. Online learning with kernels. **Trans. Sig. Proc.**, IEEE Press, v. 52, n. 8, p. 2165–2176, ago. 2004. ISSN 1053-587X.

KOPPEL, A.; PATERNAIN, S.; RICHARD, C.; RIBEIRO, A. Decentralized online learning with kernels. **IEEE Transactions on Signal Processing**, v. 66, n. 12, p. 3240–3255, June 2018. ISSN 1053-587X.

KREMER, S. C. **Field Guide to Dynamical Recurrent Networks**. 1st. ed. Piscataqay, New Jersey: Wiley-IEEE Press, 2001. ISBN 0780353692.

KUGIUMTZIS, D.; LILLEKJENDLIE, B.; CHRISTOPHERSEN, N. Chaotic time series part i: Estimation of invariant properties in state space. **Modeling, Identification and Control**, v. 15, 01 1994.

KUNG, S. Y. **Kernel Methods and Machine Learning**. Princeton University, New Jersey: Cambridge University Press, 2014.

KUO, J.-M.; PRINCIPE, J. Reconstructed dynamics and chaotic signal modeling. In: **Proceedings of IEEE Workshop on Neural Networks for Signal Processing**. Ermioni, Greece: IEEE, 1994. p. 661–670.

LAFLEUR, M.; HINRICHSEN, P.; LANDRY, P.; MOORE, R. The poisson distribution. **The Physics Teacher**, v. 10, p. 314–321, 09 1972.

LETELLIER, C.; AGUIRRE, L. A.; MAQUET, J.; GILMORE, R. Evidence for low dimensional chaos in sunspot cycles. , v. 449, p. 379–387, abr. 2006.

LI, G.; WEN, C.; Li, Z. G.; ZHANG, A.; YANG, F.; MAO, K. Model-based online learning with kernels. **IEEE Transactions on Neural Networks and Learning Systems**, v. 24, n. 3, p. 356–369, 2013.

LI, K.; PRÍNCIPE, J. C. Surprise-novelty information processing for gaussian online active learning (snip-goal). **2018 International Joint Conference on Neural Networks (IJCNN)**, p. 1–6, 2018.

LI, K.; PRINCIPE, J. C. No-trick (treat) kernel adaptive filtering using deterministic features. 2019.

LI, L.-Q.; SUN, Y.-C.; LIU, Z.-X. Maximum fuzzy correntropy kalman filter and its application to bearings-only maneuvering target tracking. **International Journal of Fuzzy Systems**, Springer Science and Business Media LLC, v. 23, n. 2, p. 405–418, fev. 2021.

LI, Y.; LI, W.; YU, W.; WAN, J.; LI, Z. Sparse adaptive channel estimation based on l p -norm-penalized affine projection algorithm. **International Journal of Antennas and Propagation**, v. 2014, p. 1–8, 07 2014.

LIU, W.; PARK, I.; PRINCIPE, J. C. An information theoretic approach of designing sparse kernel adaptive filters. **IEEE Transactions on Neural Networks**, v. 20, n. 12, p. 1950–1961, 2009.

LIU, W.; PARK, I.; WANG, Y.; PRINCIPE, J. Extended kernel recursive least squares algorithm. **IEEE Transactions on Signal Processing**, v. 57, n. 10, p. 3801–3804, 2009.

LIU, W.; POKHAREL, P.; PRINCIPE, J. The kernel least mean square algorithm. **IEEE Transactions on Signal Processing**, v. 56, n. 2, p. 543–554, 2008.

LIU, W.; POKHAREL, P. P.; PRINCIPE, J. C. Correntropy: Properties and applications in non-gaussian signal processing. **IEEE Transactions on Signal Processing**, v. 55, n. 11, p. 5286–5298, 2007.

LIU, W.; PRINCIPE, J.; HAYKIN, S. Kernel adaptive filtering: A comprehensive introduction. p. i–xxi, 03 2010.

LIU, W.; PRINCIPE, J. C. Kernel affine projection algorithms. **EURASIP Journal on Advances in Signal Processing**, v. 784292, p. 1–12, 2008.

LIU, Y.; CHEN, J. Correntropy-based kernel learning for nonlinear system identification with unknown noise: an industrial case study. **IFAC Proceedings Volumes**, v. 46, n. 32, p. 361 – 366, 2013. ISSN 1474-6670. 10th IFAC International Symposium on Dynamics and Control of Process Systems.

LIU, Y.; CHEN, J. Correntropy kernel learning for nonlinear system identification with outliers. **Industrial & Engineering Chemistry Research**, v. 53, p. 5248–5260, 2014.

LJUNG, L. **System Identification: Theory for the User**. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1999.

LJUNG, L. Perspectives on system identification. **IFAC Proceedings Volumes**, v. 41, n. 2, p. 7172–7184, 2008. ISSN 1474-6670. 17th IFAC World Congress.

LJUNG, L.; HJALMARSSON, H.; OHLSSON, H. Four encounters with system identification. **European Journal of Control**, v. 17, n. 5, p. 449 – 471, 2011. ISSN 0947-3580.

LJUNG, L.; VICINO, A. editors. **IEEE Trans. Automatic Control: Special Issue on Identification**, AC-50, 09 2005.

LUKOŠEVIČIUS, M. A practical guide to applying echo state networks. In: **Lecture Notes in Computer Science**. Berlin: Springer, 2012. p. 659–686.

LUKOSEVICIUS, M.; JAEGER, H. Reservoir computing approaches to recurrent neural network training. **Computer Science Review**, v. 3, n. 3, p. 127 – 149, 2009. ISSN 1574-0137.

MACKEY, M.; GLASS, L. Oscillation and chaos in physiological control systems. **Science**, American Association for the Advancement of Science, v. 197, n. 4300, p. 287–289, 1977. ISSN 0036-8075.

MATIAS, T.; SOUZA, F.; ARAÚJO, R.; GONÇALVES, N.; BARRETO, J. P. On-line sequential extreme learning machine based on recursive partial least squares. **Journal of Process Control**, v. 27, p. 15–21, 2015. ISSN 0959-1524.

MATTOS, C. L. C. **Recurrent gaussian processes and robust dynamical modeling**. [S.l.]: Universidade Federal do Ceará, 2017. 1 - 189 p.

MATTOS, C. L. C.; DAMIANOU, A.; BARRETO, G.; LAWRENCE, N. D. Latent autoregressive gaussian processes models for robust system identification. **IFAC-PapersOnLine**, v. 49, p. 1121–1126, 12 2016.

MENEZES, J. M. P.; BARRETO, G. A. Long-term time series prediction with the narx network: An empirical evaluation. **Neurocomputing**, v. 71, n. 16-18, p. 3355–2383, 2008.

MERCER, J.; FORSYTH, A. R. Xvi. functions of positive and negative type, and their connection the theory of integral equations. **Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character**, v. 209, n. 441-458, p. 415–446, 1909.

MICCHELLI, C. A.; XU, Y.; ZHANG, H. Universal kernels. **Mathematics**, v. 7, 12 2006.

MOOR, B. L. R. D. Daisy: Database for the identification of systems. **Department of Electrical Engineering, ESAT/STADIUS, KU Leuven, Belgium**, 2019.

MUNOZ, J. C.; CHEN, J. Removal of the effects of outliers in batch process data through maximum correntropy estimator. **Chemometrics and Intelligent Laboratory Systems**, v. 111, n. 1, p. 53 – 58, 2012. ISSN 0169-7439.

NARENDRA, K. S.; PARTHASARATHY, K. Identification and control of dynamical systems using neural networks. **IEEE Transactions on Neural Networks**, v. 1, n. 1, p. 4–27, 1990.

NEHMZOW, U. **Scientific Methods in Mobile Robotics – Quantitative Analysis of Agent Behaviour**. [S.l.: s.n.], 2006. ISBN 978-1-84628-019-1.

PAPOULIS, A. **Probability, Random Variables, and Stochastic Processes (3rd Ed.)**. New York, USA: McGraw-Hill, 1991. ISBN 0-07-048477-5.

PEARSON, R. K. Outliers in process modeling and identification. **IEEE Transactions on Control Systems Technology**, v. 10, n. 1, p. 55–63, Jan 2002. ISSN 1063-6536.

PILLONETTO, G.; NICOLAO, G. A new kernel-based approach for linear system identification. **Automatica**, v. 46, p. 81–93, 01 2010.

PLATT, J. A resource-allocating network for function interpolation. **Neural Computation**, v. 3, n. 2, p. 213–225, June 1991. ISSN 0899-7667.

PRINCIPE, J. C. **Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives**. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2010. ISBN 1441915699, 9781441915696.

PRINCIPE, J. C.; RATHIE, A.; KUO, J.-M. Prediction of chaotic time series with neural networks and the issue of dynamic modeling. **International Journal of Bifurcation and Chaos**, v. 02, n. 04, p. 989–996, 1992.

QAADAN, S.; SCHÜLER, M.; GLASMACHERS, T. Dual SVM training on a budget. **CoRR**, abs/1806.10182, 2018.

RAHIMI, A.; RECHT, B. Random features for large-scale kernel machines. In: **Proceedings of the 20th International Conference on Neural Information Processing Systems**. Red Hook, NY, USA: Curran Associates Inc., 2007. (NIPS'07), p. 1177–1184. ISBN 9781605603520.

RAHIMI, A.; RECHT, B. Random features for large-scale kernel machines. **Advances in Neural Information Processing Systems**, Curran Associates, Inc., p. 1177–1184, 2008.

RASMUSSEN, C. E. Gaussian processes in machine learning. In: ______. **Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 63–71. ISBN 978-3-540-28650-9.

RICHARD, C.; BERMUDEZ, J. C. M.; HONEINE, P. Online prediction of time series data with kernels. **IEEE Transactions on Signal Processing**, v. 57, n. 3, p. 1058–1067, 2009.

ROJO-ÁLVAREZ, J. L.; MARTÍNEZ-RAMÓN, M.; MARÍ, J. Muñoz; CAMPS-VALLS, G. A unified SVM framework for signal estimation. **Digit. Signal Process.**, Academic Press, Inc., USA, v. 26, p. 1–20, mar. 2014. ISSN 1051-2004.

ROSENSTEIN, M. T.; COLLINS, J. J.; LUCA, C. J. D. A practical method for calculating largest lyapunov exponents from small data sets. **Physica D: Nonlinear Phenomena**, v. 65, n. 1, p. 117 – 134, 1993. ISSN 0167-2789.

SAIDE, C.; LENGELLE, R.; HONEINE, P.; ACHKAR, R. Online kernel adaptive algorithms with dictionary adaptation for mimo models. **IEEE Signal Processing Letters**, v. 20, n. 5, p. 535–538, 2013.

SAIDE, C.; LENGELLE, R.; HONEINE, P.; RICHARD, C.; ACHKAR, R. Nonlinear adaptive filtering using kernel-based algorithms with dictionary adaptation. **International Journal of Adaptive Control and Signal Processing**, v. 29, 02 2015.

SAMAD, T. Neural networks for system identification: A control industry perspective. **IFAC Proceedings Volumes**, v. 30, n. 11, p. 749 – 754, 1997. ISSN 1474-6670.

SANGIORGIO, M.; DERCOLE, F. Robustness of lstm neural networks for multi-step forecasting of chaotic time series. **Chaos, Solitons  Fractals**, v. 139, p. 110045, 2020. ISSN 0960-0779.

SANTAMARIA, I.; POKHAREL, P. P.; PRINCIPE, J. C. Generalized correlation function: definition, properties, and application to blind equalization. **IEEE Transactions on Signal Processing**, v. 54, n. 6, p. 2187–2197, 2006.

SANTOS, J. D. A. Adaptive kernel-based regression for robust system identification. Universidade Federal do Ceará, 2017.

SANTOS, J. D. A.; BARRETO, G. A. An outlier-robust kernel rls algorithm for nonlinear system identification. **Nonlinear Dynamics**, v. 90, n. 3, p. 1707–1726, 2017.

SANTOS, J. D. A.; BARRETO, G. A. Novel sparse LSSVR models in primal weight space for robust system identification with outliers. **Journal of Process Control**, v. 67, p. 129–140, 2018.

SCHLAG, S.; SCHMITT, M.; SCHULZ, C. Faster support vector machines. p. 199–210, 01 2019.

SCHÖLKOPF, B.; SMOLA, A. J. **Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond**. Cambridge, MA, USA: MIT Press, 2002.

SCHOUKENS, J.; LJUNG, L. Nonlinear system identification: A user-oriented roadmap. **CoRR**, abs/1902.00683, 2019.

SCHOUKENS, J.; SUYKENS, J.; LJUNG, L. Wiener-Hammerstein benchmark. In: **Proceedings of the 15th IFAC Symposium on System Identification (SYSID'2009)**. [S.l.: s.n.], 2009. p. 1–5.

SHABESTARI, P. S.; PANAHI, S.; HATEF, B.; JAFARI, S.; SPROTT, J. C. A new chaotic model for glucose-insulin regulatory system. **Chaos, Solitons  Fractals**, v. 112, p. 44 – 51, 2018. ISSN 0960-0779.

SHALEV-SHWARTZ, S.; SINGER, Y.; SREBRO, N.; COTTER, A. Pegasos: Primal estimated sub-gradient solver for SVM. **Math. Program.**, v. 127, p. 3–30, 03 2011.

SHEN, T.; REN, W.; HAN, M. Quantized generalized maximum correntropy criterion based kernel recursive least squares for online time series prediction. **Engineering Applications of Artificial Intelligence**, v. 95, p. 103797, 2020.

SHI, Z.; HAN, M. Support vector echo-state machine for chaotic time-series prediction. **IEEE Transactions on Neural Networks**, v. 18, n. 2, p. 359–372, March 2007. ISSN 1045-9227.

SI, S.; HSIEH, C.-J.; DHILLON, I. S. Memory efficient kernel approximation. **Journal of Machine Learning Research**, v. 18, n. 6, p. 2174–2181, 2017.

SIDAK, Z.; SEN, P.; HAJEK, J. **Theory of Rank Tests**. New York: Academic Press, 1999.

SILSO World Data Center. The International Sunspot Number. **International Sunspot Number Monthly Bulletin and online catalogue**, Royal Observatory of Belgium, avenue Circulaire 3, 1180 Brussels, Belgium, 1749–2018.

SILVERMAN, B. **Density Estimation for Statistics and Data Analysis**. [S.l.]: Routledge, 2018.

SPARROW, C. **The Lorenz Equations: Bifurcations, Chaos, and Strange Attractors**. New York: Springer-Verlag, 1985. v. 27. 106-110 p.

STEINWART, I.; HUSH, D.; SCOVEL, C. Training SVMs without offset. **Journal of Machine Learning Research**, v. 12, n. 6, p. 141–202, 2011.

SU, H.; MCAVOY, T.; WERBOS, P. Long-term predictions of chemical processes using recurrent neural networks. a parallel training approach. **Industrial  Engineering Chemistry Research - IND ENG CHEM RES**, v. 31, 05 1992.

SUYKENS, J.; BRABANTER, J. D.; LUKAS, L.; VANDEWALLE, J. Weighted least squares support vector machines: robustness and sparse approximation. **Neurocomputing**, v. 48, p. 85–105, 10 2002.

TAKENS, F. Detecting strange attractors in turbulence. In: RAND, D.; YOUNG, L.-S. (Ed.). **Dynamical Systems and Turbulence, Warwick 1980**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1981. p. 366–381. ISBN 978-3-540-38945-3.

THOMANN, P.; BLASCHZYK, I.; MEISTER, M.; STEINWART, I. Spatial decompositions for large scale SVMs. 2016.

TIKHONOV, A. N.; ARSENIN, V. Y. **Solutions of Ill-posed problems**. Russia: W.H. Winston, 1977.

TRISCHLER, A. P.; D'ELEUTERIO, G. M. Synthesis of recurrent neural networks for dynamical system simulation. **Neural Networks**, v. 80, p. 67 – 78, 2016. ISSN 0893-6080.

VAN VAERENBERGH, S.; LAZARO-GREDILLA, M.; SANTAMARIA, I. Kernel recursive least-squares tracker for time-varying regression. **IEEE Transactions on Neural Networks and Learning Systems**, v. 23, n. 8, p. 1313–1326, Aug 2012. ISSN 2162-237X.

VERSTRAETEN, D.; DAMBRE, J.; DUTOIT, X.; SCHRAUWEN, B. Memory versus non-linearity in reservoirs. **The 2010 International Joint Conference on Neural Networks (IJCNN)**, p. 1–8, 2010.

WANG, S.; DANG, L.; QIAN, G.; JIANG, Y. Kernel recursive maximum correntropy with nyström approximation. **Neurocomputing**, v. 329, p. 424 – 432, 2019. ISSN 0925-2312.

WEIBULL, W. A statistical distribution function of wide applicability. **Journal of Applied Mechanics**, v. 18, p. 293–297, 1951.

WIGREN, T.; SCHOUKENS, J. Three free data sets for development and benchmarking in nonlinear system identification. **European Control Conference (ECC)**, p. 2933–2938, July 2013.

WU, Z.; SHI, J.; ZHANG, X.; MA, W.; CHEN, B. Kernel recursive maximum correntropy. **Signal Processing**, v. 117, p. 11 – 16, 2015.

XU, J.-W.; PRINCIPE, J. C. A pitch detector based on a generalized correlation function. **IEEE Transactions on Audio, Speech, and Language Processing**, v. 16, n. 8, p. 1420–1432, 2008.

YUAN, G.-X.; HO, C.-H.; LIN, C.-J. An improved glmnet for l1-regularized logistic regression. **Journal of Machine Learning Research**, v. 13, n. 64, p. 1999–2030, 2012.

ZADEH, L. A. On the identification problem. **RE Transactions onCircuit Theory**, 1956. ISSN 3:277–281.

ZHAO, J.; ZHANG, H.; WANG, G. Projected kernel recursive maximum correntropy. **IEEE Transactions on Circuits and Systems II: Express Briefs**, v. 65, n. 7, p. 963–967, July 2018. ISSN 1549-7747.

ZHAO, S.; CHEN, B.; PRINCIPE, J. C. Kernel adaptive filtering with maximum correntropy criterion. In: **The 2011 International Joint Conference on Neural Networks**. San Jose, CA: IEEE, 2011. p. 2012–2017. ISSN 2161-4407.

ZHONG, K.; MA, J.; HAN, M. Online prediction of noisy time series: Dynamic adaptive sparse kernel recursive least squares from sparse and adaptive tracking perspective. **Engineering Applications of Artificial Intelligence**, v. 91, p. 103547, 2020.

ZHOU, H.; HUANG, J.; LU, F.; THIYAGALINGAM, J.; KIRUBARAJAN, T. Echo state kernel recursive least squares algorithm for machine condition prediction. **Mechanical Systems and Signal Processing**, v. 111, p. 68 – 86, 2018. ISSN 0888-3270.

ZOU, H. The adaptive LASSO and its oracle properties. **Journal of the American Statistical Association**, v. 101, n. 476, p. 1418–1429, 2006.