



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

**JOSÉ DANIEL DE ALENCAR SANTOS**

**ADAPTIVE KERNEL-BASED REGRESSION FOR ROBUST SYSTEM  
IDENTIFICATION**

**FORTALEZA**

**2017**

JOSÉ DANIEL DE ALENCAR SANTOS

ADAPTIVE KERNEL-BASED REGRESSION FOR ROBUST SYSTEM IDENTIFICATION

Tese apresentada ao Curso de Doutorado em Engenharia de Teleinformática do 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

2017

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)

---

S235a Santos, José Daniel de Alencar.  
Adaptive Kernel-Based Regression for Robust System Identification / José Daniel de Alencar Santos. –  
2017.  
206 f.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação  
em Engenharia de Teleinformática, Fortaleza, 2017.  
Orientação: Prof. Dr. Guilherme de Alencar Barreto.

1. LSSVR Model. 2. FS-LSSVR Model. 3. KRLS Model. 4. Robustness. 5. System Identification. I.  
Título.

CDD 621.38

---

JOSÉ DANIEL DE ALENCAR SANTOS

ADAPTIVE KERNEL-BASED REGRESSION FOR ROBUST SYSTEM IDENTIFICATION

A thesis presented to the PhD course in Teleinformatics Engineering of 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 of Philosophy in Teleinformatics Engineering.  
Area of Concentration: Signals and systems

Approved on: May 22, 2017

EXAMINING COMMITTEE

---

Prof. Dr. Guilherme de Alencar Barreto  
(Supervisor)  
Universidade Federal do Ceará (UFC)

---

Prof. Dr. Luis Antonio Aguirre  
Universidade Federal de Minas Gerais (UFMG)

---

Prof. Dr. Marcello Luiz Rodrigues de Campos  
Universidade Federal do Rio de Janeiro (UFRJ)

---

Prof. Dr. Amauri Holanda de Souza Júnior  
Instituto Federal de Educação, Ciência e  
Tecnologia do Ceará (IFCE)

---

Prof. Dr. Charles Casimiro Cavalcante  
Universidade Federal do Ceará (UFC)

In dedication to my beloved son Davi.

## ACKNOWLEDGEMENTS

First and foremost, thank you God for so many blessings in my life.

To my parents José and Lucélia, for the example they set for me and for their efforts in always giving me the best that they could offer;

To my beloved wife Juliana, for her patience and unconditional love, and for accompanying me along this journey;

To my advisor, Prof. Dr. Guilherme Barreto, for trusting in me, for being available whenever I needed him and for guiding me with mastery throughout this work;

To my colleagues in GRAMA, for their cooperation and for the friendly environment that has been established since the beginning of this work. I also thank my friend César Lincoln, for our fruitful discussions and for his valuable technical collaboration during this work.

To the professors and administrative staff of DETI, for their support during the development of this PhD research;

To NUTEC and CENTAURO, for providing the laboratory infrastructure;

To IFCE, for allowing my full dedication to this work during the last years;

To the professors and colleagues of the Industry Department of IFCE - Campus of Maracanaú, for the customary partnership;

To everyone else who has contributed in some way to this work.

“Cada coisa tem sua hora e cada hora o seu cuidado.”

(Rachel de Queiroz)

## RESUMO

O estudo de sistemas dinâmicos encontra-se disseminado em várias áreas do conhecimento. Dados sequenciais são gerados constantemente por diversos fenômenos, a maioria deles não passíveis de serem explicados por equações derivadas de leis físicas e estruturas conhecidas. Nesse contexto, esta tese tem como objetivo abordar a tarefa de identificação de sistemas não lineares, por meio da qual são obtidos modelos diretamente. Os modelos de regressão por vetores-suporte via mínimos quadrados (LSSVR) e LSSVR de tamanho fixo (FS-LSSVR) são alternativas interessantes à regressão por vetores-suporte (SVR). Aqueles são derivados de funções custo baseadas em soma dos erros quadráticos (SSE) e restrições de igualdade, diferentemente do modelo SVR, cujo o problema de programação quadrática associado não apresenta bom desempenho em problemas de maior escala, além de consumir tempo considerável de processamento. Os problemas de otimização dos modelos LSSVR e FS-LSSVR tornam-se mais simples por permitir a solução de um sistema linear pelo método dos mínimos quadrados. Para o modelo LSSVR, contudo, a solução assim encontrada não é esparsa, implicando na utilização de todos os dados de treinamento como vetores-suporte. Por sua vez, a formulação do modelo FS-LSSVR é baseada no problema de otimização primal, que conduz a uma solução esparsa (i.e. uma parcela dos dados é usada pelo preditor). Entretanto, há aplicações em identificação de sistemas e processamento de sinais em que a estimação online de parâmetros é requerida para cada nova amostra. Neste sentido, a aplicação de funções de *kernel* a filtros lineares ajudou a estabelecer um campo de pesquisa emergente, o de *filtragem adaptativa kernelizada* para processamento não linear de sinais. Um algoritmo pioneiro nesse campo é o estimador de mínimos quadrados recursivo kernelizado (KRLS). Uma das contribuições desta tese consiste em utilizar o algoritmo KRLS para transformar o modelo LSSVR em um modelo adaptativo e esparsos. Além da questão da esparsidade da solução, as contribuições adicionais deste trabalho foram motivadas pelo tratamento adequado de ruído não gaussiano e *outliers*. Assim como os modelos LSSVR e FS-LSSVR, o modelo KRLS é também construído com base em uma função de custo SSE, o que garante desempenho ótimo apenas para ruído branco gaussiano. Em outras palavras, o desempenho daqueles modelos tende a degradar-se consideravelmente quando tal condição não é observada. Isto posto, nesta tese são desenvolvidas ainda quatro abordagens robustas para os modelos LSSVR, FS-LSSVR e KRLS. O arcabouço de estimação robusta de parâmetros conhecido como *estimação-M* é utilizado para este fim. Para o modelo LSSVR, uma abordagem mais heurística é seguida, em que uma versão robusta,



porém não esparsa, é obtida simplesmente trocando-se o método de estimação dos mínimos quadrados pelo algoritmo RLM (versão robusta do RLS). Para os modelos FS-LSSVR e KRLS, abordagens de cunho mais teórico são seguidas, em que se alteram as funções custo originais para se chegar aos modelos robustos propostos. Os desempenhos dos modelos propostos são avaliados e discutidos em tarefas de identificação robusta de sistemas com conjuntos de dados artificiais e reais em cenários com predição de  $k$ -passos a frente e simulação livre.

**Palavras-chave:** Modelo LSSVR. Modelo FS-LSSVR. Modelo KRLS. Esparsidade. *Outliers*. Estimação-M. Robustez. Identificação de Sistemas.

## ABSTRACT

The Least Squares Support Vector Regression (LSSVR) and Fixed Size LSSVR (FS-LSSVR) models are interesting alternatives to the Support Vector Regression (SVR). Those are derived from cost functions based on the sum-of-squared-errors (SSE) and equality constraints, unlike the SVR model, whose associated quadratic programming problem does not scale up well, and besides consumes considerable processing time. The optimization problems of the LSSVR and FS-LSSVR models become simpler because they rely on the ordinary least squares method to find a solution. For the LSSVR model, nevertheless, the solution thus found is non-sparse, implying the use of all training data as support vectors. In turn, the formulation of the FS-LSSVR model is based on the primal optimization problem, which leads to a sparse solution (i.e. a portion of the training data is used by the predictor). However, there are applications in system identification and signal processing in which online parameter estimation is required for each new sample. In this sense, the application of kernelization to linear filters has helped to establish an emerging field, that of kernel adaptive filtering for nonlinear processing of signals. A pioneering algorithm in this field is the Kernel Recursive Least Squares (KRLS) estimator. One of the contributions of this thesis consists in using the KRLS algorithm to transform the LSSVR model into an adaptive and sparse model. Beyond the question of the sparsity of the solution, the additional contributions of this work are motivated by the appropriate treatment of non-Gaussian noise and outliers. As the LSSVR and FS-LSSVR models, the KRLS model is also built upon an SSE cost function, which guarantees optimal performance only for Gaussian white noise. In other words, the performance of those models tend to considerably degrade when that condition is not observed. That said, four robust approaches for the LSSVR, FS-LSSVR and KRLS models are developed in this thesis. The framework of the robust parameter estimation known as the M-estimation is used for this purpose. For the LSSVR model, a more heuristic approach is followed, in which a robust, but non-sparse, is simply obtained by replacing the least squares estimation method by the RLM algorithm (a robust version of the RLS). For the FS-LSSVR and KRLS models, more theoretical approaches are developed, in which their original cost functions are changed in order to obtaining the proposed robust models. The performances of the proposed models are comprehensively discussed in robust system identification tasks with synthetic and real-world datasets in scenarios with  $k$ -steps ahead prediction and free simulation.

**Keywords:** LSSVR Model. FS-LSSVR Model. KRLS Model. Sparsity. Outliers. M-Estimation.

Robustness. System Identification.

## LIST OF FIGURES

Figure 1 – Example of a system identification problem, where a model is adapted in order to represent the unknown process behavior. . . . .	31
Figure 2 – Scenarios of predictions for model validation - adapted from Nelles (2001).	33
Figure 3 – Example of 2-steps ahead prediction task - adapted from Nelles (2001). . . .	34
Figure 4 – Illustration of the SVR regression curve with the $\varepsilon$ -insensitive zone and the slack variables (left); Vapnik $\varepsilon$ -insensitive loss function (right). . . . .	44
Figure 5 – Comparison between the standard LSSVR, W-LSSVR and IR-LSSVR models for estimating the sinc function in the presence of outliers. . . . .	68
Figure 6 – Basic configuration of a linear adaptive filter system. . . . .	72
Figure 7 – Basic configuration of a kernel adaptive filter system. . . . .	79
Figure 8 – Probability density functions of the Student's t and Gaussian distributions. .	102
Figure 9 – Input and output sequences of the Synthetic-1 dataset. . . . .	103
Figure 10 – Input and output sequences of the Synthetic-2 dataset. . . . .	103
Figure 11 – Input and output sequences of the Actuator dataset. . . . .	104
Figure 12 – Input and output sequences of the Dryer dataset. . . . .	104
Figure 13 – RMSE values for the test samples over 20 independent runs in free simulation.	106
Figure 14 – Predicted outputs by the best models in Table 6 with their worst performances in RMSE over the 20 runs - Actuator dataset. . . . .	108
Figure 15 – Predicted outputs by the best models in Table 7 with their worst performances in RMSE over the 20 runs - Dryer dataset. . . . .	110
Figure 16 – Input and output sequences - Silverbox dataset. . . . .	119
Figure 17 – Input and output sequences of the Industrial dryer dataset - MIMO system. .	119
Figure 18 – RMSE values for the test samples over 20 independent runs in free simulation.	121
Figure 19 – Predicted outputs by the best models in Table 9 with their worst performances in RMSE over the 20 runs - Actuator dataset. . . . .	123
Figure 20 – Predicted outputs by the best models in Table 10 with their worst performances in RMSE over the 20 runs - Dryer dataset. . . . .	125
Figure 21 – Boxplots for the RMSE values of test samples after 20 independent runs - Silverbox dataset. . . . .	126
Figure 22 – RMSE values for the test samples over 20 independent runs. . . . .	141

Figure 23 – Boxplots for the RMSE values of the test samples after 20 independent runs - Actuator dataset. . . . .	142
Figure 24 – Predicted outputs by the model with worst performance in RMSE along the 20 runs for the outlier-free scenario - Actuator dataset. . . . .	144
Figure 25 – Predicted outputs by the model with worst performance in RMSE along the 20 runs for the scenario with 10% of outliers - Actuator dataset. . . . .	144
Figure 26 – Correlation tests for input-output models - Synthetic-1 dataset with 30% of outliers. . . . .	146
Figure 27 – Correlation tests for input-output models - Synthetic-2 dataset with 30% of outliers. . . . .	147
Figure 28 – Correlation tests for input-output models - Actuator dataset with 10% of outliers. . . . .	148
Figure 29 – Correlation tests for input-output models - Silverbox dataset with 5% of outliers.	149
Figure 30 – Convergence curves for the evaluated online models. . . . .	151
Figure 31 – Obtained RMSE values and predicted outputs for the sinc function. . . . .	159
Figure 32 – Prediction of the laser time-series with 100 test samples. . . . .	161
Figure 33 – Prediction of the laser time-series with 500 test samples. . . . .	161
Figure 34 – Prediction of the laser time-series with 500 test samples for LSSVR and OS-LSSVR models. . . . .	161
Figure 35 – Obtained RMSE values and predicted outputs for the Synthetic-1 dataset - system identification task. . . . .	163
Figure 36 – Obtained RMSE values and predicted outputs for the Synthetic-2 dataset - system identification task. . . . .	164
Figure 37 – Convergence curves for all the datasets: Fig. (37a) - Sinc function. Fig. (37b) - Laser time series. Fig. (37c) - Artificial 1. Fig. (37d) - Artificial 2. Fig. (37e) - Silverbox. . . . .	166

## LIST OF TABLES

Table 1 – Examples of kernel functions. . . . .	47
Table 2 – Comparison of some important characteristics of the SVR, LSSVR and FS-LSSVR models. . . . .	57
Table 3 – Examples of the objective $\rho(\cdot)$ and weight $\nu(\cdot)$ functions for the M-estimators: OLS, Huber, Hampel, Logistic and Myriad. . . . .	63
Table 4 – List of kernel adaptive algorithms related to the KAPA family - adapted from Liu and Príncipe (2008). . . . .	91
Table 5 – Important features of the evaluated datasets and models for the computational experiments. . . . .	104
Table 6 – RMSE for the test samples over 20 independent runs in 3-step-ahead prediction scenario - Actuator dataset. . . . .	107
Table 7 – RMSE for the test samples over 20 independent runs in free simulation scenario - Dryer dataset. . . . .	109
Table 8 – Important features of the evaluated datasets and models for the computational experiments. . . . .	120
Table 9 – RMSE values for the test samples and number of PVs for each evaluated model - Actuator dataset. . . . .	123
Table 10 – RMSE values for the test samples and number of PVs for each evaluated model - Dryer dataset. . . . .	125
Table 11 – RMSE values for test samples for the Industrial Dryer dataset (MIMO system) in outlier-free scenario. . . . .	126
Table 12 – RMSE values for test samples for Industrial Dryer dataset (MIMO system) in scenario with 5% f outliers. . . . .	128
Table 13 – Important features of the evaluated datasets and models for the computational experiments. . . . .	140
Table 14 – Average number of training input vectors chosen as SVs for the synthetic datasets. . . . .	141
Table 15 – RMSE values for test samples and number of SVs for each evaluated model - Silverbox dataset. . . . .	145
Table 16 – Important features of the evaluated datasets for the computational experiments.	160

Table 17 – RMSE values for the test samples and number of SVs for each evaluated model - Silverbox dataset. . . . .	165
Table 18 – AIC and BIC information criteria for all the evaluated models. . . . .	167
Table 19 – Average norm of the parameter vector $\alpha$ for the LSSVR model and $\tilde{\alpha}_t$ for the KRLS and OS-LSSVR models. . . . .	168
Table 20 – Summary of the main characteristics of each proposed model. . . . .	174

## LIST OF ALGORITHMS

Algorithm 1	- Pseudo-code for the LSSVR model. . . . .	51
Algorithm 2	- Pseudo-code for the FS-LSSVR model. . . . .	56
Algorithm 3	- Pseudo-code for the IRLS algorithm. . . . .	62
Algorithm 4	- Pseudo-code for the W-LSSVR model. . . . .	66
Algorithm 5	- Pseudo-code for the IR-LSSVR model. . . . .	68
Algorithm 6	- Pseudo-code for the LMS algorithm. . . . .	75
Algorithm 7	- Pseudo-code for the RLS algorithm. . . . .	78
Algorithm 8	- Pseudo-code for the KLMS algorithm. . . . .	81
Algorithm 9	- Pseudo-code for the KRLS model. . . . .	86
Algorithm 10	- Pseudo-code for the RLM-SVR model. . . . .	100
Algorithm 11	- Pseudo-code for the RFS-LSSVR model. . . . .	113
Algorithm 12	- Pseudo-code for the R <sup>2</sup> FS-LSSVR model. . . . .	114
Algorithm 13	- Pseudo-code for the ROB-KRLS model. . . . .	138
Algorithm 14	- Pseudo-code for the OS-LSSVR model. . . . .	158



## LIST OF ABBREVIATIONS AND ACRONYMS

AIC	Akaike's Information Criterion
ALD	Approximate Linear Dependency
APA	Affine Projection Algorithms
BIC	Bayesian Information Criterion
FS-LSSVR	Fixed Size Least Squares Support Vector Regression
i.i.d.	Independent and Identically Distributed
IQR	Interquatile Range
IRLS	Iteratively Reweighted Least Squares
IR-LSSVR	Iteratively Reweighted Least Squares Support Vector Regression
KAF	Kernel Adaptive Filtering
KAPA	Kernel Affine Projection Algorithms
KKT	Karush-Kuhn-Tucker
KLMS	Kernel Least Mean Squares
KMC	Kernel Maximum Correntropy
KRLS	Kernel Recursive Least Squares
KRMC	Kernel Recursive Maximum Correntropy
LMS	Least Mean Squares
LS	Least Squares
LSSVM	Least Squares Support Vector Machine
LSSVR	Least Squares Support Vector Regression
MIMO	Multiple-Input Multiple-Output
MSE	Mean Squared Error
NARX	Nolinear Auto-Regressive with eXogenous input
NC	Novelty Criterion
OLS	Ordinary Least Squares
OSA	One-Step Ahead
OS-LSSVR	Online Sparse Least Squares Support Vector Regression
PV	Prototype Vector
QP	Quadratic Programming
RFS-LSSVR	Robust Fixed Size LSSVR
R <sup>2</sup> FS-LSSVR	Robust Reweighted Fixed Size LSSVR

RKHS	Reproducing Kernel Hilbert Spaces
RLS	Recursive Least Squares
RLM	Recursive Least M-estimate
RLM-SVR	Recursive Least M-estimate Support Vector Regression
RMSE	Root Mean Square Error
ROB-KRLS	ROBust Kernel Recursive Least Squares
SISO	Single-Input Single-Output
SSE	Sum of Squared Errors
SV	Support Vector
SVM	Support Vector Machine
SVR	Support Vector Regression
TSP	Time Series Prediction
W-LSSVR	Weighted Least Squares Support Vector Regression

## LIST OF SYMBOLS

$\mathcal{D}$	Set of input and output training/estimation pairs
$\mathcal{D}_t$	Set of input and output training/estimation pairs built up the instant $t$
$\mathcal{D}^{pv}$	Set of prototype vectors (FS-LSSVR)
$\mathcal{D}_t^{sv}$	Dictionary of support vectors built up the instant $t$ (KRLS)
$m_t$	Cardinality of the dictionary
$\mathbb{R}$	Set of the real numbers
$N$	Number of data samples used for training/estimation
$N'$	Number of data samples used for test/validation
$\mathbf{x}_n$	$n$ -th input vector
$\mathbf{x}_t$	Input vector built up the instant $t$
$u_n, u_t$	$n$ -th or $t$ -th control input (system identification task)
$n_t$	$t$ -th random noise component
$d$	Dimensionality of the input space
$d_h$	Dimensionality of the feature space
$y_n, y_t$	$n$ -th or $t$ -th real output
$\mathbf{y}_t$	Vector of outputs built up the instant $t$
$\hat{y}_n, \hat{y}_t$	$n$ -th or $t$ -th predicted output
$L_u$	Memory order of the system input
$L_y$	Memory order of the system output
$\hat{L}_u$	Estimate for the memory order of the system input
$\hat{L}_y$	Estimate for the memory order of the system output
$f(\cdot)$	Linear/nonlinear regression function
$\hat{f}(\cdot)$	Approximate solution for the function $f$
$\phi(\cdot)$	Nonlinear map into the feature space
$\hat{\phi}(\cdot)$	Approximation of the feature mapping $\phi$
$\hat{\phi}_i(\cdot)$	$i$ -th component of the approximation map $\hat{\phi}$

$\hat{\Phi}$	Approximated feature matrix (FS-LSSVR)
$\Phi_t$	Matrix of the projected vectors built up the instant $t$ (KRLS)
$\mathbf{w}$	Vector of unknown parameters
$\hat{\mathbf{w}}$	Approximate solution for $\mathbf{w}$
$\hat{\mathbf{w}}^r$	Robust solution for $\hat{\mathbf{w}}$
$b$	Bias of the regression model
$b^r$	Robust solution for $b$
$\hat{b}$	Approximate solution for $b$
$\hat{b}^r$	Robust solution for $\hat{b}$
$e_n, e_t$	$n$ -th or $t$ -th prediction error
$v_n, v_t$	Weight associated to the $n$ -th or $t$ -th prediction error (M-estimators)
$\mathbf{V}$	Diagonal matrix of the weights $v_n$ 's
$\mathbf{V}_t$	Diagonal matrix of the weights $v_t$ 's built up the instant $t$
$J_p$	Functional of an optimization problem in the primal space
$J_d$	Functional of an optimization problem in the dual space
$\mathcal{L}$	Lagrangian
$\gamma$	Regularization parameter
$\alpha_n$	$n$ -th Lagrange multiplier or $n$ -th coefficient of the solution vector
$\alpha_n^r$	$n$ -th robust version for $\alpha_n$
$\boldsymbol{\alpha}_t$	Vector of coefficients built up the instant $t$
$\tilde{\boldsymbol{\alpha}}_t$	Reduced vector of $m_t$ coefficients built up the instant $t$
$\tilde{\boldsymbol{\alpha}}_t^r$	Robust solution for the vector $\tilde{\boldsymbol{\alpha}}_t$
$k(\cdot, \cdot)$	Kernel function
$\sigma$	Bandwidth of Gaussian kernel function
$\mathbf{K}$	Kernel (or Gram) matrix
$\mathbf{K}_t$	Kernel matrix built up the instant $t$
$\tilde{\mathbf{K}}_t$	Kernel matrix built with the dictionary samples up the instant $t$
$\bar{\mathbf{K}}$	Reduced kernel matrix (FS-LSSVR)

$M$	Number of prototype vector (FS-LSSVR)
$H_R$	Quadratic Rényi's entropy
$\rho(\cdot)$	Objective function or loss function for M-Estimators
$\psi(\cdot)$	Score function for M-estimators
$\hat{\sigma}$	Error (outlier) threshold or tuning constant
$E\{\cdot\}$	Statistical expectation operator
$(\cdot)^\dagger$	Pseudo-inverse matrix
$\ \cdot\ $	Euclidean norm
$\mathbf{p}_t$	$t$ -th cross-correlation vector between the output and input signals
$\mathbf{R}_t$	$t$ -th input signal correlation matrix
$\mu$	Step-size parameter or learning step
$\lambda$	Forgetting factor
$\nabla$	Gradient vector
$\nu$	sparsity level parameter
$\delta_t$	$t$ -th solution of the ALD criterion
$\mathbf{a}_t$	$t$ -th vector of coefficients of the ALD criterion
$\mathbf{A}_t$	Matrix formed by the $\mathbf{a}_t$ 's vectors

## CONTENTS

<b>1</b>	<b>INTRODUCTION</b> . . . . .	<b>25</b>
<b>1.1</b>	<b>General and Specific Objectives</b> . . . . .	<b>29</b>
<b>1.2</b>	<b>The System Identification Problem</b> . . . . .	<b>30</b>
<i>1.2.1</i>	<i>Classes of Models</i> . . . . .	<i>31</i>
<i>1.2.2</i>	<i>Model Validation</i> . . . . .	<i>32</i>
<i>1.2.3</i>	<i>Additional Remarks</i> . . . . .	<i>35</i>
<b>1.3</b>	<b>Scientific Production</b> . . . . .	<b>36</b>
<b>1.4</b>	<b>Thesis Structure</b> . . . . .	<b>38</b>
<b>2</b>	<b>SUPPORT VECTOR REGRESSION MODELS</b> . . . . .	<b>40</b>
<b>2.1</b>	<b>Introduction</b> . . . . .	<b>40</b>
<b>2.2</b>	<b>Support Vector Regression - A Brief Discussion</b> . . . . .	<b>42</b>
<i>2.2.1</i>	<i>SVR for Linear Regression</i> . . . . .	<i>42</i>
<i>2.2.2</i>	<i>SVR for Nonlinear Regression</i> . . . . .	<i>45</i>
<b>2.3</b>	<b>Least Squares Support Vector Regression</b> . . . . .	<b>48</b>
<i>2.3.1</i>	<i>Estimation in the Dual Space</i> . . . . .	<i>48</i>
<b>2.4</b>	<b>Fixed Size Least Squares Support Vector Regression</b> . . . . .	<b>51</b>
<i>2.4.1</i>	<i>Estimation in Primal Weight Space</i> . . . . .	<i>51</i>
<i>2.4.1.1</i>	<i>Approximation to the Feature Map</i> . . . . .	<i>52</i>
<i>2.4.1.2</i>	<i>Imposing Sparseness and Subset Selection</i> . . . . .	<i>53</i>
<i>2.4.1.3</i>	<i>The FS-LSSVR Solution</i> . . . . .	<i>54</i>
<b>2.5</b>	<b>Concluding Remarks</b> . . . . .	<b>56</b>
<b>3</b>	<b>ROBUSTNESS IN LSSVR DERIVED MODELS</b> . . . . .	<b>58</b>
<b>3.1</b>	<b>Introduction</b> . . . . .	<b>58</b>
<b>3.2</b>	<b>Fundamentals of the <math>M</math>-Estimators</b> . . . . .	<b>60</b>
<i>3.2.1</i>	<i>Objective and Weigthing Functions</i> . . . . .	<i>61</i>
<b>3.3</b>	<b>Robustness in LSSVR Models</b> . . . . .	<b>65</b>
<i>3.3.1</i>	<i>The IR-LSSVR Model</i> . . . . .	<i>66</i>
<b>3.4</b>	<b>Example of the Influence of Outliers</b> . . . . .	<b>67</b>
<b>3.5</b>	<b>Some Other Robust LSSVR Approaches</b> . . . . .	<b>68</b>
<b>3.6</b>	<b>Concluding Remarks</b> . . . . .	<b>70</b>
<b>4</b>	<b>ONLINE KERNEL-BASED MODELS</b> . . . . .	<b>71</b>

<b>4.1</b>	<b>Introduction</b> . . . . .	71
<b>4.2</b>	<b>Linear Adaptive Filtering</b> . . . . .	72
<b>4.2.1</b>	<i>Wiener Filter</i> . . . . .	73
<b>4.2.2</b>	<i>Least Mean Square Algorithm</i> . . . . .	74
<b>4.2.3</b>	<i>Recursive Least Squares Algorithm</i> . . . . .	75
<b>4.3</b>	<b>Kernel Adaptive Filtering</b> . . . . .	78
<b>4.3.1</b>	<i>Kernel LMS Algorithm</i> . . . . .	79
<b>4.3.2</b>	<i>Kernel RLS Algorithm</i> . . . . .	80
<b>4.3.2.1</b>	<i>Case 1 - Unchanged Dictionary</i> . . . . .	83
<b>4.3.2.2</b>	<i>Case 2 - Updating the Dictionary</i> . . . . .	84
<b>4.4</b>	<b>Other Sparsification Criteria</b> . . . . .	86
<b>4.5</b>	<b>Other Online Kernel Models</b> . . . . .	88
<b>4.6</b>	<b>Concluding Remarks</b> . . . . .	92
<b>5</b>	<b>NOVEL ROBUST KERNEL-BASED MODELS IN BATCH MODE</b> . .	94
<b>5.1</b>	<b>Introduction</b> . . . . .	94
<b>5.2</b>	<b>Proposed Robust Approach in Dual Space</b> . . . . .	96
<b>5.2.1</b>	<i>The RLM-SVR Model</i> . . . . .	97
<b>5.2.2</b>	<i>Computational Experiments</i> . . . . .	99
<b>5.2.2.1</b>	<i>Evaluated Datasets</i> . . . . .	101
<b>5.2.2.2</b>	<i>Results and Discussion</i> . . . . .	103
<b>5.2.2.2.1</b>	<i>Experiments with Synthetic Datasets</i> . . . . .	105
<b>5.2.2.2.2</b>	<i>Experiments with Real-World Datasets</i> . . . . .	105
<b>5.3</b>	<b>Proposed Robust Approaches in Primal Space</b> . . . . .	107
<b>5.3.1</b>	<i>The RFS-LSSVR Model</i> . . . . .	110
<b>5.3.2</b>	<i>The R<sup>2</sup>FS-LSSVR Model</i> . . . . .	112
<b>5.3.3</b>	<i>Extension to Nonlinear MIMO Systems</i> . . . . .	113
<b>5.3.3.1</b>	<i>The Regression Vector - MIMO Case</i> . . . . .	116
<b>5.3.4</b>	<i>Final Remarks on the Proposed Models</i> . . . . .	117
<b>5.3.5</b>	<i>Computational Experiments</i> . . . . .	117
<b>5.3.5.1</b>	<i>Evaluated Datasets</i> . . . . .	118
<b>5.3.5.2</b>	<i>Results and Discussion</i> . . . . .	120
<b>5.3.5.2.1</b>	<i>Experiments with Synthetic Datasets</i> . . . . .	120

5.3.5.2.2	<i>Experiments with Real-World Datasets</i>	121
5.3.5.2.3	<i>Experiments with a Large-Scale Dataset</i>	124
5.3.5.2.4	<i>Experiments with a MIMO Dataset</i>	124
<b>5.4</b>	<b>Concluding Remarks</b>	126
<b>6</b>	<b>NOVEL ONLINE KERNEL-BASED MODELS</b>	130
<b>6.1</b>	<b>Introduction</b>	130
<b>6.2</b>	<b>Proposed Robust KAF Model</b>	132
<b>6.2.1</b>	<b><i>The ROB-KRLS Model</i></b>	133
6.2.1.1	<i>Case 1 - Unchanged Dictionary</i>	135
6.2.1.2	<i>Case 2 - Updating the Dictionary</i>	136
6.2.1.3	<i>On the Choice of the Weight Function</i>	137
<b>6.2.2</b>	<b><i>Computational Experiments</i></b>	139
6.2.2.1	<i>Results and Discussion</i>	140
6.2.2.1.1	<i>Experiments with Synthetic Datasets</i>	140
6.2.2.1.2	<i>Experiments with Real-World Datasets</i>	143
6.2.2.2	<i>Correlation Analysis of the Residuals</i>	145
6.2.2.3	<i>Convergence Analysis</i>	150
<b>6.3</b>	<b>Proposed Online LSSVR Model</b>	152
<b>6.3.1</b>	<b><i>The OS-LSSVR Model</i></b>	153
6.3.1.1	<i>Case 1 - Unchanged Dictionary</i>	155
6.3.1.2	<i>Case 2 - Updating the Dictionary</i>	156
<b>6.3.2</b>	<b><i>Computational Experiments</i></b>	159
6.3.2.1	<i>Results and Discussion</i>	160
6.3.2.1.1	<i>Function Approximation</i>	160
6.3.2.1.2	<i>Long-Term Time Series Prediction</i>	162
6.3.2.1.3	<i>Free-Simulation System Identification</i>	164
6.3.2.1.4	<i>Performance on a Large-Scale Dataset</i>	165
6.3.2.2	<i>Convergence Analysis</i>	166
6.3.2.3	<i>Model Efficiency</i>	167
6.3.2.4	<i>A Note on the Norm of the Solution Vector</i>	169
<b>6.4</b>	<b>Concluding Remarks</b>	169
<b>7</b>	<b>CONCLUSIONS AND FUTURE DIRECTIONS</b>	171



<b>7.1</b>	<b>Conclusions and Discussion</b> . . . . .	171
<b>7.2</b>	<b>Future Directions</b> . . . . .	174
	<b>Bibliography</b> . . . . .	176
	<b>APPENDIX</b> . . . . .	189
	<b>APPENDIX A – The ROB-KRLS Model</b> . . . . .	189
	<b>APPENDIX B – The OS-LSSVR Model</b> . . . . .	196
	<b>APPENDIX C – Bound on Generalization Error - OS-LSSVR Model</b> .	203

## 1 INTRODUCTION

“Work gives you meaning and purpose and life is empty without it.”

(Stephen Hawking)

Nonlinear phenomena are commonly encountered in most practical problems in areas such as Engineering, Physics, Chemistry, Biology, Economy, etc. This nonlinear nature can be revealed by several sources of non-linearity, including harmonic distortion, chaos, limit cycle, bifurcation and hysteresis (ZHANG; BILLINGS, 2015). Therefore, the data analysis in these cases often requires nonlinear regression methods to detect the kind of dependencies that allow for the successful prediction of properties of interest (HOFMANN *et al.*, 2008).

In this scenario, the kernel-based methods have been proven to be suitable tools to deal with nonlinear regression problems. Inspired by the statistical learning theory (VAPNIK, 1995; VAPNIK, 1998), the Support Vector Regression (SVR) (SMOLA; VAPNIK, 1997) and Least Squares Support Vector Regression (LSSVR) (SUYKENS *et al.*, 2002b; SAUNDERS *et al.*, 1998) non-parametric models, among others, have achieved acceptable performance in several regression tasks, including function approximation, time series prediction and system identification.

The basic idea behind the kernel methods is that they build a linear model in the so called feature space, where the original pattern inputs are transformed by means of a high-dimensional or even infinite-dimensional nonlinear mapping  $\phi$ . Then, the problem is converted from the primal space to the dual space by applying Mercer’s theorem and a positive kernel function (SCHÖLKOPF; SMOLA, 2002), without the need for explicitly computing the mapping  $\phi$ . This appealing property of the kernel-based methods is referred to as the *kernel trick*.

It should be noted that the term *kernel* may assume different meanings, depending on the nonlinear modeling paradigm. In Volterra series models (ZHANG; BILLINGS, 2017), for example, the kernel can be regarded as a higher-order impulse response of the dynamic system. In reproducing kernel Hilbert spaces (RKHS)-based techniques, such as the ones followed in the current thesis, the kernel *per se* does not have a signal processing connotation. It is related to Mercer-type kernels for nonlinear transformation on Machine Learning algorithms, popularized by Vapnik (1998) via the Support Vector Machine (SVM) classifier. In such kernels, there are no memory (i.e. dynamical) issues involved. It is a purely static transformation.

Notwithstanding the differences between the SVR and LSSVR techniques with

respect to the sparsity of the solution, since in the LSSVR model all the training examples are used as support vectors in order to make new predictions, the learning process in both kernel-based models is carried out in *batch mode*. This means that all training samples are stored in computer memory for building the kernel (Gram) matrix used during the parameter optimization phase. If computer memory cannot hold all of the training data, one can resort to disk storage, but it tends to slow down the learning speed. In addition, if a new data sample is to be added to the model, the kernel matrix computation and the respective parameter optimization process have to be carried out again.

From the 2000s decade, much attention has been focused on developing LSSVR derived methods in order to obtain sparse solutions and, consequently, achieve greater scalability in the inherent batch learning. This problem was addressed, for example, by using pruning (SUYKENS *et al.*, 2002a) and fixed-size approaches (SUYKENS *et al.*, 2002b). The latter, called the *Fixed-Size LSSVR* (FS-LSSVR) model, provides an alternative solution for the LSSVR formulation in the primal space, which is sparse since it relies on a selected subsample of pattern inputs. The FS-LSSVR model has been widely used to solve static (DE-BRABANTER *et al.*, 2010; MALL; SUYKENS, 2015) and dynamical (DE-BRABANTER *et al.*, 2009a; CASTRO *et al.*, 2014; LE *et al.*, 2011) regression problems.

However, in several specific application domains, such as time series prediction, control systems, system identification and channel equalization, online (i.e. sample-by-sample) learning strategies are required, where the predictor model is modified following the arrival of each new sample. It should be noted that those are classical signal processing applications, but 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; SLAVAKIS *et al.*, 2014) and also data nonstationarity (DITZLER *et al.*, 2015).

Strongly motivated by the success of the SVR/LSSVR models in nonlinear regression, the application of the kernelization philosophy to linear adaptive filters has produced powerful nonlinear extensions of well-known signal processing algorithms (ROJO-ÁLVAREZ *et al.*, 2014), including the Least Mean Squares (LMS) (LIU *et al.*, 2008; CHAUDHARY; RAJA, 2015), the Normalized LMS (SAIDE *et al.*, 2015), and the Recursive Least Squares (RLS) (ENGEL *et al.*, 2004; LIU *et al.*, 2009b; ZHU *et al.*, 2012; SAIDE *et al.*, 2013; FAN; SONG, 2013) algorithms. These contributions have helped to establish another emerging field, that of *Kernel Adaptive Filtering* (KAF) for nonlinear signal processing (LIU *et al.*, 2009a; SAIDE *et al.*, 2015).

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. In this work, the term “online” means that the learning process can be updated one-by-one, without re-training using all the learning data 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).

Of particular interest to this thesis is the work of Engel *et al.* (2004), who introduced the *Kernel Recursive Least-Squares* (KRLS) algorithm as a kernelized version of the famed RLS algorithm (HAYKIN, 2013; LI *et al.*, 2014), based on an online sparsification method called *approximate linear dependence* (ALD) (ENGEL *et al.*, 2002), that sequentially admits into a dictionary of kernel representation only those samples that cannot be approximately represented by a linear combination of samples that were previously admitted as *support vectors* (SVs). In summary, the ALD criterion, as well as other sparsification rules, aims at identifying kernel functions whose removal is expected to have negligible effect on the quality of the model solution (RICHARD *et al.*, 2009).

Regarding the discussion above, a common characteristic of the batch LSSVR and FS-LSSVR models, and of the online KRLS algorithm, is that they are built upon *sum of squared errors* (SSE) cost functions. Then, their optimality is guaranteed only for normally distributed errors<sup>1</sup> and, consequently, the performance of its solution can drastically degrade or even completely break down when the evaluated data is corrupted with outlier-like<sup>2</sup> samples, e.g. impulsive disturbances (SUYKENS *et al.*, 2002b; SUYKENS *et al.*, 2002a; PAPAGEORGIU *et al.*, 2015; ZOUBIR *et al.*, 2012).

In this scenario, M-estimation is a broad robust statistics framework (HUBER; RONCHETTI, 2009; ROUSSEEUW; LEROY, 1987) successfully used for parameter estimation in signal processing problems (ZHANG *et al.*, 2014; ZOUBIR *et al.*, 2012), when the Gaussianity assumption for the prediction errors (i.e. residuals) does not hold. According to this framework, robustness to outliers is achieved by minimizing another function than the sum of the squared errors.

Although M-estimators have been widely used to robustify the standard dual LSSVR

---

<sup>1</sup>The least squares estimate is the best linear unbiased estimate of the true parameters of a model if the noise is white. This means that the covariance matrix of the estimated parameters is the smallest of all possible linear unbiased estimators. If the white noise is also Gaussian, then there does not even exist a better nonlinear unbiased estimator (NELLES, 2001).

<sup>2</sup>Although there is no universal definition of outlier (HODGE; AUSTIN, 2004), in this thesis we consider an outlier as an observed output  $y$  that differs markedly from the other observations of the sample which it belongs.

model (SUYKENS *et al.*, 2002a; DE-BRABANTER *et al.*, 2009b; DEBRUYNE ANDREAS CHRISTMANN; SUYKENS, 2010; DE-BRABANTER *et al.*, 2012; DEBRUYNE *et al.*, 2008), the adopted robust strategies usually modify the original LSSVR loss function applying weighted procedures. In addition, the LSSVR derived models have been little used for robust system identification tasks, and generally only in one-step ahead prediction scenarios (FALCK *et al.*, 2009; LIU; CHEN, 2013a; LIU; CHEN, 2013b). Regarding the primal LSSVR formulation, the development of robust strategies for the FS-LSSVR model is still an entirely open issue.

As before mentioned, the KRLS model inherits from the RLS algorithm its high sensitivity to outliers. It is then desirable for the KRLS algorithm to be endowed with some inherent mechanism to handle outliers smoothly. One may argue that the simplest approach would be to employ some of the robust strategies used for the linear RLS-based models in the KRLS algorithm. However, such direct transfer is not always straightforward because the resulting KRLS algorithm must satisfy the following three requirements: (i) it must be capable of handling nonlinearities; (ii) it must be suitable for online learning; and (iii) it must provide a sparse solution to the parameter vector. Preferably, the sparsification and online learning issues should be unified in the same framework.

Due to these requirements, strategies for building outlier-robust kernel adaptive filtering algorithms are not widely available yet, despite their importance to real-world applications. An important attempt in this direction has been recently introduced by Wu *et al.* (2015), who developed a robust kernel adaptive algorithm based on the maximum correntropy criterion (MCC) (LIU; WANG, 2007). MCC is built upon information theoretic concepts and aims at substituting conventional 2nd-order statistical figures of merit, such as *Mean Squared Error* (MSE), in order to capture higher-order statistics and offer significant performance improvement, especially when data contain large outliers or are disturbed by impulsive noises.

Despite the rapidly growing number of successful applications of the aforementioned kernel-based online learning models, their use for nonlinear online system identification has not been fully explored yet. This is particularly true for scenarios contaminated with outliers and/or non-Gaussian errors, where just a few works can be found (WU *et al.*, 2015; SANTOS *et al.*, 2015; LIU; CHEN, 2013b; QUAN *et al.*, 2010; TANG *et al.*, 2006; SUYKENS *et al.*, 2002a). However, most of these previous works (SANTOS *et al.*, 2015; LIU; CHEN, 2013b; QUAN *et al.*, 2010; SUYKENS *et al.*, 2002a) cannot be applied to online nonlinear system identification because they do not simultaneously offer sparse solutions under a recursive parameter estimation

scheme. In other words, they use all the available training input-output pairs for model building purposes. Sparse solutions, when offered, are achieved via post-training pruning procedures (SUYKENS *et al.*, 2002a).

Finally, an additional difficulty in dealing with (online) system identification problems in the presence of outliers is due to the own dynamic nature of the adopted model. In this case, even considering a possible contamination of outliers only in the output variables  $y$ , their previously values are used to build the regressor vectors that will be used as input samples for the next time iterations, and so on. Then, the effect of outliers may be felt in both input and outputs of the system. In this scenario, one can conclude that the system identification task becomes even more difficult.

## 1.1 General and Specific Objectives

In a broader sense, the overall objective in this thesis is to propose novel robust and/or online kernel-based approaches for dynamic system identification tasks in the presence of outliers. Furthermore, one expects that the proposed models, which are derived from the M-estimators theory and kernel adaptive filtering algorithms, achieve suitable performances in terms of generalization capacity when compared to classical methods of the state-of-the-art. In order to accomplish that, we may specifically follow the next steps:

- Propose robust versions (based on the M-estimators) of the LSSVR model in batch learning mode, following its standard dual formulation and the primal space formulation of the FS-LSSVR model;
- Present a novel robust kernel adaptive filtering model, based on the M-estimators and derived from the original KRLS model, for online signal processing in the presence of outliers;
- Propose a sparse and online version of the standard LSSVR model, based on the original KRLS learning procedure;
- Evaluate all the proposed approaches in robust system identification tasks, in scenarios with long-term prediction, using synthetic and real-world datasets, from SISO and MIMO systems, and a large-scale dataset.

## 1.2 The System Identification Problem

The framework of mathematical modeling involves strategies in developing and implementing mathematical models of real systems. According to Billings (2013), a mathematical model of a system can be used to emulate this system, predict its outputs for given inputs and investigate different design scenarios. However, these objectives can only be fully achieved if the model of the system is perfectly known. This case corresponds to the *white-box modeling* (SJÖBERG *et al.*, 1995), where it is possible to construct the original model entirely from prior knowledge and physical insight.

As an alternative to white-box modeling, system identification emerges as an important area that studies different modeling techniques, where no physical insight is available or used. These techniques are referred to as *black-box modeling*, wherein the challenge of describing the dynamic behavior of the systems consists of searching for a mathematical description from empirical data or measurements in the system (SJÖBERG *et al.*, 1995). Therefore, the black-box modeling framework corresponds to the focus of this thesis.

System identification is a regression-like problem where the input and output observations come from time series data. In other words, information about the dynamics (i.e. temporal behavior) of the system of interest must be learned from time series data. When nonlinear dynamics has to be captured from data, many modeling paradigms are available in the literature, including neural networks (NARENDRA; PARTHASARATHY, 1990), kernel-based models (ROJO-ÁLVAREZ *et al.*, 2004; CAI *et al.*, 2013; GRETTON *et al.*, 2001), fuzzy systems (TAKAGI; SUGENO, 1985), neuro-fuzzy systems (JANG, 1993), Volterra series representation (ZHANG; BILLINGS, 2017), Hammerstein-Wiener models (WILLS *et al.*, 2013), and Chebyshev polynomials approximation (ZHANG *et al.*, 2013), in addition to others.

Mathematically, the representation of a typical nonlinear system identification problem can be expressed, for the sake of simplicity and without loss of generality, by a *single-input single-output* (SISO) dynamic system from a finite set of  $N$  pairs of observed measurements  $\mathcal{D} = \{u_t, y_t\}_{t=1}^N$ , where  $u_t \in \mathbb{R}$  is the  $t$ -th input and  $y_t \in \mathbb{R}$  is the output at time  $t$ . Then, the input-output discrete time relationship can be given by

$$y_t = g(\mathbf{x}_t) + n_t = g_t + n_t, \quad (1.1)$$

where  $g_t \in \mathbb{R}$  is the true (noiseless) output of the system,  $g(\cdot)$  is the unknown, and usually nonlinear, target function and  $\{n_t\}_{t=1}^N$  corresponds to a noise sequence which is assumed to be

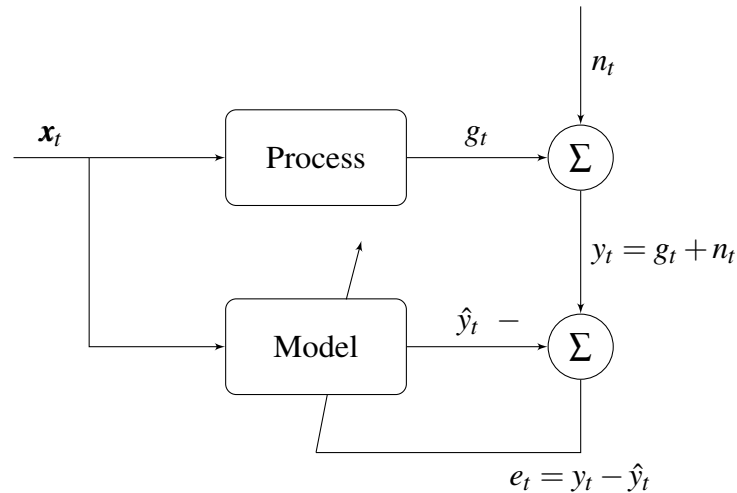


Figure 1 – Example of a system identification problem, where a model is adapted in order to represent the unknown process behavior.

independent and identically distributed (i.i.d.) with zero mean and finite variance.  $\mathbf{x}_t$  denotes the *regression vector* at sampling time  $t$  and its components are given as a function of the relevant variables of the system at a previous time.

An example of a system identification problem for a SISO system is illustrated in Fig. (1), where the model predictions should represent the dynamical behavior of the process as closely as possible. The model performance is typically evaluated in terms of a function of the error  $e_t$  between the noisy process output  $y_t$  and the model output  $\hat{y}_t$ . Then, this error is used to adjust the parameters of the model.

### 1.2.1 Classes of Models

In order to construct or select models in system identification scenarios, the first step is determining a class of models within which the search for the most suitable model is to be conducted. For this purpose, a number of different linear models have been proposed (LJUNG, 1999; BILLINGS, 2013; AGUIRRE, 2007), such as Auto-Regressive (AR), Auto-Regressive with eXogenous inputs (ARX), Moving Average (MA), Auto-Regressive Moving Average (ARMA), Auto-Regressive Moving Average with eXogenous inputs (ARMAX), Output Error (OE), Finite Impulse Response (FIR) and Box-Jenkins (BJ). One should note that the respective nonlinear versions follow a similar nomenclature of these linear models.

Among the nonlinear dynamic models, the *Nonlinear Auto-Regressive with eXogenous inputs* (NARX) covers a wide class of nonlinear systems and it is normally the standard approach pursued in most engineering applications (NELLES, 2001). Because of this and since



the major goal in this thesis is not to find the best model for a given application, but rather, evaluate the performance of parameter estimation techniques for kernel-based models, the NARX dynamic model will be used for all the applications in system identification throughout this work.

The NARX model can be defined by rewriting Eq. (1.1) as (BILLINGS, 2013; AGUIRRE, 2007; NELLES, 2001)

$$y_t = g(y_{t-1}, \dots, y_{t-L_y}, u_{t-\tau_d}, \dots, u_{t-L_u}) + n_t, \quad (1.2)$$

whose regression vector is given by

$$\mathbf{x}_t = [y_{t-1}, \dots, y_{t-L_y}, u_{t-\tau_d}, \dots, u_{t-L_u}]^\top, \quad (1.3)$$

where it is assumed that the random noise  $n_t$  in Eq. (1.2) is i.i.d. and is purely additive to the model. The  $L_u \geq 1$  and  $L_y \geq 1$  values are the maximum lags for the system input and output, respectively, denoting their memory orders. The target nonlinear function  $g(\cdot) : \mathbb{R}^{L_y+L_u} \rightarrow \mathbb{R}$  is assumed to be unknown and  $\tau_d$  is a time delay, also called dead time, typically set to  $\tau_d = 1$ .

In the next step, the observed data in the form of input-output time series should be used to build an approximating model  $f(\cdot)$  for the target function  $g(\cdot)$ , which is expressed by

$$\hat{y}_t = f(y_{t-1}, \dots, y_{t-\hat{L}_y}, u_{t-1}, \dots, u_{t-\hat{L}_u}), \quad (1.4)$$

where  $\hat{L}_u$  and  $\hat{L}_y$  are estimates of the real system memory. Clearly, the regressors  $\mathbf{x}_t$ , and consequently the model  $f(\cdot)$ , depend on  $\hat{L}_u$  and  $\hat{L}_y$ , but these dependencies have been omitted in order to simplify the notation, as was done by SOUZA-JÚNIOR (2014).

The techniques for developing nonlinear mappings  $f(\cdot)$  strategies, derived from kernel-based models, and their required procedures for parameter estimation are the main purpose of this thesis and, therefore, they will be discussed in detail throughout next chapters.

### 1.2.2 Model Validation

In order to assess model performance in system identification tasks, a commonly used practice involves splitting the available data samples into two parts: one for model estimation or training (or model building), which is referred to as the training dataset, and another part for model testing (or validation), which is referred to as the test dataset. Therefore, the model obtained from the training dataset must be evaluated over an independent test dataset, whose samples are unknown by the model so far. In other words, one wants to know if the obtained

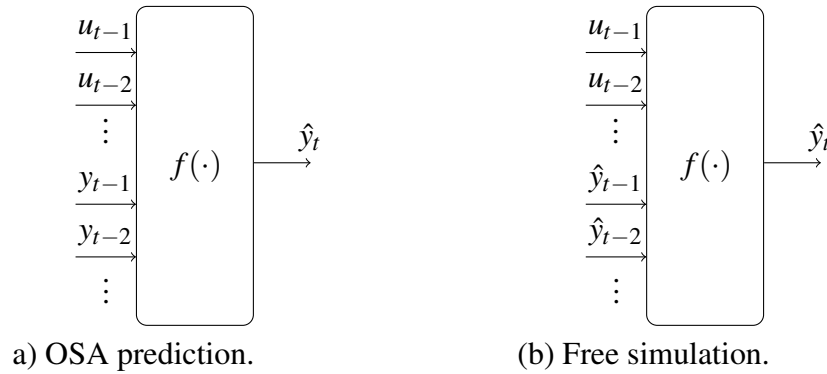


Figure 2 – Scenarios of predictions for model validation - adapted from Nelles (2001).

model serves to explain another observed dataset of the same system (AGUIRRE, 2007). This strategy is used to measure the generalization capacity of the model.

Since the model was properly obtained, it may be used to simulate the dynamical output of the identified system through iterative predictions. In this sense, there are two possible scenarios: *prediction* and *simulation*. Prediction means that, on the basis of previous inputs  $u_{t-i}$  (for  $i = 1, \dots, \hat{L}_u$ ) and outputs  $y_{t-j}$  (for  $j = 1, \dots, \hat{L}_y$ ), the model predicts one or several steps into the future, while simulation means that the model simulates future outputs only on the basis of previous inputs  $u_{t-i}$  (NELLES, 2001). Simply stated, prediction uses previous values of inputs and observed outputs to predict the current output, whereas simulation uses previous values of inputs and predictions (rather than observed outputs) to predict the current output.

The simplest concept in prediction scenarios is that of *One-Step Ahead (OSA)* prediction, in which the vector of regressors  $\mathbf{x}_{t-1}$  at time instant  $t - 1$  is first built, based on previous values of observed outputs and inputs, only to compute the predicted output  $\hat{y}_t$  for the next time instant  $t$ . However, it is noteworthy that this prediction  $\hat{y}_t$  is not used to obtain the next OSA prediction  $\hat{y}_{t+1}$ . The output given by Eq. (1.4) is considered as a OSA prediction, because  $\hat{y}_t$  is computed from measured outputs up to time instant  $t - 1$ . Fig. (2a) illustrates the task of one-step ahead prediction.

Although OSA predictions are customarily used in certain applications, such as short term stock market and weather forecasts (NELLES, 2001), where the current state of the system can be measured, they do not correspond to a proper representation of the true system (BILLINGS, 2013). In this sense, Aguirre (2007) states that OSA predictions are not a good test to validate a model, since “bad” models normally present “good” OSA predictions. Based on practical examples, Billings (2013) shows how misleading model validation based on OSA predictions can be, because even though the model is incorrect, the OSA predictions over

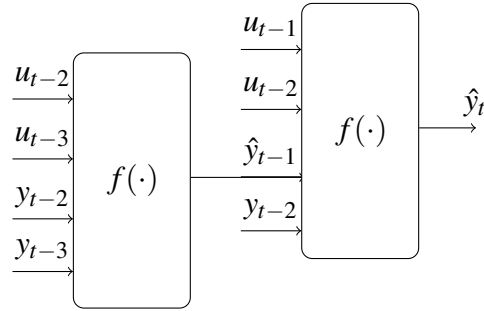


Figure 3 – Example of 2-steps ahead prediction task - adapted from Nelles (2001).

the test samples can be almost perfect. Therefore, it can be concluded that OSA predictions are not appropriate to explain the dynamic behavior of the identified system.

In contrast to OSA, in simulation scenarios the previous predictions are used to build the vector of regressors in order to continue making predictions. Thus, the vector of regressors is expressed as

$$\mathbf{x}_t = [\hat{y}_{t-1}, \dots, \hat{y}_{t-\hat{L}_y}, u_{t-1}, \dots, u_{t-\hat{L}_u}]^\top, \quad (1.5)$$

where the process inputs  $u_{t-i}$ , for  $i = 1, \dots, \hat{L}_u$ , are always available, as in the case of OSA. The output simulated according to Eq. (1.5) is called *free simulation* or *infinite-steps ahead prediction*, and it is illustrated in Fig. (2b).

Free simulation tasks are often required for applications in optimization, control, system identification and fault detection, where in some circumstances it becomes difficult (or even impossible) to measure the process outputs. In free simulation, as may be inferred, predictions rely on previous predictions and, consequently, the prediction errors tend to propagate through nonlinear transformations. This makes the modeling and identification phase harder, and requires additional care in order to ensure the stability of the model (NELLES, 2001). Even in this hostile scenario, free simulation is, unlike the OSA predictions, a suitable way to test if the model can explain the measured observations (AGUIRRE, 2007).

Regarding the discussion above, one can say that the prediction scenarios of OSA and free simulation are two extreme cases. Nevertheless, there is an intermediate case when more than one step is to be predicted into the future. This case is called the *multiple-steps ahead* or *k-steps ahead prediction*, which consists of using the obtained model as a free predictor only for  $k$  sampling intervals, restarting it immediately afterward with measured data (AGUIRRE, 2007). The number of steps predicted into the future  $k$  denotes the *prediction horizon*.

Based on Eq. (1.4) for OSA prediction, it can be used to predict a further step ahead by replacing  $t - 1$  for  $t$  and utilizing the result  $\hat{y}_{t-1}$  from the previous prediction step.

This procedure can be repeated  $k$  times in order to predict  $k$  steps ahead altogether. Then, for  $k = 2, 3, \dots$ , the  $k$ -step prediction is recursively computed by (ZHANG; LJUNG, 2007)

$$\hat{y}_t^{(k)} = f(\hat{y}_{t-1}^{(k-1)}, \dots, \hat{y}_{t-\hat{L}_y}^{(k-1)}, u_{t-1}, \dots, u_{t-\hat{L}_u}), \quad (1.6)$$

where  $\hat{y}_t^{(1)} \triangleq \hat{y}_t$ , as defined by Eq. (1.4). One should note that the sequence  $\hat{y}_t^{(1)}$  (for  $t = 1, \dots, N'$ ), where  $N'$  denotes the number of test samples used for model validation, must be first computed before  $\hat{y}_t^{(2)}$  be computed. Then,  $\hat{y}_t^{(3)}$  is computed, and so on. Therefore, the predicted output  $\hat{y}_t^{(k)}$ , computed with the recurrent expression in Eq. (1.6), can be viewed as a  $k$ -steps ahead prediction of  $y_t$ . This scenario is illustrated in Fig. (3) for  $k = 2$ .

Among the prediction scenarios, it is worth mentioning that  $k$ -steps ahead prediction task is equivalent to free simulation when the prediction horizon approaches infinity ( $k \rightarrow +\infty$ ). In this thesis, we do not apply OSA predictions for model validation. For this purpose, we only use  $k$ -steps ahead prediction and, in most applications, free simulation.

Finally, a common and useful choice in the quantification of predictions is the *Root Mean Squared Error* (RMSE), calculated between the predicted outputs and the real observations over the whole test dataset. Mathematically, the RMSE value is given by

$$RMSE = \sqrt{\frac{1}{N'} \sum_{t=1}^{N'} (y_t - \hat{y}_t)^2}. \quad (1.7)$$

In principle, the smaller the RMSE value, the better the reconstruction of the system dynamics.

Next, some additional remarks are presented in order to clarify and facilitate the understanding about the system identification problems treated throughout this thesis. For more details in the framework of system identification, the reference text-books by Billings (2013), Ljung (1999), Aguirre (2007), Nelles (2001) and Söderström and Stoica (1989) are recommended.

### 1.2.3 Additional Remarks

**Remark 1** - The first step, and probably the most important (mainly for black-box modeling), when dealing with system identification tasks is collecting data from the real system to be identified. This phase includes important decisions, such as on the choice of excitation signals of (control) input and output, and on the choice of the sampling time. The chosen estimation dataset should be suitable to represent the dynamics behavior of the system.

Otherwise, the obtained model will not be able to produce reliable predictions of this system.

Regarding the sampling time, a value that is too small (oversampling) causes successive data samples to be very similar, so that the data become redundant. If a sampling time that is too large (undersampling) is chosen, then successive data points tend to become independent from each other so that the dynamical information about the original system is lost (ZHANG *et al.*, 2013).

In this thesis, we abstract away from these particular issues because we use benchmarking datasets for the computational experiments. In this case, we assume that the available estimation datasets are sufficiently representative of the system of interest and, besides that, the sampling rate is suitably chosen and normalized, so that we can treat the systems of interest as intrinsically nonlinear discrete-time systems.

**Remark 2** - Several nomenclatures commonly used in system identification and control theories have synonyms somewhat different in the field of Machine Learning. Therefore, in order to avoid confusion with some of them, we provide a glossary of terms (as done in Sjöberg *et al.* (1995), SOUZA-JÚNIOR (2014)) that are used interchangeably with their equivalents in this thesis:

- estimate = train, learn, model building, structure selection;
- validate = test, generalize;
- estimation dataset = training dataset, model building dataset;
- validation dataset = test dataset, generalization dataset.

### 1.3 Scientific Production

Throughout the development of this work, a reasonable number of articles have been published by the author and respective co-authors. Among them, the published articles related to the thesis are:

1. José Daniel A. Santos and Guilherme A. Barreto, **Novel Sparse LSSVR Models in Primal Weight Space for Robust System Identification with Outliers**. *Journal of Process Control*, to appear (<http://doi.org/10.1016/j.jprocont.2017.04.001>), 2017.
2. José Daniel A. Santos and Guilherme A. Barreto, **A Regularized Estimation Framework for Online Sparse LSSVR Models**. *Neurocomputing*, vol. 238, 114-125, 2017.
3. José Daniel A. Santos and Guilherme A. Barreto, **An Outlier-Robust Kernel RLS Al-**

**gorithm for Nonlinear System Identification.** *Nonlinear Dynamics*, vol. 90, issue 3, 1707-1726, 2017.

4. José Daniel A. Santos and Guilherme A. Barreto, **A Novel Recursive Solution to LS-SVR for Robust Identification of Dynamical Systems.** *In Lecture Notes in Computer Science: Intelligent Data Engineering and Automated Learning - IDEAL 2015*, K. Jackowski, R. Burduk, K. Walkowiak, M. Wozniak and H. Yin Eds., vol. 9375, 191-198, 2015.
5. José Daniel A. Santos, César Lincoln C. Mattos and Guilherme A. Barreto, **Performance Evaluation of Least Squares SVR in Robust Dynamical System Identification.** *In Lecture Notes in Computer Science: International Work Conference on Artificial Neural Networks - IWANN 2015*, I. Rojas, G. Joya and A. Catala Eds., vol. 9095, 422-435, 2015.

Furthermore, the following published articles were produced during the thesis period as result of research collaborations:

1. César Lincoln C. Mattos, José Daniel A. Santos and Guilherme A. Barreto, **An Empirical Evaluation of Robust Gaussian Process Models for System Identification.** *In Lecture Notes in Computer Science: Intelligent Data Engineering and Automated Learning - IDEAL 2015*, K. Jackowski, R. Burduk, K. Walkowiak, M. Wozniak and H. Yin Eds., vol. 9375, 172-180, 2015.
2. José Daniel A. Santos and Guilherme A. Barreto, **Uma Nova Solução Recursiva para LS-SVR em Identificação Robusta de Sistemas Dinâmicos.** *In: XII Simpósio Brasileiro de Automação Inteligente (SBAI 2015)*, Natal (Brazil), 25-28 October 2015, p. 642-647.
3. César Lincoln C. Mattos, José Daniel A. Santos and Guilherme A. Barreto, **Uma Avaliação Empírica de Modelos de Processos Gaussianos para Identificação Robusta de Sistemas Dinâmicos.** *In: XII Simpósio Brasileiro de Automação Inteligente (SBAI 2015)*, Natal (Brazil), 25-28 October 2015, p. 712-717.
4. José Daniel A. Santos, César Lincoln C. Mattos and Guilherme A. Barreto, **A Novel Recursive Kernel-Based Algorithm for Robust Pattern Classification.** *In Lecture Notes in Computer Science: Intelligent Data Engineering and Automated Learning - IDEAL 2014*, E. Corchado, J. A. Lozano, H. Quintián and H. Yin Eds., vol. 8669, 150-157, 2014.
5. César Lincoln C. Mattos, José Daniel A. Santos and Guilherme A. Barreto, **Improved Adaline Networks for Robust Pattern Classification.** *In Lecture Notes in Computer Science: International Conference on Artificial Neural Networks - ICANN 2014*, S. Wermter, C. Weber, W. Duch, T. Honkela, P. Koprinkova-Hristova and S. Magg Eds., vol. 8681,

579-586, 2014.

6. Davi N. Coelho, Guilherme A. Barreto, Cláudio M. S. Medeiros and José Daniel A. Santos, **Performance Comparison of Classifiers in the Detection of Short Circuit Incipient Fault in a Three-Phase Induction Motor**. *In: IEEE Symposium on Computational Intelligence for Engineering Solutions (IEEE-CIES 2014), Orlando (EUA), 9-12 December 2014, vol. 01, p. 42-48.*
7. César Lincoln C. Mattos, José Daniel A. Santos and Guilherme A. Barreto, **Classificação de Padrões Robusta com Redes Adaline Modificadas**. *In: Encontro Nacional de Inteligência Artificial e Computacional (ENIAC 2014), São Carlos (Brazil), 19-23 October 2014.*
8. Edmilson Q. Santos Filho, José Daniel A. Santos and Guilherme A. Barreto, **Estudo Comparativo de Métodos de Extração de Características para Classificação da Qualidade de Peles de Caprinos com Opção de Rejeição**. *In: XI Simpósio Brasileiro de Automação Inteligente (SBAI 2013), Fortaleza (Brazil), 13-17 October 2013, p. 01-06.*

#### 1.4 Thesis Structure

The remainder of this thesis is organized as follows.

Chapter 2 presents a brief framework of the kernel-based methods, more specifically, the support vector theory applied to nonlinear regression problems. Besides containing the basic ideas of the SVR model, it discusses in more detail the theoretical development of the LSSVR model in its standard dual formulation and in the primal space, which is referred to as the FS-LSSVR model. Finally, a comparative framework, with the main features of each evaluated kernel-based model: SVR, LSSVR and FS-LSSVR, is presented.

Chapter 3 is dedicated to outlier robustness. It discusses the theoretical foundation of M-estimators, which are presented as an alternative to least squares methods for linear regression problems. Next, some strategies in splice M-estimators to standard LSSVR formulation, giving rise its robust versions W-LSSVR and IR-LSSVR models, are discussed. Moreover, a computational experiment of outliers influence using the above LSSVR derived models is presented. Finally, a brief state-of-the-art about some robust LSSVR approaches is described.

Chapter 4 briefly covers an overview of kernel adaptive filtering models, which are implementations of linear adaptive filters in feature space. Initially, a basic notion of linear adaptive filters, focusing on classical LMS and RLS algorithms, is presented. Then, their

respective kernelized versions KLMS and KRLS are discussed in the sequence. Afterwards, a description of the family of kernel affine projection algorithms (KAPA) and some other sparsification criteria, different from ALD of the KRLS model, are presented.

Chapter 5 introduces the first part of the proposed approaches of this thesis, namely the RLM-SVR, RFS-LSSVR and  $R^2$ FS-LSSVR models, which correspond to robust kernel-based methods derived from the LS-SVR and FS-LSSVR standard models and based on M-estimators. This chapter also reports the performance results of the novel proposals in several system identification tasks using synthetic, real-world, large-scale and MIMO datasets in the presence of outliers.

Chapter 6 presents the second part of the contributions of this thesis, which comprises two novel kernel adaptive filtering methods derived from the original KRLS model. The first proposal is called ROB-KRLS and corresponds to a robust version of the KRLS model based on M-estimators. The second one, called the OS-LSSVR model, is based on the standard LSSVR formulation and is solved according to the KRLS learning procedure. This chapters also encompasses the results of computational experiments with the proposed approaches in several regression/system identification tasks using synthetic, real-world and large-scale datasets.

Chapter 7 is dedicated to the conclusions of this work as well to the directions for future research on issues related to the thesis.

Appendix A provides a step-by-step mathematical formulation for the ROB-KRLS proposed approach.

Appendix B provides a step-by-step mathematical formulation for the OS-LSSVR proposed approach.

Appendix C presents a study of the generalization error bound applicable to the proposed OS-LSSVR model.



## 2 SUPPORT VECTOR REGRESSION MODELS

“Things don’t have to change the world to be important.”

(Steve Jobs)

This chapter covers the kernel-based models theory, with an emphasis on support vector machines, when applied in nonlinear regression problems. Initially, Section 2.1 makes some considerations on the topics that are treated throughout the chapter. Then, Section 2.2 briefly discusses the basic ideas of Support Vector Regression (SVR), which was the pioneer among the kernel-based methods. Next, the theoretical foundations of the Least Squares Support Vector Regression (LSSVR) (Section 2.3) and Fixed Size Least Squares Support Vector Regression (FS-LSSVR) (Section 2.4) models are developed in detail, since they directly provide the backgrounds for some contributions of this thesis. Finally, Section 2.5 presents the final remarks of the chapter.

### 2.1 Introduction

The Support Vector Machines (SVM) framework comprises a powerful class of learning algorithms, derived from the results of the statistical learning theory (VAPNIK, 1995). Its development was first presented as a nonlinear generalization of the *Generalized Portrait* algorithm (VAPNIK, 1963; VAPNIK; CHERVONENKIS, 1964) to solve pattern recognition problems (VAPNIK; CHERVONENKIS, 1974; BOSER *et al.*, 1992). Later, it was extended by Smola and Vapnik (1997) to the domain of regression problems, which will be referred to as *Support Vector Regression* (SVR) from now on.

The SVR model works with a constrained quadratic optimization problem, which minimizes a combination of the empirical risk (employed by the conventional neural networks, e.g.) and a regularization term responsible for the smoothness of the model solution. This solution is obtained by solving a convex *quadratic programming* (QP) problem in dual space and, thus, has the noteworthy advantage of being a global and sparse solution. By sparseness, one means that the final regressor can be written as a combination of a relatively small number of training patterns, called the *support vectors* (SVs). Therefore, the SVR intrinsic characteristics, resulting from its strong conceptual formulation, have made it achieve satisfactory performance in a wide variety of regression problems (BAO *et al.*, 2014; KARTHIK *et al.*, 2016; KAZEM *et al.*, 2013; CHEN; YU, 2014).

In spite of success in regression applications, the SVR model solutions have been

empirically shown to be not maximally sparse (BURGES; SCHÖLKOPF, 1997; DOWNS *et al.*, 2001). In addition, an SVR optimization problem involves twice as many variables as the number of training samples and, moreover, the methods to solve the QP problem do not have a trivial implementation. In general, these methods present a super-linear dependence on the computation time of the number of available patterns and require repeated access to the training samples, making them suitable only for batch learning (ENGEL *et al.*, 2002).

In this context, the *Least Squares Support Vector Regression* (LSSVR) model (SAUNDERS *et al.*, 1998; SUYKENS *et al.*, 2002b) was proposed as a simpler alternative method to the standard Vapnik's SVR model, since learning in LSSVR relies on an SSE cost function and a set of equality constraints, instead of the original QP problem in SVR. Consequently, the model solution in LSSVR problem is easier to obtain by solving a set of linear equations in its dual formulation. In addition, the LSSVR model inherits from the SVR technique the convexity of the optimization problem and the global solution. On the other hand, the sparsity common to the SVR model is lost in the LSSVR problem, since every training sample contributes to the model solution as an SV.

At present, the LSSVR model has been widely applied to regression tasks, such as time series forecasting (ORMÁNDI, 2008; GESTEL *et al.*, 2001; MELLIT *et al.*, 2013; CHEN; LEE, 2015), control (SUYKENS *et al.*, 2001; KHALIL; EL-BARDINI, 2011; ZHANG *et al.*, 2014) and dynamical system identification (FALCK *et al.*, 2012; LAURAIN *et al.*, 2015; CAI *et al.*, 2013).

A key property behind the SVR and LSSVR models is that they apply kernel functions, which only require the evaluation of inner products between pairs of pattern inputs. Replacing inner products with a Mercer kernel provides an efficient way to implicitly map the data into a high, even infinite, dimensional RKHS (SCHÖLKOPF; SMOLA, 2002) by means of a nonlinear feature map  $\phi$ . Then, the calculations are carried out without making direct reference to the mapping of the input vectors, a property referred to as the kernel trick.

Although the LSSVR model is mostly solved in the dual formulation through the kernel trick, its optimization problem can also be solved in the primal space by estimating the feature map  $\phi$ . This attempt was introduced by Suykens *et al.* (2002b), resulting in a parametric and sparse representation called the *Fixed-Size Least Squares Support Vector Regression* (FS-LSSVR) model. In this method, an explicit expression of  $\phi$  or an approximation to it is required, which can be found using the Nyström approximation (WILLIAMS; SEEGER, 2001).

In the FS-LSSVR model, it is possible to compute sparse and approximated model solution by using only a subset of selected support vectors, also called *prototype vectors* (PVs), from the entire training data. One distinct advantage of this approach is that it can easily handle nonlinear estimation problems that require large amounts of data. Successful applications of the FS-LSSVR model can be found in De-Brabanter *et al.* (2009a), Espinoza *et al.* (2006), De-Brabanter *et al.* (2010) and Castro *et al.* (2014).

Finally, one should consider that the LSSVR and FS-LSSVR models serve as basis for some contributions of this thesis. Therefore, they are treated in detail throughout this chapter. On the other hand, the SVR theory is presented as a brief discussion, because of its importance in the development of the area of support vector methods and also for a better understanding in developing the LSSVR and FS-LSSVR models.

## 2.2 Support Vector Regression - A Brief Discussion

This section presents a brief discussion on linear and nonlinear SVR model, in which the origins of the derivation of the LSSVR formulation will be highlighted. Further details on SVR theory can also be found in Smola and Schölkopf (2004), Basak *et al.* (2007), Schölkopf and Smola (2002), Bishop (2006), Cherkassky and Mulier (2007) and Vapnik *et al.* (1996).

### 2.2.1 SVR for Linear Regression

Consider an estimation/training dataset  $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$  formed by  $N$  pairs of inputs  $\mathbf{x}_n \in \mathbb{R}^d$  and corresponding outputs  $y_n \in \mathbb{R}$ , for  $n = 1, \dots, N$ . In a regression problem, the goal is to search for a function  $f(\cdot)$  that approximates the outputs  $y_n$  for all instances of the available data. Initially, for the linear case,  $f(\cdot)$  usually takes the form

$$f(\mathbf{x}_n) = \mathbf{w}^\top \mathbf{x}_n + b, \quad (2.1)$$

where  $\mathbf{w} \in \mathbb{R}^d$  is a parameter vector and  $b \in \mathbb{R}$  is a bias term. In order to seek flatness in Eq. (2.1), minimizing the Euclidean norm  $\|\cdot\|_2$ , or simply  $\|\cdot\|$ , of  $\mathbf{w}$  is required (SMOLA, 1998; SCHÖLKOPF; SMOLA, 2002). Then, the optimal parameters of the linear function  $f(\cdot)$  are searched by minimization of the following constrained functional:

$$\min_{\mathbf{w}, b} J_p(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2, \quad (2.2)$$

$$\text{subject to } \begin{cases} y_n - \mathbf{w}^\top \mathbf{x}_n - b \leq \varepsilon, & \text{for } n = 1, \dots, N, \\ \mathbf{w}^\top \mathbf{x}_n + b - y_n \leq \varepsilon, & \text{for } n = 1, \dots, N. \end{cases} \quad (2.3)$$

The convex optimization problem in Eq. (2.2) with the inequality constraints in Eq. (2.3) is feasible in cases where the function  $f(\cdot)$  exists and has, at most,  $\varepsilon$  deviation from all the training outputs  $y_n$  and, at the same time, is as smooth as possible. However, sometimes errors are allowed in introducing slack variables, which describe the penalty for the training patterns lying outside the  $\varepsilon$  boundary. Therefore, the formulation in Eqs. (2.2), (2.3) becomes

$$\min_{\mathbf{w}, b, \xi, \xi^*} J_p(\mathbf{w}, \xi, \xi^*) = \frac{1}{2} \|\mathbf{w}\|^2 + \gamma \sum_{n=1}^N (\xi_n + \xi_n^*), \quad (2.4)$$

$$\text{subject to } \begin{cases} y_n - \mathbf{w}^\top \mathbf{x}_n - b \leq \varepsilon + \xi_n, \\ \mathbf{w}^\top \mathbf{x}_n + b - y_n \leq \varepsilon + \xi_n^*, \\ \xi_n, \xi_n^* \geq 0, \end{cases} \quad (2.5)$$

for  $n = 1, \dots, N$ . The values of  $\xi_n$  and  $\xi_n^*$  are slack variables corresponding to the points for which  $f(\mathbf{x}_n) - y_n > \varepsilon$  (points above the curve) and  $y_n - f(\mathbf{x}_n) > \varepsilon$  (points below the curve), respectively. The constant  $\gamma > 0$  is a regularization parameter that establishes a trade-off between the smoothness of  $f(\cdot)$  and the amount up to which deviations larger than  $\varepsilon$  are tolerated. This corresponds to dealing with a so called  $\varepsilon$ -insensitive loss function, or Vapnik loss function (VAPNIK, 1995), described by

$$|\xi|_\varepsilon := \begin{cases} 0, & \text{if } |\xi| < \varepsilon \\ |\xi| - \varepsilon, & \text{otherwise,} \end{cases} \quad (2.6)$$

and illustrated on the right side of Fig. (4). One should note that only the points outside the  $\varepsilon$ -insensitive zone contribute to the cost function, as can be seen in the blue point close to the symbol  $\xi$ , on the left size of Fig. (4).

The parametric formulation in Eq. (2.4) with the inequality constraints in Eq. (2.5) is an optimization problem in the primal weight space, also referred to as a the *primal problem*, since it is defined in the original space  $\mathbb{R}^d$  of the input data  $\mathbf{x}_n$  and the parameter vector  $\mathbf{w}$ . The solution of this primal problem can be achieved by constructing a Lagrange function (FLETCHER, 2013)

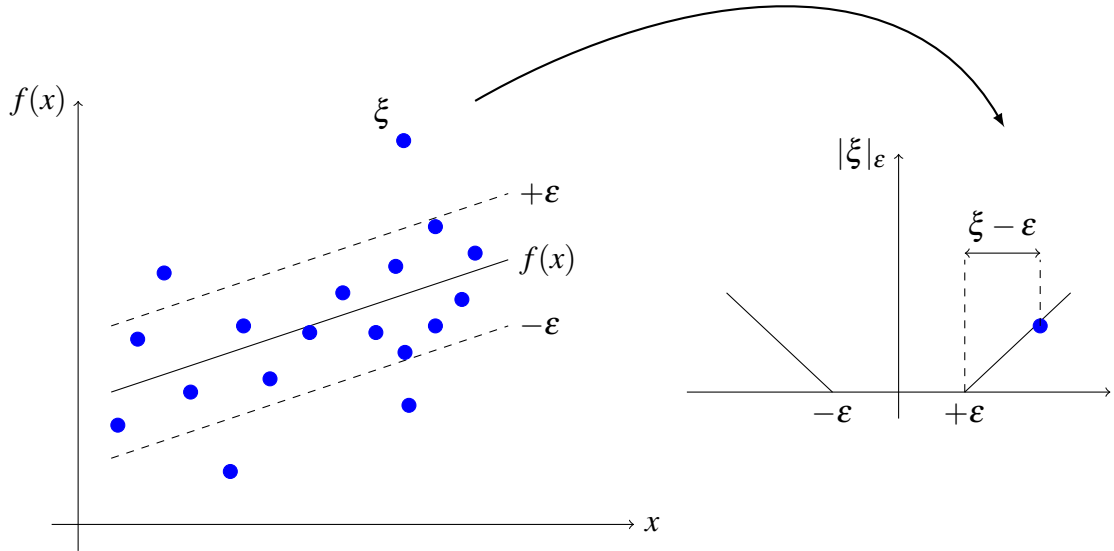


Figure 4 – Illustration of the SVR regression curve with the  $\varepsilon$ -insensitive zone and the slack variables (left); Vapnik  $\varepsilon$ -insensitive loss function (right).

from Eqs. (2.4) and (2.5), as follows:

$$\begin{aligned} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*; \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\zeta}, \boldsymbol{\zeta}^*) &:= \frac{1}{2} \|\mathbf{w}\|^2 + \gamma \sum_{n=1}^N (\xi_n + \xi_n^*) - \sum_{n=1}^N \alpha_n (\mathbf{w}^\top \mathbf{x}_n + b - y_n + \varepsilon + \xi_n) \\ &\quad - \sum_{n=1}^N \alpha_n^* (y_n - \mathbf{w}^\top \mathbf{x}_n - b + \varepsilon + \xi_n^*) - \sum_{n=1}^N (\zeta_n \xi_n + \zeta_n^* \xi_n^*), \end{aligned} \quad (2.7)$$

where  $\alpha_n \geq 0$ ,  $\alpha_n^* \geq 0$ ,  $\zeta_n \geq 0$ ,  $\zeta_n^* \geq 0$  are the Lagrange multipliers and  $\mathcal{L}$  denotes the Lagrangian, whose saddle point is characterized by

$$\max_{(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\zeta}, \boldsymbol{\zeta}^* \geq 0)} \min_{(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*)} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*; \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\zeta}, \boldsymbol{\zeta}^*), \quad (2.8)$$

where the gradients related to the primal variables are set to zero, according to Karush-Kuhn-Tucker (KKT) conditions for optimality (KARUSH, 1939; KUHN; TUCKER, 1951), such as:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{0} \Rightarrow \mathbf{w} = \sum_{n=1}^N (\alpha_n - \alpha_n^*) \mathbf{x}_n,$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{n=1}^N (\alpha_n - \alpha_n^*) = 0,$$

$$\frac{\partial \mathcal{L}}{\partial \xi_n} = 0 \Rightarrow \gamma - \alpha_n - \zeta_n = 0,$$

$$\frac{\partial \mathcal{L}}{\partial \xi_n^*} = 0 \Rightarrow \gamma - \alpha_n^* - \zeta_n^* = 0.$$

(2.9)

Using the results of Eqs. (2.9) to eliminate the corresponding variables from the Lagrangian, a new optimization problem can be formulated by the maximization of

$$\max_{\boldsymbol{\alpha}, \boldsymbol{\alpha}^*} J_d(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = -\frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N (\alpha_n - \alpha_n^*)(\alpha_m - \alpha_m^*) \mathbf{x}_n^\top \mathbf{x}_m - \varepsilon \sum_{n=1}^N (\alpha_n + \alpha_n^*) + \sum_{n=1}^N (\alpha_n - \alpha_n^*) y_n, \quad (2.10)$$

$$\text{subject to } \begin{cases} \sum_{n=1}^N (\alpha_n - \alpha_n^*) = 0, \\ 0 \leq \alpha_n \leq \gamma, \\ 0 \leq \alpha_n^* \leq \gamma, \end{cases} \quad (2.11)$$

for  $n = 1, \dots, N$ . The problem in Eqs. (2.10) and (2.11) corresponds to the formulation of a *quadratic programming* (QP) problem in the dual weight space, and can also be referred to as the *dual problem*. The term ‘‘dual’’ is used because now the problem is solved in the space of  $2N$  dual variables  $\alpha_n$  and  $\alpha_n^*$ , instead of  $\mathbf{w}$  in the primal space. Using Eq. (2.1) and the expression for  $\mathbf{w}$  in Eq. (2.9), the SVR model solution is given by

$$f(\mathbf{x}^*) = \sum_{n=1}^N (\alpha_n - \alpha_n^*) \mathbf{x}_n^\top \mathbf{x}^* + b. \quad (2.12)$$

An important property of the QP problem in Eqs. (2.10), (2.11) is that many resulting values of  $\alpha_n, \alpha_n^*$  are equal to zero. Hence, the obtained solution is sparse, i.e. the sum in Eq. (2.12) should be taken only over non-zero values of  $(\alpha_n - \alpha_n^*)$ , instead of all training patterns. The input examples  $\mathbf{x}_n$ , for which the corresponding Lagrange multipliers are not null, are the support vectors. In addition, note that, unlike the primal space solution in Eq. (2.1), the expressions in Eqs. (2.10)-(2.12) are independent of the parameter vector  $\mathbf{w}$ . Therefore, one should say that the SVR dual problem corresponds to a non-parametric model, although its original formulation in Eqs. (2.4) and (2.5) corresponds to a parametric model.

### 2.2.2 SVR for Nonlinear Regression

The majority of regression problems in machine learning are essentially nonlinear. Therefore, the linear formulation of the SVR model in Section 2.2.1 should be extended to the nonlinear case. Then, given the estimation dataset  $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$ , with  $\mathbf{x}_n \in \mathbb{R}^d$  and  $y_n \in \mathbb{R}$ , the function  $f(\cdot)$  in Eq. (2.1) now takes the form

$$f(\mathbf{x}_n) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b, \quad (2.13)$$

where  $\mathbf{w} \in \mathbb{R}^{d_h}$ ,  $b \in \mathbb{R}$  and  $\boldsymbol{\phi}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^{d_h}$  is a nonlinear map into a higher dimensional feature space (whose dimensionality  $d_h$  might be infinite), where the problem is linearly solvable. Then, the nonlinear version of the SVR optimization problem in the primal space (Eqs. (2.4) and (2.5)) is given by

$$\min_{\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*} J_p(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\xi}^*) = \frac{1}{2} \|\mathbf{w}\|^2 + \gamma \sum_{n=1}^N (\xi_n + \xi_n^*), \quad (2.14)$$

$$\text{subject to } \begin{cases} y_n - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) - b \leq \varepsilon + \xi_n \\ \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b - y_n \leq \varepsilon + \xi_n^* \\ \xi_n, \xi_n^* \geq 0, \text{ for } n = 1, \dots, N, \end{cases} \quad (2.15)$$

where the only difference between it and the previous linear formulation is the replacement of  $\mathbf{x}_n$  by  $\boldsymbol{\phi}(\mathbf{x}_n)$  into the constraints of Eq. (2.15). Again, one can apply the Lagrangian in Eqs. (2.14) and (2.15) to obtain the formulation of the dual (and QP) problem, as

$$\max_{\boldsymbol{\alpha}, \boldsymbol{\alpha}^*} J_d(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = -\frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N (\alpha_n - \alpha_n^*)(\alpha_m - \alpha_m^*) k(\mathbf{x}_n, \mathbf{x}_m) - \varepsilon \sum_{n=1}^N (\alpha_n + \alpha_n^*) + \sum_{n=1}^N (\alpha_n - \alpha_n^*) y_n, \quad (2.16)$$

$$\text{subject to } \begin{cases} \sum_{n=1}^N (\alpha_n - \alpha_n^*) = 0 \\ 0 \leq \alpha_n \leq \gamma \\ 0 \leq \alpha_n^* \leq \gamma, \text{ for } n = 1, \dots, N, \end{cases} \quad (2.17)$$

where  $k(\cdot, \cdot)$  is known as a *kernel function*, which is used to replace the inner product  $\boldsymbol{\phi}(\cdot)^\top \boldsymbol{\phi}(\cdot)$  in the nonlinear versions of the optimality conditions in Eq. (2.9), without the need of explicitly computing the mapping  $\boldsymbol{\phi}$ . The equality  $k(\mathbf{x}_n, \mathbf{x}_m) = \boldsymbol{\phi}(\mathbf{x}_n)^\top \boldsymbol{\phi}(\mathbf{x}_m)$  for  $n, m = 1, \dots, N$ , is an appealing property of the kernel methods, referred to as *kernel trick*. This result is derived from Mercer's conditions (MERCER, 1909), given in the following theorem:

**Theorem 2.1** (MERCER, 1909) *Let the functions  $k(\cdot) \in L^2(\mathcal{C})$  and  $g(\cdot) \in L^2(\mathcal{C})$ , where  $L^2$  is a space of square integrable functions,  $\mathcal{C}$  is a compact subset of  $\mathbb{R}^d$ , and  $k(\mathbf{x}, \mathbf{x}')$  describes an inner product in some feature space and let the vectors  $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ . To guarantee that a continuous symmetric function  $k(\cdot, \cdot)$  has an expansion*

$$k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} c_i \varphi_i(\mathbf{x}) \varphi_i(\mathbf{x}')$$

Table 1 – Examples of kernel functions.

kernel	Description	
	Function	Hiperparameters
Linear	$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$	-
Polynomial	$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + \tau)^p$	$p$ (degree) and $\tau$ .
Gaussian	$k(\mathbf{x}, \mathbf{x}') = \exp\{-\frac{\ \mathbf{x}-\mathbf{x}'\ _2^2}{2\sigma^2}\}$	$\sigma$ (bandwidth).
Sigmoid	$k(\mathbf{x}, \mathbf{x}') = \tanh\{\kappa_1 \mathbf{x}^\top \mathbf{x}' + \kappa_2\}$	$\kappa_1, \kappa_2$ .

with coefficients  $c_i > 0$ , it is necessary and sufficient that the condition

$$\iint_{\mathcal{C}} k(\mathbf{x}, \mathbf{x}') g(\mathbf{x}) g(\mathbf{x}') d\mathbf{x} d\mathbf{x}' \geq 0$$

is valid for any square integrable function  $g(\cdot)$ .

Then, using the Mercer's condition one can write  $k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{n_f} \sqrt{c_i} \phi_i(\mathbf{x}) \sqrt{c_i} \phi_i(\mathbf{x}')$ , where  $n_f \in \mathbb{N}$  or  $n_f \rightarrow +\infty$ , and define  $\phi_i(\mathbf{x}) = \sqrt{c_i} \phi_i(\mathbf{x})$  and  $\phi_i(\mathbf{x}') = \sqrt{c_i} \phi_i(\mathbf{x}')$ , such that the kernel function can be expressed as the inner product

$$k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}') \Rightarrow \text{kernel trick.} \quad (2.18)$$

Hence, if  $k(\cdot, \cdot)$  is positive semidefinite, then the equality in Eq. (2.18) holds. Several functions can be used as suitable kernel functions, among which some common choices are summarized in Table 1.

For the kernel functions in Table 1, Mercer's condition holds for all values of  $\sigma > 0$  in Gaussian kernel and for  $\tau \geq 0$  in polynomial cases, but not for all the possible choices of  $\kappa_1, \kappa_2$  in the sigmoid kernel (SUYKENS *et al.*, 2002b). A further discussion about the kernel functions can be found in Schölkopf and Smola (2002) and Burges (1998).

Finally, the predicted output of the nonlinear SVR model, for a novel incoming vector  $\mathbf{x}^*$ , is given by

$$f(\mathbf{x}^*) = \sum_{n=1}^N (\alpha_n - \alpha_n^*) k(\mathbf{x}_n, \mathbf{x}^*) + b, \quad (2.19)$$

where  $\alpha_n, \alpha_n^*$  are the solution of the QP problem in Eqs. (2.16) and (2.17), and  $b$  follows from the complementary KKT conditions. As in the linear case, many values of  $\alpha_n, \alpha_n^*$  are null and, therefore, the model solution is sparse and can be expressed only in terms of its SVs. Again, the dual problem results in a nonparametric model, which does not depend on the dimensionality  $d$  of the input data, but only on the number  $N$  of training samples. Then, the computational



complexity of the SVR model is  $\mathcal{O}(N^3)$  and its memory demand is  $\mathcal{O}(N^2)$ , because although its solution is sparse, its algorithm starts by considering all training samples as potential SVs.

The next section discusses another important kernel-based method, the LSSVR model, derived from the original SVR formulation through some modifications in its primal optimization problem in Eqs. (2.14) and (2.15).

### 2.3 Least Squares Support Vector Regression

Despite being a powerful kernel method successfully applied to non-linear regression problems, SVR works with a QP problem, which is complex and time-consuming, demanding high computational resources for its implementation. In this context, the Least Squares SVR (LSSVR) model, which was introduced by Suykens *et al.* (2002b) and is closely related to kernel ridge regression (SAUNDERS *et al.*, 1998), was proposed as an alternative kernel method to the original Vapnik's SVR model in order to simplify its formulation, while attempting to keep its main advantages. Therefore, this section explores the development of the theoretical foundation of the LSSVR model.

#### 2.3.1 Estimation in the Dual Space

Consider again the nonlinear regression model, described in Eq. (2.13) and inserted here for a better understanding, such as

$$f(\mathbf{x}_n) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b, \quad (2.20)$$

where  $\mathbf{w} \in \mathbb{R}^{d_h}$ ,  $b \in \mathbb{R}$ ,  $\mathbf{x}_n \in \mathbb{R}^d$ ,  $y_n \in \mathbb{R}$ , for  $n = 1, \dots, N$ .  $\boldsymbol{\phi}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^{d_h}$  is the mapping to the high dimensional feature space, as previously defined in Section 2.2.2 for the SVR model.

Basically, there are two modifications that convert the SVR primal problem into the LSSVR formulation. First, instead of using the  $\varepsilon$ -insensitive loss function in Eq. (2.6), one may choose to use an SSE cost function. The second modification is replacing the inequality constraints in Eq. (2.15) by equality constraints. Thereby, the primal optimization problem in LSSVR model leads to minimize the following functional:

$$\min_{\mathbf{w}, b, \mathbf{e}} J_p(\mathbf{w}, \mathbf{e}) = \frac{1}{2} \|\mathbf{w}\|^2 + \gamma \frac{1}{2} \sum_{n=1}^N e_n^2, \quad (2.21)$$

$$\text{subject to } \begin{cases} y_n = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b + e_n, \text{ for } n = 1, \dots, N, \end{cases} \quad (2.22)$$

where  $e_n = y_n - f(\mathbf{x}_n)$  is the error due to the  $n$ -th input pattern, which plays a similar role as the slack variables  $\xi_n$  in the SVR formulation.  $\gamma > 0$  is again a regularization parameter, whose tuning involves finding a trade-off between the minimization of training errors and smoothness of the estimation function. The problem in Eq. (2.21) with the equality constraints in Eq. (2.22) was treated in the kernel ridge regression by Saunders *et al.* (1998), but without using a bias term.

Regarding the case of nonlinear regression, one cannot exactly solve the primal problem in Eqs. (2.21) and (2.22), since the dimension  $d_h$  of  $\mathbf{w}$  and  $\boldsymbol{\phi}$  is unknown (it might even become infinite). However, it is possible to derive the dual problem by constructing the Lagrangian, which is given by

$$\mathcal{L}(\mathbf{w}, b, \mathbf{e}, \boldsymbol{\alpha}) := \frac{1}{2} \|\mathbf{w}\|^2 + \gamma \frac{1}{2} \sum_{n=1}^N e_n^2 - \sum_{n=1}^N \alpha_n (\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b + e_n - y_n), \quad (2.23)$$

where, different from the SVR model, the Lagrange multipliers in Eq. (2.23) may assume positive or negative values because of the equality constraints in Eq. (2.22). The next step to obtain the solution of Eq. (2.23) is setting the gradients of the primal variables to zero, according to Karush-Kuhn-Tucker (KKT) (FLETCHER, 2013) conditions for optimality, leading to

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{0} &\Rightarrow \mathbf{w} = \sum_{n=1}^N \alpha_n \boldsymbol{\phi}(\mathbf{x}_n), \\ \frac{\partial \mathcal{L}}{\partial b} = 0 &\Rightarrow \sum_{n=1}^N \alpha_n = 0, \end{aligned} \quad (2.24)$$

$$\frac{\partial \mathcal{L}}{\partial e_n} = 0 \Rightarrow \alpha_n = \gamma e_n,$$

$$\frac{\partial \mathcal{L}}{\partial \alpha_n} = 0 \Rightarrow \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b + e_n - y_n = 0,$$

for  $n = 1, \dots, N$ .

In order to eliminate  $\mathbf{w}$  and  $e_n$  from the optimization problem, it is possible to use the first and third expressions in Eq. (2.24) to rewrite the last one as

$$\sum_{n=1}^N \alpha_n \boldsymbol{\phi}(\mathbf{x}_n)^\top \boldsymbol{\phi}(\mathbf{x}_n) + b + \frac{1}{\gamma} \alpha_n = y_n. \quad (2.25)$$

Moreover, the second equality in Eq. (2.24) and Eq. (2.25) can be arranged in corresponding matrix forms, respectively, such as

$$\mathbf{1}_N^\top \boldsymbol{\alpha} = 0, \quad (2.26)$$

and

$$b\mathbf{1}_N + (\mathbf{K} + \gamma^{-1}\mathbf{I}_N)\boldsymbol{\alpha} = \mathbf{y}, \quad (2.27)$$

where  $\mathbf{1}_N \in \mathbb{R}^N$  is a vector of ones,  $\mathbf{I}_N \in \mathbb{R}^{N \times N}$  is an identity matrix,  $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_N]^\top$ ,  $\mathbf{y} = [y_1, \dots, y_N]^\top$  and  $\mathbf{K} \in \mathbb{R}^{N \times N}$  is a symmetric and positive semidefinite (or positive definite) matrix, called *kernel matrix* (or Gram matrix), whose entries are

$$K_{i,j} := k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i)^\top \boldsymbol{\phi}(\mathbf{x}_j), \text{ for } i, j = 1, \dots, N, \quad (2.28)$$

where  $k(\cdot, \cdot)$  is the chosen kernel function. The last equality in Eq. (2.28) is derived from the kernel trick in Eq. (2.18).

Therefore, one can write Eqs. (2.26) and (2.27) as a system of  $N + 1$  linear equations given by

$$\underbrace{\begin{bmatrix} 0 & \mathbf{1}_N^\top \\ \mathbf{1}_N & \mathbf{K} + \gamma^{-1}\mathbf{I}_N \end{bmatrix}}_{\boldsymbol{\Omega}} \underbrace{\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix}}_{\boldsymbol{\alpha}_o} = \underbrace{\begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix}}_{\mathbf{y}_o}, \quad (2.29)$$

where the lagrange multipliers  $\alpha_n$  and the bias  $b$  are the unknown variables. The solution of the linear system in Eq. (2.29) corresponds to the LSSVR model solution.

Alternatively, the linear system in Eq. (2.29) can be written as  $\boldsymbol{\Omega}\boldsymbol{\alpha}_o = \mathbf{y}_o$ , with  $\boldsymbol{\Omega} \in \mathbb{R}^{(N+1) \times (N+1)}$ ,  $\boldsymbol{\alpha}_o \in \mathbb{R}^{N+1}$  and  $\mathbf{y}_o \in \mathbb{R}^{N+1}$ , and can simply be solved by least squares method, in its batch mode, as

$$\boldsymbol{\alpha}_o = \boldsymbol{\Omega}^\dagger \mathbf{y}_o, \quad (2.30)$$

where  $\boldsymbol{\Omega}^\dagger = (\boldsymbol{\Omega}^\top \boldsymbol{\Omega})^{-1} \boldsymbol{\Omega}^\top$  is the Moore-Penrose inverse of the matrix  $\boldsymbol{\Omega}$  (GOLUB; VAN LOAN, 2013). Then, the computational complexity in solving Eq. (2.30) is  $\mathcal{O}(N^3)$ , and its memory demand is  $\mathcal{O}(N^2)$ .

Different methods can be used to calculate the Moore-Penrose generalized inverse of a matrix, namely orthogonal projection, orthogonalization and singular value decomposition (SVD) methods (HUANG, 2014). However, for large-scale datasets, the use of iterative methods is recommended, such as successive over-relaxation (SOR), conjugate gradient (CG) and generalized minimal residual (GMRES) (SUYKENS *et al.*, 2002b). In this work, the SVD approach is adopted.

Finally, the LSSVR model solution is expressed by

$$f(\mathbf{x}) = \sum_{n=1}^N \alpha_n k(\mathbf{x}, \mathbf{x}_n) + b. \quad (2.31)$$

As in the SVR case, the dual problem in Eq. (2.29) has a global solution. In addition, it is easier to compute, compared to the QP problem in SVR. However, a drawback of the LSSVR simplified formulation is the lack of sparsity, which is immediately clear from the third condition for optimality in Eq. (2.24),  $\alpha_n = \gamma e_n$ , for  $n = 1, \dots, N$ . Therefore, in the LSSVR model every input sample is an SV since, normally, no  $\alpha_n$  will be exactly equal to zero. The LSSVR model is summarized in the form of a pseudo-code in Algorithm 1.

---

**Algorithm 1:** - Pseudo-code for the LSSVR model.

---

**Require:** Set  $\gamma$  and the kernel hiperparameters ( $\sigma$  for the Gaussian kernel, for example)  
 From the estimation dataset  $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$ :  
 Compute the kernel matrix  $\mathbf{K}$ ;  
 Build the linear system of Eq. (2.29);  
 Compute  $\boldsymbol{\alpha}$  and  $b$  from Eq. (2.30); % LSSVR solution

---

Next, an LSSVR derived approach will be considered - the so called FS-LSSVR model - which, instead of solving the dual optimization problem in Eq. (2.29), searches for an approximate solution of the primal problem in Eqs. (2.21) and (2.22).

## 2.4 Fixed Size Least Squares Support Vector Regression

As previously presented in Section 2.3, the standard LSSVR comprises a dual optimization problem with a nonparametric model, resulting in a non sparse solution. However, in some applications, especially for big data analysis, it is often advantageous to work in the primal space solving the optimization problem in Eqs. (2.21) and (2.22), where the dimension  $d_h$  of the parameter vector  $\mathbf{w}$  may be smaller compared to the dimension  $N$  of the vector  $\boldsymbol{\alpha}$  of Lagrange multipliers.

Therefore, this section discusses a framework developed by Suykens *et al.* (2002b) to solve the LSSVR problem in primal space. This approach is called Fixed Size LSSVR (FS-LSSVR) model, which is of considerable relevance to this work.

### 2.4.1 Estimation in Primal Weight Space

A challenge that appears in solving nonlinear problems in primal space is that the feature map  $\boldsymbol{\phi}$  may be not explicitly known. For instance, this is the case for the Gaussian kernel, where the dimension  $d_h$  of  $\boldsymbol{\phi}$  is infinite (VAPNIK, 1998). The ability to work in primal

space requires either an explicit expression for  $\phi$  (e.g. linear kernel) or an approximation to the feature map  $\hat{\phi}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^M$ , based on a sampled set of  $M \ll N$  instances, the *prototype vectors* (PVs), from the whole training dataset. This feature approximation is obtained by the Nyström method (BAKER, 1977; WILLIAMS; SEEGER, 2001), as seen below.

#### 2.4.1.1 Approximation to the Feature Map

Recalling the LSSVR primal problem in Eq. (2.21), it is possible to rewrite it as

$$\min_{\mathbf{w}, b} J_p(\mathbf{w}, b) = \frac{1}{2} \|\mathbf{w}\|^2 + \gamma \frac{1}{2} \sum_{n=1}^N (y_n - \mathbf{w}^\top \phi(\mathbf{x}_n) - b)^2, \quad (2.32)$$

where the error variables were replaced by  $y_n - \mathbf{w}^\top \phi(\mathbf{x}_n) - b = e_n$ , according to the constraints in Eq. (2.22). As previously mentioned, it is necessary to find a finite approximation  $\hat{\phi}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^M$  (in general,  $M \ll N$ ) of  $\phi$  to solve the primal optimization problem in Eq. (2.32).

Thus, let the input vector  $\mathbf{x}$  be a random sample from an unknown probability density  $p(\mathbf{x})$ . Moreover, consider the following eigenfunction expansion of a kernel function (WILLIAMS; SEEGER, 2001):

$$k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{N_f} \lambda'_i \varphi_i(\mathbf{x}) \varphi_i(\mathbf{x}'), \quad (2.33)$$

where  $N_f \leq \infty$ ,  $\lambda'_1 \geq \dots \geq \lambda'_{N_f} \geq 0$  and  $\varphi_1, \dots, \varphi_{N_f}$  denote, respectively, the eigenvalues and eigenfunctions of the operator whose kernel function is  $k(\cdot, \cdot)$ , which is in agreement with the Fredholm integral equation of the first kind

$$\int k(\mathbf{x}, \mathbf{x}') \varphi_i(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} = \lambda'_i \varphi_i(\mathbf{x}'). \quad (2.34)$$

The integral in Eq. (2.34) can be discretized on a finite and i.i.d. set of pattern inputs  $\{\mathbf{x}_n\}_{n=1}^N$ , where it is possible to rewrite it by replacing the integral over  $p(\mathbf{x})$  with an empirical average, such as

$$\frac{1}{N} \sum_{n=1}^N k(\mathbf{x}_n, \mathbf{x}') \mathbf{u}_i(\mathbf{x}_n) = \lambda_i^s \mathbf{u}_i(\mathbf{x}'), \text{ for } i = 1, \dots, N, \quad (2.35)$$

where the eigenvalues  $\lambda'_i$  and eigenfunctions  $\varphi_i$  from the continuous problem can be approximated by the sample eigenvalues  $\lambda_i^s$  and eigenvectors  $\mathbf{u}_i$  as (WILLIAMS; SEEGER, 2001)

$$\hat{\phi}_i(\mathbf{x}') \approx \sqrt{N} \mathbf{u}_i \text{ and } \hat{\lambda}'_i \approx \frac{1}{N} \lambda_i^s, \text{ for } i = 1, \dots, N. \quad (2.36)$$

Furthermore, the matrix eigenproblem associated to the expression in Eq. (2.35) can be denoted by

$$\mathbf{K}\mathbf{U} = \mathbf{U}\mathbf{\Lambda}, \quad (2.37)$$

where  $\mathbf{K} \in \mathbb{R}^{N \times N}$  is the kernel matrix,  $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_N] \in \mathbb{R}^{N \times N}$  is a matrix of the eigenvectors of  $\mathbf{K}$  and  $\mathbf{\Lambda} = \text{diag}\{\lambda_n^s\}_{n=1}^N \in \mathbb{R}^{N \times N}$  is a diagonal matrix formed by the respective nonnegative eigenvalues arranged in a decreasing order. Then, substituting the expressions of Eq. (2.36) into Eq. (2.35), one obtains the Nyström approximation to the  $i$ -th eigenfunction (BAKER, 1977)

$$\hat{\phi}_i(\mathbf{x}') \approx \frac{\sqrt{N}}{\lambda_i^s} \sum_{n=1}^N (u_i)_n k(\mathbf{x}_n, \mathbf{x}'), \quad (2.38)$$

where  $(u_i)_n$  is the  $n$ -th element of the  $i$ -th eigenvector in Eq. (2.37). Thus, based on Nyström approximation, one can write an expression for the  $i$ -th entry of the approximated finite feature map  $\hat{\phi} : \mathbb{R}^d \rightarrow \mathbb{R}^N$  as (DE-BRABANTER *et al.*, 2010; SUYKENS *et al.*, 2002b)

$$\hat{\phi}_i(\mathbf{x}') = \sqrt{\hat{\lambda}_i'} \hat{\phi}_i(\mathbf{x}') = \frac{1}{\sqrt{\lambda_i^s}} \sum_{n=1}^N (u_i)_n k(\mathbf{x}_n, \mathbf{x}'), \text{ for } i = 1, \dots, N. \quad (2.39)$$

#### 2.4.1.2 Imposing Sparseness and Subset Selection

Note that the approximation in Eq. (2.39) is carried out over all the  $N$  available input samples. However, this can be a problem if one deals with a large-scale dataset, since the memory demand and the computational complexity can become too large. Then, it is advantageous to use only a subsample of  $M \ll N$  instances to compute the approximation  $\hat{\phi} : \mathbb{R}^d \rightarrow \mathbb{R}^M$  for the feature map, such as

$$\hat{\phi}_i(\mathbf{x}') = \frac{1}{\sqrt{\bar{\lambda}_i}} \sum_{m=1}^M (\bar{u}_i)_m k(\mathbf{z}_m, \mathbf{x}'), \text{ for } i = 1, \dots, M, \quad (2.40)$$

where  $\bar{\lambda}_i$  and  $\bar{\mathbf{u}}_i$  are, respectively, the eigenvalues and the eigenvectors of the reduced kernel matrix  $\bar{\mathbf{K}} \in \mathbb{R}^{M \times M}$ , with  $\bar{K}_{ij} = k(\mathbf{z}_i, \mathbf{z}_j)$ . The vectors  $\mathbf{z}_i$  and  $\mathbf{z}_j$  belong to a subset of  $M$  training instances, the PVs, randomly selected. One should note that in this methodology it is not necessary to calculate the full kernel matrix  $\mathbf{K}$  with all the training samples.

In order to perform a more suitable selection of the  $M$  prototype vectors instead of a random selection, one can adopt an entropy based selection method, as proposed by Suykens *et al.* (2002b). In this method, the quadratic Rényi's entropy  $H_R$  is used as (GIROLAMI, 2002)

$$H_R = -\log \int p(\mathbf{x})^2 d\mathbf{x}, \quad (2.41)$$

which can be approximated by

$$\int \hat{p}(\mathbf{x})^2 d\mathbf{x} = \frac{1}{M^2} \mathbf{1}_M^\top \bar{\mathbf{K}} \mathbf{1}_M, \quad (2.42)$$

where  $\mathbf{1}_M = [1, \dots, 1]^\top$ .

In general, the procedure to select the PVs begins with randomly choosing a fixed size working set  $\mathcal{D}^{pv} = \{\mathbf{x}_m\}_{m=1}^M$  from the training data, and another set  $\mathcal{D}_1$  with the  $N - M$  remaining training samples. The next step is to randomly select a pattern  $\mathbf{x}_m$  from  $\mathcal{D}^{pv}$  and replace it by a randomly selected pattern  $\mathbf{x}^*$  from  $\mathcal{D}_1$ . If the entropy increases by taking the pattern  $\mathbf{x}^*$  instead of  $\mathbf{x}_m$ , then  $\mathbf{x}^*$  is accepted in  $\mathcal{D}^{pv}$ . Otherwise, it is rejected (and returns to  $\mathcal{D}_1$ ) and  $\mathbf{x}_m$  stays in the working set. This procedure continues until the change in entropy value is very small or the maximum number of iterations  $N_{iter}$  is achieved.

It is important to highlight that higher the entropy found in the selected subset of the PVs, better it will represent the whole training dataset. Simply stated, the optimality of this subset selection is related to the final accuracy that can be obtained by the model predictions (ESPINOZA *et al.*, 2004).

#### 2.4.1.3 The FS-LSSVR Solution

As long as the PVs subset is properly selected and given the finite approximation  $\hat{\phi}$  of the feature map, the FS-LSSVR model solution is obtained by minimizing the following functional in the primal space:

$$J_p(\hat{\mathbf{w}}, \hat{b}) = \frac{1}{2} \hat{\mathbf{w}}^\top \hat{\mathbf{w}} + \gamma \frac{1}{2} \sum_{n=1}^N (y_n - \hat{\mathbf{w}}^\top \hat{\phi}(\mathbf{x}_n) - \hat{b})^2, \quad (2.43)$$

where  $\hat{\phi}(\mathbf{x}_n) = [\hat{\phi}_1(\mathbf{x}_n), \dots, \hat{\phi}_m(\mathbf{x}_n)]^\top \in \mathbb{R}^M$ ,  $\hat{\mathbf{w}} \in \mathbb{R}^M$  and  $\hat{b} \in \mathbb{R}$  are approximate solutions for  $\mathbf{w}$  and  $b$ , respectively.

In order to solve the primal problem in Eq. (2.43), it is necessary to compute the gradient vectors,  $\nabla_{\hat{\mathbf{w}}} J_p(\hat{\mathbf{w}}, \hat{b}) = \partial J_p(\hat{\mathbf{w}}, \hat{b}) / \partial \hat{\mathbf{w}}$  and  $\nabla_{\hat{b}} J_p(\hat{\mathbf{w}}, \hat{b}) = \partial J_p(\hat{\mathbf{w}}, \hat{b}) / \partial \hat{b}$ , and set them to zero. Then, one gets

$$\frac{\partial J_p}{\partial \hat{\mathbf{w}}} = \hat{\mathbf{w}} - \gamma \sum_{n=1}^N [y_n - (\hat{\mathbf{w}}^\top \hat{\phi}(\mathbf{x}_n) + \hat{b})] \hat{\phi}(\mathbf{x}_n) = \mathbf{0}_M, \quad (2.44)$$

and

$$\frac{\partial J_p}{\partial \hat{b}} = -\gamma \sum_{n=1}^N [y_n - (\hat{\mathbf{w}}^\top \hat{\phi}(\mathbf{x}_n) + \hat{b})] = 0, \quad (2.45)$$

where  $\mathbf{0}_M \in \mathbb{R}^M$  is a vector of zeros. The expressions in Eqs. (2.44) and (2.45) can be arranged in corresponding matrix forms, respectively, as

$$\frac{1}{\gamma} \hat{\mathbf{w}} + \hat{\Phi}^\top \hat{\Phi} \hat{\mathbf{w}} - \hat{\Phi}^\top \mathbf{y} + \hat{\Phi}^\top \mathbf{1}_N \hat{b} = \mathbf{0}_M,$$

$$\left( \hat{\Phi}^\top \hat{\Phi} + \frac{1}{\gamma} \mathbf{I}_M \right) \hat{\mathbf{w}} + \hat{\Phi}^\top \mathbf{1}_N \hat{b} = \hat{\Phi}^\top \mathbf{y}, \quad (2.46)$$

and

$$-\mathbf{1}_N^\top \mathbf{y} + \mathbf{1}_N^\top \hat{\Phi} \hat{\mathbf{w}} + \mathbf{1}_N^\top \mathbf{1}_N \hat{b} = 0,$$

$$\mathbf{1}_N^\top \hat{\Phi} \hat{\mathbf{w}} + \mathbf{1}_N^\top \mathbf{1}_N \hat{b} = \mathbf{1}_N^\top \mathbf{y}, \quad (2.47)$$

where  $\hat{\Phi} = [\hat{\phi}(\mathbf{x}_1), \dots, \hat{\phi}(\mathbf{x}_N)]^\top \in \mathbb{R}^{N \times M}$  is the approximated feature matrix for all the training instances, which is given by

$$\hat{\Phi} = \begin{bmatrix} \hat{\phi}_1(\mathbf{x}_1) & \dots & \hat{\phi}_M(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \hat{\phi}_1(\mathbf{x}_N) & \dots & \hat{\phi}_M(\mathbf{x}_N) \end{bmatrix}. \quad (2.48)$$

Finally, the optimization problem in Eq. (2.43) can be solved by arranging Eqs. (2.46) and (2.47) in the following linear system of  $M + 1$  equations:

$$\begin{bmatrix} \hat{\Phi}^\top \hat{\Phi} + \frac{1}{\gamma} \mathbf{I}_M & \hat{\Phi}^\top \mathbf{1}_N \\ \mathbf{1}_N^\top \hat{\Phi} & \mathbf{1}_N^\top \mathbf{1}_N \end{bmatrix} \begin{bmatrix} \hat{\mathbf{w}} \\ \hat{b} \end{bmatrix} = \begin{bmatrix} \hat{\Phi}^\top \mathbf{y} \\ \mathbf{1}_N^\top \mathbf{y} \end{bmatrix}. \quad (2.49)$$

Therefore, the corresponding FS-LSSVR model solution is given by

$$\hat{f}(\mathbf{x}) = \hat{\mathbf{w}}^\top \hat{\phi}(\mathbf{x}) + \hat{b}, \quad (2.50)$$

where  $\hat{f}(\cdot)$  is an approximate solution for  $f(\cdot)$  of Eq. (2.20). The sparsity of the FS-LSSVR model is previously ensured in the choice of the  $M$  prototype vectors. Consequently, its computational complexity and memory demand are  $\mathcal{O}(NM^2)$  and  $\mathcal{O}(NM)$  (DE-BRABANTER, 2011; MALL; SUYKENS, 2013), respectively. The pseudo-code for the FS-LSSVR model is summarized in Algorithm 2.



---

**Algorithm 2:** - Pseudo-code for the FS-LSSVR model.

---

**Require:**  $M, \gamma, \sigma, N_{iter}$ ;  
 Choose a working set  $\mathcal{D}^{pv} = \{\mathbf{x}_m\}_{m=1}^M$ ;  
 Select  $\mathcal{D}_1$  with the  $N - M$  remaining training samples;  
**for**  $i = n_{iter} : N_{iter}$ , **do**  
   Compute  $H_R$  of  $\mathcal{D}^{pv}$  from Eq. (2.42);  
   Randomly select  $\mathbf{x}_m$  from  $\mathcal{D}^{pv}$ ;  
   Randomly select  $\mathbf{x}^*$  from  $\mathcal{D}_1$ ;  
   Replace  $\mathbf{x}_m$  by  $\mathbf{x}^*$  in  $\mathcal{D}^{pv}$ ;  
   Compute  $H_R^*$  of  $\mathcal{D}^{pv}$  from Eq. (2.42);  
   **if**  $H_R^* > H_R$ , **then**  
      $\mathbf{x}^*$  is accepted for  $\mathcal{D}^{pv}$ ;  
   **else**  
      $\mathbf{x}^*$  is rejected by  $\mathcal{D}^{pv}$ ;  
   **end if**  
**end for**  
 Compute  $\hat{\Phi}$  from Eq. (2.40) and Eq. (2.48);  
 Build the linear system of Eq.(2.49);  
 Compute  $\hat{\mathbf{w}}$  and  $\hat{\mathbf{b}}$  from Eq.(2.30); % FS-LSSVR solution

---

## 2.5 Concluding Remarks

This chapter has been an overview of support vector models, especially those used in nonlinear regression tasks. Initially, a brief discussion about the SVR theory was presented, since it was a pioneer among the kernel models and covers important aspects that make it a widely used tool in solving regression problems. The SVR model is based on Vapnik's  $\varepsilon$ -insensitive loss function and solves a convex QP optimization problem, whose solution is a global minimum and preserves sparseness. Moreover, the size of this QP problem does not depend on the dimension  $d$  of the input space.

In addition, a framework of the LSSVR model was discussed, which is an alternative kernel method to the standard SVR model, since it works with an SSE loss function instead the  $\varepsilon$ -insensitive function of the SVR model. Also the inequality constraints of SVR are replaced by equality constraints in the LSSVR formulation. Consequently, the global optimum in LSSVR problem is easier to obtain by solving a set of linear equations. On the other hand, the sparsity common to the SVR model is lost in the solution of the LSSVR problem.

Finally, the basis of the FS-LSSVR model was presented, which is derived from the standard LSSVR formulation. However, instead of solving the dual optimization problem, it uses an approximation to the feature map  $\phi$  and solves the optimization problem in the primal space. The feature approximation  $\hat{\phi}$  was obtained using the Nyström method based on a subsample of

Table 2 – Comparison of some important characteristics of the SVR, LSSVR and FS-LSSVR models.

Characteristics	Models		
	SVR	LSSVR	FS-LSSVR
Error cost function	$\epsilon$ -insensitive	SSE	SSE
Obtaining the model solution	QP problem	Linear system	Linear system
Number of model variables	$2N + 1$	$N + 1$	$M + 1$
Computational complexity	$\mathcal{O}(N^3)$	$\mathcal{O}(N^3)$	$\mathcal{O}(NM^2)$
Memory demand	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(NM)$
Model hiperparameters (for Gaussian kernel)	$\epsilon, \gamma, \sigma$	$\gamma, \sigma$	$M, \gamma, \sigma$
Sparsity solution	YES	NO	YES
Parametric/Nonparametric model	Nonparametric	Nonparametric	Parametric

$M$  prototype vectors, which are responsible for the sparseness of the solution.

It is worth highlighting that specifically the LSSVR and FS-LSSVR models are going to be further used for the contributions of this thesis. As an additional remark, Table 2 summarizes some of the main characteristics of the SVR, LSSVR and FS-LSSVR models, discussed throughout this chapter.

Bearing in mind the kernel-based models theory, the next chapter will discuss the application of robust strategies to the kernel models, specially to the LSSVR, in order to minimize the effect of the presence of outliers in estimation data.

### 3 ROBUSTNESS IN LSSVR DERIVED MODELS

“Education is not the learning of the facts,  
but the training of the mind to think.”

(Albert Einstein)

This chapter discusses a framework of robust regression based on the M-Estimators and applied to kernel-based models, specifically those ones derived from the LSSVR approaches. First, Section 3.1 addresses the problem of the presence of outliers in estimation data, and some strategies to overcome this issue. Next, Section 3.2 presents the theoretical fundamentals of M-Estimators and, in Section 3.3, these estimators are used to develop robust versions of the LSSVR model, namely, the Weighted LSSVR (W-LSSVR) and Iteratively Reweighted LSSVR (IR-LSSVR) models. Then, Section 3.4 shows a computational experiment with outliers and their influence in these LSSVR derived models. Section 3.3 presents a brief discussion about different robust strategies applied to the LSSVR model. Finally, some important remarks of the chapter are mentioned in Section 3.6.

#### 3.1 Introduction

In real-world applications of parameter estimation for regression tasks, the available data are often contaminated by random noise. Some of its main sources are due to measurement errors, human mistakes, rounding errors, system failures and changes in the system set-point (ROUSSEEUW; LEROY, 2005; HAMPEL *et al.*, 2011), to name a few. In some extreme situations<sup>1</sup>, the noisy data may be of heavy-tail<sup>1</sup>, resulting in a different sample distribution than the desired normal (or Gaussian) distribution.

According to Bamnett and Lewis (1994), these observations, which appear to be inconsistent with the remainder of their original dataset, are called *outliers*. As defined by Barros (2013), an outlier is an observation (scalar or vector) which differs markedly from other observations of the sample to which it belongs, as indicated by the chosen model to represent this sample. Although there is no universal definition of outlier (HODGE; AUSTIN, 2004), the above definition is suitable for robust regression problems and, therefore, is adopted from now on.

---

<sup>1</sup>Heavy-tailed or fat-tailed distributions are those whose density tails tend to zero more slowly than the normal density tails (MARONNA; YOHAI, 2006).

In general, Stevens (1984) distinguished two type of outliers; the first one, which contaminates the independent variables  $\mathbf{x}$  (inputs), is called *leverage point* (ROUSSEEUW; LEROY, 1987). The second type usually contaminates the dependent variables  $y$  (outputs) and, therefore, is the type of outlier that one should bear in mind when treating regression problems in this thesis.

In order to fit an estimation model to contaminated data (either in  $\mathbf{x}$  or  $y$ ), one should adopt one of the following strategies: *regression diagnostic* or *robust regression* (ROUSSEEUW; LEROY, 2005). Diagnostic methods focus on detecting and removing the outliers to, in the next step, fit the “good data” by applying the standard least squares technique, for example. However, it would be misleading to always think of outliers as “bad data” (MARONNA; YOHAI, 2006), since they can contain unexpected relevant information or even the most important samples in the available data (BEN-GAL, 2005). Therefore, the action of simply removing them may not be the most appropriate decision in many practical situations, such as in Kandel (1992).

On the other hand, the robust regression approach seeks to develop estimators that are not so strongly affected by outliers. These estimators discover the outliers as those points which possess large residuals from that robust solution (ROUSSEEUW; LEROY, 2005). From robust regression, Huber *et al.* (1964) developed a class of estimators based on the robust statistics framework, known as *M-estimators*, where *M* stands for maximum likelihood-type. Simply stated, the *M-estimators* suggest minimizing a different cost function, instead of SSE, in order to obtain more robust estimates in the presence of outliers. Other important contributions in the field of robust regression, especially in *M-estimators*, are also found in the works developed by Tukey (1960) and Hampel (1971).

Although the *M-estimators* were originally proposed for linear and parametric regression tasks, their application in nonparametric kernelized models, e.g. LSSVR model, has aroused interest of the scientific community in the last two decades (SUYKENS *et al.*, 2002a; DEBRUYNE *et al.*, 2008; DE-BRABANTER *et al.*, 2009b; CHRISTMANN; STEINWART, 2007; DEBRUYNE *et al.*, 2010). These techniques apply weighting strategies (based on the *M-estimation*) to the errors obtained in the initial stage, in order to minimize the effects of the larger ones (caused by outliers) and, then, make their solutions robust. The LSSVR robust models can be useful in nonlinear regression tasks, such as time series prediction, control and system identification in the presence of outliers.

### 3.2 Fundamentals of the M-Estimators

Since linear regression is one of the most important statistical data analysis tools, one may initially consider a sequence of observations  $\{\mathbf{x}_n, y_n\}_{n=1}^N$ , where  $\mathbf{x}_n \in \mathbb{R}^d$  is the input vector and  $y_n \in \mathbb{R}$  is the corresponding output. In order to understand how the outputs  $y_n$  are related to the inputs  $\mathbf{x}_n$ , one can adopt the linear model

$$y_n = \mathbf{w}^\top \mathbf{x}_n + \tau_n, \quad (3.1)$$

where  $\mathbf{w} \in \mathbb{R}^d$  is the unknown parameter vector and  $\{\tau_n\}_{n=1}^N$  is a sequence of additive noise. An estimator  $\hat{\mathbf{w}}$  (for  $\mathbf{w}$ ) can be expressed by  $y_n = \hat{\mathbf{w}}^\top \mathbf{x}_n + e_n$ , where  $e_n = y_n - \hat{y}_n$  is the error made in approaching the real output  $y_n$  by its prediction  $\hat{y}_n = \hat{\mathbf{w}}^\top \mathbf{x}_n$ .

The most commonly used estimator for  $\mathbf{w}$  is the Ordinary Least Squares (OLS) criterion, initially proposed by Legendre (1805), then independently, by Gauss (1809). The OLS estimate minimizes the sum of squared errors, according to the following functional:

$$J(\hat{\mathbf{w}}) = \sum_{n=1}^N e_n^2 = \sum_{n=1}^N (y_n - \hat{\mathbf{w}}^\top \mathbf{x}_n)^2. \quad (3.2)$$

Under the assumption that the additive noise  $\tau_n$  in the regression model is normally distributed with zero mean and unknown variance  $\sigma_e^2$ , the OLS estimates are the most efficient unbiased estimates of  $\mathbf{w}$ . However, if this noise is not normally distributed, e.g. in the presence of impulsive noise or outliers, the OLS performance significantly reduces, since a single outlier can spoil its estimates completely.

The M-Estimation framework was proposed by Huber *et al.* (1964) as an alternative to overcome this situation. In short, the M-estimators replace the cost function based on SSE by a robust version that is not as vulnerable as the least squares to the presence of outliers. Then, based on the Huber theory, a general M-estimator minimizes the following functional (FOX, 2002):

$$J(\hat{\mathbf{w}}) = \sum_{n=1}^N \rho(e_n) = \sum_{n=1}^N \rho(y_n - \hat{\mathbf{w}}^\top \mathbf{x}_n), \quad (3.3)$$

where the loss function  $\rho(\cdot)$  computes a contribution of each prediction error  $e_n$  to the functional  $J(\cdot)$ . One should note that the standard OLS estimator is achieved if one sets  $\rho(\cdot) = \|\cdot\|^2$ . A reasonable choice for  $\rho(\cdot)$  should have the following properties (FOX, 2002):

- $\rho(e) \geq 0$ ;
- $\rho(0) = 0$ ;

- $\rho(e) = \rho(-e)$ ;
- $\rho(e_i) \geq \rho(e_j)$  for  $|e_i| > |e_j|$ .

In order to minimize  $J(\hat{\mathbf{w}})$  in Eq. (3.3), an optimal solution vector  $\mathbf{w}$  is searched by computing the gradient vector  $\nabla_{\hat{\mathbf{w}}} J(\hat{\mathbf{w}}) = \partial J(\hat{\mathbf{w}}) / \partial \hat{\mathbf{w}}$  and equaling it to zero, which is given by

$$\frac{\partial J(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}} = \sum_{n=1}^N \frac{\partial \rho(e_n)}{\partial e_n} \frac{\partial e_n}{\partial \hat{\mathbf{w}}} = \sum_{n=1}^N \psi(e_n) \frac{\partial e_n}{\partial \hat{\mathbf{w}}} = \mathbf{0}, \quad (3.4)$$

where  $\psi(e) = d\rho(e)/de$  is a bounded, continuous, differentiable and odd function, usually called *score function* (DEBRUYNE ANDREAS CHRISTMANN; SUYKENS, 2010). In the OLS case,  $\rho(e) = e^2/2$  and  $\psi(e) = e$ , i.e. the influence of certain data linearly increases with the magnitude of its corresponding prediction error, which confirms the non-robustness of the least squares estimation (ZHANG, 1997; FREIRE, 2015).

If one defines a weight function for each error  $e_n$  as

$$v(e_n) = \frac{1}{e_n} \frac{d\rho(e_n)}{de} = \frac{\psi(e_n)}{e_n}, \quad (3.5)$$

where  $e_n = y_n - \hat{\mathbf{w}}^\top \mathbf{x}_n$  and, therefore,  $\partial e_n / \partial \hat{\mathbf{w}} = -\mathbf{x}_n$ , it is possible to rewrite Eq. (3.4) as

$$\sum_{n=1}^N v(e_n) (y_n - \hat{\mathbf{w}}^\top \mathbf{x}_n) \mathbf{x}_n = \mathbf{0}. \quad (3.6)$$

The system of equations in Eq. (3.6) corresponds to the weighted least squares problem, which minimizes a weighted norm of the error (BJÖRCK, 1996; BEN-ISRAEL; GREVILLE, 2003), expressed by

$$J(\hat{\mathbf{w}}) = \mathbf{e}^\top \mathbf{V} \mathbf{e} = \sum_{n=1}^N v_n e_n^2, \quad (3.7)$$

where  $\mathbf{e} \in \mathbb{R}^N$  is a vector formed by the errors  $e_n$ , and  $\mathbf{V} \in \mathbb{R}^{N \times N}$  is a diagonal matrix with the weights  $v_n$  along its diagonal. Nevertheless, one should note that the weights depend upon the errors, the errors depend upon the estimated coefficients, and the estimated coefficients depend upon the weights (FOX, 2002). Consequently, a closed-form equation for estimating  $\hat{\mathbf{w}}$  is not available. Thus, an iterative solution, named *Iteratively Reweighted Least Squares* (IRLS) algorithm (FOX, 1997), is required. The pseudo-code is shown in Algorithm 3, where  $\Delta$  is a small convergence threshold.

### 3.2.1 Objective and Weighing Functions

In order to apply the M-estimators in regression problems, many weight functions have been proposed in the literature, especially for linear regression (ROUSSEEUW; LEROY,

---

**Algorithm 3:** - Pseudo-code for the IRLS algorithm.

---

**Require:** Initial least square estimate  $\hat{\mathbf{w}}^{(0)}$  from Eq. (2.30);  
 Set the convergence threshold  $\Delta$ ;  
**while**  $\max(|\hat{\mathbf{w}}^{(i)} - \hat{\mathbf{w}}^{(i-1)}|) > \Delta$  **do**  
   **for**  $n = 1 : N$ , **do**  
     Compute  $e_n^{(i-1)} = y_n - \hat{\mathbf{w}}^{(i-1)\top} \mathbf{x}_n$  and the respective  $v_n^{(i-1)}$ ;  
   **end for**  
 Compute  $\hat{\mathbf{w}}^{(i)} = [\mathbf{X}^\top \mathbf{V}^{(i-1)} \mathbf{X}]^{-1} \mathbf{X}^\top \mathbf{V}^{(i-1)} \mathbf{y}$ ;  
 $i=i+1$ ;  
**end while**  
 Output  $\hat{\mathbf{w}}$ .

---

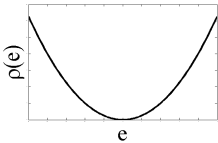
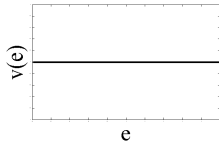
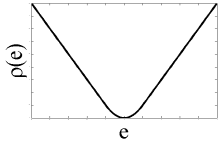
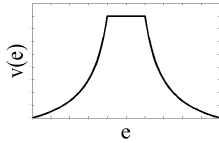
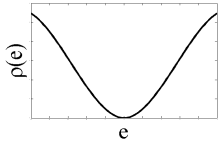
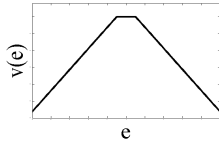
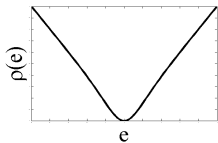
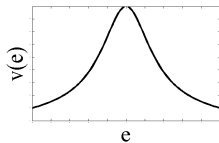
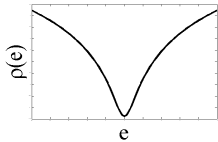
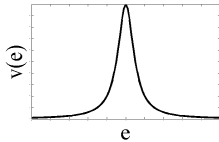
1987). As they are fundamental to this work, five of these functions, OLS, Huber, Hampel, Logistic and Myriad, are shown in Table 3, with the corresponding loss functions  $\rho(\cdot)$ , weight functions  $v(\cdot)$  and their respective graphics.

Analyzing the weight functions in Table 3, one can easily see that the OLS estimator assigns unit weight to each prediction error. Therefore, its performance significantly decreases in the presence of outliers because, although the errors caused by the outliers are larger than those ones caused by the other input samples, the former are treated with the same importance as the latter. On the other hand, for the Huber function (HUBER *et al.*, 1964), for instance, the weight values decrease when the errors lie beyond a certain threshold,  $|e| > \hat{s}$ . In this case, the Huber estimator reduces the weights of the errors caused by the outliers and, consequently, decreasing their influence. In addition, the errors within the defined threshold ( $|e| \leq \hat{s}$ ) are treated exactly as the OLS estimator.

The above mentioned  $\hat{s}$  is a positive parameter to be adjusted, and is known as the error (outlier) threshold or tuning constant. Smaller values of  $\hat{s}$  produce more resistance to outliers, but at the expense of lower efficiency when the errors are normally distributed (FOX, 2002; BARROS; BARRETO, 2013). According to Fox (2002), this tuning constant is usually chosen to give reasonable high efficiency in the normal case. For example, one has  $\hat{s} = 1.345\sigma_e$  for the Huber function and  $\hat{s} = 1.960\sigma_e$  for the Hampel function (ROUSSEUW; LEROY, 1987; DAVID, 1998), where  $\sigma_e$  is the standard deviation of the errors, in order to produce 95% of efficiency when the errors are normally distributed, while still offering protection against outliers (FOX, 1997).

In practical situations, it is necessary to compute a robust estimate for the standard

Table 3 – Examples of the objective  $\rho(\cdot)$  and weight  $v(\cdot)$  functions for the M-estimators: OLS, Huber, Hampel, Logistic and Myriad.

M-estimators	Functions	
	Objective $\rho(e)$	Weight $v(e)$
OLS	$\frac{e^2}{2}$ 	$1$ 
Huber	$\begin{cases} e^2/2, &  e/\hat{s}  \leq 1, \\ \hat{s} e  - \frac{1}{2}\hat{s}^2, &  e/\hat{s}  > 1. \end{cases}$ 	$\begin{cases} 1, &  e/\hat{s}  \leq 1, \\ \hat{s}/ e , &  e/\hat{s}  > 1. \end{cases}$ 
Hampel	$\begin{cases} e^2/2, &  e/\hat{s}  \leq c_1, \\ \frac{c_2 e/\hat{s} ^2 -  e/\hat{s} ^3}{c_2 - c_1}, & c_1 <  e/\hat{s}  \leq c_2, \\ 10^{-4}, &  e/\hat{s}  > c_2. \end{cases}$ 	$\begin{cases} 1, &  e/\hat{s}  \leq c_1, \\ \frac{c_2 -  e/\hat{s} }{c_2 - c_1}, & c_1 <  e/\hat{s}  \leq c_2, \\ 10^{-4}, &  e/\hat{s}  > c_2. \end{cases}$ 
Logistic	$\frac{\tanh(e)}{e}$ 	$e \tanh(e)$ 
Myriad	$\frac{\delta^2}{\delta^2 + e^2}$ 	$\log(\delta^2 + e^2)$ 



deviation of the error variables  $e_n$  as

$$\hat{\sigma} = \frac{\text{MAR}}{0.6745}, \quad (3.8)$$

for the Huber function, where MAR is the median of the absolute values of the errors. Already, for the Hampel function, one gets

$$\hat{\sigma} = \frac{\text{IQR}}{1.3490}, \quad (3.9)$$

where the *interquatile range* IQR is the difference between the 75-th percentile and the 25-th percentile. The  $\hat{\sigma}$  estimate takes into account how much the estimated error distribution deviates from a Gaussian distribution. The constant values 0.6745 (for the Huber function) and 1.3490 (for the Hampel function) make  $\hat{\sigma}$  an unbiased estimate for Gaussian errors (BARROS, 2013). Another feasible robust estimation of the standard deviation is  $\hat{\sigma} = 1.483\text{MAD}(\mathbf{e})$ , where MAD stands for the median absolute deviation (HAMPEL *et al.*, 2011).

Besides the robust estimate  $\hat{\sigma}$ , there are two additional parameters,  $c_1$  and  $c_2$ , for the Hampel objective function, as can be seen in Table 3. According to Rousseeuw and Leroy (1987), the values  $c_1 = 2.5$  and  $c_2 = 3.0$  are reasonable choices, whereas for a Gaussian distribution, there will be very few values larger than  $2.5\hat{\sigma}$  (SUYKENS *et al.*, 2002a). Another possibility is computing  $c_1, c_2$  from a density estimation of the  $e_n$  distribution.

The last two objective functions shown in Table 3 are the Logistic and Myriad. The Logistic function is often used in regression problems (ROUSSEEUW; LEROY, 2005; DEBRUYNE *et al.*, 2008), but with no tuning parameters. In general, it is a good choice between convergence and robustness (DE-BRABANTER *et al.*, 2009b). In turn, the Myriad function with parameter  $\delta \in \mathbb{R}_0^+$  has been proposed in the field of statistical nonlinear signal processing (ARCE, 2005). It is derived from the maximum likelihood estimation of a Cauchy distribution and is used as a robust location estimator in stable noise environments. When the noise is Gaussian, large values of the parameter  $\delta$  can provide the optimal performance associated with the sample mean, whereas for highly impulsive noise statistics, the resistance of mode-type estimators can be achieved by setting low values of  $\delta$  (DE-BRABANTER *et al.*, 2009b).

Some other common choices of the weight functions for M-estimators, such as Andrews, Bisquare, Cauchy, Fair, Talwar and Welsch can be found in Barros (2013). From the next section moving forward, the M-estimators theory will be applied to the standard LSSVR model in order to robustify its solution in the presence of outliers.

### 3.3 Robustness in LSSVR Models

As outlined in Section 3.2 for the OLS estimates, instead of using robust cost functions, one can also obtain a robust estimate of the LSSVR model based upon its previous solution and a suitable weighting procedure. In this sense, considering again the training data  $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ , with  $\mathbf{x}_n \in \mathbb{R}^d$  and  $y_n \in \mathbb{R}$ , Suykens *et al.* (2002a) proposed the *Weighted LSSVR* (W-LSSVR) model, which is formulated by the minimization of the following functional:

$$\min_{\mathbf{w}, b, \mathbf{e}} J_p(\mathbf{w}, \mathbf{e}) = \frac{1}{2} \|\mathbf{w}\|^2 + \gamma \frac{1}{2} \sum_{n=1}^N v_n e_n^2, \quad (3.10)$$

$$\text{subject to } \begin{cases} y_n = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b + e_n, \text{ for } n = 1, \dots, N, \end{cases} \quad (3.11)$$

where  $\mathbf{w}$  is the parameter vector,  $\gamma$  is the regularization parameter,  $\boldsymbol{\phi}$  is the nonlinear map and  $b$  is the bias. The only difference between the robust formulation in Eqs. (3.10)-(3.11) and the standard LSSVR model in Eqs. (2.21)-(2.22), is that the error variables  $e_n$  from the unweighted LSSVR are now weighted by a vector  $\mathbf{v} = [v_1, \dots, v_N]^\top$ , computed in this case according to the Hampel's function, as shown in Table 3.

The equivalent dual problem for the primal formulation in Eqs. (3.10)-(3.11) is obtained by constructing the Lagrangian as

$$\mathcal{L}(\mathbf{w}, b^r, \mathbf{e}, \boldsymbol{\alpha}^r) := \frac{1}{2} \|\mathbf{w}\|^2 + \gamma \frac{1}{2} \sum_{n=1}^N v_n e_n^2 - \sum_{n=1}^N \alpha_n^r (\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b^r + e_n - y_n), \quad (3.12)$$

where  $\alpha_n^r$  denote the Lagrange multipliers of the robust solution and  $b^r$  is its bias. As in the LSSVR formulation (see Chapter 2), from the Karush-Kuhn-Tucker (KKT) (FLETCHER, 2013) conditions for optimality and elimination of  $\mathbf{w}$  and  $\mathbf{e}$ , one gets the following linear system of equations:

$$\begin{bmatrix} 0 & \mathbf{1}_N^\top \\ \mathbf{1}_N & \mathbf{K} + \gamma^{-1} \mathbf{V} \end{bmatrix} \begin{bmatrix} b^r \\ \boldsymbol{\alpha}^r \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix}, \quad (3.13)$$

where  $\boldsymbol{\alpha}^r = [\alpha_1^r, \dots, \alpha_N^r]^\top$ ,  $\mathbf{y} = [y_1, \dots, y_N]^\top$  and  $\mathbf{K} \in \mathbb{R}^{N \times N}$  is the kernel matrix. The diagonal matrix  $\mathbf{V} \in \mathbb{R}^{N \times N}$  is given by

$$\mathbf{V} = \text{diag} \left\{ \frac{1}{v_1}, \dots, \frac{1}{v_N} \right\}, \quad (3.14)$$

where each weight  $v_n$  is determined based on the error variables  $e_n = \alpha_n / \gamma$ , from the original (unweighted) LSSVR model.

Finally, the linear system in Eq. (3.13) can be solved by the Moore-Penrose inverse using Eq. (2.30). Then, the W-LSSVR model solution is given by

$$f(\mathbf{x}) = \sum_{n=1}^N \alpha_n^r k(\mathbf{x}, \mathbf{x}_n) + b^r. \quad (3.15)$$

As in the LSSVR case, the W-LSSVR solution is not sparse, since all the training samples are used as support vectors to make new predictions. Furthermore, the computational complexity of the W-LSSVR model is  $\mathcal{O}(N^3)$  and its memory demand is  $\mathcal{O}(N^2)$ , which are the same ones as those in the standard LSSVR model. The W-LSSVR model is summarized in Algorithm 4.

---

**Algorithm 4:** - Pseudo-code for the W-LSSVR model.

---

**Require:** Set  $\gamma$  and the kernel hiperparameters ( $\sigma$  for the Gaussian kernel, for example)

  Compute the kernel matrix  $\mathbf{K}$ ;

  Build the linear system of Eq. (2.29);

  Compute  $\boldsymbol{\alpha}$ ,  $b$  from Eq. (2.30), and  $e_n = \alpha_n/\gamma$ ; % unweighted LSSVR solution

  Compute  $\hat{s}$  from the  $e_n$  distribution in Eq. (3.9);

  Determine  $v_n$  from the Hampel's function in Table 3;

  Build the linear system of Eq. (3.13);

  Compute  $\boldsymbol{\alpha}^r$  and  $b^r$  from Eq. (2.30);

  Output  $\boldsymbol{\alpha}^r$ ,  $b^r$  % W-LSSVR solution.

---

### 3.3.1 The IR-LSSVR Model

In practice, Suykens *et al.* (2002a) claim that one single additional weighted LSSVR step is often sufficient in most cases. However, in order to also obtain a robust estimate, this weighting procedure can be iteratively repeated, giving rise to the *Iteratively Reweighted* LSSVR (IR-LSSVR) model (DE-BRABANTER *et al.*, 2009b). At each iteration  $i$ , one can weight the error variables  $e_n^{(i)} = \alpha_n^{(i)}/\gamma$ , for  $n = 1, \dots, N$ . Again, the weights  $v_n^{(i)}$  are calculated based upon  $e_n^{(i)}/\hat{s}$ , using the Hampel function.

Thus, one solves the linear system in Eq. (3.13) for each iteration  $i$  as

$$\begin{bmatrix} 0 & \mathbf{1}_N^\top \\ \mathbf{1}_N & \mathbf{K} + \gamma^{-1} \mathbf{V}^{(i)} \end{bmatrix} \begin{bmatrix} b^{r(i)} \\ \boldsymbol{\alpha}^{r(i)} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix}, \quad (3.16)$$

where

$$\mathbf{V}^{(i)} = \text{diag} \left\{ \frac{1}{v_1^{(i)}}, \dots, \frac{1}{v_N^{(i)}} \right\}. \quad (3.17)$$

The resulting model after the  $i$ -th iteration is given by

$$f^{(i)}(\mathbf{x}) = \sum_{n=1}^N \alpha_n^{r(i)} k(\mathbf{x}, \mathbf{x}_n) + b^{r(i)}. \quad (3.18)$$

Following, one sets  $i = i + 1$  and the new weights  $v_n^{(i+1)}$ , the vector  $\boldsymbol{\alpha}^{r(i+1)}$  and the bias  $b^{r(i+1)}$  are calculated. This iterative procedure should continue until  $\max_n (|\alpha_n^{r(i)} - \alpha_n^{r(i-1)}|) \leq \Delta$ , where  $\Delta$  is a convergence threshold pre-established by the user. This seeks to ensure that consecutive estimates  $\alpha_n^{r(i-1)}$  and  $\alpha_n^{r(i)}$  are sufficiently close to each other  $\forall n = 1, \dots, N$ . The pseudo-code for the IR-LSSVR model is summarized in Algorithm 5.

### 3.4 Example of the Influence of Outliers

In order to verify the outliers influence over the model estimation in nonlinear regression, one considers now an example of the 1-dimensional sinc function  $y = \sin(\pi x)/\pi x$ , defined on the interval  $[-5, 5]$ . The training outputs were corrupted by additive Gaussian noise with zero-mean and a standard deviation of 0.05, as illustrated by the blue circles in Fig (5a). Moreover, the predicted output for the standard unweighted LSSVR model (red curve) also can be seen in Fig (5a), where one may observe that its generalization performance is, in general, appropriate for this case.

In the next step, the training outputs were purposely contaminated with two outliers, which are represented by the black asterisks in Fig (5b). Due to the addition of these outliers, the difference between the LSSVR performances in Figs. (5a) and (5b) is remarkable. In other words, the outliers produce large prediction errors and, consequently, draw the regression curve towards them. This effect distorts the LSSVR predictions regarding the other training samples, as can be seen in Fig. (5b) by the data points below both outliers and the LSSVR predicted output.

The robustness of the W-LSSVR and IR-LSSVR approaches are presented in Figs. (5c) and (5d), respectively, considering the scenario with the outliers. Observing these figures, it is clear that the generalization performance obtained by the standard LSSVR model was further improved by applying the weighted procedure. This can be explained by the fact that, for the robust models (W-LSSVR and IR-LSSVR), the large errors are compensated with weights of small magnitude, minimizing the effect of the outliers. For this case, it is also possible to see in Figs. (5c) and (5d) that the W-LSSVR and IR-LSSVR models achieved performances close to each other.

---

**Algorithm 5:** - Pseudo-code for the IR-LSSVR model.

---

**Require:**  $\gamma, \Delta, \sigma$  (for the Gaussian kernel);

Compute  $e_n^{(0)} = \alpha_n^{(0)} / \gamma$  from the standard unweighted LSSVR model;

**while**  $\max_n(|\alpha_n^{r(i-1)} - \alpha_n^{r(i)}|) > \Delta$ , **do**

    Compute  $\hat{s} = 1.483\text{MAD}(e_n^{(i)})$  from the  $e_n^{(i)}$  distribution;

    Determine  $v_n$  from the Hampel's function in Table 3;

    Build the linear system of Eq. (3.13);

    Compute  $\alpha^{r(i)}$  and  $b^{r(i)}$  from Eq. (2.30);

    Set  $i = i + 1$ ;

**end while**

Output  $\alpha^{r(i)}, b^{(i)}$  % IR-LSSVR solution.

---

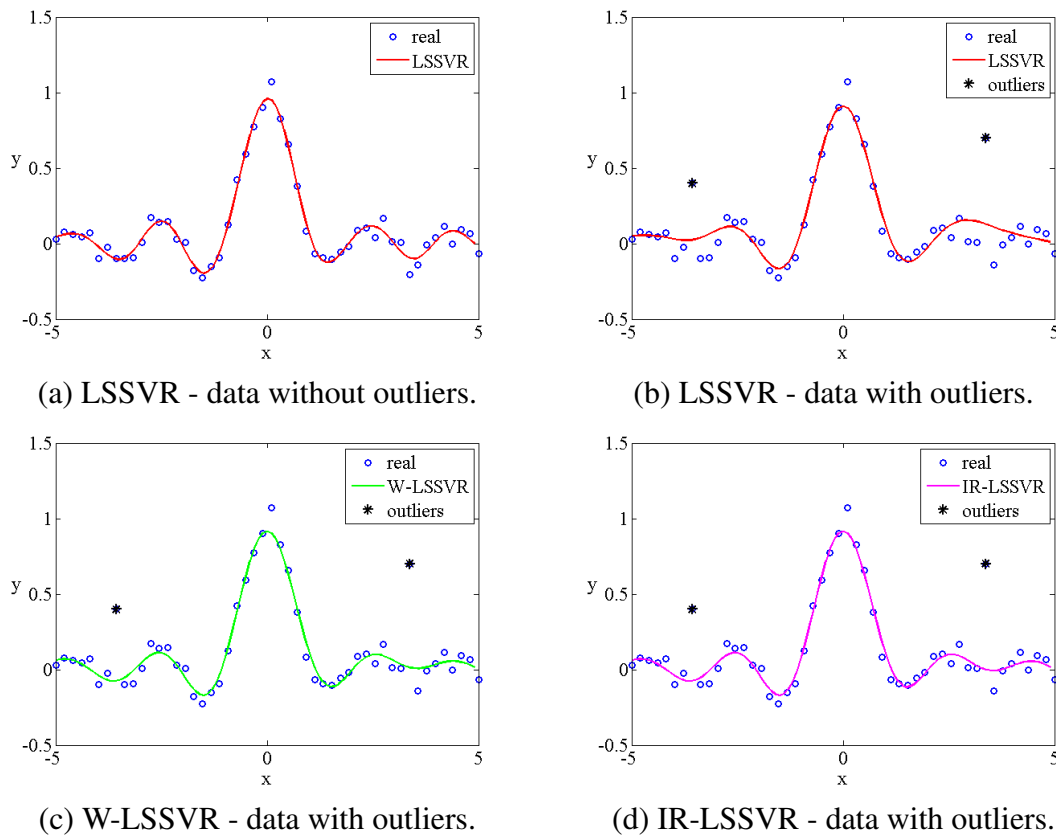


Figure 5 – Comparison between the standard LSSVR, W-LSSVR and IR-LSSVR models for estimating the sinc function in the presence of outliers.

### 3.5 Some Other Robust LSSVR Approaches

When observing the W-LSSVR and IR-LSSVR strategies, it is clear that robustness in the nonparametric LSSVR model can be achieved by using the least squares cost function by means of weighting or iterative reweighting procedures. Thus, besides working with a convex optimization problem solved by a simple linear system of equations, it was showed by Debruyne *et al.* (2010) that this reweighting methodology does not only improve robustness

against outliers, but also improves the stability of the LSSVR derived models, especially at heavy tailed distributions.

In this context, some authors have devoted attention to developing weighted LSSVR models to mitigate the influence of outliers. For instance, Chen *et al.* (2015) presented a two-stage robust weighted LSSVR approach based first on the Least Trimmed Squares (LTS) technique, in order to obtain simulation results at the cost of losing statistical efficiency to some extent. Then, in the second stage, a weighted LSSVR model is employed to optimize these simulation results. Wen *et al.* (2008) already proposed an iterative and heuristic weight setting algorithm, derived from the idea of outlier mining, independent of the unweighted standard LSSVR model at an early stage.

Following the same line of the weighting procedure, Quan *et al.* (2010) designed a robust algorithm for nonlinear time series prediction using the W-LSSVR strategy and local models. In the methodology proposed by Valyon and Horváth (2006), a geometric view of the LSSVR was presented in order to obtain a sparse solution and reduce the effect of non Gaussian noise, with the aid of the weighted LSSVR model.

Many researchers argued that first the outliers should be filtered out, by some advanced techniques, and then the standard LSSVR should be used to train the remaining data samples. In this sense, Wen *et al.* (2010) implemented a qualitative criterion derived from the LTS algorithm, based on robust linear regression, for eliminating the outliers within a recursive framework. Similarly, Cui and Yan (2009) combined strategies of outlier detection and adaptive weighted LSSVR algorithm for the training samples. Motivated by the Vapnik's theory (VAPNIK, 1998), Chuang and Lee (2011) proposed a hybrid strategy with which to treat outliers, where a data preprocessing stage is performed, by using the SVR model to filter out the outliers in the training dataset. Then, a standard LSSVR model is used with the data samples without outliers.

In order to avoid setting weights, some authors preferred to use robust cost functions. For example, Yang *et al.* (2014) presented a robust LSSVR variant based on a truncated least squares loss function, which is neither differentiable nor convex. Therefore, the authors proposed an iterative algorithm based on the concave-convex procedure (CCCP) (YUILLE; RANGARAJAN, 2003) and the Newton algorithm (CHAPELLE, 2007) to solve the resulting optimization problem. On the other hand, Wang *et al.* (2014) proposed an absolute deviation loss function to derive a robust regression model termed as Least Absolute Deviation SVR (LAD-SVR). Since its proposed loss function is not differentiable, it was necessary to approximate it by constructing

a smooth function and developing a Newton algorithm to solve the optimization problem.

Derived from the information theoretic learning (ITL) (PRÍNCIPE, 2010), Chen *et al.* (2012) built a regression model in the kernel space based on the maximum correntropy criterion and regularization technique. An iterative algorithm derived from the half-quadratic optimization was developed to solve the problem with theoretically guaranteed convergence. Feng *et al.* (2015) conducted a detailed analysis of the connections between the regression model associated with the correntropy induced loss and the least squares regression model, as well as the study of its convergence property.

### 3.6 Concluding Remarks

The present chapter presented an overview about robust regression techniques applied to parameter estimation problems. These evaluated techniques are based on the M-estimators theory, which replace the SSE cost function by a more general version, that minimizes the effect of outliers by assigning to them small weights for their large errors. In addition, the W-LSSVR and IR-LSSVR models were discussed, since they are the state-of-the-art in robust approaches derived from the standard LSSVR model and the M-estimators. Some possible choices for their objective and weight functions, such as OLS, Huber, Hampel, Logistic and Myriad, were briefly described throughout the chapter.

This chapter also presented a numerical example of the influence of outliers in nonlinear regression, specifically in function approximation, as proof of concept. For this purpose, the sinc function was used, and was purposely contaminated with outliers. In this case, the degradation of the standard LSSVR model when the training output were corrupted with outliers can be seen. On the other hand, the W-LSSVR and IR-LSSVR models were less affected in the presence of outliers, achieving satisfactory performances.

Finally, some other developed strategies to robustify the LSSVR model were commented. Notably, some of them are first interested in filtering the outliers and training a non robust model with the remaining data. Other models are based on weighting procedures similar to those of W-LSSVR and IR-LSSVR models. Moreover, some of them prefer to use robust cost functions instead of those based on SSE. The latter usually occurs in non convex optimization problems, whose resolution procedures are quite tedious.

The next chapter will discuss the theory of the kernel adaptive filtering methods applied to nonlinear signal processing problems.

## 4 ONLINE KERNEL-BASED MODELS

“Have no fear of perfection, you’ll never reach it.”

(Salvador Dali)

This chapter presents an overview of kernel adaptive filtering models, which are direct implementations of linear adaptive filters in feature space. Initially, Section 4.1 contextualizes the issue of online learning for solving nonlinear problems, by using kernelized versions of linear adaptive algorithms. Next, Section 4.2 presents a brief theory of linear adaptive filtering, with an emphasis on the LMS and RLS algorithms. Then, the respective online kernel versions of the LMS (KLMS) and RLS (KRLS) algorithms are treated in detail in Section 4.3. Section 4.4 discusses some sparsification criteria, different from the approximate linear dependency (ALD) procedure, as defined for the KRLS model. Next, Section 4.5 briefly describes the family of the Kernel Affine Projection Algorithms (KAPA), as an alternative to the original KLMS and KRLS models. Finally, the chapter is concluded in Section 4.6.

### 4.1 Introduction

The kernel-based models, as SVR (VAPNIK, 1995; VAPNIK, 1998), LSSVR (SAUNDERS *et al.*, 1998; SUYKENS *et al.*, 2002b) gaussian processes (RASMUSSEN, 1996) and regularization networks (GIROSI *et al.*, 1995), have achieved considerable success in nonlinear regression problems in the batch learning scenario, i.e., when all of the training data are available in advance.

However, batch learning is a limiting factor impairing widespread use of the above models to suitably handle the challenges when, for instance, the amount of data is too high to apply batch algorithms, where the respective solution has to be updated sequentially to account for all processed data. A second example is when the scenario that requires online updating of the solution occurs in dynamic environments, where the model solution changes over time.

In this scenario, the field of Kernel Adaptive Filtering (KAF) has become a powerful tool in solving nonlinear signal processing problems (LIU *et al.*, 2009a), since it consists of the application of the kernelization strategy, common to the support vector theory, to linear adaptive filters producing their nonlinear versions, including the Kernel Least Mean Squares (KLMS) (LIU *et al.*, 2008), Kernel Recursive Least Squares (KRLS) (ENGEL *et al.*, 2004),



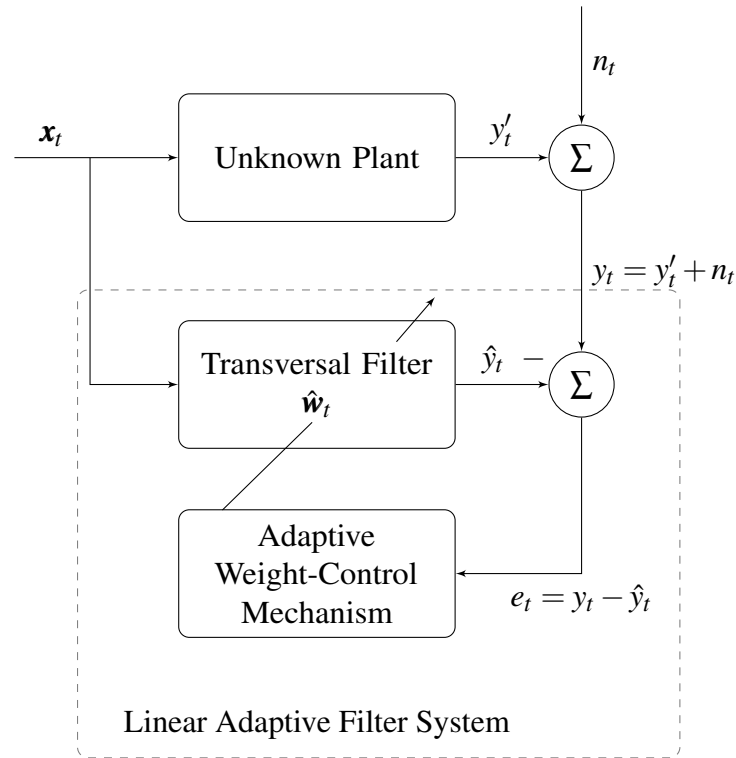


Figure 6 – Basic configuration of a linear adaptive filter system.

EXtended Kernel Recursive Least Squares (EX-KRLS) (LIU *et al.*, 2009b; ZHU *et al.*, 2012) and Kernel Affine Projections Algorithms (KAPA) (LIU; PRÍNCIPE, 2008).

Those KAF algorithms generalize the classical linear adaptive filters in RKHS, thus, combining the adaptive characteristics of these linear filters with the capabilities of the kernel methods in solving nonlinear problems by means of a convex learning process with no local minima. In addition, they aim to sequentially search a function  $f(\cdot)$  that maps a series of input patterns  $\{\mathbf{x}_t\}_{t=1}^N$  onto their respective desired images  $\{y_t\}_{t=1}^N$ , where  $N$  is the amount of available training instances, maintaining its computational load as moderate during each iteration  $t$ . This means that the amount of computations required per new sample must not necessarily increase as the number of samples increases.

## 4.2 Linear Adaptive Filtering

In general, the problem in linear filtering consists of adjusting the parameters of the filter as time advances, in order to adapt their performance according to some prespecified criterion (cost function), mostly driven by an error signal. Adaptive filters that use the *mean squared-error* (MSE) as cost function are the most common in practice, since the involved mathematical complexities are relatively easy to handle. In addition, according to Haykin (2013),

a filter is said to be linear if the filtered, smoothed or predicted quantity at the output of the device is a linear function of the observations applied to the filter input.

The basic configuration of a linear adaptive filter is illustrated in Fig. (6), where the index  $t$  denotes the time step,  $\mathbf{x}_t \in \mathbb{R}^d$  is the input signal,  $y_t' \in \mathbb{R}$  is the true (noiseless) output signal and  $y_t \in \mathbb{R}$  is the observed output signal, which is contaminated with an additive source of noise  $n_t$ . Moreover, the signal  $\hat{y}_t = \mathbf{x}_t^\top \hat{\mathbf{w}}_t \in \mathbb{R}$  is the adaptive filter output and  $\hat{\mathbf{w}}_t \in \mathbb{R}^d$  is the unknown parameter vector at time  $t$ . The prediction error is given by  $e_t = y_t - \hat{y}_t$  and it acts, in turn, as a guide to adapt the weights of  $\hat{\mathbf{w}}_t$  by an incremental adjustment. Therefore, the values of the transversal filter  $\hat{\mathbf{w}}_t$  are computed by an adaptive weight-control mechanism, according to the MSE criterion. Normally, the algorithm starts with an initial guess of  $\hat{\mathbf{w}}_t$ , which may be based on some available information about the system and, then, refines its solution in successive iterations, such that each refinement tends to improve its performance in terms of the chosen cost function.

#### 4.2.1 Wiener Filter

Initially, consider an adaptive linear filter of length  $d$ , whose output at the instant  $t$  is given as

$$\hat{y}_t = \sum_{i=0}^{d-1} \hat{w}_t^{(i)} x_{t-i} = \hat{\mathbf{w}}_t^\top \mathbf{x}_t, \quad (4.1)$$

where  $\mathbf{x}_t = [x_t, x_{t-1}, \dots, x_{t-d+1}]^\top$  and  $\hat{\mathbf{w}}_t = [\hat{w}_t^0, \hat{w}_t^1, \dots, \hat{w}_t^{d-1}]^\top$ . Then, the prediction error  $e_t = y_t - \hat{y}_t$  is used to minimize the following MSE cost function:

$$J_{\hat{\mathbf{w}}_t}(e_t) = E[e_t^2], \quad (4.2)$$

where  $E[\cdot]$  denotes the statistical expectation operator. Also considering a filter with fixed coefficients in a stationary environment, the functional in Eq. (4.2) can be rewritten as (DINIZ, 2008)

$$\begin{aligned} J_{\hat{\mathbf{w}}_t}(e_t) &= E[(y_t - \hat{\mathbf{w}}_t^\top \mathbf{x}_t)^2], \\ &= E[y_t^2 - 2y_t \hat{\mathbf{w}}_t^\top \mathbf{x}_t + \hat{\mathbf{w}}_t^\top \mathbf{x}_t \mathbf{x}_t^\top \hat{\mathbf{w}}_t], \\ &= E[y_t^2] - 2E[y_t \hat{\mathbf{w}}_t^\top \mathbf{x}_t] + E[\hat{\mathbf{w}}_t^\top \mathbf{x}_t \mathbf{x}_t^\top \hat{\mathbf{w}}_t], \\ &= E[y_t^2] - 2\hat{\mathbf{w}}_t^\top E[y_t \mathbf{x}_t] + \hat{\mathbf{w}}_t^\top E[\mathbf{x}_t \mathbf{x}_t^\top] \hat{\mathbf{w}}_t, \\ &= E[y_t^2] - 2\hat{\mathbf{w}}_t^\top \mathbf{p}_t + \hat{\mathbf{w}}_t^\top \mathbf{R}_t \hat{\mathbf{w}}_t, \end{aligned} \quad (4.3)$$

where  $\mathbf{p}_t = E[y_t \mathbf{x}_t] \in \mathbb{R}^d$  is a vector of cross-correlation coefficients between the desired output and input signals, and  $\mathbf{R}_t = E[\mathbf{x}_t \mathbf{x}_t^\top] \in \mathbb{R}^{d \times d}$  is the input signal correlation matrix. In order to search for the optimal vector  $\mathbf{w}_t^{opt}$ , that minimizes the functional in Eq. (4.3), one should compute the gradient vector of  $J_{\hat{\mathbf{w}}_t}$  as

$$\nabla_{\hat{\mathbf{w}}_t} J_{\hat{\mathbf{w}}_t}(e_t) = \frac{\partial J_{\hat{\mathbf{w}}_t}(e_t)}{\partial \hat{\mathbf{w}}_t} = -2\mathbf{p}_t + 2\mathbf{R}_t \hat{\mathbf{w}}_t. \quad (4.4)$$

By setting the gradient vector in Eq. (4.4) to zero and assuming that  $\mathbf{R}_t$  is nonsingular, the optimal solution is given by

$$\mathbf{w}_t^{opt} = \mathbf{R}_t^{-1} \mathbf{p}_t, \quad (4.5)$$

where  $\mathbf{w}_t^{opt}$  is called the *Wiener solution*. One should note that, to obtain the Wiener solution in Eq. (4.5), it is necessary to know exactly the matrix  $\mathbf{R}_t^{-1}$  and the vector  $\mathbf{p}_t$ . However, in practice, one does not have access to the statistics  $E[\cdot]$  to compute them, since only sample values of the input  $\mathbf{x}_t$  and the output  $y_t$  signals are available. An alternative to overcome this issue is solving the filtering problem iteratively, by applying the steepest descent method (HAYKIN, 2013) as

$$\begin{aligned} \hat{\mathbf{w}}_{t+1} &= \hat{\mathbf{w}}_t - \mu \nabla_{\hat{\mathbf{w}}_t} J_{\hat{\mathbf{w}}_t}(e_t), \\ &= \hat{\mathbf{w}}_t + 2\mu(\hat{\mathbf{p}}_t - \hat{\mathbf{R}}_t \hat{\mathbf{w}}_t), \end{aligned} \quad (4.6)$$

where  $\hat{\mathbf{p}}_t$  and  $\hat{\mathbf{R}}_t$  are estimates for  $\mathbf{p}_t$  and  $\mathbf{R}_t$ , respectively, and  $\mu$  is the *step-size parameter* (or learning step) for updating the weights of  $\hat{\mathbf{w}}_t$ . This parameter determines the stability, convergence speed and steady-state misalignment of the algorithm.

#### 4.2.2 Least Mean Square Algorithm

The simplest and most commonly choice for linear adaptive filtering is the LMS algorithm (WIDROW; HOFF, 1960), which replaces the MSE cost function in Eq. (4.2) with another version, based on the instantaneous squared error, expressed in an online manner by

$$J_{\hat{\mathbf{w}}_t}(e_t) = e_t^2. \quad (4.7)$$

This means that the LMS solution is adapted one step at a time when a new data sample  $(\mathbf{x}_t, y_t)$  becomes available.

As before mentioned, the statistical variables  $\mathbf{R}_t$  and  $\mathbf{p}_t$  are usually unknown. Then, the LMS algorithm uses their instantaneous estimates, respectively, as follows:

$$\hat{\mathbf{R}}_t = \mathbf{x}_t \mathbf{x}_t^\top, \quad (4.8)$$

$$\hat{\mathbf{p}}_t = y_t \mathbf{x}_t. \quad (4.9)$$

Thus, substituting the estimates of Eqs. (4.8) and (4.9) into Eq. (4.6), one gets the update rule for the LMS algorithm as

$$\begin{aligned} \hat{\mathbf{w}}_{t+1} &= \hat{\mathbf{w}}_t + 2\mu(\hat{\mathbf{p}}_t - \hat{\mathbf{R}}_t \hat{\mathbf{w}}_t), \\ &= \hat{\mathbf{w}}_t + 2\mu \mathbf{x}_t (y_t - \mathbf{x}_t^\top \hat{\mathbf{w}}_t), \\ &= \hat{\mathbf{w}}_t + 2\mu \mathbf{x}_t e_t, \end{aligned} \quad (4.10)$$

where  $\mathbf{x}_t e_t$  represents an instantaneous estimate of the gradient, and  $\hat{\mathbf{w}}_t$  is an iterative approximation of the Wiener solution  $\mathbf{w}_t^{opt}$ . One can easily observe that, unlike the Wiener filter, the LMS algorithm avoids the computation of the inverse matrix of  $\mathbf{R}_t$ .

Examining the operations in Eq. (4.10), it is possible to see the simplicity of the LMS algorithm, whose computational complexity is  $\mathcal{O}(d)$ . However, its main limitation is the relatively slow rate of convergence (HAYKIN, 2013). To minimize this effect, one should choose a relatively large value of  $\mu$ . Finally, the resulting LMS pseudo-code is summarized in Algorithm 6.

---

**Algorithm 6:** - Pseudo-code for the LMS algorithm.

---

**Require:** Set  $\mu$ ;  
 $\hat{\mathbf{w}}_0 = \mathbf{0}$ ;  
**for**  $t = 1, 2, \dots$  **do**  
     $\hat{y}_t = \mathbf{x}_t^\top \hat{\mathbf{w}}_{t-1}$ ;  
     $e_t = y_t - \hat{y}_t$ ;  
     $\hat{\mathbf{w}}_t = \hat{\mathbf{w}}_{t-1} + 2\mu e_t \mathbf{x}_t$ ;  
**end for**  
Output  $\hat{\mathbf{w}}_t$ .

---

### 4.2.3 Recursive Least Squares Algorithm

A suitable alternative to overcome the slow rate of convergence of the LMS algorithm is the classical RLS algorithm (HAYKIN, 2013; DINIZ, 2008). Although both the LMS and RLS present a criterion of weight adaptation based on the error-correction learning, there is

a substantial difference relative to their cost functions. While the LMS algorithm minimizes the instantaneous squared error in Eq. (4.7), the RLS algorithm aims at minimizing the sum of squared estimation errors up to and including the current time step  $t$ .

Therefore, the resulting cost function for the RLS algorithm is defined by

$$\begin{aligned} J_{\hat{\mathbf{w}}_t}(e_t) &= \sum_{i=0}^t \lambda^{t-i} |e_i|^2, \\ &= \sum_{i=0}^t \lambda^{t-i} |y_i - \mathbf{x}_i^\top \hat{\mathbf{w}}_t|^2, \end{aligned} \quad (4.11)$$

where the parameter  $0 \ll \lambda < 1$  is called the *forgetting factor*, since the information of the distant past has an increasingly negligible effect on the coefficients updating. In the special case when  $\lambda = 1$ , one has the OLS algorithm, as seen in Chapter 3.

Thus, taking into account the gradient of  $J_{\hat{\mathbf{w}}_t}$  in Eq. (4.11), one gets

$$\nabla_{\hat{\mathbf{w}}_t} J_{\hat{\mathbf{w}}_t}(e_t) = \frac{\partial J_{\hat{\mathbf{w}}_t}(e_t)}{\partial \hat{\mathbf{w}}_t} = -2 \sum_{i=0}^t \lambda^{t-i} \mathbf{x}_i (y_i - \mathbf{x}_i^\top \hat{\mathbf{w}}_t). \quad (4.12)$$

By setting the gradient in Eq. (4.12) to zero, it is possible to write

$$-\sum_{i=0}^t \lambda^{t-i} \mathbf{x}_i \mathbf{x}_i^\top \hat{\mathbf{w}}_t + \sum_{i=0}^t \lambda^{t-i} \mathbf{x}_i y_i = \mathbf{0}, \quad (4.13)$$

where  $\mathbf{0}$  is a vector of zeros. Isolating  $\hat{\mathbf{w}}_t$  in Eq. (4.13), the resulting expression for its optimal value is given by

$$\hat{\mathbf{w}}_t = \left[ \sum_{i=0}^t \lambda^{t-i} \mathbf{x}_i \mathbf{x}_i^\top \right]^{-1} \sum_{i=0}^t \lambda^{t-i} \mathbf{x}_i y_i. \quad (4.14)$$

Defining analogous expressions to Eqs.(4.8) and (4.9), one has

$$\bar{\mathbf{R}}_t = \sum_{i=0}^t \lambda^{t-i} \mathbf{x}_i \mathbf{x}_i^\top, \quad (4.15)$$

$$\bar{\mathbf{p}}_t = \sum_{i=0}^t \lambda^{t-i} \mathbf{x}_i y_i, \quad (4.16)$$

where  $\bar{\mathbf{R}}_t$  and  $\bar{\mathbf{p}}_t$  are called the deterministic correlation matrix of the input signal and the deterministic cross-correlation vector between the input and the desired signals, respectively. Then, the resulting solution is given by

$$\hat{\mathbf{w}}_t = \bar{\mathbf{R}}_t^{-1} \bar{\mathbf{p}}_t, \quad (4.17)$$

where it was assumed in Eq. (4.14) that  $\bar{\mathbf{R}}_t$  is nonsingular. Nevertheless, the recursive adaptation of the RLS algorithm would face the problem in computing the inverse matrix  $\bar{\mathbf{R}}_t^{-1}$  at each

iteration. In order to avoid this costly procedure, the RLS algorithm makes use of the matrix inversion lemma (see Lemma 4.1 below), which allows the calculation of the inverse matrix in a recursive manner.

**Lemma 4.1** (*Matrix Inversion Lemma*). *Let  $\mathbf{A} \in \mathbb{R}^{L \times L}$  and  $\mathbf{B} \in \mathbb{R}^{L \times L}$  be two positive-definite matrices satisfying*

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^T,$$

where  $\mathbf{D} \in \mathbb{R}^{M \times M}$  is positive-definite and  $\mathbf{C} \in \mathbb{R}^{L \times M}$ . Then, the inverse of the matrix  $\mathbf{A}$  can be expressed as

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}(\mathbf{D} + \mathbf{C}^T\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{B}. \quad (4.18)$$

This relationship is also referred to in the literature as *Woodbury's formula* or the *Sherman-Morrison-Woodbury formula* (GOLUB; LOAN, 2012; HAGER, 1989).

Thus, the expressions for  $\bar{\mathbf{R}}_t$  and  $\bar{\mathbf{p}}_t$  in Eqs. (4.15) and (4.16) can be developed, respectively, as

$$\begin{aligned} \bar{\mathbf{R}}_t &= \mathbf{x}_t\mathbf{x}_t^\top + \lambda \sum_{i=0}^{t-1} \lambda^{t-i-1} \mathbf{x}_i\mathbf{x}_i^\top, \\ &= \mathbf{x}_t\mathbf{x}_t^\top + \lambda \bar{\mathbf{R}}_{t-1}, \end{aligned} \quad (4.19)$$

and

$$\begin{aligned} \bar{\mathbf{p}}_t &= \mathbf{x}_t y_t + \lambda \sum_{i=0}^{t-1} \lambda^{t-i-1} \mathbf{x}_i y_i, \\ &= \mathbf{x}_t y_t + \lambda \bar{\mathbf{p}}_{t-1}. \end{aligned} \quad (4.20)$$

Considering the equalities  $\mathbf{A} = \bar{\mathbf{R}}_t$ ,  $\mathbf{B} = \lambda \bar{\mathbf{R}}_{t-1}$ ,  $\mathbf{C} = \mathbf{x}_t$  and  $\mathbf{D} = 1$  in Eq. (4.19), defining  $\mathbf{S}_t \triangleq \bar{\mathbf{R}}_t^{-1}$  and applying the matrix inversion lemma of Eq. (4.18), one can write the expression

$$\mathbf{S}_t = \frac{1}{\lambda} \left( \mathbf{S}_{t-1} - \frac{\mathbf{S}_{t-1} \mathbf{x}_t \mathbf{x}_t^\top \mathbf{S}_{t-1}}{\lambda + \mathbf{x}_t^\top \mathbf{S}_{t-1} \mathbf{x}_t} \right), \quad (4.21)$$

with which it becomes possible to compute the inverse matrix of  $\bar{\mathbf{R}}_t$  iteratively.

Therefore, the resulting RLS pseudo-code is summarized in Algorithm 7, where a positive constant  $\delta$ , weighting the identity matrix  $\mathbf{I}$ , is required for the initialization of the algorithm. According to Haykin (2013), a recommended choice of  $\delta$  is that it should be small compared to  $0.01\sigma_x^2$ , where  $\sigma_x^2$  is the variance of  $\mathbf{x}_t$ .

Notably, the convergence rate of the RLS algorithm is typically an order of magnitude faster than the LMS algorithm (LIU *et al.*, 2011). On the other hand, its performance is achieved at the expense of a large increase in its computational complexity  $\mathcal{O}(d^2)$ , since it propagates an error-covariance matrix.

---

**Algorithm 7:** - Pseudo-code for the RLS algorithm.

---

**Require:** Set  $\lambda, \delta$ ;  
 $\hat{\mathbf{w}}_0 = \mathbf{0}, \mathbf{S}_0 = \delta^{-1}\mathbf{I}, \mathbf{p}_0 = \mathbf{0}$ ;  
**for**  $t = 1, 2, \dots$  **do**  
 $\mathbf{S}_t = \frac{1}{\lambda} \left( \mathbf{S}_{t-1} - \frac{\mathbf{S}_{t-1} \mathbf{x}_t \mathbf{x}_t^\top \mathbf{S}_{t-1}}{\lambda + \mathbf{x}_t^\top \mathbf{S}_{t-1} \mathbf{x}_t} \right)$ ;  
 $\bar{\mathbf{p}}_t = \lambda \bar{\mathbf{p}}_{t-1} + \mathbf{x}_t y_t$ ;  
 $\hat{\mathbf{w}}_t = \mathbf{S}_t \bar{\mathbf{p}}_t$ ;  
 $\hat{y}_t = \hat{\mathbf{w}}_t^\top \mathbf{x}_t$ ;  
 $e_t = y_t - \hat{y}_t$ ;  
**end for**  
Output  $\mathbf{w}_t$ .

---

For a more detailed analysis of the LMS and RLS algorithms and their main properties, such as convergence, stability, steady-state and transient behaviors, etc., one can consult the references Haykin (2013), Diniz (2008), Bellanger (2001) and Farhang-Boroujeny (2013).

### 4.3 Kernel Adaptive Filtering

The most common existing approaches in the field of adaptive filtering focus on linear models, due to their inherent simplicity in terms of concepts and implementations. However, as previously mentioned, there are many real-world applications that require more expressive hypothesis spaces than linear functions (LIU *et al.*, 2011). In this sense, the KAF models have dawned as suitable tools in solving nonlinear regression problems.

The general structure of a kernel adaptive filter system is illustrated in Fig. (7), where the signals  $\mathbf{x}_t \in \mathbb{R}^d$ ,  $n_t$ ,  $y'_t$  and  $y_t$  were already defined for the linear filter in Section 4.2. The cost function of the kernel adaptive models is also based on the minimization of the MSE (as in the linear case), but now the filter output is given by  $\hat{y}_t = \mathbf{w}_{t-1}^\top \boldsymbol{\phi}(\mathbf{x}_t)$ , where  $\boldsymbol{\phi}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^{d_h}$  is a nonlinear map into a higher dimensional feature space, where the problem is linearly solvable thanks to the kernel trick. Still, in Fig. (7), the learning algorithm denotes the technique for how to search for the optimal solution, according to the cost function. Common choices for it are the classical LMS and RLS algorithms, whose kernelized versions are treated below.

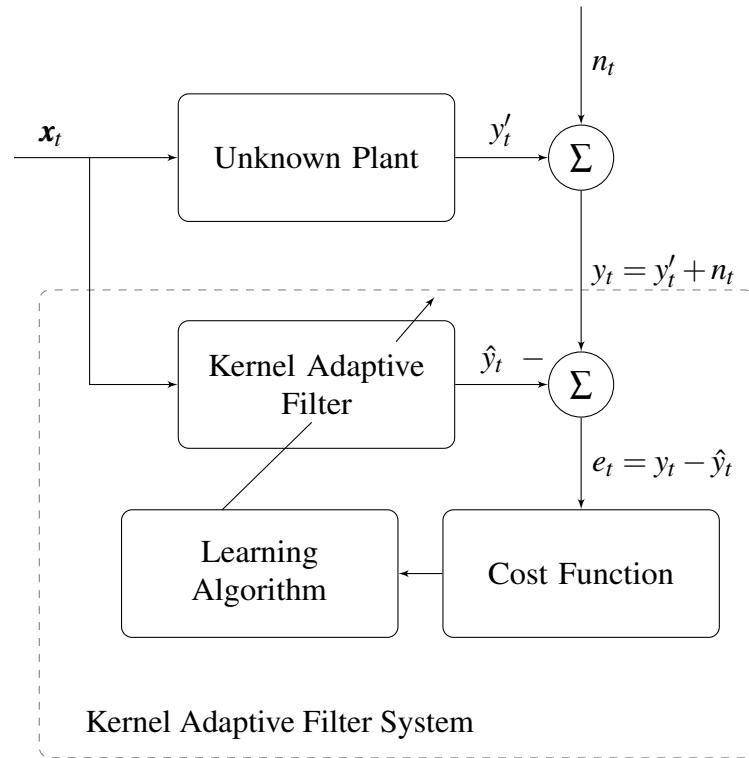


Figure 7 – Basic configuration of a kernel adaptive filter system.

#### 4.3.1 Kernel LMS Algorithm

If the mapping between the inputs  $\mathbf{x}_t$ 's and the corresponding outputs  $y_t$ 's is nonlinear, it is expected that the performance of the linear algorithms (e.g. the LMS algorithm) considerably decreases. In order to overcome this limitation in the LMS algorithm, Liu *et al.* (2008) proposed the *Kernel LMS* (KLMS) algorithm, which provides a sample-by-sample update for an adaptive filter in RKHS.

The KLMS algorithm uses the gradient descent method to search for the optimal solution, by minimizing the instantaneous cost function

$$\begin{aligned} J_{\mathbf{w}}(e_t) &= e_t^2 \\ &= \|y_t - \mathbf{w}_{t-1}^\top \boldsymbol{\phi}(\mathbf{x}_t)\|^2, \end{aligned} \quad (4.22)$$

where  $e_t = y_t - \mathbf{w}_{t-1}^\top \boldsymbol{\phi}(\mathbf{x}_t)$  is the prediction error at the time step  $t$ . Then, using the LMS algorithm to update  $\mathbf{w}_t$ , one gets

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \mu e_t \boldsymbol{\phi}(\mathbf{x}_t). \quad (4.23)$$

Note that the vector  $\mathbf{w}_t$  is of the same dimensionality  $d_h$  as the map  $\boldsymbol{\phi}$ . However, this dimensionality  $d_h$  is very high, and can be even infinite in the case of Gaussian kernel. Thus, although



the KLMS algorithm uses the stochastic gradient in the training phase, as it is still an online algorithm, it loses the original LMS simplicity due to the inability to work directly with the weights in the RKHS (LIU *et al.*, 2008). Then, it is necessary to have an alternative for carrying out the iterative procedure. It is possible to repeatedly apply Eq. (4.23) as follows:

$$\begin{aligned}
\mathbf{w}_t &= \mathbf{w}_{t-1} + \mu e_t \boldsymbol{\phi}(\mathbf{x}_t), \\
&= (\mathbf{w}_{t-2} + \mu e_{t-1} \boldsymbol{\phi}(\mathbf{x}_{t-1})) + \mu e_t \boldsymbol{\phi}(\mathbf{x}_t), \\
&= \mathbf{w}_{t-2} + \mu (e_{t-1} \boldsymbol{\phi}(\mathbf{x}_{t-1}) + e_t \boldsymbol{\phi}(\mathbf{x}_t)), \\
&\vdots \\
&= \mathbf{w}_0 + \mu \sum_{n=1}^{t-1} e_n \boldsymbol{\phi}(\mathbf{x}_n), \\
&= \mu \sum_{n=1}^{t-1} e_n \boldsymbol{\phi}(\mathbf{x}_n),
\end{aligned} \tag{4.24}$$

assuming  $\mathbf{w}_0 = \mathbf{0}$ . Thus, the output  $\hat{y}_t$  of the KLMS algorithm for an input  $\mathbf{x}_t$  can be expressed by

$$\begin{aligned}
\hat{y}_t = \mathbf{w}_t^\top \boldsymbol{\phi}(\mathbf{x}_t) &= \left[ \mu \sum_{n=1}^{t-1} e_n \boldsymbol{\phi}(\mathbf{x}_n)^\top \right] \boldsymbol{\phi}(\mathbf{x}_t), \\
&= \mu \sum_{n=1}^{t-1} e_n \left[ \boldsymbol{\phi}(\mathbf{x}_n)^\top \boldsymbol{\phi}(\mathbf{x}_t) \right].
\end{aligned} \tag{4.25}$$

According to the kernel trick, the output in Eq. (4.25) can be computed in the input space by kernel evaluations, such as

$$\hat{y}_t = \mu \sum_{n=1}^{t-1} e_n k(\mathbf{x}_n, \mathbf{x}_t). \tag{4.26}$$

One should note in Eq. (4.26) that the present output is determined by the sum of all the previous predicted errors weighted by the kernel evaluations on the previously received training inputs. Therefore, the KLMS algorithm creates a growing network, since their complexity and memory storage increase linearly with each incoming sample. However, KLMS is a simple recursive algorithm, whose computational complexity and memory demand are  $\mathcal{O}(N)$ , resulting in reliable convergence without the hassles of local minima (POKHAREL *et al.*, 2007). The standard KLMS algorithm is summarized in Algorithm 8.

### 4.3.2 Kernel RLS Algorithm

Following the same line of a kernelized adaptive filter as the KLMS algorithm, the *Kernel RLS* (KRLS) model, developed by Engel *et al.* (2004), can be understood as a technique

---

**Algorithm 8:** - Pseudo-code for the KLMS algorithm.

---

**Require:** Set  $\mu$ ;  
 $e_1 = y_1, \hat{y}_1 = \mu e_1$ ;  
**for**  $t = 2, 3, \dots$  **do**  
 $\hat{y}_t = \mu \sum_{n=1}^{t-1} e_n k(\mathbf{x}_n, \mathbf{x}_t)$ ;  
 $e_t = y_t - \hat{y}_t$ ;  
**end for**  
Output  $\hat{y}_t, e_t$ .

---

for performing the standard RLS algorithm in feature space. Therefore, the KRLS model is capable of efficiently solving least squares prediction problems in a recursive and online manner. This section discusses in detail, the KRLS overview, due to its importance for this thesis.

Firstly, consider a sequential stream  $\mathcal{D}_t = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_t, y_t)\}$  of input-output training pairs. Then, the KRLS algorithm assumes a functional form  $f(\mathbf{x}_t) = \boldsymbol{\phi}^\top(\mathbf{x}_t)\mathbf{w}$  and minimizes, at each instant  $t$ , the cost function

$$\begin{aligned} J(\mathbf{w}) &= \sum_{i=1}^t (y_i - f(\mathbf{x}_i))^2, \\ &= \|\mathbf{y}_t - \boldsymbol{\Phi}_t^\top \mathbf{w}\|^2, \end{aligned} \quad (4.27)$$

where  $\boldsymbol{\phi}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^{d_h}$ ,  $\mathbf{w} \in \mathbb{R}^{d_h}$ ,  $\boldsymbol{\Phi}_t = [\boldsymbol{\phi}(\mathbf{x}_1), \dots, \boldsymbol{\phi}(\mathbf{x}_t)] \in \mathbb{R}^{d_h \times t}$  is a matrix storing all the projected vectors and  $\mathbf{y}_t = [y_1, \dots, y_t]^\top \in \mathbb{R}^t$  is the vector of outputs. Ordinarily, the solution of the minimization of  $J(\mathbf{w})$ , in Eq. (4.27), with respect to  $\mathbf{w}$  is given by

$$\begin{aligned} \mathbf{w}_t &= \operatorname{argmin}_{\mathbf{w}} \|\mathbf{y}_t - \boldsymbol{\Phi}_t^\top \mathbf{w}\|^2, \\ &= (\boldsymbol{\Phi}_t^\top)^\dagger \mathbf{y}_t, \end{aligned} \quad (4.28)$$

where  $(\cdot)^\dagger$  is the pseudo-inverse matrix (see Eq. (2.30)).

As previously mentioned, the feature map  $\boldsymbol{\phi}$  may be of very high dimension, making the manipulation of the matrix  $\boldsymbol{\Phi}_t$  difficult, or even impracticable in some situations, such as when using the Gaussian kernel. An alternative to overcome this problem is to express the optimal vector  $\mathbf{w}_t$  as (ENGEL *et al.*, 2004)

$$\mathbf{w}_t = \sum_{i=1}^t \alpha_i \boldsymbol{\phi}(\mathbf{x}_i) = \boldsymbol{\Phi}_t \boldsymbol{\alpha}_t, \quad (4.29)$$

where  $\boldsymbol{\alpha}_t = [\alpha_1, \dots, \alpha_t]^\top \in \mathbb{R}^t$  is a vector of coefficients. Because of its analogy with the vector of Lagrange multipliers in SVR (Section 2.2) and LSSVR (Section 2.3) dual formulations, the same symbol  $\boldsymbol{\alpha}$  to denote the vector of coefficients is used in this work. Thus, the cost function

in Eq. (4.27) can be rewritten as

$$J(\boldsymbol{\alpha}_t) = \|\mathbf{y}_t - \mathbf{K}_t \boldsymbol{\alpha}_t\|^2, \quad (4.30)$$

where  $\mathbf{K}_t = \boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t$  is the kernel matrix built up the instant  $t$ , whose entries are calculated by a kernel function as  $\mathbf{K}_{t(i,j)} = k(\mathbf{x}_i, \mathbf{x}_j)$ . Theoretically, the analytic solution for the minimization of  $J(\boldsymbol{\alpha}_t)$  in Eq. (4.30) is given by  $\boldsymbol{\alpha}_t = \mathbf{K}_t^{-1} \mathbf{y}_t$ . However, in order to avoid recomputation of the inverse matrix  $\mathbf{K}_t^{-1}$  at each time step  $t$ ,  $\boldsymbol{\alpha}_t$  may be calculated recursively applying the RLS algorithm.

Nevertheless, a problem arises in this sequential updating scenario. Each new incoming pair  $(\mathbf{x}_t, y_t)$  must be incorporated into  $\mathbf{K}_t$  and  $\mathbf{y}_t$ , respectively, increasing their dimensions and, by extension, increasing the complexity of the KRLS algorithm. An elegant way to overcome this shortcoming, coping with limited resources of memory/computation for online operation and requiring that the computational cost per sample does not necessarily increase with the size of the dataset, involves the use of the sparsification procedure based on the *approximate linear dependency* (ALD) criterion, introduced by Engel *et al.* (2002).

The basic idea behind online sparsification methods, especially the ALD criterion, consists in, at any point in time, the model verifying and deciding whether to add the current sample to its representation, or discard it (ENGEL *et al.*, 2003). Then, one assumes that at a time step  $t$  ( $2 \leq t \leq N$ ), after having sequentially observed  $t - 1$  training samples  $\{\mathbf{x}_i\}_{i=1}^{t-1}$ , one has collected a support vector (SV) dictionary comprised of a subset of training inputs  $\mathcal{D}_{t-1}^{sv} = \{\tilde{\mathbf{x}}_j\}_{j=1}^{m_{t-1}}$ . The symbol  $\sim$  is used from now on in this thesis to refer to the vectors and matrices of the dictionary patterns.

When a new incoming sample  $\mathbf{x}_t$  is available, one must verify if  $\boldsymbol{\phi}(\mathbf{x}_t)$  is approximately linearly dependent on the dictionary vectors. If the test reveals that  $\boldsymbol{\phi}(\mathbf{x}_t)$  is independent on the dictionary vectors,  $\mathbf{x}_t$  must be added to the dictionary. Thus, to test if a training vector  $\mathbf{x}_t$  should be added or not to the dictionary, it is necessary to estimate a vector of coefficients  $\mathbf{a} = [a_1, \dots, a_{m_{t-1}}]^\top$  satisfying the ALD criterion

$$\delta_t := \min_{\mathbf{a}} \left\| \sum_{m=1}^{m_{t-1}} a_m \boldsymbol{\phi}(\tilde{\mathbf{x}}_m) - \boldsymbol{\phi}(\mathbf{x}_t) \right\|^2 \leq \nu, \quad (4.31)$$

where  $\nu$  is the *sparsity level parameter*. Developing the minimization in Eq. (4.31) and using  $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$ , one can write

$$\delta_t = \min_{\mathbf{a}} \{ \mathbf{a}^\top \tilde{\mathbf{K}}_{t-1} \mathbf{a} - 2 \mathbf{a}^\top \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) + k_{tt} \}, \quad (4.32)$$

where  $\tilde{\mathbf{K}}_{t-1} \in \mathbb{R}^{m_{t-1} \times m_{t-1}}$  is the kernel matrix calculated with the dictionary samples,  $\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \in \mathbb{R}^{m_{t-1}}$  and  $k_{tt} = k(\mathbf{x}_t, \mathbf{x}_t) \in \mathbb{R}$ . The  $(i, j)$ -th entry of  $\tilde{\mathbf{K}}_{t-1}$  is computed as  $[\tilde{\mathbf{K}}_{t-1}]_{i,j} = k(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j)$ , while the  $i$ -th component of  $\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)$  is computed as  $(\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t))_i = k(\tilde{\mathbf{x}}_i, \mathbf{x}_t)$ , for  $i, j = 1, \dots, m_{t-1}$ . The solution of Eq. (4.32) is given by

$$\mathbf{a}_t = \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t), \quad (4.33)$$

for which one gets

$$\delta_t = k_{tt} - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \mathbf{a}_t \leq \nu. \quad (4.34)$$

If otherwise  $\delta_t > \nu$ , the current dictionary must be expanded by adding  $\mathbf{x}_t$ . Thus,  $\mathcal{D}_t^{\text{sv}} = \mathcal{D}_{t-1}^{\text{sv}} \cup \{\mathbf{x}_t\}$  and  $m_t = m_{t-1} + 1$ .

From the above exposed and from Engel *et al.* (2004), it is possible to rewrite the problem of Eq. (4.30) as

$$J(\tilde{\boldsymbol{\alpha}}_t) = \|\mathbf{y}_t - \mathbf{A}_t \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t\|^2, \quad (4.35)$$

where  $\mathbf{A}_t = [\mathbf{a}_1 \ \mathbf{a}_2 \ \dots \ \mathbf{a}_t]^\top \in \mathbb{R}^{t \times m_t}$  and  $\tilde{\boldsymbol{\alpha}}_t \in \mathbb{R}^{m_t}$  is a reduced vector of  $m_t$  coefficients. Thus, the optimal solution vector  $\tilde{\boldsymbol{\alpha}}_t$ , which is searched by computing the gradient vector  $\nabla_{\tilde{\boldsymbol{\alpha}}_t} J(\tilde{\boldsymbol{\alpha}}_t) = \partial J(\tilde{\boldsymbol{\alpha}}_t) / \partial \tilde{\boldsymbol{\alpha}}_t$  and equaling it to zero, is given by

$$\tilde{\boldsymbol{\alpha}}_t = \tilde{\mathbf{K}}_t^{-1} (\mathbf{A}_t^\top \mathbf{A}_t)^{-1} \mathbf{A}_t^\top \mathbf{y}_t. \quad (4.36)$$

By defining a matrix  $\mathbf{P}_t$  as

$$\mathbf{P}_t = (\mathbf{A}_t^\top \mathbf{A}_t)^{-1}, \quad (4.37)$$

one finally gets

$$\tilde{\boldsymbol{\alpha}}_t = \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t \mathbf{A}_t^\top \mathbf{y}_t. \quad (4.38)$$

The next step requires the iterative computation of the inverse matrices in Eqs. (4.37) and (4.38) to estimate the vector  $\tilde{\boldsymbol{\alpha}}_t$ . For this purpose, there are two possible situations, which are described below.

#### 4.3.2.1 Case 1 - Unchanged Dictionary

In this case,  $\delta_t \leq \nu$ , meaning that  $\boldsymbol{\phi}(\mathbf{x}_t)$  is approximately linearly dependent on the dictionary vectors. Hence,  $\mathbf{x}_t$  is not added to the dictionary and the kernel matrix is not changed. Mathematically,  $\mathcal{D}_t^{\text{sv}} = \mathcal{D}_{t-1}^{\text{sv}}$  and  $\tilde{\mathbf{K}}_t = \tilde{\mathbf{K}}_{t-1}$ .

Since  $\mathbf{a}_t$  needs to be computed by Eq. (4.33) to determine  $\delta_t$ , the matrix  $\mathbf{A}_t$  is iteratively built by inclusion of  $\mathbf{a}_t$ , i.e.  $\mathbf{A}_t = [\mathbf{A}_{t-1}^\top \ \mathbf{a}_t]^\top$ . Therefore,  $\mathbf{A}_t^\top \mathbf{A}_t = \mathbf{A}_{t-1}^\top \mathbf{A}_{t-1} + \mathbf{a}_t \mathbf{a}_t^\top$ . Then, one can use the standard RLS algorithm based on the matrix inversion lemma (see Eq. (4.18)) to recursively compute the matrix  $\mathbf{P}_t$  as

$$\mathbf{P}_t = (\mathbf{A}_t^\top \mathbf{A}_t)^{-1} = \mathbf{P}_{t-1} - \frac{\mathbf{P}_{t-1} \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1}}{1 + \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t}. \quad (4.39)$$

A gain vector  $\mathbf{q}_t$  is defined as

$$\mathbf{q}_t = \frac{\mathbf{P}_{t-1} \mathbf{a}_t}{1 + \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t}, \quad (4.40)$$

and consequently

$$\mathbf{P}_t = \mathbf{P}_{t-1} - \mathbf{q}_t \mathbf{a}_t^\top \mathbf{P}_{t-1}. \quad (4.41)$$

Finally, using the fact that  $\mathbf{A}_t^\top \mathbf{y}_t = \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} + \mathbf{a}_t \mathbf{y}_t$ , the KRLS update rule for  $\tilde{\boldsymbol{\alpha}}_t$  can be written by

$$\begin{aligned} \tilde{\boldsymbol{\alpha}}_t &= \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t \mathbf{A}_t^\top \mathbf{y}_t, \\ &= \tilde{\mathbf{K}}_t^{-1} (\mathbf{P}_{t-1} - \mathbf{q}_t \mathbf{a}_t^\top \mathbf{P}_{t-1}) (\mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} + \mathbf{a}_t \mathbf{y}_t), \\ &= \tilde{\boldsymbol{\alpha}}_{t-1} + \tilde{\mathbf{K}}_t^{-1} (\mathbf{P}_t \mathbf{a}_t \mathbf{y}_t - \mathbf{q}_t \mathbf{a}_t^\top \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_{t-1}), \\ &= \tilde{\boldsymbol{\alpha}}_{t-1} + \tilde{\mathbf{K}}_t^{-1} (\mathbf{q}_t \mathbf{y}_t - \mathbf{q}_t \mathbf{a}_t^\top \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_{t-1}), \\ &= \tilde{\boldsymbol{\alpha}}_{t-1} + \tilde{\mathbf{K}}_t^{-1} \mathbf{q}_t (\mathbf{y}_t - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}), \end{aligned} \quad (4.42)$$

where the last equalities are based on  $\mathbf{q}_t = \mathbf{P}_t \mathbf{a}_t$  and  $\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) = \tilde{\mathbf{K}}_t \mathbf{a}_t$ .

#### 4.3.2.2 Case 2 - Updating the Dictionary

In this case, one gets  $\delta_t > \nu$ , 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  $\mathbf{x}_t$  must be added to the dictionary, i.e.  $\mathcal{D}_t^{\text{sv}} = \mathcal{D}_{t-1}^{\text{sv}} \cup \{\mathbf{x}_t\}$  and  $m_t = m_{t-1} + 1$ . As a consequence of this inclusion, the kernel matrix must be updated accordingly.

The challenge here is to compute  $\tilde{\mathbf{K}}_t$  (and hence  $\tilde{\mathbf{K}}_t^{-1}$ ) recursively using  $\tilde{\mathbf{K}}_{t-1}$  and the information provided by the new sample. For this purpose, based on Golub and Loan (2012), one computes the matrices  $\tilde{\mathbf{K}}_t$  and  $\tilde{\mathbf{K}}_t^{-1}$  as

$$\tilde{\mathbf{K}}_t = \begin{bmatrix} \tilde{\mathbf{K}}_{t-1} & \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \\ \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top & k_{tt} \end{bmatrix}, \quad (4.43)$$

and

$$\tilde{\mathbf{K}}_t^{-1} = \frac{1}{\delta_t} \begin{bmatrix} \delta_t \tilde{\mathbf{K}}_{t-1}^{-1} + \mathbf{a}_t \mathbf{a}_t^\top & -\mathbf{a}_t \\ -\mathbf{a}_t^\top & 1 \end{bmatrix}. \quad (4.44)$$

Consider that, for Case 2, not only does the dimension of the kernel matrix  $\tilde{\mathbf{K}}_t$  increases due to the inclusion of sample  $\mathbf{x}_t$  in the dictionary, but also the dimensions of the matrices  $\mathbf{A}_t$  and  $\mathbf{P}_t$ . Hence, one can write

$$\mathbf{A}_t = \begin{bmatrix} \mathbf{A}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad (4.45)$$

$$\mathbf{A}_t^\top \mathbf{A}_t = \begin{bmatrix} \mathbf{A}_{t-1}^\top \mathbf{A}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad (4.46)$$

and

$$\mathbf{P}_t = (\mathbf{A}_t^\top \mathbf{A}_t)^{-1} = \begin{bmatrix} \mathbf{P}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad (4.47)$$

where  $\mathbf{0}$  is a vector of zeros. Then, Eqs. (4.44)-(4.47) are used to calculate  $\tilde{\boldsymbol{\alpha}}_t$  as

$$\begin{aligned} \tilde{\boldsymbol{\alpha}}_t &= \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t \mathbf{A}_t^\top \mathbf{y}_t, \\ &= \tilde{\mathbf{K}}_t^{-1} \begin{bmatrix} (\mathbf{A}_{t-1}^\top \mathbf{A}_{t-1})^{-1} \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} \\ y_t \end{bmatrix}, \\ &= \begin{bmatrix} \tilde{\boldsymbol{\alpha}}_{t-1} - \frac{\mathbf{a}_t}{\delta_t} (y_t - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}) \\ \frac{1}{\delta_t} (y_t - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}) \end{bmatrix}, \end{aligned} \quad (4.48)$$

where for the final equality one uses  $\mathbf{a}_t^\top \tilde{\mathbf{K}}_{t-1}^{-1} = \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top$ . Since the parameter vector  $\tilde{\boldsymbol{\alpha}}_t$  has been updated, the sparse solution for the KRLS model is given by

$$\hat{y}_t = f(\mathbf{x}) = \sum_{m=1}^{m_t} \tilde{\alpha}_m k(\mathbf{x}, \mathbf{x}_m) = \tilde{\mathbf{k}}_{m_t}(\mathbf{x})^\top \tilde{\boldsymbol{\alpha}}_t. \quad (4.49)$$

As a final remark, the computational complexity of the KRLS algorithm is bound by  $\mathcal{O}(m_t^2)$  and its memory demand is  $\mathcal{O}(Nm_t)$ . The KRLS pseudo-code is summarized in Algorithm (9).

---

**Algorithm 9:** - Pseudo-code for the KRLS model.

---

**Require:**  $\nu, \sigma$  (for Gaussian kernel);  
**Set:**  $\tilde{\mathbf{K}}_1 = k_{11}; \tilde{\mathbf{K}}_1^{-1} = 1/\tilde{\mathbf{K}}_1; \mathbf{P}_1 = 1; \tilde{\boldsymbol{\alpha}}_1 = y_1/k_{11}; \mathbf{A}_1 = 1; m_1 = 1;$   
**for**  $t = 2 : N$ , **do**  
    Get new sample  $(\mathbf{x}_t, y_t)$  and compute  $\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t);$   
     $\mathbf{a}_t = \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t);$   
     $\delta_t = k_{tt} - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \mathbf{a}_t;$   
    **if**  $\delta_t > \nu$  **then**  
         $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \cup \{\mathbf{x}_t\};$  % add  $\mathbf{x}_t$  to the dictionary  
        Compute  $\tilde{\mathbf{K}}_t^{-1}$  from Eq. (4.44);  
        Compute  $\mathbf{P}_t$  from Eq. (4.47);  
        Compute  $\tilde{\boldsymbol{\alpha}}_t$  from Eq. (4.48);  
         $m_t = m_{t-1} + 1;$   
    **else**  
         $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv};$  % unchanged dictionary  
        Compute  $\mathbf{q}_t$  from Eq. (4.40);  
        Compute  $\mathbf{P}_t$  from Eq. (4.41);  
        Compute  $\tilde{\boldsymbol{\alpha}}_t$  from Eq. (4.42);  
    **end if**  
**end for**  
Output  $\tilde{\boldsymbol{\alpha}}_t, \mathcal{D}_t^{sv}.$

---

#### 4.4 Other Sparsification Criteria

As discussed in Section 4.3.2, the ALD criterion corresponds to an online constructive sparsification procedure. This means that the algorithm starts with an empty representation, in which, initially, there are no coefficients, and gradually adds input samples to the dictionary according to some criterion. On that note, there are several sparsification techniques available in the literature to obtain sparse solutions in kernel adaptive filters. Among them, the following should be noted:

1. **Novelty Criterion** - The novelty criterion (NC) was proposed by Platt (1991) in order to find a compact representation for resource-allocating networks. It corresponds to a simple procedure that, first, computes the distance of the present input sample  $\mathbf{x}_t$  to the dictionary, such as

$$\text{dis}_1 = \min_{\mathbf{x}_j \in \mathcal{D}_{t-1}^{sv}} \|\mathbf{x}_t - \mathbf{x}_j\|, \quad (4.50)$$

where  $\mathcal{D}_{t-1}^{sv}$  is the dictionary built up the time step  $t - 1$ . Then, if  $\text{dis}_1 < \delta_1$ , the input  $\mathbf{x}_t$  should not be added to the dictionary. Otherwise, one computes the prediction error  $e_t = y_t - \hat{y}_t$  and, only if  $|e_t| > \delta_2$ ,  $\mathbf{x}_t$  should be accepted for the dictionary. The parameters

$\delta_1$  and  $\delta_2$  should be previously selected.

2. **Coherence Criterion** - The coherence parameter was introduced as a quantity of heuristic for matching pursuit (MALLAT; ZHANG, 1993). Later, Richard *et al.* (2009) proposed a coherence criterion to control the number of selected inputs in kernel models as well as limiting the size of the kernel matrix. Mathematically, the coherence parameter is defined as

$$\delta_c = \max_{i \neq j} |k(\mathbf{x}_i, \mathbf{x}_j)|, \quad (4.51)$$

where, in this case,  $k(\cdot)$  is a unit-norm kernel function<sup>1</sup>, that is,  $k(\mathbf{x}_i, \mathbf{x}_i) = 1$  for all value of  $i$ . The parameter  $\delta_c$  is the largest absolute value of the off-diagonal entries in the kernel matrix, which reveals the largest cross-correlations in the dictionary. Therefore, this criterion suggests that an input  $\mathbf{x}_t$  be inserted in the dictionary if its coherence remains below a certain threshold  $\delta_0$ , given by (RICHARD *et al.*, 2009)

$$\max_{\mathbf{x}_j \in \mathcal{D}_{t-1}^{sv}} |k(\mathbf{x}_t, \mathbf{x}_j)| \leq \delta_0, \quad (4.52)$$

where  $0 \leq \delta_0 < 1$  determines both the level of sparsity and the coherence of the dictionary  $\mathcal{D}_{t-1}^{sv}$  up the step  $t - 1$ . At a time step  $t$ , if  $k(\cdot, \mathbf{x}_t)$  satisfies the condition of Eq. (4.52), then  $\mathbf{x}_t$  should be introduced into the dictionary, which becomes  $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \cup \{\mathbf{x}_t\}$ .

3. **Surprise Criterion** - Recently, Liu *et al.* (2009a) developed an information theoretic measure called surprise. The concept of surprise quantifies how much information an input pattern contains relative to the current knowledge of the learning system (LIU *et al.*, 2011). According to Liu *et al.* (2009a), the surprise of a pair  $(\mathbf{x}, y)$ , denoted by  $S_{\mathcal{T}}(\mathbf{x}, y)$ , is defined as the negative log likelihood (NLL) of the input-output pair, given the learning system's hypothesis on the data distribution

$$S_{\mathcal{T}}(\mathbf{x}, y) = -\ln p(\mathbf{x}, y | \mathcal{T}), \quad (4.53)$$

where  $p(\mathbf{x}, y | \mathcal{T})$  is the subjective probability of  $(\mathbf{x}, y)$  hypothesized by  $\mathcal{T}$ . One can say that  $S_{\mathcal{T}}$  measures how "surprising" an input-output pair is to the learning system. The definition in Eq. (4.53) can be used in the context of online learning problems, where the surprise of  $(\mathbf{x}_t, y_t)$  to the current learning system  $\mathcal{T}_{t-1}$  is given by

$$S_{\mathcal{T}_{t-1}}(\mathbf{x}_t, y_t) = -\ln p(\mathbf{x}_t, y_t | \mathcal{T}_{t-1}), \quad (4.54)$$

---

<sup>1</sup>If  $k(\cdot)$  is not a unit-normal kernel,  $k(\mathbf{x}_i, \mathbf{x}_j)$  can be replaced by  $k(\mathbf{x}_i, \mathbf{x}_j) / \sqrt{k(\mathbf{x}_i, \mathbf{x}_i)k(\mathbf{x}_j, \mathbf{x}_j)}$ .



where  $p(\mathbf{x}_t, y_t | \mathcal{T}_{t-1})$  is the a posterior distribution of  $(\mathbf{x}_t, y_t)$  hypothesized by  $\mathcal{T}_{t-1}$ .

Based on Gaussian Processes Regression (GPR) theory (RASMUSSEN; WILLIAMS, 2006), denoting  $S_t = S_{\mathcal{T}_{t-1}}(\mathbf{x}_t, y_t)$  for simplicity, and assuming  $\mathcal{T}_{t-1} = \{\mathbf{x}_i, y_i\}_{i=1}^{t-1}$ , the surprise measure is computed as (LIU *et al.*, 2009a)

$$\begin{aligned} S_t &= -\ln[p(\mathbf{x}_t, y_t | \mathcal{T}_{t-1})], \\ &= \ln \sigma_t + \frac{(y_t - \bar{y}_t)^2}{2\sigma_t^2} - \ln[p(\mathbf{x}_t, | \mathcal{T}_{t-1})] + \ln \sqrt{2\pi}, \end{aligned} \quad (4.55)$$

where  $\sigma_t$  is the predicted variance and  $\bar{y}_t$  is the maximum a posteriori (MAP) estimation of  $y_t$  by the current learning system  $\mathcal{T}_{t-1}$ .

In general, one can say that there are similar features between some of the above mentioned sparsification methods. For instance, according to Liu *et al.* (2009a), the coherence criterion can be viewed as an approximation to ALD. Moreover, under a memoryless uniform input assumption, the surprise criterion becomes the ALD criterion. Therefore, the ALD criterion is a special case of the surprise criterion (LIU *et al.*, 2009a; LIU *et al.*, 2011).

Finally, it is important to note that the above mentioned sparsification criteria, as well as the ALD, can be generally applied to the KLMS, KRLS and other kernel-based and online models to obtain sparse solutions. The next section briefly explains some of other online kernel-based models.

#### 4.5 Other Online Kernel Models

The KLMS and KRLS are the most common recursive algorithms used as kernel adaptive filters. However, some other approaches have been developed as efficient nonlinear extensions of simple linear filtering algorithms. Particularly, the *Kernelized Affine Projection Algorithms* (KAPA) were presented by Liu and Príncipe (2008) to reformulate the conventional *Affine Projection Algorithms* (APA) (SAYED, 2003) in RKHS.

As in the LMS case, the APA are simple and online, but they reduce the gradient noise using instantaneous approximations from a window comprised of the  $L$  most recent inputs and respective outputs as

$$\begin{aligned} \mathbf{X}_t &= [\mathbf{x}_{t-L+1}, \dots, \mathbf{x}_t], \\ \mathbf{y}_t &= [y_{t-L+1}, \dots, y_t], \end{aligned} \quad (4.56)$$

where  $\mathbf{X}_t \in \mathbb{R}^{d \times L}$  is the input matrix and  $\mathbf{y}_t \in \mathbb{R}^L$  is the vector of outputs. From the steepest

descent method in Eq. (4.10), which is repeated here for greater clarity

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mu(\hat{\mathbf{p}}_t - \hat{\mathbf{R}}_t \mathbf{w}_t), \quad (4.57)$$

where, for simplicity, the factor 2 in Eq. (4.6) was inserted in the step-size parameter  $\mu$ . Thus, assuming

$$\hat{\mathbf{R}}_t = \frac{1}{L} \mathbf{X}_t \mathbf{X}_t^\top, \quad (4.58)$$

for the input correlation matrix, and

$$\hat{\mathbf{p}}_t = \frac{1}{L} \mathbf{X}_t \mathbf{y}_t, \quad (4.59)$$

for the cross-correlation vector, Eq. (4.57) becomes

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mu \mathbf{X}_t (\mathbf{y}_t - \mathbf{X}_t^\top \mathbf{w}_t). \quad (4.60)$$

The recursion expression in Eq. (4.60) is called the APA-1 . A similar procedure can be performed using the regularized Newton's recursion method (DINIZ, 2008), which is given by

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mu (\mathbf{R}_t + \gamma_0 \mathbf{I})^{-1} (\hat{\mathbf{p}}_t - \hat{\mathbf{R}}_t \mathbf{w}_t), \quad (4.61)$$

where  $\gamma_0$  is a regularization parameter. Again, using the equalities in Eqs. (4.58) and (4.59), and the matrix inversion lemma, Eq. (4.61) becomes

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mu \mathbf{X}_t (\mathbf{X}_t^\top \mathbf{X}_t + \gamma_0 \mathbf{I})^{-1} (\mathbf{y}_t - \mathbf{X}_t \mathbf{w}_t), \quad (4.62)$$

which is called the APA-2.

In some circumstances, instead of using the instantaneous squared error as the cost function (see Eq. (4.7)), a regularized version of the least squares (LS) problem is necessary, which is given by

$$J_{\mathbf{w}}(e_t) = e_t^2 + \gamma \|\mathbf{w}\|^2, \quad (4.63)$$

where the regularization parameter  $\gamma$  is different from  $\gamma_0$  of Newton's method. Then, the gradient method in Eq. (4.57) can be rewritten as

$$\begin{aligned} \mathbf{w}_{t+1} &= \mathbf{w}_t + \mu [\hat{\mathbf{p}}_t - (\gamma \mathbf{I} + \hat{\mathbf{R}}_t) \mathbf{w}_t], \\ &= (1 - \mu \gamma) \mathbf{w}_t + \mu [\hat{\mathbf{p}}_t - \hat{\mathbf{R}}_t \mathbf{w}_t]. \end{aligned} \quad (4.64)$$

Likewise, the Newton's recursion in Eq. (4.61), with  $\gamma_0 = 0$ , becomes

$$\begin{aligned}\mathbf{w}_{t+1} &= \mathbf{w}_t + \mu(\gamma\mathbf{I} + \mathbf{R}_t)^{-1}[\hat{\mathbf{p}}_t - (\gamma\mathbf{I} + \hat{\mathbf{R}}_t)\mathbf{w}_t], \\ &= (1 - \mu)\mathbf{w}_t + \mu(\gamma\mathbf{I} + \mathbf{R}_t)^{-1}\mathbf{p}_t.\end{aligned}\quad (4.65)$$

Applying the approximations in Eqs. (4.58)-(4.59) and again the matrix inversion lemma, one can write Eqs. (4.64) and (4.65), respectively, as

$$\mathbf{w}_{t+1} = (1 - \mu\gamma)\mathbf{w}_t + \mu\mathbf{X}_t[\mathbf{y}_t - \mathbf{X}_t^\top \mathbf{w}_t], \quad (4.66)$$

and

$$\mathbf{w}_{t+1} = (1 - \mu)\mathbf{w}_t + \mu\mathbf{X}_t[\gamma\mathbf{I} + \mathbf{X}_t^\top \mathbf{X}_t]^{-1}\mathbf{y}_t. \quad (4.67)$$

The recursive expressions in Eqs. (4.66) and (4.67) are called the APA-3 and APA-4, respectively.

In general, the APA family (APA-1, APA-2, APA-3 and APA-4) of linear filters appears as intermediate complexity algorithms between the standard LMS and RLS algorithms (LIU *et al.*, 2011).

Regarding the discussion above and based on the same methodology outlined for the KLMS algorithm, it is possible to transform the input data into the feature space through the nonlinear map  $\phi$  and applying the kernel trick. This procedure gives rise to the KAPA family (LIU; PRÍNCIPE, 2008). Thereby, one formulates the affine projection algorithms on the sequences  $\{y_1, y_2, \dots\}$  and  $\{\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \dots\}$  to estimate the weight vector  $\mathbf{w}$  that minimizes the cost function

$$J_{\mathbf{w}}(e_t) = \|y_t - \mathbf{w}^\top \phi(\mathbf{x}_t)\|^2, \quad (4.68)$$

as seen in Eq. (4.22). Then, by straightforward manipulation, one can rewrite Eqs. (4.60) and (4.62), respectively, as

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mu\Phi_t(\mathbf{y}_t - \Phi_t^\top \mathbf{w}_t), \quad (4.69)$$

and

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mu\Phi_t(\Phi_t^\top \Phi_t + \gamma_0\mathbf{I})^{-1}(\mathbf{y}_t - \Phi_t^\top \mathbf{w}_t), \quad (4.70)$$

where  $\Phi_t = [\phi(\mathbf{x}_{t-L+1}), \dots, \phi(\mathbf{x}_t)]$ . The recursive expressions in Eqs. (4.69) and (4.70) correspond to the Kernel APA-1 (KAPA-1) and Kernel APA-2 (KAPA-2), respectively.

Algorithm	Recursive Equation	Relation to KAPA
KLMS	$\mathbf{w}_{t+1} = \mathbf{w}_t + \mu \boldsymbol{\phi}_t (y_t - \boldsymbol{\phi}_t^\top \mathbf{w}_t)$	KAPA-1 ( $L = 1$ )
NKLMS	$\mathbf{w}_{t+1} = \mathbf{w}_t + \frac{\mu \boldsymbol{\phi}_t}{k_{tt} + \gamma_0} (y_t - \boldsymbol{\phi}_t^\top \mathbf{w}_t)$	KAPA-2 ( $L = 1$ )
Norma	$\mathbf{w}_{t+1} = (1 - \mu \gamma) \mathbf{w}_t + \mu \boldsymbol{\phi}_t [y_t - \boldsymbol{\phi}_t^\top \mathbf{w}_t]$	KAPA-3 ( $L = 1$ )
Kernel ADALINE	$\mathbf{w}_{t+1} = \mathbf{w}_t + \mu \boldsymbol{\Phi} (\mathbf{y} - \boldsymbol{\Phi}^\top \mathbf{w}_t)$	KAPA-1 ( $L = N$ )
RA-RBF	$\mathbf{w}_{t+1} = \mu \boldsymbol{\Phi} [\mathbf{y} - \boldsymbol{\Phi}^\top \mathbf{w}_t]$	KAPA-3 ( $L = N, \mu \gamma = 1$ )
SW-KRLS	$\mathbf{w}_{t+1} = \boldsymbol{\Phi}_t [\gamma \mathbf{I} + \boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t]^{-1} \mathbf{y}_t$	KAPA-4 ( $\mu = 1$ )
RegNet	$\mathbf{w}_{t+1} = \boldsymbol{\Phi} [\gamma \mathbf{I} + \boldsymbol{\Phi}^\top \boldsymbol{\Phi}]^{-1} \mathbf{y}$	KAPA-4 ( $L = N, \mu = 1$ )

Table 4 – List of kernel adaptive algorithms related to the KAPA family - adapted from Liu and Príncipe (2008).

Accordingly, Eqs. (4.66) and (4.67) become

$$\mathbf{w}_{t+1} = (1 - \mu \gamma) \mathbf{w}_t + \mu \boldsymbol{\Phi}_t [\mathbf{y}_t - \boldsymbol{\Phi}_t^\top \mathbf{w}_t], \quad (4.71)$$

and

$$\mathbf{w}_{t+1} = (1 - \mu) \mathbf{w}_t + \mu \boldsymbol{\Phi}_t [\gamma \mathbf{I} + \boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t]^{-1} \mathbf{y}_t. \quad (4.72)$$

The expressions in Eq. (4.71) and (4.72) are the KAPA-3 and KAPA-4, respectively.

As observed, the KAPA family corresponds to a stochastic gradient methodology to solve the LS problem in RKHS. According to Liu and Príncipe (2008), their performance is somewhere between the KLMS and KRLS models, which is specified by the window length  $L$ . Similar approaches to the KAPA were discussed in Richard *et al.* (2009), Slavakis and Theodoridis (2008) from different perspectives.

As a final remark, an interesting feature of the KAPA family is that it provides an unifying model for several existing neural networks techniques and kernel adaptive filters (LIU *et al.*, 2011), as shown in Table 4, which summarizes all the above related algorithms. In this table one assumes, for simplicity, that  $\boldsymbol{\phi}_t = \boldsymbol{\phi}(\mathbf{x}_t)$ ,  $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1), \dots, \boldsymbol{\phi}(\mathbf{x}_N)]$  and  $k_{tt} = k(\mathbf{x}_t, \mathbf{x}_t)$ .

It is possible to see in Table 4 that, for example, if the window length in KAPA-1 is unitary  $L = 1$ , it reduces to the original KLMS (LIU *et al.*, 2008). Also with  $L = 1$ , a normalized version of KLMS (NKLMS) (MODAGHEGH *et al.*, 2009) is obtained from KAPA-2 and, similarly, KAPA-3 reduces to the Norma algorithm, introduced by Kivinen *et al.* (2004).

Assuming that the window length includes all the available training samples ( $L = N$ ), the update rule of KAPA-1 becomes the kernel ADALINE algorithm (FRIESS; HARRISON, 1999). On the other hand, if one sets  $L = N$  and  $\mu\gamma = 1$ , the update expression of KAPA-3 becomes the same recursively adapted radial basis function (RA-RBF) network, introduced by Liu *et al.* (2007). Likewise, if  $L = N$  and  $\mu = 1$ , KAPA-4 directly becomes the regularization network (RegNet) (GIROSI *et al.*, 1995). One should note that ADALINE, RA-RBF and RegNet are kernel models but they are not online methods. Finally, in KAPA-4 with  $\mu = 1$ , one gets the sliding-window kernel RLS (SW-KRLS), introduced by Vaerenbergh *et al.* (2006), Vaerenbergh *et al.* (2007).

#### 4.6 Concluding Remarks

This chapter has presented a general framework of the kernel adaptive filtering algorithms for nonlinear signal processing problems. This field of machine learning consists of applying the kernelization philosophy to linear adaptive filters in order to produce powerful online and nonlinear extensions of the well-known linear filter algorithms, such as the LMS and RLS algorithms.

Initially, a brief review of the LMS and RLS adaptive algorithms was presented, for further understanding of their nonlinear kernel extensions. The LMS algorithm uses the steepest descent method as the adaptive rule to minimize the instantaneous squared error. Despite its simplicity and low computational complexity  $\mathcal{O}(d)$ , its rate of convergence is relatively slow. On the other hand, the RLS algorithm minimizes the sum of squared errors up to actual instant  $t$ . Therefore, its rate of convergence is one order of magnitude faster than the LMS algorithm, and the achieved training MSE value is, in general, lower than that obtained by the LMS algorithm. However, the RLS computational complexity  $\mathcal{O}(d^2)$  is typically large, since it needs to propagate the error-covariance matrix.

This chapter also discussed two of the most important kernel adaptive filtering models, namely the KLMS and KRLS models, which are nonlinear versions of their equivalent linear filters. The KLMS algorithm solves the least squares problem in the feature space, implying that the gradient search is on a smooth quadratic performance surface. Unlike most kernel methods, the KLMS algorithm does not need to build large kernel matrices because it uses the input patterns one at a time. However, it creates a growing network because it uses all the past training instances to compute the actual output. Its computational complexity and memory

demand are  $\mathcal{O}(N)$ .

In addition, the KRLS algorithm corresponds to the performance of the linear RLS algorithm in feature space. In order to limit the growth of the computational complexity with a new incoming input sample, it applies a sparsification procedure, called the ALD criterion, which verifies if an actual input sample is approximately linearly dependent on the dictionary, which is formed by a reduced amount of input patterns acting as support vectors. Because of that, the computational complexity of the KRLS model is  $\mathcal{O}(m_t^2)$  and its memory demand is  $\mathcal{O}(Nm_t)$ .

Finally, a few comments about other sparsification methods (different from the ALD criterion) were presented, which are usual alternatives for kernel online models to achieved sparse solutions. The sparsification methods that were commented are novelty criterion (NC), coherence and surprise criterion. Besides that, a brief discussion was also presented about some other kernel adaptive models, in particular, the Kernel Affine Projection Algorithms (KAPA). In general, the KAPA family has features, such as performance and rate of convergence, between the standard KLMS and KRLS models.

In the next chapter, the contributions of this thesis will be developed. Specifically, novel robust kernel-based models, in batch mode, will be presented and derived from the LSSVR and FS-LSSVR models.

## 5 NOVEL ROBUST KERNEL-BASED MODELS IN BATCH MODE

“I haven’t failed.  
I’ve just found 10,000 ways that won’t work.”  
(Thomas Edison)

This chapter presents the first part of the contributions of this thesis, which are robust kernel methods derived from the LSSVR and FS-LSSVR models, and are also based on the M-estimators theory. Section 5.1 discusses the motivation in developing the novel proposals. Next, Section 5.2 presents the first proposed robust approach, called the RLM-SVR model, whose formulation is derived from the standard LSSVR formulation (solved in dual space). The corresponding computational experiments using the RLM-SVR model are also presented and discussed in Section 5.2. Then, Section 5.3 presents two novel robust approaches, the RFS-LSSVR and  $R^2$ FS-LSSVR models, which are derived from the original FS-LSSVR model (solved in primal space). Section 5.3 also discusses the obtained results by computational experiments using the RFS-LSSVR and  $R^2$ FS-LSSVR proposals. Finally, Section 5.4 gives the closing remarks of the chapter.

### 5.1 Introduction

The LSSVR model stands for a powerful alternative kernel method to the original Vapnik’s SVR model. As discussed in detail in Chapter 2, learning in LSSVR relies on a SSE cost function and equality constraints instead of the typical quadratic programming problem in SVR, which on its turn is complex and time-consuming, demanding high computational resources for its implementation. A global optimal solution for the LSSVR problem is simpler to obtain since it requires only a set of linear equations, which can be solved by least squares procedure, using the Moore-Penrose inverse matrix in *batch mode*, i.e., all the available pattern samples are stored in computer memory during the training phase.

However, despite its computationally attractive feature, the LSSVR model has some potential limitations, specially for large-scale datasets. The first one is the lack of sparsity in the solution vector, implying that all training examples will be used as support vectors to make new predictions. The second drawback is that the optimality of the SSE cost functions, in the sense of obtain unbiased solutions with minimal variances, is guaranteed only for normally distributed errors. Then, the performance of the LSSVR solution may be considerably affected when the

estimation data is corrupted with non-Gaussian noise, e.g. outliers (SUYKENS *et al.*, 2002a).

In order to properly handle outliers, some authors have developed strategies to robustify the standard LSSVR model for nonlinear regression applications. A first work in this area was proposed by Suykens *et al.* (2002a), who presented a robust weighted version of the LSSVR model (W-LSSVR) based on weighted least squares algorithm and  $M$ -estimators. Later, an iteratively reweighting procedure was applied into the W-LSSVR approach giving rise to the iteratively reweighted LSSVR (IR-LSSVR) model (DE-BRABANTER *et al.*, 2009b). A brief discussion on both W-LSSVR and IR-LSSVR robust approaches was presented in Chapter 3.

In essence, while these robust variants modify the original LSSVR loss function to penalize large errors, they still rely on the LS procedure (using Moore-Penrose inverse matrix) to provide a solution for the resulting linear system. In this scenario, we introduce a different approach to add outlier robustness to the original LSSVR model in dual space. Instead of modifying its loss function, we decide to solve the resulting LSSVR linear system for the parameter estimation by using the *recursive least M-estimate* (RLM) algorithm (ZOU *et al.*, 2000), which is a robust variant (based on the  $M$ -estimators) of the standard RLS algorithm. Our proposed approach, referred to as the *Recursive Least M-estimate Support Vector Regression* (RLM-SVR) model, updates recursively the vector of Lagrange multipliers and the corresponding bias for each input sample. We are strongly inclined to think that this recursive procedure can improve the performance of the model solution in the presence of outliers.

As before mentioned, besides the performance decay in the presence of non-Gaussian noise, non-sparsity is another limiting factor to use the LSSVR model in some modeling applications as, for example, those requiring large amounts of data. An attractive alternative to overcome this aspect is solving the LSSVR optimization problem in the primal space, by applying an approximation for the nonlinear map  $\phi$  using the Nyström method (WILLIAMS; SEEGER, 2001), which corresponds to the FS-LSSVR model, properly discussed in Chapter 2.

Even though being widely used to solve static (DE-BRABANTER *et al.*, 2010; MALL; SUYKENS, 2015) and dynamical (ESPINOZA *et al.*, 2005; ESPINOZA *et al.*, 2004; DE-BRABANTER *et al.*, 2009a; CASTRO *et al.*, 2014; LE *et al.*, 2011; ESPINOZA *et al.*, 2006) regression problems, the FS-LSSVR model is still sensitive to the presence of outliers, as in the standard (dual space) LSSVR case, and the development of robust strategies for its solution is still an open issue.

From the discussion above, we also propose in this chapter two robust approaches



derived from the original FS-LSSVR model, which are based on the  $M$ -estimators and the weighted least squares algorithm. Following the same line of reasoning of the W-LSSVR and IR-LSSVR models in dual space, we present a theoretical development which corresponds to their versions for the FS-LSSVR model in primal space. The proposed approaches, henceforth called the *Robust FS-LSSVR* (RFS-LSSVR) and *Reweighted Robust FS-LSSVR* (R<sup>2</sup>FS-LSSVR) models, keep the same sparsity and computational complexity of the FS-LSSVR model, do not require extra parameters for tuning and can also handle system identification problems with large-scale data.

## 5.2 Proposed Robust Approach in Dual Space

Initially, we address again the standard LSSVR model, whose main steps of its formulation are repeated here for a better understanding of our proposal. Thus, consider the training dataset  $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$ , with inputs  $\mathbf{x}_n \in \mathbb{R}^d$  and corresponding outputs  $y_n \in \mathbb{R}$ . In a nonlinear regression problem, the function  $f(\cdot)$  to be searched (as established by Eq. (2.13)), that approximates the outputs  $y_n$  for all available input instances  $\mathbf{x}_n$ , is given by

$$f(\mathbf{x}_n) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b, \quad (5.1)$$

where  $\mathbf{w} \in \mathbb{R}^{d_h}$ ,  $b \in \mathbb{R}$  and  $\boldsymbol{\phi}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^{d_h}$  were already defined.

The formulation of the LSSVR primal optimization problem leads to minimize the following functional:

$$\min_{\mathbf{w}, b, \mathbf{e}} J_p(\mathbf{w}, \mathbf{e}) = \frac{1}{2} \|\mathbf{w}\|^2 + \gamma \frac{1}{2} \sum_{n=1}^N e_n^2, \quad (5.2)$$

$$\text{subject to } \begin{cases} y_n = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b + e_n, \text{ for } n = 1, \dots, N, \end{cases} \quad (5.3)$$

as already seen in Eqs. (2.21) and (2.22), where  $e_n = y_n - f(\mathbf{x}_n)$  is the  $n$ -th error.

Mostly, the problem in Eqs. (5.2)-(5.3) can be solved by constructing the Lagrangian, making it possible to rewrite the primal optimization problem in dual space as

$$\begin{aligned} \mathcal{L}(\mathbf{w}, b, \mathbf{e}, \boldsymbol{\alpha}) &:= J_p(\mathbf{w}, \mathbf{e}) - \sum_{n=1}^N \alpha_n [\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b + e_n - y_n], \\ &= \frac{1}{2} \|\mathbf{w}\|^2 + \gamma \frac{1}{2} \sum_{n=1}^N e_n^2 - \sum_{n=1}^N \alpha_n [\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b + e_n - y_n]. \end{aligned} \quad (5.4)$$

Then, the solution of Eq. (5.4) can be obtained by the KKT conditions for optimality, and by solving the following linear system of  $N + 1$  equations:

$$\underbrace{\begin{bmatrix} 0 & \mathbf{1}_N^\top \\ \mathbf{1}_N & \mathbf{K} + \gamma^{-1} \mathbf{I}_N \end{bmatrix}}_{\mathbf{\Omega}} \underbrace{\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix}}_{\boldsymbol{\alpha}_o} = \underbrace{\begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix}}_{\mathbf{y}_o}, \quad (5.5)$$

where  $\mathbf{y} = [y_1, \dots, y_N]^\top$ ,  $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_N]^\top$  and  $\mathbf{K} \in \mathbb{R}^{N \times N}$  is the kernel matrix. The solution of Eq. (5.5) is computed using Eq. (2.30) as

$$\boldsymbol{\alpha}_o = \mathbf{\Omega}^\dagger \mathbf{y}_o, \quad (5.6)$$

where  $\mathbf{\Omega}^\dagger = (\mathbf{\Omega}^\top \mathbf{\Omega})^{-1} \mathbf{\Omega}^\top$ .

### 5.2.1 The RLM-SVR Model

From now on, instead of modifying the LSSVR cost function by inserting a weighting procedure, as done for the W-LSSVR and IR-LSSVR models, the proposed RLM-SVR approach works with the same standard LSSVR formulation in Eqs. (5.2) and (5.3). Therefore, the starting point for developing the RLM-SVR model is the previous linear system of Eq. (5.5) in dual space, simply expressed by  $\mathbf{\Omega} \boldsymbol{\alpha}_o = \mathbf{y}_o$ .

The RLM-SVR model proposes a recursive estimation of the model solution  $\boldsymbol{\alpha}_o$ , which encompasses the vector of Lagrange multipliers  $\boldsymbol{\alpha}$  and the bias  $b$ , instead of using the batch mode procedure of Eq. (5.6). Hence, we consider the matrix  $\mathbf{\Omega}$  as an array of column vectors expressed by  $\mathbf{\Omega} = [\boldsymbol{\omega}_1 | \dots | \boldsymbol{\omega}_n | \dots | \boldsymbol{\omega}_{N+1}] \in \mathbb{R}^{(N+1) \times (N+1)}$ , where  $\boldsymbol{\omega}_n \in \mathbb{R}^{N+1}$  denotes the  $n$ -th input vector. Then, for each value of  $n = 1, \dots, N + 1$ , we should compute  $\boldsymbol{\alpha}_o^{r(n)}$ , which is a robust version of  $\boldsymbol{\alpha}_o$ , after the presentation of the  $n$ -th input vector  $\boldsymbol{\omega}_n$ , whose corresponding output is  $y_o^{(n)}$ .

In order to robustify the RLM-SVR solution against the hostile effect of outliers, we decide to apply a linear, robust and recursive algorithm to update  $\boldsymbol{\alpha}_o^{r(n)}$ . The chosen option was the RLM algorithm, presented by Zou *et al.* (2000), which is based on the M-estimators theory and is derived from the RLS algorithm. Because of that, the RLM features, such as initial convergence, steady-state error and computational complexity, are similar to the ones of the RLS algorithm in the Gaussian noise cases (ZOU *et al.*, 2000).

Different from the SSE cost function of the RLS algorithm (see Eq. (4.11)), the

resulting cost function for the RLM algorithm is given by

$$J_\rho(e_n) = \sum_{i=0}^n \lambda^{n-i} \rho(e_i), \quad (5.7)$$

where  $0 \ll \lambda < 1$  is the forgetting factor and  $\rho(\cdot)$  is an  $M$ -estimate function. In this case, the Hampel's three-part function (ROUSSEEUW; LEROY, 1987) was originally adopted by Zou *et al.* (2000) for the RLM algorithm, and is computed by

$$\rho(e_n) = \begin{cases} \frac{e_n^2}{2} & 0 \leq |e_n| < t_1, \\ t_1|e_n| - \frac{t_1^2}{2} & t_1 \leq |e_n| < t_2, \\ \frac{t_1}{2}(t_3 + t_2) - \frac{t_1^2}{2} + \frac{t_1}{2} \frac{(|e_n| - t_3)^2}{t_2 - t_3} & t_2 \leq |e_n| < t_3, \\ \frac{t_1}{2}(t_3 + t_2) - \frac{t_1^2}{2} & |e_n| \geq t_3, \end{cases} \quad (5.8)$$

where  $t_1$ ,  $t_2$  and  $t_3$  are threshold parameters that need to be estimated continuously. In order to obtain 95% of confidence to down weight the error signal in the interval  $[t_1, t_2]$ , 97.5% of confidence to down weight the error in the interval  $[t_2, t_3]$ , and 99% of confidence to reject it when  $|e_n| > t_3$ , these thresholds are determined as  $t_1 = 1.96\hat{\sigma}_n$ ,  $t_2 = 2.24\hat{\sigma}_n$  and  $t_3 = 2.576\hat{\sigma}_n$ , where  $\hat{\sigma}_n$  is the standard deviation of the "impulse-free" estimation error, which is treated below.

Since the distribution of the signal error  $\mathbf{e} = \{e_n\}_{n=1}^{N+1}$  is in general unknown, one assumes, for simplicity, that it is Gaussian distributed possibly corrupted with some impulsive noise. Thus, according to Zou *et al.* (2000), the error variance  $\sigma_n^2$  at the step  $n$  is estimated as follows:

$$\hat{\sigma}_n^2 = \lambda_e \hat{\sigma}_{n-1}^2 + c(1 - \lambda_e) \text{med}(F_n), \quad (5.9)$$

where  $0 \ll \lambda_e \leq 1$  is a forgetting factor,  $\text{med}(\cdot)$  is the median operator and  $F_n = \{e_n^2, e_{n-1}^2, \dots, e_{n-N_w+1}^2\}$ . Moreover,  $N_w$  is the fixed window length for the median operation and  $c = 1.483(1 + 5/(N_w - 1))$  is the estimator's correction factor.

In order to search for the optimal solution in Eq. (5.7), it is necessary to compute the gradient vector,  $\nabla_{\boldsymbol{\alpha}_o^{r(n)}} J_\rho(e_n) = \partial J_\rho(e_n) / \partial \boldsymbol{\alpha}_o^{r(n)}$ , and set it to zero. Then, in a similar procedure to the RLS algorithm in Eq. (4.17), one gets

$$\boldsymbol{\alpha}_o^{r(n)} = (\mathbf{R}_\rho^{(n)})^{-1} \mathbf{p}_\rho^{(n)}, \quad (5.10)$$

with

$$\mathbf{R}_\rho^{(n)} = \sum_{i=0}^n \lambda^{n-i} v(e_i) \boldsymbol{\omega}_i \boldsymbol{\omega}_i^\top, \quad (5.11)$$

$$\mathbf{p}_\rho^{(n)} = \sum_{i=0}^n \lambda^{n-i} v(e_i) y_o^{(i)} \boldsymbol{\omega}_i, \quad (5.12)$$

where  $\mathbf{R}_\rho^{(n)} \in \mathbb{R}^{(N+1) \times (N+1)}$  and  $\mathbf{p}_\rho^{(n)} \in \mathbb{R}^{N+1}$  are called the  $M$ -estimate correlation matrix of  $\boldsymbol{\omega}_n$  and the  $M$ -estimate cross-correlation vector of  $\boldsymbol{\omega}_n$  and  $y_o^{(n)}$ , respectively. The weights  $v(e_i)$  are computed according to the  $M$ -estimate function as  $v(e_i) = \psi(e_i)/e_i$ , where  $\psi(e_i) = d\rho(e_i)/de_i$  is the score function, as discussed in Section 3.2.

Then, in order to solve Eq. (5.10) for the RLM-SVR model, it is necessary to compute the inverse matrix  $(\mathbf{R}_\rho^{(n)})^{-1}$  recursively. This is obtained by defining  $\mathbf{S}_\rho^{(n)} = (\mathbf{R}_\rho^{(n)})^{-1}$  and applying the matrix inversion lemma (Eq. (4.18)), as done for the RLS algorithm (see Section 4.2.3), such as

$$\mathbf{S}_\rho^{(n)} = \lambda^{-1}(\mathbf{I}_{N+1} - \mathbf{g}_n \boldsymbol{\omega}_n^\top) \mathbf{S}_\rho^{(n-1)}, \quad (5.13)$$

where  $\mathbf{g}_n \in \mathbb{R}^{N+1}$  is the  $M$ -estimate gain vector, defined by

$$\mathbf{g}_n = \frac{v(e_n) \mathbf{S}_\rho^{(n-1)} \boldsymbol{\omega}_n}{\lambda + v(e_n) \boldsymbol{\omega}_n^\top \mathbf{S}_\rho^{(n-1)} \boldsymbol{\omega}_n}. \quad (5.14)$$

Then, the RLM-SVR robust solution at each step  $n$  is computed as

$$\boldsymbol{\alpha}_o^{r(n)} = \boldsymbol{\alpha}_o^{r(n-1)} + (y_o^{(n)} - \boldsymbol{\omega}_n^\top \boldsymbol{\alpha}_o^{r(n-1)}) \mathbf{g}_n. \quad (5.15)$$

Therefore, the RLM-SVR model corresponds to sequentially applying the adaptive Eqs. (5.13), (5.14) and (5.15). One should note that its computational complexity is  $\mathcal{O}(N^3)$  and its memory demand is  $\mathcal{O}(N^2)$ , which are the same ones of the standard LSSVR, W-LSSVR and IR-LSSVR models.

As a final remark, it is known that the maximum number of iterations in RLM-SVR model is  $N + 1$ , which corresponds to vanish the whole training dataset. However, it may be necessary to cycle over the data samples  $N_e > 1$  times in order to ensure the convergence of the algorithm. The resulting RLM-SVR pseudo-code is summarized in Algorithm 10, where  $\delta$  is a pre-established positive parameter, similar to the one defined for the RLS algorithm.

### 5.2.2 Computational Experiments

In this section, we report and discuss the results of comprehensive computer simulations involving the application of the proposed RLM-SVR approach to nonlinear system identification tasks in the presence of outliers. For the sake of completeness, we compare the RLM-SVR performance to the ones obtained by the standard LSSVR model and two of its robust variants in the dual space, the W-LSSVR and IR-LSSVR models. Following, we describe

---

**Algorithm 10:** Pseudo-code for the RLM-SVR model.
 

---

**Require:**  $\gamma, \sigma$  (for Gaussian kernel),  $\delta, N_e, \lambda, \lambda_e, N_w$   
 Build the linear system of Eq. (5.5); % LSSVR linear system  
 Set  $\mathbf{S}_\rho^{(0)} = \delta \mathbf{I}, \mathbf{g}_0 = \boldsymbol{\alpha}_o^{r(0)} = \mathbf{0}$ ;  
**for**  $i = 1 : N_e$ , **do**  
   **for**  $n = 1 : N + 1$ , **do**  
      $e_n = y_o^{(n)} - \boldsymbol{\omega}_n^\top \boldsymbol{\alpha}_o^{r(n-1)}$ ;  
     Compute  $\hat{\sigma}_n^2$  from Eq. (5.9);  
     Compute  $v(e_n) = \frac{1}{e_n} \frac{d\rho(e_n)}{de_n}$  from Eq. (5.8);  
     Compute  $\mathbf{S}_\rho^{(n)}$  from Eq. (5.13);  
     Compute  $\mathbf{g}_n$  from Eq. (5.14);  
     Compute  $\boldsymbol{\alpha}_o^{r(n)}$  from Eq. (5.15);  
   **end for**  
**end for**  
 Output:  $\boldsymbol{\alpha}_o^{r(n)}$ .

---

in detail the experimental setup and methodology, the procedure of outlier contamination, the evaluated datasets and the performance results.

1. Methodology - Firstly, as usual in Machine Learning applications, the evaluated samples of each dataset are split into training and test sets. Then, we perform a *5-fold cross-validation* strategy, in an automatic model selection procedure, in the search for optimal values for the hyperparameters of each model as well as the input and output regression orders  $\hat{L}_u$  and  $\hat{L}_y$ .

The standard LSSVR model has, in general, two hyperparameters, the regularization term  $\gamma$  and the kernel width  $\sigma$  (for the Gaussian kernel). By the way, the Gaussian kernel is adopted from now on for all the evaluated models in this thesis. In addition, one should highlight that the W-LSSVR and IR-LSSVR models, as were originally proposed, use the same optimal hyperparameters searched for the standard LSSVR model. Therefore,  $\gamma$  is optimized from the range  $\{2^0, \dots, 2^{20}\}$  and  $\sigma$  from the range  $\{2^{-10}, \dots, 2^0\}$  for the RLM-SVR and LSSVR (W-LSSVR and IR-LSSVR) models.

Regarding the proposed RLM-SVR model, besides  $\gamma$  and  $\sigma$ , we still optimize the forgetting factor  $\lambda$  from the range  $\{0.99, 0.999, 0.9999\}$ . Its other parameters are fixed at  $\lambda_e = 0.95$ ,  $N_w = 14$ ,  $\delta = 1,000$  and, based on some preliminary simulations, we also set  $N_e = 20$  for all the experiments, in order to ensure convergence in the training phase.

For the synthetic datasets, the input and output regression orders ( $\hat{L}_u$  and  $\hat{L}_y$ ) are set according to their largest delays, as inferred from Eqs. (5.16) and (5.17). On the other

hand, for the real-world datasets,  $\hat{L}_u$  and  $\hat{L}_y$  are optimized from the range  $\{1, 2, 3, 4, 5\}$  (via 5-fold cross-validation strategy) for the standard LSSVR model. Then, their resulting values for the Actuator and Dryer datasets are also used for all the other models. The Actuator and Dryer datasets are properly described in next subsection.

The figure of merit for evaluating the numerical performance of the RLM-SVR model is the RMSE value computed for the  $N'$  test samples over 20 independent runs, since this model is sensitive to the initial conditions. However, the other evaluated models (LSSVR, W-LSSVR and IR-LSSVR) solve a convex optimization problem and, consequently, they do not present dispersion of the RMSE values over the runs.

2. **Outlier Contamination** - For the experiments in robust system identification, in addition to the standard Gaussian noise added to training samples (for the synthetic datasets), we also contaminate the outputs of the training data by replacing them randomly with outliers covering 0%, 5%, 10%, 15%, 20%, 25% and 30% of the total of estimation samples. We use the same procedure to contaminate with 10% of outliers the training outputs of the real-world datasets.

The adopted outlier generation methodology is similar to the one performed by Mattos *et al.* (2016), who considered a non-Gaussian noise model. For that matter, outliers were generated by sampling from a Student's  $t$ -distribution with zero mean and 2 degrees of freedom, whose probability density function (PDF) is illustrated in Fig. (8). By doing this, we guarantee that the distribution of the residuals (i.e. estimation errors) is non-Gaussian<sup>1</sup>. One can see in Fig. (8) that the  $t$ -distribution (in blue color) is symmetric and bell-shaped like the normal distribution (in red color), but it has heavier tails, meaning that ranges of too large or too small values have higher probability to occur.

### 5.2.2.1 Evaluated Datasets

For the experiments in system identification tasks, we use two synthetic datasets (Synthetic-1 and Synthetic-2) and two real-world datasets (Actuator and Dryer), from benchmark plants, which are described below.

The first synthetic dataset (Synthetic-1) is reported in Kocijan *et al.* (2005), being

---

<sup>1</sup>In real-world applications outliers are unknown in advance, but for the sake of evaluating the robustness of estimation algorithms it is common approach to introduce fake outliers in the form of samples whose distribution does not follow the normal distribution. Some estimation algorithms, such as the ordinary least squares (OLS), are optimal only under the assumption of Gaussianity of the residuals. If this assumption is not met, the performance of the corresponding model degrades considerably.

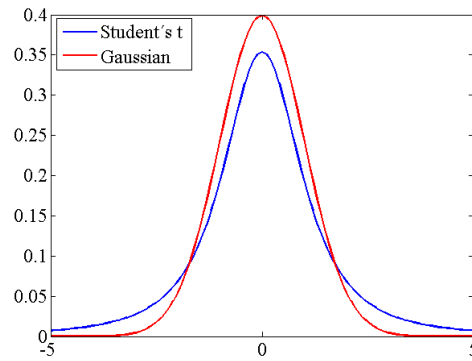


Figure 8 – Probability density functions of the Student's t and Gaussian distributions.

generated according to the following discrete-time dynamical equation:

$$y_n = y_{n-1} - 0.5 \tanh(y_{n-1} + u_{n-1}^3), \quad (5.16)$$

with  $u_n$  drawn from a truncated Gaussian distribution (in the  $\pm 1$  range) of zero mean and unity variance. The corresponding input  $u_n$  and output  $y_n$  sequences of the Synthetic-1 dataset are illustrated in Fig. (9) where, of a total of 300 samples, we separate the first 150 of each sequence  $u_n$  and  $y_n$  for training and the remaining for testing. Moreover, the training samples are further corrupted with additive Gaussian noise with zero mean and variance 0.0025.

The second synthetic dataset (Synthetic-2) was introduced in the seminal work of Narendra and Parthasarathy (1990), which is given by the following equation:

$$y_n = \frac{y_{n-1}}{1 + y_{n-1}^2} + u_{n-1}^3, \quad (5.17)$$

where

$$u_n = \begin{cases} U(-2, 2), & \text{for training data} \\ \sin(2\pi n/25) + \sin(2\pi n/10), & \text{for test data} \end{cases}$$

where  $U(-2, 2)$  denotes a uniformly distributed random number in the specified range. Its input and output sequences are shown in Fig. (10), where the first 300 samples of each sequence are used for training and the remaining for test. The training data is also corrupted with additive Gaussian noise with zero mean and variance 0.65.

The two real-world datasets, called *Actuator* and *Dryer*, used in the experiments are described in Sjöberg *et al.* (1995) and Ljung (1987), respectively, and are publicly available for download from the DaISy repository website at the *Katholieke Universiteit Leuven*<sup>2</sup>. The Actuator dataset corresponds to a SISO system, where the position of a robot arm is controlled

<sup>2</sup><http://www.iau.dtu.dk/nbook/systems.html>

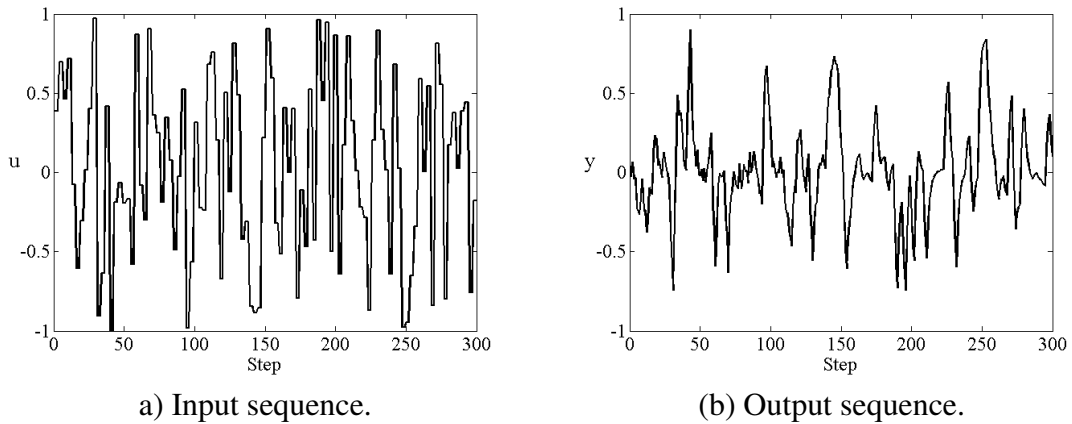


Figure 9 – Input and output sequences of the Synthetic-1 dataset.

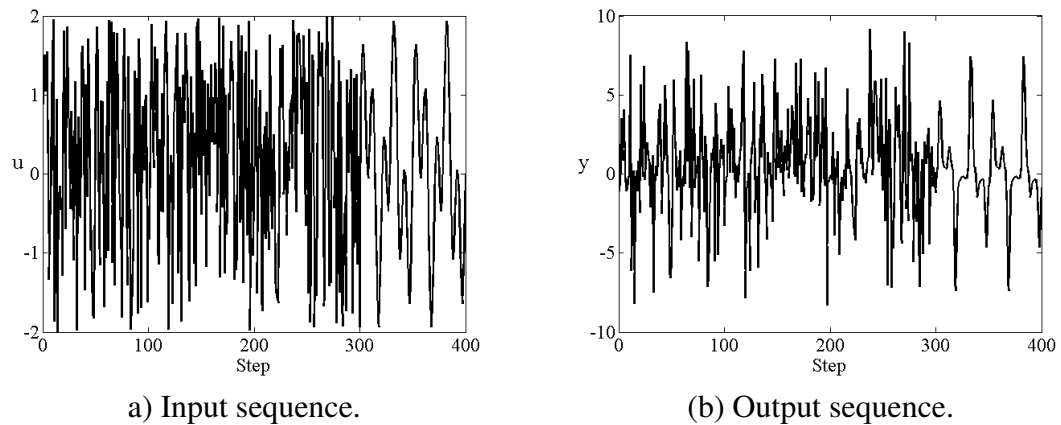


Figure 10 – Input and output sequences of the Synthetic-2 dataset.

by a hydraulic actuator. In this case, the input  $u_n$  is the size of the actuator's valve opening and the output  $y_n$  is the corresponding oil pressure. From the total of 1,024 samples of each time series  $u_n$  and  $y_n$ , which are shown in Fig. (11), we used 512 samples for model building (i.e. training) and the other half for model evaluation (i.e. testing the predictor's performance).

The Dryer dataset corresponds to a mechanical SISO system from a laboratory setup acting like a hair dryer. In this system, the air temperature  $y_n$  is measured by a thermocouple at the output. The input  $u_n$  is the voltage over the heating device. From the 1,000 available samples of  $u_n$  and  $y_n$ , which are illustrated in Fig. (12), we use the first half for model building and the remaining for model evaluation.

### 5.2.2.2 Results and Discussion

In order to assess the performance of the proposed approach in nonlinear system identification for k-step-ahead prediction and in the presence of outliers, we carried out computer experiments using the above described datasets. For the two synthetic datasets (Synthetic-1 and



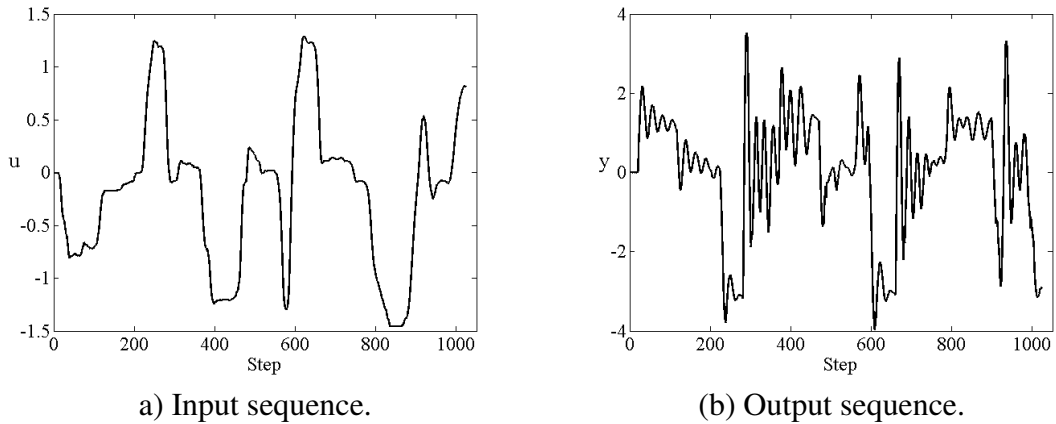


Figure 11 – Input and output sequences of the Actuator dataset.

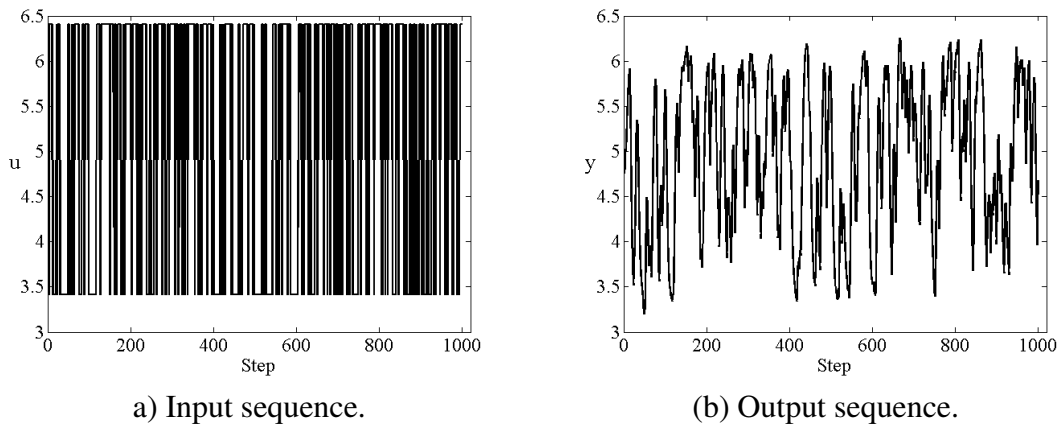


Figure 12 – Input and output sequences of the Dryer dataset.

Table 5 – Important features of the evaluated datasets and models for the computational experiments.

<b>Important Features of the Evaluated Datasets</b>					
Dataset	Prediction Scenario	Training ( $N$ )	Test ( $N'$ )	$\hat{L}_u$	$\hat{L}_y$
Synthetic-1	Free Simulation	150	150	1	1
Synthetic-2	Free Simulation	300	100	1	1
Actuator	3-step ahead	512	512	4	4
Dryer	Free Simulation	500	500	5	5

Synthetic-2) and for the Dryer dataset, we set  $k \rightarrow +\infty$ , which correspond to a free simulation scenario. For the Actuator dataset, we set  $k = 3$ , based on the obtained performance in previous experiments. Some important features of the datasets and of the corresponding evaluated models are summarized in Table 5.

#### 5.2.2.2.1 Experiments with Synthetic Datasets

We report in Fig. (13) the RMSE values obtained by each evaluated model for the synthetic datasets with increasing scenarios from 0% to 30% of outliers contamination. Error bars corresponding to one standard deviation below and above the mean are shown.

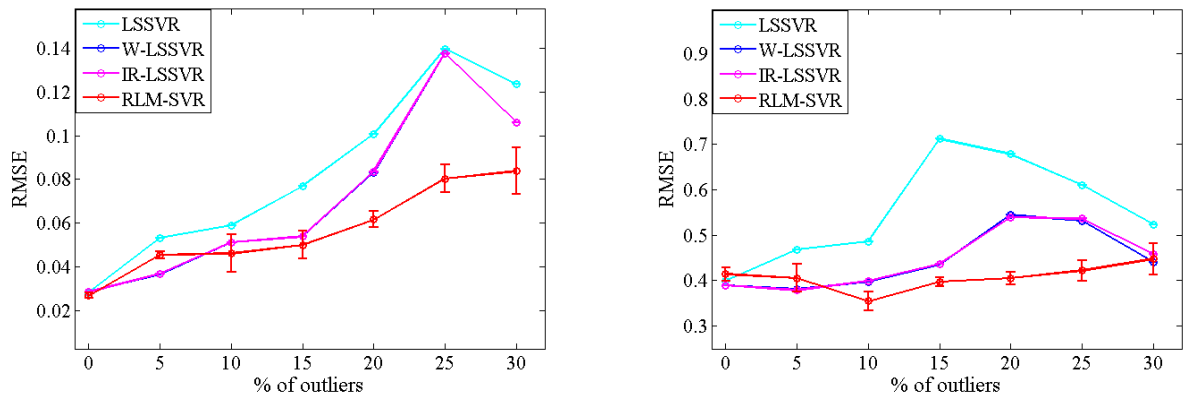
More specifically, the RMSE values produced by the evaluated models for the Synthetic-1 dataset are shown in Fig. (13a). A closer analysis of these results reveals that for almost all the scenarios of contamination, the proposed RLM-SVR model achieved much lower values of RMSE in comparison to the other models. This is particularly true for the scenarios with high levels of outlier contamination (higher than 15% of outliers), where even presenting some variance for the RMSE values over the runs, the maximum values achieved by the RLM-SVR model were lower than those obtained by the other (robust) approaches.

The RMSE values for the Synthetic-2 dataset are shown in Fig. (13b). It can be easily noted that, except for the scenarios up 5% and with 30% of outliers contamination, the RLM-SVR model achieved the lowest values of RMSE among the evaluated models. A point to be highlighted in the RLM-SVR performance is that, as already occurred for the Synthetic-1 and now for the Synthetic-2 dataset, it has become better (compared to the other models) as the number of outliers increased. This is an evidence that the recursive updating procedure of the RLM-SVR model, which individually treats each input sample eventually contaminated with outliers, may be the responsible for the improvement in the quality of the solution.

#### 5.2.2.2.2 Experiments with Real-World Datasets

In the next set of experiments, we assess the achieved results obtained by the evaluated models for the Actuator dataset in 3-step-ahead prediction considering two cases: (i) outlier-free scenario, and (ii) scenario with 10% of outliers. For each case we compare the best state-of-the-art model (LSSVR, W-LSSVR and IR-LSSVR), i.e. the one that achieves the lowest RMSE value, with the proposed RLM-SVR model.

We report in Table 6 the RMSE values obtained by each model for the Actuator dataset. For the outlier-free scenario, one can see that the W-LSSVR model, which presented the lowest RMSE values among the state-of-the-art models, achieved a better performance when compared to the RLM-SVR model. On the other hand, for the scenario with 10% of outliers, even with a certain dispersion in the RMSE values, the RLM-SVR model presented lower RMSE



a) Synthetic-1 dataset.

(b) Synthetic-2 dataset.

Figure 13 – RMSE values for the test samples over 20 independent runs in free simulation.

values than the W-LSSVR model (the best one among the LSSVR, W-LSSVR and IR-LSSVR models).

In order to assess the effect of the outliers in the prediction performance of the models, we report the worst-case results in Fig. (14) for the predicted outputs of the models whose results are in bold in Table 6. In other words, we report the predicted outputs that led to the highest RMSE values among the 20 runs, for both scenarios.

A careful analysis of Figs. (14a) and (14b), for the case without outliers, reveals that the W-LSSVR and RLM-SVR models produced equivalent predictions, except between the steps 300 and 400, where the W-LSSVR predicted output is closer to the real one. However, for the case with 10% of outliers showed in Figs. (14c) and (14d), it becomes evident the superiority of the proposed approach. Even considering the output that obtained the higher value of RMSE, the RLM-SVR model outperformed the W-LSSVR model, mainly along the top peaks of the signal between the time steps 150 – 250 and 400 – 500, where its predictions are closer to the real output than the W-LSSVR ones.

The obtained results from the experiments with the Dryer dataset in free simulation are shown in Table 7 and Fig. (15). In Table 7 we report the RMSE values for each evaluated model, where it is possible to note that, for the outlier-free scenario, the RLM-SVR model achieved lower performance than the other models. For the outlier-corrupted scenario, again the RLM-SVR proposed approach achieved the lowest RMSE values (even with a certain dispersion) among the evaluated models.

In Fig. (15) we report the worst predicted outputs obtained from the models whose results are in bold in Table 7. For the scenario without outliers, it is possible to observe in Figs. (15a) and (15b) that the predicted outputs for the LSSVR and RLM-SVR models were quite

Table 6 – RMSE for the test samples over 20 independent runs in 3-step-ahead prediction scenario - Actuator dataset.

Models	0% of outliers	10% of outliers
	RMSE	RMSE
LSSVR	2.39E-1 ± 0.00E0	6.30E-1 ± 0.00E0
W-LSSVR	<b>2.36E-1 ± 0.00E0</b>	<b>6.21E-1 ± 0.00E0</b>
IR-LSSVR	2.37E-1 ± 0.00E0	6.21E-1 ± 0.00E0
RLM-SVR	<b>2.67E-1 ± 1.70E-3</b>	<b>5.79E-1 ± 5.10E-3</b>

close to each other and they were able to follow, in general, the dynamical behavior of the real output. In the presence of outliers, we can observe in Figs. (15c) and (15d) that the predictions of both robust models (W-LSSVR and RLM-SVR) did not suffer a notable degradation with the insertion of outliers.

Finally, as we could see in the obtained results of the computational experiments, the achieved performances of the RLM-SVR proposal were in general suitable, since it outperformed the other robust models in most scenarios with outliers. This is more one evidence that the recursive procedure of the RLM-SVR model may improve the robustness of its solution against outliers, through the individual treatment of each input sample possibly contaminated. However, the RLM-SVR approach has inherited from the standard LSSVR model the non-sparsity of its solution. This can be a hindrance for applications that require the use of big datasets, for example. Furthermore, as well as the IR-LSSVR model, the robustness of the RLM-SVR model is obtained after a relatively large number of iterations, which makes the training phase computationally expensive.

From the discussion above, the next section presents two novel robust approaches derived from the LSSVR formulation in the primal space, whose solutions preserve, at the same time, robustness against outliers and sparsity.

### 5.3 Proposed Robust Approaches in Primal Space

The dual formulation expressed in Eq. (5.4), and earlier in Eq. (2.23), is by far the most common strategy for building LSSVR models. However, in applications for which the processing of large amount of data is mandatory, it may be advantageous to work in primal space instead. One clear advantage is that solving the primal optimization problem in Eqs. (5.2)-(5.3) corresponds to estimate directly the parameter vector  $\mathbf{w} \in \mathbb{R}^d$  to be used in Eq. (5.1) for the prediction. It turns out to be that the dimension of  $\mathbf{w}$  is usually much smaller than the dimension

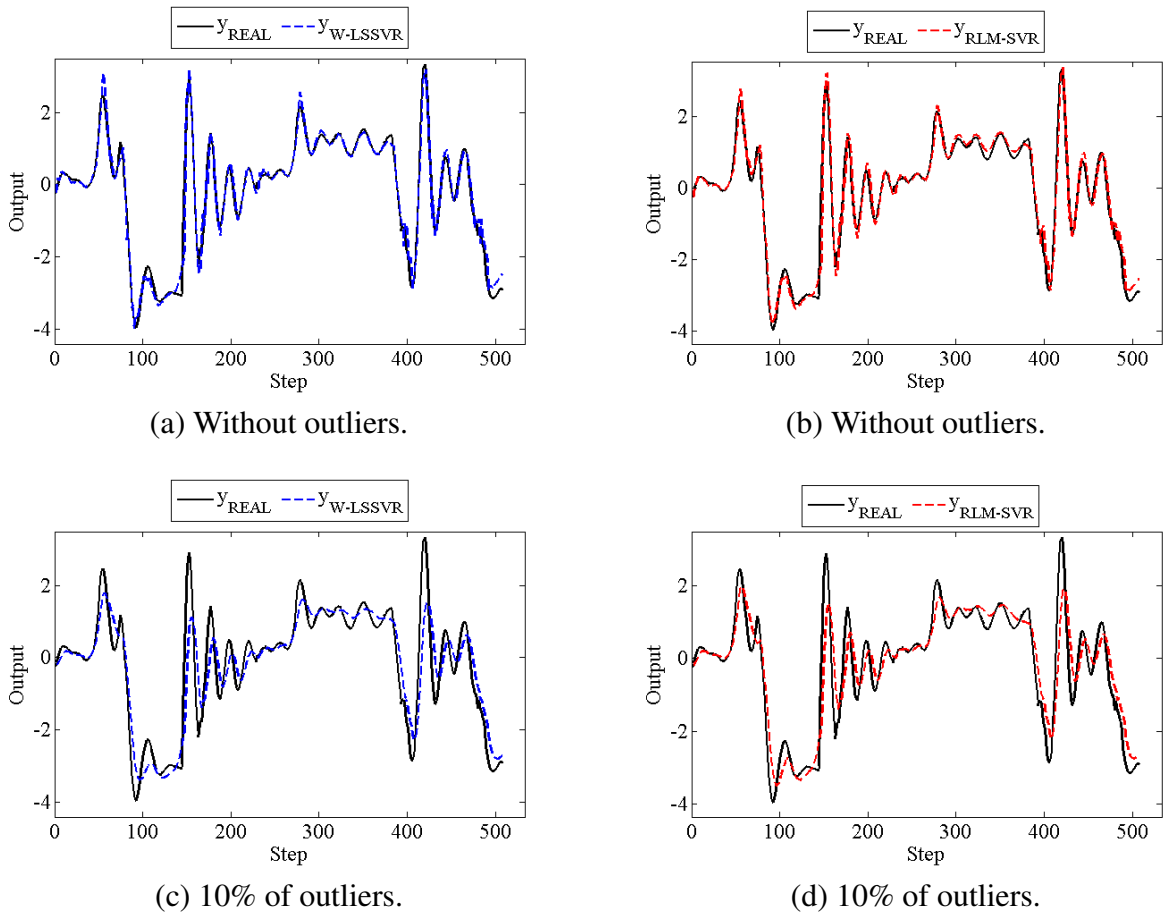


Figure 14 – Predicted outputs by the best models in Table 6 with their worst performances in RMSE over the 20 runs - Actuator dataset.

of  $\alpha \in \mathbb{R}^N$ .

In order to adopt the primal optimization problem, explicit knowledge of the nonlinear feature mapping  $\phi$  is required; 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*, where the inner product in feature space  $\phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$  can be exactly computed by means of kernel function evaluations, such as for the Gaussian kernel.

The kernel trick though cannot be used in the primal problem. In this case, it is required either an explicit expression for  $\phi$  (only possible for the linear or polynomial case) or an approximation to the feature map  $\hat{\phi}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^M$ , where  $d$  is the space dimension of  $\mathbf{x}_n$ . This approximation is carried out using a subset of  $M$  ( $M \ll N$ ) prototype vectors (PV) extracted from the whole training dataset, which is obtained by the Nyström method (BAKER, 1977; WILLIAMS; SEEGER, 2001) as

$$\hat{\phi}_i(\mathbf{x}') = \frac{1}{\sqrt{\lambda_i}} \sum_{m=1}^M (u_i)_m k(\mathbf{z}_m, \mathbf{x}'), \text{ for } i = 1, \dots, M, \quad (5.18)$$

Table 7 – RMSE for the test samples over 20 independent runs in free simulation scenario - Dryer dataset.

Models	0% of outliers	10% of outliers
	RMSE	RMSE
LSSVR	<b>1.12E-1 ± 0.00E0</b>	1.48E-1 ± 0.00E0
W-LSSVR	1.13E-1 ± 0.00E0	<b>1.41E-1 ± 0.00E0</b>
IR-LSSVR	1.13E-1 ± 0.00E0	1.41E-1 ± 0.00E0
RLM-SVR	<b>1.24E-1 ± 1.30E-3</b>	<b>1.29E-1 ± 2.80E-3</b>

where  $\lambda_i$  are the eigenvalues and  $\mathbf{u}_i$  the respective eigenvectors of the reduced kernel matrix  $\bar{\mathbf{K}} \in \mathbb{R}^{M \times M}$ , with  $\bar{K}_{ij} = k(\mathbf{z}_i, \mathbf{z}_j)$ . The vectors  $\mathbf{z}_i$  and  $\mathbf{z}_j$  belong to the subset of  $M$  PV, randomly selected.

In order to avoid a simple random selection of  $M$  prototype vectors, one can choose a procedure based on an approximation  $H_R$  of quadratic Rényi's entropy, computed by (SUYKENS *et al.*, 2002b)

$$H_R = \frac{1}{M^2} \mathbf{1}_M^\top \bar{\mathbf{K}} \mathbf{1}_M, \quad (5.19)$$

where  $\mathbf{1}_M = [1, \dots, 1]^\top$ . By using such criterion, one can be sure that the selected subsample is well spread over the entire data region.

Conceptually, the general formulation of the LSSVR problem solved in primal space, called FS-LSSVR model, was properly discussed in Chapter 2. However, we repeat next only the main steps for a better understanding in deriving our proposals. Thus, the FS-LSSVR model consists in minimizing the following functional:

$$J_p(\hat{\mathbf{w}}, \hat{b}) = \frac{1}{2} \hat{\mathbf{w}}^\top \hat{\mathbf{w}} + \gamma \frac{1}{2} \sum_{n=1}^N (y_n - \hat{\mathbf{w}}^\top \hat{\boldsymbol{\phi}}(\mathbf{x}_n) - \hat{b})^2, \quad (5.20)$$

where  $\hat{\boldsymbol{\phi}}(\mathbf{x}_n) = [\hat{\phi}_1(\mathbf{x}_n), \dots, \hat{\phi}_M(\mathbf{x}_n)]^\top \in \mathbb{R}^M$ ,  $\hat{\mathbf{w}} \in \mathbb{R}^M$  and  $\hat{b} \in \mathbb{R}$  are the approximate solutions for  $\mathbf{w}$  and  $b$ , respectively. Thus, by constructing an approximated feature matrix for all the training instances as

$$\hat{\boldsymbol{\Phi}} = \begin{bmatrix} \hat{\phi}_1(\mathbf{x}_1) & \dots & \hat{\phi}_M(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \hat{\phi}_1(\mathbf{x}_N) & \dots & \hat{\phi}_M(\mathbf{x}_N) \end{bmatrix}_{N \times M} = \begin{bmatrix} \hat{\boldsymbol{\phi}}^\top(\mathbf{x}_1) \\ \vdots \\ \hat{\boldsymbol{\phi}}^\top(\mathbf{x}_N) \end{bmatrix}_{N \times M}, \quad (5.21)$$

and solving the primal problem in Eq. (5.20) with the approximated feature matrix  $\hat{\boldsymbol{\Phi}}$  leads to

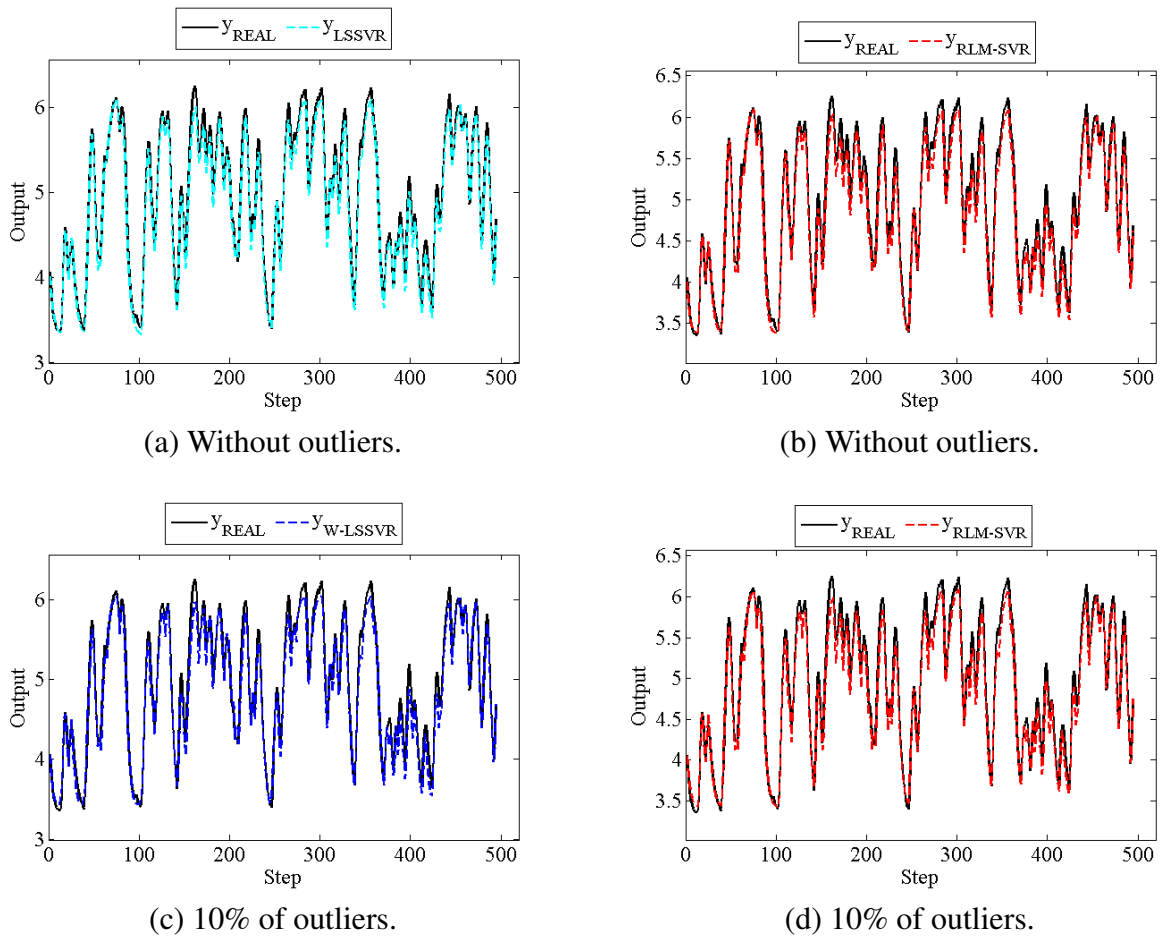


Figure 15 – Predicted outputs by the best models in Table 7 with their worst performances in RMSE over the 20 runs - Dryer dataset.

the following linear system of  $M + 1$  equations:

$$\underbrace{\begin{bmatrix} \hat{\Phi}^\top \hat{\Phi} + \frac{1}{\gamma} \mathbf{I}_M & \hat{\Phi}^\top \mathbf{1}_N \\ \mathbf{1}_N^\top \hat{\Phi} & \mathbf{1}_N^\top \mathbf{1}_N \end{bmatrix}}_{(M+1) \times (M+1)} \underbrace{\begin{bmatrix} \hat{w} \\ \hat{b} \end{bmatrix}}_{(M+1) \times 1} = \underbrace{\begin{bmatrix} \hat{\Phi}^\top \mathbf{y} \\ \mathbf{1}_N^\top \mathbf{y} \end{bmatrix}}_{(M+1) \times 1}, \quad (5.22)$$

as already obtained in Eq. (2.49).

From now, our goal is to develop robust LSSVR predictor models derived from its primal space formulation.

### 5.3.1 The RFS-LSSVR Model

The starting point for the development of our first primal approach, called the Robust FS-LSSVR (RFS-LSSVR) model, is the previous FS-LSSVR linear system given by Eq. (5.22). In order to make it robust to outliers we adapt to the primal problem the robust approach previously developed by Suykens *et al.* (2002a) for the dual problem.

For that sake, we collect the prediction error values ( $e_n = y_n - \hat{\mathbf{w}}^\top \hat{\boldsymbol{\phi}}(\mathbf{x}_n) - \hat{b}$ ) produced by the FS-LSSVR model and compute the corresponding weighting factors  $v_n$  using the Hampel's weight function (ROUSSEEUW; LEROY, 1987) as follows:

$$v_n = \begin{cases} 1, & \text{if } |e_n/\hat{s}| \leq c_1 \\ \frac{c_2 - |e_n/\hat{s}|}{c_2 - c_1}, & \text{if } c_1 < |e_n/\hat{s}| \leq c_2 \\ 10^{-4}, & \text{otherwise.} \end{cases} \quad (5.23)$$

where  $c_1 = 2.5$ ,  $c_2 = 3.0$  and  $\hat{s} = \text{IQR}/1.349$  is a robust estimate of the standard deviation of the FS-LSSVR error variables  $e_n$ .

By doing so, the RFS-LSSVR optimization problem can be reformulated as

$$J_p(\hat{\mathbf{w}}^r, \mathbf{e}) = \frac{1}{2}(\hat{\mathbf{w}}^r)^\top \hat{\mathbf{w}}^r + \frac{\gamma}{2} \sum_{n=1}^N v_n e_n^2, \quad (5.24)$$

subject to

$$y_n = (\hat{\mathbf{w}}^r)^\top \hat{\boldsymbol{\phi}}(\mathbf{x}_n) + \hat{b}^r + e_n, \quad n = 1, \dots, N, \quad (5.25)$$

where  $\hat{\mathbf{w}}^r$  and  $\hat{b}^r$  are used to denote robust solutions of  $\hat{\mathbf{w}}$  and  $\hat{b}$ , respectively.

In order to solve Eq. (5.24) with the constraints of Eq. (5.25), one must first write it as

$$J_p(\hat{\mathbf{w}}^r, \hat{b}^r) = \frac{1}{2}(\hat{\mathbf{w}}^r)^\top \hat{\mathbf{w}}^r + \frac{\gamma}{2} \sum_{n=1}^N v_n (y_n - (\hat{\mathbf{w}}^r)^\top \hat{\boldsymbol{\phi}}(\mathbf{x}_n) - \hat{b}^r)^2, \quad (5.26)$$

and search for its solution by computing the gradient vectors,  $\nabla_{\hat{\mathbf{w}}^r} J_p(\hat{\mathbf{w}}^r, \hat{b}^r) = \partial J_p(\hat{\mathbf{w}}^r, \hat{b}^r) / \partial \hat{\mathbf{w}}^r$  and  $\nabla_{\hat{b}^r} J_p(\hat{\mathbf{w}}^r, \hat{b}^r) = \partial J_p(\hat{\mathbf{w}}^r, \hat{b}^r) / \partial \hat{b}^r$ , and setting them to zero. Then, we get

$$\frac{\partial J_p}{\partial \hat{\mathbf{w}}^r} = \hat{\mathbf{w}}^r - \gamma \sum_{n=1}^N v_n [y_n - ((\hat{\mathbf{w}}^r)^\top \hat{\boldsymbol{\phi}}(\mathbf{x}_n) + \hat{b}^r)] \hat{\boldsymbol{\phi}}(\mathbf{x}_n) = \mathbf{0}_M, \quad (5.27)$$

and

$$\frac{\partial J_p}{\partial \hat{b}^r} = -\gamma \sum_{n=1}^N v_n [y_n - ((\hat{\mathbf{w}}^r)^\top \hat{\boldsymbol{\phi}}(\mathbf{x}_n) + \hat{b}^r)] = 0, \quad (5.28)$$

where  $\mathbf{0}_M \in \mathbb{R}^M$  is a vector of zeros. The expressions in Eqs. (5.27) and (5.28) can be arranged in corresponding matrix forms, respectively, as

$$\frac{1}{\gamma} \hat{\mathbf{w}}^r + \hat{\boldsymbol{\Phi}}^\top \mathbf{V} \hat{\boldsymbol{\Phi}} \hat{\mathbf{w}}^r - \hat{\boldsymbol{\Phi}}^\top \mathbf{V} \mathbf{y} + \hat{\boldsymbol{\Phi}}^\top \mathbf{V} \mathbf{1}_N \hat{b}^r = \mathbf{0}_M,$$



$$\left( \hat{\Phi}^\top \mathbf{V} \hat{\Phi} + \frac{1}{\gamma} \mathbf{I}_M \right) \hat{\mathbf{w}}^r + \hat{\Phi}^\top \mathbf{V} \mathbf{1}_N \hat{\mathbf{b}}^r = \hat{\Phi}^\top \mathbf{V} \mathbf{y}, \quad (5.29)$$

and

$$-\mathbf{1}_N^\top \mathbf{V} \mathbf{y} + \mathbf{1}_N^\top \mathbf{V} \hat{\Phi} \hat{\mathbf{w}}^r + \mathbf{1}_N^\top \mathbf{V} \mathbf{1}_N \hat{\mathbf{b}}^r = 0,$$

$$\mathbf{1}_N^\top \mathbf{V} \hat{\Phi} \hat{\mathbf{w}}^r + \mathbf{1}_N^\top \mathbf{V} \mathbf{1}_N \hat{\mathbf{b}}^r = \mathbf{1}_N^\top \mathbf{V} \mathbf{y}, \quad (5.30)$$

with  $\mathbf{V} = \text{diag}\{v_n\}_{n=1}^N \in \mathbb{R}^{N \times N}$  being a diagonal matrix, whose diagonal elements are the weights  $v_n$ . For instance,  $\mathbf{V} = \mathbf{I}_N$  for the (unweighted) FS-LSSVR model.

Finally, to determine the RFS-LSSVR model solution, we can arrange the expressions in Eqs. (5.29) and (5.30) into a single linear system of  $M + 1$  equations, which is given by

$$\underbrace{\begin{bmatrix} \hat{\Phi}^\top \mathbf{V} \hat{\Phi} + \frac{1}{\gamma} \mathbf{I}_M & \hat{\Phi}^\top \mathbf{V} \mathbf{1}_N \\ \mathbf{1}_N^\top \mathbf{V} \hat{\Phi} & \mathbf{1}_N^\top \mathbf{V} \mathbf{1}_N \end{bmatrix}}_{(M+1) \times (M+1)} \underbrace{\begin{bmatrix} \hat{\mathbf{w}}^r \\ \hat{\mathbf{b}}^r \end{bmatrix}}_{(M+1) \times 1} = \underbrace{\begin{bmatrix} \hat{\Phi}^\top \mathbf{V} \mathbf{y} \\ \mathbf{1}_N^\top \mathbf{V} \mathbf{y} \end{bmatrix}}_{(M+1) \times 1}, \quad (5.31)$$

where one can easily see the similarities with the unweighted FS-LSSVR formulation in Eq. (5.22). Therefore, we can conclude that the computational complexity  $\mathcal{O}(NM^2)$  and memory demand  $\mathcal{O}(NM)$  of the RFS-LSSVR are the same ones of the original FS-LSSVR model. The RFS-LSSVR model is summarized in Algorithm 11.

### 5.3.2 The $R^2$ FS-LSSVR Model

In general, a single run of the RFS-LSSVR model should suffice to obtain good predictions in most practical applications. However, we have observed in practice that it may be convenient to re-run the RFS-LSSVR prediction model in the search for better performances, specially when the training samples are contaminated with a high number of outliers. Bearing this in mind, iterative application of the RFS-LSSVR model gives rise to the second robust primal approach proposed in this thesis, called the Reweighted RFS-LSSVR ( $R^2$ FS-LSSVR) model.

More specifically, at each iteration  $k$ , we need to compute the error variables  $e_n^{(k)} = y_n - (\hat{\mathbf{w}}^{r(k)})^\top \hat{\Phi}(\mathbf{x}_n) - \hat{\mathbf{b}}^{r(k)}$ . Then, the corresponding weights  $v_n^{(k)}$  are computed by means of

---

**Algorithm 11:** - Pseudo-code for the RFS-LSSVR model.

---

**Require:**  $M, \gamma, \sigma, N_{iter}$ ;  
 Choose a working set  $\mathcal{D}^{pv} = \{\mathbf{x}_m\}_{m=1}^M$ ;  
 Select  $\mathcal{D}_1$  with the  $N - M$  remaining training samples;  
**for**  $i = n_{iter} : N_{iter}$ , **do**  
   Compute  $H_R$  of  $\mathcal{D}^{pv}$  from (2.42);  
   Randomly select  $\mathbf{x}_m$  from  $\mathcal{D}^{pv}$ ;  
   Randomly select  $\mathbf{x}^*$  from  $\mathcal{D}_1$ ;  
   Replace  $\mathbf{x}_m$  by  $\mathbf{x}^*$  in  $\mathcal{D}^{pv}$ ;  
   Compute  $H_R^*$  of  $\mathcal{D}^{pv}$  from (2.42);  
   **if**  $H_R^* > H_R$ , **then**  
      $\mathbf{x}^*$  is accepted for  $\mathcal{D}^{pv}$ ;  
   **else**  
      $\mathbf{x}^*$  is rejected by  $\mathcal{D}^{pv}$ ;  
   **end if**  
**end for**  
 Compute  $\hat{\Phi}$  from Eq. (5.21);  
 Build the linear system of Eq. (5.22);  
 Compute  $\hat{\mathbf{w}}$  and  $\hat{b}$  from Eq. (5.6); % FS-LSSVR solution  
**for**  $n = 1 : N$ , **do**  
   Compute  $e_n = y_n - \hat{\mathbf{w}}^\top \hat{\Phi}(\mathbf{x}_n) - \hat{b}$ ;  
   Compute  $v_n$  from Eq. (5.23);  
**end for**  
 Build the linear system of Eq. (5.31);  
 Compute  $\hat{\mathbf{w}}^r$  and  $\hat{b}^r$  from Eq. (5.6); % RFS-LSSVR solution

---

$e_n^{(k)}/\hat{s}$  and using the weight function in Eq. (5.23). The final step requires the solution of Eq. (5.31) to compute the parameter vector  $\hat{\mathbf{w}}^{r(k)}$  and the bias term  $\hat{b}^{r(k)}$ .

Then, we set  $k = k + 1$ , and repeat the procedure for the new iteration so that new weights  $v_n^{(k+1)}$ , a new model solution  $\hat{\mathbf{w}}^{r(k+1)}$ , and a new bias term  $\hat{b}^{r(k+1)}$  are calculated. It is necessary to establish a maximum number of iterations for the algorithm. Moreover, we adopt  $\max(|\hat{\mathbf{w}}^{r(k)} - \hat{\mathbf{w}}^{r(k-1)}|) > 10^{-4}$  as stopping criterion. The pseudocode in Algorithm 12 brings all the steps necessary to implement the R<sup>2</sup>FS-LSSVR model correctly.

### 5.3.3 Extension to Nonlinear MIMO Systems

The previous formulations for the RFS-LSSVR and R<sup>2</sup>FS-LSSVR models are intended to be applied to SISO systems. However, many practical applications of system modeling and identification techniques, specially in industrial environments, are designed for Multiple-Input Multiple-Output (MIMO) systems. Bearing this in mind, in this section we extend and generalize both RFS-LSSVR and R<sup>2</sup>FS-LSSVR models for MIMO regression/identification

---

**Algorithm 12:** - Pseudo-code for the R<sup>2</sup>FS-LSSVR model.

---

**Require:**  $M, \gamma, \sigma, N_{iter}$ ;  
 Choose a working set  $\mathcal{D}^{pv} = \{\mathbf{x}_m\}_{m=1}^M$ ;  
 Select  $\mathcal{D}_1$  with the  $N - M$  remaining training samples;  
**for**  $n_{iter} = 1 : N_{iter}$ , **do**  
   Compute  $H_R$  of  $\mathcal{D}^{pv}$  from (2.42);  
   Randomly select  $\mathbf{x}_m$  from  $\mathcal{D}^{pv}$ ;  
   Randomly select  $\mathbf{x}^*$  from  $\mathcal{D}_1$ ;  
   Replace  $\mathbf{x}_m$  by  $\mathbf{x}^*$  in  $\mathcal{D}^{pv}$ ;  
   Compute  $H_R^*$  of  $\mathcal{D}^{pv}$  from (2.42);  
   **if**  $H_R^* > H_R$ , **then**  
      $\mathbf{x}^*$  is accepted for  $\mathcal{D}^{pv}$ ;  
   **else**  
      $\mathbf{x}^*$  is rejected by  $\mathcal{D}^{pv}$ ;  
   **end if**  
**end for**  
 Compute  $\hat{\Phi}$  from Eq. (5.21);  
 Build the linear system of Eq. (5.22);  
 Compute  $\hat{\mathbf{w}}$  and  $\hat{b}$  from Eq. (5.6); % FS-LSSVR solution  
**for**  $n = 1 : N$ , **do**  
   Compute  $e_n = y_n - \hat{\mathbf{w}}^\top \hat{\Phi}(\mathbf{x}_n) - \hat{b}$ ;  
   Compute  $v_n$  from Eq. (5.23);  
**end for**  
 Build the linear system of Eq. (5.31);  
 Compute  $\hat{\mathbf{w}}^{r(1)}$  and  $\hat{b}^{r(1)}$  from Eq. (5.6); % RFS-LSSVR solution  
**while**  $\max(|\hat{\mathbf{w}}^{r(i)} - \hat{\mathbf{w}}^{r(i-1)}|) > 10^{-4}$  **do**  
   **for**  $n = 1 : N$ , **do**  
     Compute  $e_n^{(k)} = y_n - (\hat{\mathbf{w}}^{r(k)})^\top \hat{\Phi}(\mathbf{x}_n) - \hat{b}^{r(k)}$ ;  
     Compute  $v_n^{(k)}$  from Eq. (5.23);  
   **end for**  
   Build the linear system of Eq. (5.31);  
   Compute  $\hat{\mathbf{w}}^{r(k)}$  and  $\hat{b}^{r(k)}$  from Eq. (5.6);  
    $k=k+1$ ;  
**end while**  
 $\hat{\mathbf{w}}^{r(k)}$  and  $\hat{b}^{r(k)}$ ; % R<sup>2</sup>FS-LSSVR solution

---

problems.

Now, let us consider a training dataset  $\{\mathbf{x}'_n, \mathbf{y}_n\}_{n=1}^N$ , with input  $\mathbf{x}'_n \in \mathbb{R}^{d'}$ , where  $d'$  denotes the input dimension. The corresponding outputs are  $\mathbf{y}_n = [y_n^{(1)}, \dots, y_n^{(n_y)}]^\top \in \mathbb{R}^{n_y}$ , where  $n_y$  is the number of outputs. For a new incoming vector  $\mathbf{x}^*$ , the predictions for each output are

computed in accordance with  $\hat{f}(\mathbf{x}^*) = \hat{\mathbf{w}}^\top \hat{\boldsymbol{\phi}}(\mathbf{x}^*) + \hat{b}$  as follows:

$$\begin{aligned} \hat{f}_1(\mathbf{x}^*) &= \hat{\mathbf{w}}_1^\top \hat{\boldsymbol{\phi}}(\mathbf{x}^*) + \hat{b}_1, \\ &\vdots \\ \hat{f}_{n_y}(\mathbf{x}^*) &= \hat{\mathbf{w}}_{n_y}^\top \hat{\boldsymbol{\phi}}(\mathbf{x}^*) + \hat{b}_{n_y}, \end{aligned} \quad (5.32)$$

where  $\hat{\mathbf{w}}_j \in \mathbb{R}^M$  and  $\hat{b}_j \in \mathbb{R}$ ,  $j = 1, \dots, n_y$ , are the solutions for the  $j$ -th output.

For the MIMO case, let us arrange the  $n_y$  parameter vectors  $\hat{\mathbf{w}}_j \in \mathbb{R}^M$  along the columns of the matrix  $\hat{\mathbf{W}} = [\hat{\mathbf{w}}_1 \mid \dots \mid \hat{\mathbf{w}}_{n_y}] \in \mathbb{R}^{M \times n_y}$ , and the  $n_y$  biases into the vector  $\hat{\mathbf{b}} = [\hat{b}_1, \dots, \hat{b}_{n_y}]^\top \in \mathbb{R}^{n_y}$ . Then, the vectors  $\hat{\mathbf{w}}_j$  and the biases  $\hat{b}_j$  can be computed at once by solving the following extended version of the linear system in Eq. (5.22):

$$\underbrace{\begin{bmatrix} \hat{\boldsymbol{\Phi}}^\top \hat{\boldsymbol{\Phi}} + \frac{1}{\gamma} \mathbf{I}_M & \hat{\boldsymbol{\Phi}}^\top \mathbf{1}_N \\ \mathbf{1}_N^\top \hat{\boldsymbol{\Phi}} & \mathbf{1}_N^\top \mathbf{1}_N \end{bmatrix}}_{(M+1) \times (M+1)} \underbrace{\begin{bmatrix} \hat{\mathbf{W}} \\ \hat{\mathbf{b}}^\top \end{bmatrix}}_{(M+1) \times n_y} = \underbrace{\begin{bmatrix} \hat{\boldsymbol{\Phi}}^\top \mathbf{Y} \\ \mathbf{1}_N^\top \mathbf{Y} \end{bmatrix}}_{(M+1) \times n_y}, \quad (5.33)$$

where  $\mathbf{Y}^\top = [\mathbf{y}_1 \mid \dots \mid \mathbf{y}_N] \in \mathbb{R}^{n_y \times N}$ . One clear advantage of the kernel-based approach is that the dimension of the first matrix on the left-hand side of Eq. (5.33) is the same as that in Eq. (5.22). This is true, even though the input vector  $\mathbf{x}'_n \in \mathbb{R}^d$  for the MIMO case has higher dimensionality compared to the input vector  $\mathbf{x}_n \in \mathbb{R}^d$  of the SISO case, because the kernel computations required by the reduced kernel matrix  $\bar{\mathbf{K}}$  and for the feature map approximation  $\hat{\phi}_i(\cdot)$  in Eq. (5.18) are independent of the dimensionality of the input data. Accordingly, the approximated feature matrix  $\hat{\boldsymbol{\Phi}}$  for the MIMO case also remains with the same dimensionality as that one of the SISO case.

Since the FS-LSSVR solutions  $\hat{\mathbf{w}}_j$  and  $\hat{b}_j$  have been found, the next step is to compute the prediction errors due each output as

$$\begin{aligned} e_n^{(1)} &= y_n^{(1)} - \hat{\mathbf{w}}_1^\top \hat{\boldsymbol{\phi}}(\mathbf{x}_n) - \hat{b}_1, \\ &\vdots \\ e_n^{(n_y)} &= y_n^{(n_y)} - \hat{\mathbf{w}}_{n_y}^\top \hat{\boldsymbol{\phi}}(\mathbf{x}_n) - \hat{b}_{n_y}, \end{aligned} \quad (5.34)$$

for  $n = 1, \dots, N$ . Thus, we use Eq. (5.23) to determine the weights  $v_n^{(j)}$  and the respective diagonal matrices  $\mathbf{V}_j = \text{diag}\{v_n^{(j)}\}_{n=1}^N$  for the outputs  $y_n^{(j)}$ .

Finally, the RFS-LSSVR solutions for MIMO systems can be computed by solving the linear system in Eq. (5.31) for each output. Therefore, one gets

$$\underbrace{\begin{bmatrix} \hat{\Phi}^\top \mathbf{V}_j \hat{\Phi} + \frac{1}{\gamma} \mathbf{I}_M & \hat{\Phi}^\top \mathbf{V}_j \mathbf{1}_N \\ \mathbf{1}_N^\top \mathbf{V}_j \hat{\Phi} & \mathbf{1}_N^\top \mathbf{V}_j \mathbf{1}_N \end{bmatrix}}_{(M+1) \times (M+1)} \underbrace{\begin{bmatrix} \hat{\mathbf{w}}_j^r \\ \hat{\mathbf{b}}_j^r \end{bmatrix}}_{(M+1) \times 1} = \underbrace{\begin{bmatrix} \hat{\Phi}^\top \mathbf{V}_j \mathbf{y}^{(j)} \\ \mathbf{1}_N^\top \mathbf{V}_j \mathbf{y}^{(j)} \end{bmatrix}}_{(M+1) \times 1}, \quad (5.35)$$

where  $\mathbf{y}^{(j)} = [y_1^{(j)}, \dots, y_N^{(j)}]^\top \in \mathbb{R}^N$ , for  $j = 1, \dots, n_y$ . As in the SISO case, the procedure involving the computations demanded in Eqs. (5.23), (5.34) and (5.35) can be iteratively repeated, much the same way as stated in Section 5.3.2, to obtain the R<sup>2</sup>FS-LSSVR solutions for identification of nonlinear MIMO systems. By the way, we discuss next how to build the input regression vectors for a MIMO system.

### 5.3.3.1 The Regression Vector - MIMO Case

Since we are also interested in MIMO dynamical system identification, the regression vector should contain the lagged values (for a certain memory order) for all inputs and output variables of interest. In this respect, let us define the input regression subvector associated with the  $i$ -th input as

$$\mathbf{u}_{n-1}^{(i)} = \left[ u_{n-1}^{(i)} \ u_{n-2}^{(i)} \ \cdots \ u_{n-L_u^{(i)}}^{(i)} \right]^\top, \quad i = 1, \dots, n_u, \quad (5.36)$$

where  $L_u^{(i)}$  denotes the memory order of the  $i$ -th input variable. By the same token, let  $\mathbf{y}_{n-1}^{(j)}$  us define the output regression subvector associated with the  $j$ -th output, being defined as

$$\mathbf{y}_{n-1}^{(j)} = \left[ y_{n-1}^{(j)} \ y_{n-2}^{(j)} \ \cdots \ y_{n-L_y^{(j)}}^{(j)} \right]^\top, \quad j = 1, \dots, n_y, \quad (5.37)$$

where  $L_y^{(j)}$  denotes the order of the regression of the  $j$ -th output variable.

Without loss of generality, we assume the same memory order values for all the input and output variables involved; i.e.  $L_u^{(i)} = L_u, \forall i$ ; and  $L_y^{(j)} = L_y, \forall j$ . That said, the vector of regressors is then given by (BILLINGS, 2013)

$$\mathbf{x}'_n = [\mathbf{y}_{n-1}^{(1)} \ \mathbf{y}_{n-1}^{(2)} \ \cdots \ \mathbf{y}_{n-1}^{(n_y)}, \ \mathbf{u}_{n-1}^{(1)} \ \mathbf{u}_{n-1}^{(2)} \ \cdots \ \mathbf{u}_{n-1}^{(n_u)}]^\top, \quad (5.38)$$

where  $\mathbf{x}'_n \in \mathbb{R}^{(n_y \times L_y + n_u \times L_u)}$ . It is worth mentioning that since we assume that each output of the MIMO systems obeys a NARX-type dynamics, the definition of the regression vector in Eq. (5.38) may raise high-dimensionality concerns in system identification tasks. However, as stated in Subsection 5.3.3, the dimension of the input vector has no effect on the dimensions of the kernel operations required by the linear system in Eq. (5.33).

### 5.3.4 Final Remarks on the Proposed Models

**Remark 1** - It should be noted that the parameter estimation problem for the original W-LSSVR and IR-LSSVR models are formulated as an optimization problem in dual space. There is no relevant difference in their training times when compared to the original LSSVR model (also formulated in dual space), but the resulting models are not sparse (just like the original LSSVR). By the same token, the parameter estimation problem of the proposed RFS-LSSVR and R<sup>2</sup>FS-LSSVR are developed in the primal space, as it is done for the original FS-LSSVR model. There is absolutely no difference in their training times, when compared to the original FS-LSSVR.

**Remark 2** - One clear advantage of the FS-LSSVR (primal space) model over the original LSSVR (dual space) model is that the final solution is already sparse. The proposed RFS-LSSVR and R<sup>2</sup>FS-LSSVR models inherit sparsity from the original FS-LSSVR model, a property that enables them to be applied to large-scale datasets.

**Remark 3** - In the models formulated in the primal space, there is an extra time spent in the search of the  $M$  prototype vectors, but this extra computational effort is a minor issue if we take into account that the resulting solution vector (i.e. parameter vector) is sparse.

### 5.3.5 Computational Experiments

From this section on we report and discuss the results of comprehensive computer simulations involving the application of the proposed approaches (RFS-LSSVR and R<sup>2</sup>FS-LSSVR models) to nonlinear system identification with outliers. We compare their performances to those achieved by their equivalent robust approaches in dual space (W-LSSVR and IR-LSSVR) and by the standard LSSVR model. Next, we describe some important details about the experimental methodology and the evaluation of the obtained results.

1. Methodology - Except for the large-scale dataset, we performed a 5-fold cross-validation in the search for optimal values for the hyperparameters  $\gamma$  (the regularization term) from the range  $\{2^0, \dots, 2^{20}\}$  and  $\sigma$  (the Gaussian kernel radius) from the range  $\{2^{-10}, \dots, 2^0\}$  for all the evaluated kernel models, similarly as done in Section 5.2.2. Furthermore, for the primal models, we set  $N_{iter} = 500$  and select the amount of prototype vectors  $M$  in preliminary experiments with each dataset.

As in Section 5.2.2, again the orders  $\hat{L}_u$  and  $\hat{L}_y$  are set according the largest delays of

Eqs. (5.16) and (5.17) for the synthetic datasets. For the real-world datasets (except for the large-scale and MIMO datasets),  $\hat{L}_u$  and  $\hat{L}_y$  are optimized from the range  $\{1, 2, 3, 4, 5\}$  for the standard LSSVR model. Thus, their resulting values are used for the other models. The figure of merit for evaluating the numerical performance of our RFS-LSSVR and R<sup>2</sup>FS-LSSVR models is the RMSE computed for the  $N'$  test samples over 20 independent runs, since these models are sensitive to the initial conditions.

2. **Outlier Contamination** - Following the same outlier generation methodology addressed in Mattos *et al.* (2016) and properly described in Section 5.2.2, in addition to Gaussian noise we contaminate the outputs of the training data (for the synthetic datasets) by replacing them randomly with outliers covering 0%, 5%, 10%, 15%, 20%, 25% and 30% of the total of estimation samples. We use the same procedure to contaminate with 5% or 10% of outliers the estimation outputs of the real datasets.

#### 5.3.5.1 Evaluated Datasets

For the experiments with the proposed approaches in nonlinear system identification tasks, we initially use all the synthetic (Synthetic-1 and Synthetic-2) and real-world (Actuator and Dryer) datasets previously described in Section 5.2.2.1. In addition, since the RFS-LSSVR and R<sup>2</sup>FS-LSSVR models have inherited the high sparsity property of the FS-LSSVR model, we decided to evaluate their performances on a large-scale dataset. For that matter, we choose the Silverbox dataset, which was introduced in Schoukens *et al.* (2003) and represents an electrical circuit simulating a mass-spring damper system. It corresponds to a nonlinear dynamical system with feedback, presenting a dominant linear behavior (ESPINOZA *et al.*, 2004). More information about this dataset can be found in Pintelon and Schoukens (2012).

The input  $u_n$  and output  $y_n$  voltage sequences of the Silverbox dataset are shown in Figs. (16a) and (16b), respectively. This dataset contains a total of 131,072 samples of each sequence  $u_n$  and  $y_n$ , where the first 40,000 samples of  $u_n$  and  $y_n$  were used for model evaluation (testing) and the remaining 91,072 for model building (training). From the model building samples, we separate the first 45,000 for estimating the model hyperparameters  $\gamma$  and  $\sigma$ , and the remaining 46,072 samples for validate the parameters of the models. The same cross-validation strategy for this dataset was adopted in Espinoza *et al.* (2004). The values  $M = 500$ ,  $L_u = L_y = 10$  were chosen for the experiments, as was also suggested in Espinoza *et al.* (2004).

Finally, we also test our proposals on a benchmarking dataset corresponding to a

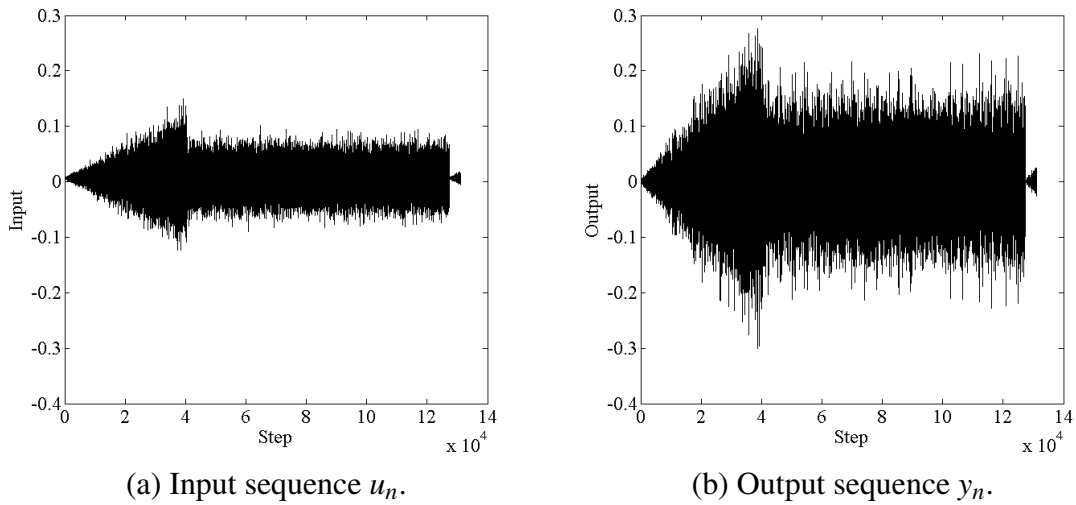


Figure 16 – Input and output sequences - Silverbox dataset.

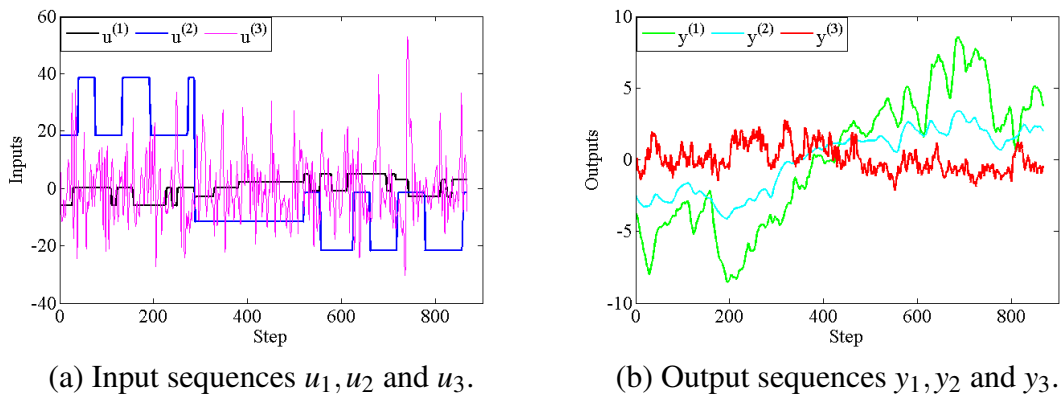


Figure 17 – Input and output sequences of the Industrial dryer dataset - MIMO system.

MIMO system. For this purpose, we selected the Industrial Dryer dataset described in Maciejowski (1996), Chou and Maciejowski (1997) and available at the DaISy repository. This system has  $n_u = 3$  inputs, namely: the fuel flow rate ( $u^{(1)}$ ), the hot gas exhaust fan speed ( $u^{(2)}$ ) and the rate of flow of raw material ( $u^{(3)}$ ). In addition, it has  $n_y = 3$  outputs: the temperature of the dry bulb ( $y^{(1)}$ ), the temperature of the wet bulb ( $y^{(2)}$ ) and the moisture content of raw material ( $y^{(3)}$ ). These inputs and outputs are shown in Figs. (17a) and (17b), respectively. For better viewing, the amplitudes of the input  $u^{(3)}$ , in Fig. (17a), were divided by a factor of 20. From a total of 867 available samples, we use the first 600 for estimating the values of the hyperparameters  $\gamma$  and  $\sigma$ , and the remaining 267 for testing the models.

Based on preliminary experimentation with the Industrial Dryer dataset for the standard LSSVR model, we chose the order values to be  $L_u = L_y = 8$ . Then, we use these values for all other models. In this case, the input regression vector  $\mathbf{x}'_n \in \mathbb{R}^{d'}$  was previously defined in



Table 8 – Important features of the evaluated datasets and models for the computational experiments.

Dataset	Important Features of the Evaluated Datasets					
	System Type	Prediction Scenario	Train ( $N$ )	Test ( $N'$ )	$\hat{L}_u$	$\hat{L}_y$
Silverbox	SISO	Free Simulation	91,072	40,000	10	10
Industrial Dryer	MIMO	3-step ahead	600	267	8	8

Eq. (5.38) as

$$\mathbf{x}'_n = [\mathbf{y}_{n-1}^{(1)} \mathbf{y}_{n-1}^{(2)} \cdots \mathbf{y}_{n-1}^{(n_y)}, \mathbf{u}_{n-1}^{(1)} \mathbf{u}_{n-1}^{(2)} \cdots \mathbf{u}_{n-1}^{(n_u)}]^\top, \quad (5.39)$$

where  $d' = n_y \times L_y + n_u \times L_u = (3)(8) + (3)(8) = 48$ , which may raise high-dimensionality concerns in system identification tasks. However, as before mentioned, the dimension of the input vector has no effect on the dimensions of the kernel operations required by the linear system in Eq. (5.33).

### 5.3.5.2 Results and Discussion

Computer experiments involving all datasets correspond to the task of iterative  $k$ -step ahead prediction. For the Actuator and Industrial Dryer (MIMO) datasets, we set  $k = 3$  and for the remaining datasets we set  $k \rightarrow +\infty$ , which correspond to infinite horizon prediction (a.k.a. free simulation) scenario. Some important features of the synthetic (Synthetic-1 and Synthetic-2), Actuator and Dryer datasets were already shown in Table 5, whereas the corresponding features and parameters of the Silverbox and Industrial Dryer datasets are shown in Table 8.

#### 5.3.5.2.1 Experiments with Synthetic Datasets

Initially, we report in Fig. (18) the RMSE values, given in the form of error bars, achieved by the evaluated models for the synthetic datasets with increasing scenarios from 0% to 30% of outliers contamination. Furthermore, the number of prototype vectors (PV) for the proposed approaches and the total number of training samples are also included in the figures.

For the Synthetic-1 dataset, the RMSE values produced by the evaluated models are shown in Fig. (18a). A detailed analysis of these results reveals that for almost all the scenarios of contamination, including the outlier-free case, the RFS-LSSVR and R<sup>2</sup>FS-LSSVR models achieved lower values of RMSE in comparison to the other models. This is particularly more evident for scenarios with high levels of outlier contamination (higher than 15% of outliers). Furthermore, an important aspect to be highlighted is the significant level of sparsity achieved by

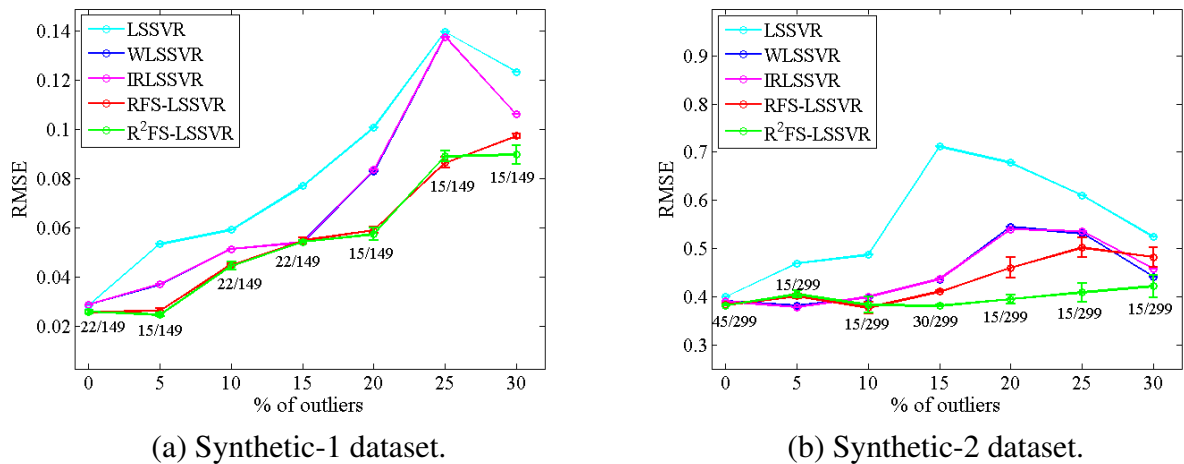


Figure 18 – RMSE values for the test samples over 20 independent runs in free simulation.

the proposed approaches. While the standard LSSVR, W-LSSVR and IR-LSSVR models used all of the training data (149 samples in this case), the RFS-LSSVR and  $R^2$ FS-LSSVR models used at most 15% (22 samples) of the total training instances as PV.

The RMSE values for the Synthetic-2 dataset are shown in Fig. (18b). It can be easily observed that the proposed approaches achieved in general lower values of RMSE in comparison to the other models. One can also note that the performances of the dual-space models (i.e. LSSVR, W-LSSVR and IR-LSSVR) improves as the level of outlier contamination increases, but it must be emphasized that this is happening while those models are using *all* training samples (a total of 299 samples) as PVs. The proposed primal-space models (i.e. RFS-LSSVR and  $R^2$ FS-LSSVR) achieve better results using only a small number of PVs (at most 45 samples) from the total number of training samples. As a final remark, one can note that the  $R^2$ FS-LSSVR model is highly resilient to outliers, since its performance is practically not affected by the presence of outliers (even for a high number of them).

### 5.3.5.2.2 Experiments with Real-World Datasets

In the next set of computer experiments, we assess the achieved results for the Actuator (in 3-step-ahead prediction) and for the Dryer (in free simulation) datasets considering two different cases: (i) outlier-free scenario, and (ii) scenario with 10% of outliers. For each case we compare the best dual model, i.e. the dual model that achieves the lowest RMSE value, with the best primal model (RFS-LSSVR or  $R^2$ FS-LSSVR). The best primal model corresponds to the one that simultaneously achieves the lowest average RMSE value and the lowest dispersion along the 20 runs.

We report in Table 9 the RMSE values and the number of PV obtained by each model for the Actuator dataset. For the outlier-free scenario, one can see that W-LSSVR and RFS-LSSVR models, which presented the lowest RMSE values among the dual and primal models, respectively, achieved performances quite close to each other. However, the proposed RFS-LSSVR model achieved such a superior performance using only about 10% (51 samples) of the training samples used by the best dual-space model (W-LSSVR).

For the scenario with 10% of outliers, even with a certain dispersion in the RMSE values, the  $R^2$ FS-LSSVR model (best performance between the primal models) presented much lower RMSE values than the best dual model (W-LSSVR). Moreover, the performance of the  $R^2$ FS-LSSVR model was achieved using only about 5% (25 samples) of the training data as PV. This is a clear indicator that the proposed approaches achieve very good performances in the presence of outliers with a parsimonious use of the available training samples. This is an appealing property of the proposed models which will be explored later for the sake of large-scale data modeling.

We report the worst-case results (the predicted outputs that led to the highest RMSE values among the independent runs) in Fig. (19) for the predictions of the best models showed in Table 9. The Figs. (19a) and (19b), which correspond to the case without outliers, reveals that the W-LSSVR and RFS-LSSVR models produced equivalent predictions, being visually indistinguishable in the graphics since the predicted dynamical behaviors are quite close to the real one. However, for the case with 10% of outliers showed in Figs. (19c) and (19d), it becomes evident the superiority of the proposed approach. Even considering the output that obtained the higher value of RMSE, the  $R^2$ FS-LSSVR model outperformed the W-LSSVR model along the whole duration of the signal. It is worth evaluating the performance of the (dual-space) W-LSSVR model between the time steps 150 – 250 and 400 – 500, where the predictions are notably more distorted comparing to that ones achieved by the (primal space)  $R^2$ FS-LSSVR model.

We can see in Table 10 and Fig. (20) the obtained results from the experiments with the Dryer dataset. In Table 10 we report the RMSE values and the number of PV for each evaluated model, where it is possible to note that, for the outlier-free scenario, all the primal and dual models achieved performances which were very close to each other. However, it is important to highlight once again, the performances of the RFS-LSSVR and  $R^2$ FS-LSSVR were achieved using only 15% (74 samples) of the training data. For the outlier-corrupted scenario,

Table 9 – RMSE values for the test samples and number of PVs for each evaluated model - Actuator dataset.

Models	0% of outliers		10% of outliers	
	RMSE	#PV	RMSE	#PV
LSSVR	$2.39\text{E-}1 \pm 0.00\text{E}0$	508	$6.30\text{E-}1 \pm 0.00\text{E}0$	508
W-LSSVR	<b><math>2.36\text{E-}1 \pm 0.00\text{E}0</math></b>	508	<b><math>6.21\text{E-}1 \pm 0.00\text{E}0</math></b>	508
IR-LSSVR	$2.37\text{E-}1 \pm 0.00\text{E}0$	508	$6.21\text{E-}1 \pm 0.00\text{E}0$	508
RFS-LSSVR	<b><math>2.38\text{E-}1 \pm 5.00\text{E-}4</math></b>	<b>51</b>	$6.05\text{E-}1 \pm 2.70\text{E-}3$	<b>25</b>
R <sup>2</sup> FS-LSSVR	$2.79\text{E-}1 \pm 1.30\text{E-}3$	<b>51</b>	<b><math>4.85\text{E-}1 \pm 8.60\text{E-}3</math></b>	<b>25</b>

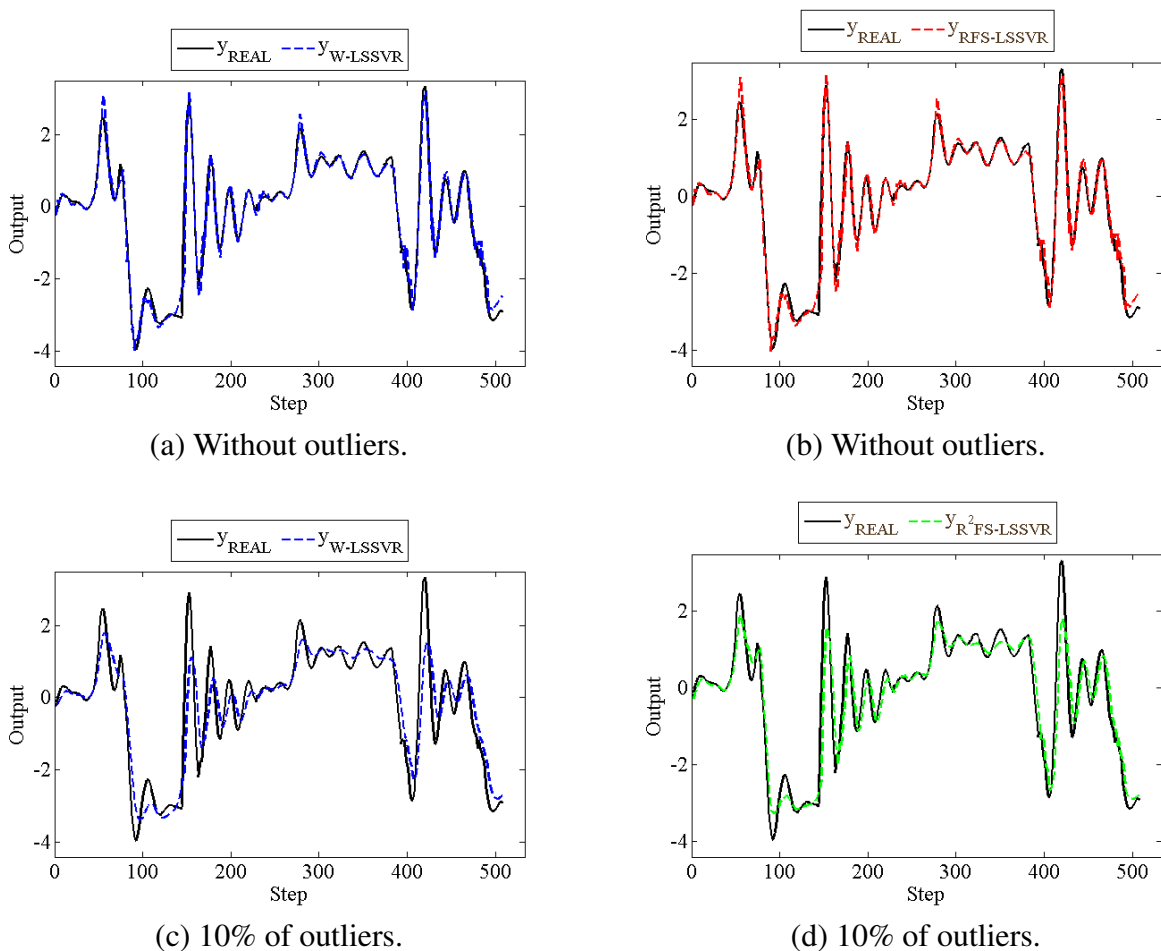


Figure 19 – Predicted outputs by the best models in Table 9 with their worst performances in RMSE over the 20 runs - Actuator dataset.

again the robust primal models used an even smaller amount of PVs, just 50 samples, compared to the other models. Furthermore, the RFS-LSSVR model achieved lower RMSE values (even with a certain dispersion) than the W-LSSVR model.

In Fig. (20) we report the worst predicted outputs obtained from the models with the best results in Table 10. For the scenario without outliers, one can note in Figs. (20a) and (20b)

that the predicted outputs for the LSSVR and RFS-LSSVR models were quite close to each other, both following the dynamical behavior of the actual output. In the presence of outliers, we can observe in Figs. (20c) and (20d) that the predictions of both robust models (W-LSSVR and RFS-LSSVR) did not suffer a notable degradation with the insertion of outliers. Therefore, even for the worst-case predictions, the RFS-LSSVR model was able to follow closely the dynamical behavior of the system of interest.

#### 5.3.5.2.3 Experiments with a Large-Scale Dataset

We report in Fig. (21) the boxplots of the RMSE values for the proposed robust primal approaches and the FS-LSSVR model in free simulation scenario. A remarkable aspect to be highlighted in these experiments is the absence of results from the dual models: LSSVR, W-LSSVR and IR-LSSVR. It is not possible to train these models in such a large-scale dataset using the standard batch mode techniques for solving the required linear systems.

For the case with no outliers, it can be seen in Fig. (21a) that the obtained results by the FS-LSSVR model were compatible with those ones found in Espinoza *et al.* (2004). We can also observe that, in general, the RMSE values achieved by the robust approaches and the FS-LSSVR model were relatively close each other.

Finally, the Fig. (21b) shows the RMSE boxplots when the predicted outputs are contaminated with 5% of outliers. One can note that the performances of all the models have been degraded with the insertion of outliers, as expected. However, our robust approaches were considerably less affected, since their RMSE values were much smaller than those ones obtained by the non-robust FS-LSSVR model. In time, The R<sup>2</sup>FS-LSSVR model achieved RMSE values that were approximately half of those ones obtained by the FS-LSSVR model.

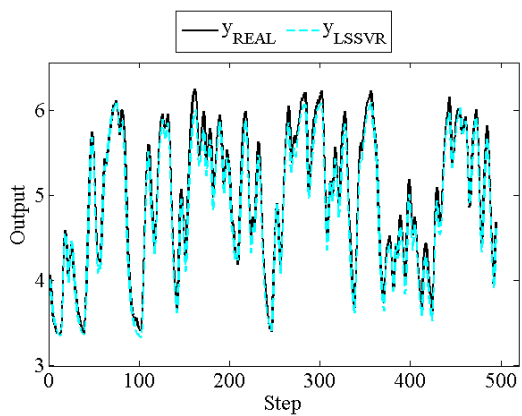
#### 5.3.5.2.4 Experiments with a MIMO Dataset

We report in Tables 11 and 12 the obtained RMSE values (mean value and standard deviation) and the number of PV by each model for the Industrial Dryer dataset in 3-step-ahead prediction. The achieved results correspond to the scenarios without outliers (Table 11) and with 5% of contamination (Table 12), considering each predicted output. In the tables, the lowest RMSE values obtained by the best dual model and by the best primal model are highlighted in bold for each output.

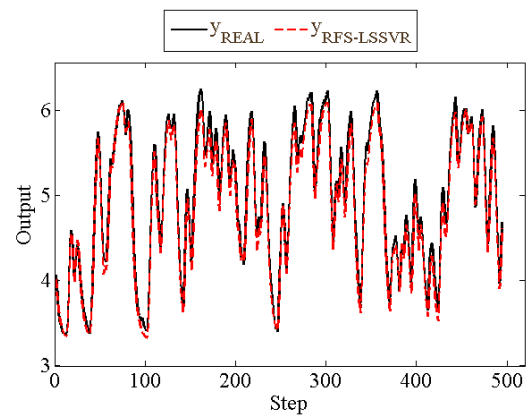
For the scenario without outliers in Table 11, the dual and primal models (be they

Table 10 – RMSE values for the test samples and number of PVs for each evaluated model - Dryer dataset.

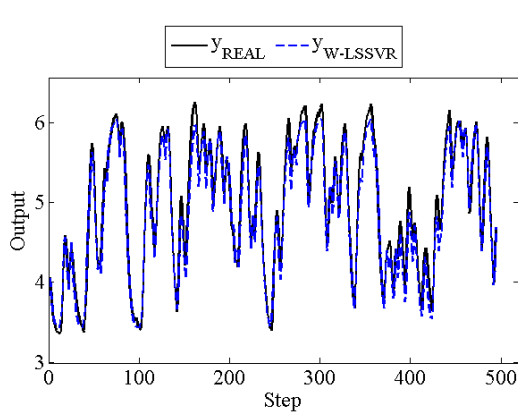
Models	0% of outliers		10% of outliers	
	RMSE	#PV	RMSE	#PV
LSSVR	<b>1.12E-1</b> ± 0.00E0	495	1.48E-1 ± 0.00E0	495
W-LSSVR	1.13E-1 ± 0.00E0	495	<b>1.41E-1</b> ± 0.00E0	495
IR-LSSVR	1.13E-1 ± 0.00E0	495	1.41E-1 ± 0.00E0	495
RFS-LSSVR	<b>1.13E-1</b> ± <b>7.00E-4</b>	<b>74</b>	<b>1.31E-1</b> ± <b>6.80E-3</b>	<b>50</b>
R <sup>2</sup> FS-LSSVR	1.13E-1 ± 7.00E-4	<b>74</b>	1.29E-1 ± 1.05E-2	<b>50</b>



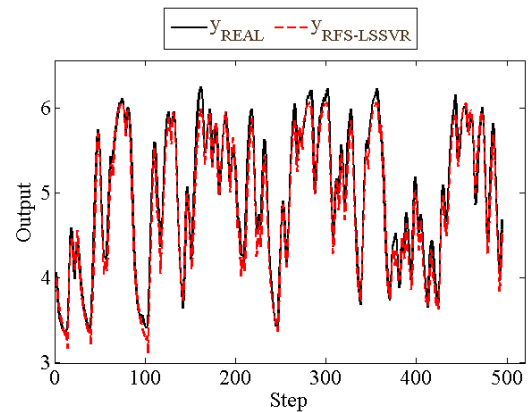
(a) Without outliers.



(b) Without outliers.



(c) 10% of outliers.



(d) 10% of outliers.

Figure 20 – Predicted outputs by the best models in Table 10 with their worst performances in RMSE over the 20 runs - Dryer dataset.

robust or not) achieved similar RMSE values for the three outputs, as expected. A case worth mentioning was that involving the  $\hat{y}^{(2)}$  output, for which the standard LSSVR achieved lower values of RMSE compared to the other models. However, one should note that, in this case, the performances of the proposed approaches were achieved using only 25% (i.e.  $M = 150$  samples) of the training data as prototype vectors.

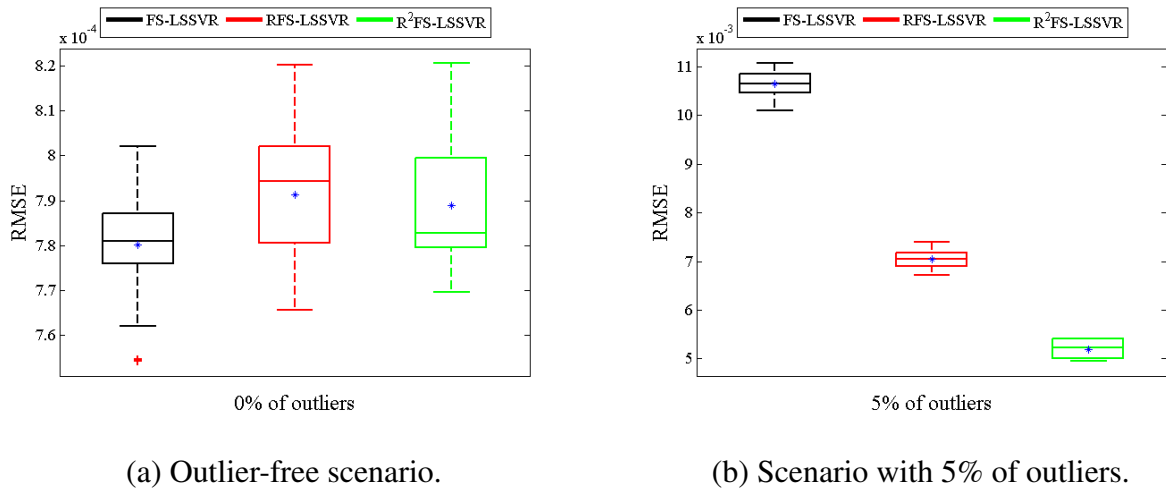


Figure 21 – Boxplots for the RMSE values of test samples after 20 independent runs - Silverbox dataset.

Table 11 – RMSE values for test samples for the Industrial Dryer dataset (MIMO system) in outlier-free scenario.

Models	0% of outliers			
	RMSE - $\hat{y}^{(1)}$	RMSE - $\hat{y}^{(2)}$	RMSE - $\hat{y}^{(3)}$	#PV
LSSVR	<b>6.51E-1</b> $\pm$ <b>0.00E0</b>	<b>1.49E-1</b> $\pm$ <b>0.00E0</b>	<b>4.49E-1</b> $\pm$ <b>0.00E0</b>	600
W-LSSVR	6.69E-1 $\pm$ 0.00E0	1.86E-1 $\pm$ 0.00E0	4.59E-1 $\pm$ 0.00E0	600
IR-LSSVR	6.68E-1 $\pm$ 0.00E0	1.86E-1 $\pm$ 0.00E0	4.59E-1 $\pm$ 0.00E0	600
FS-LSSVR	6.35E-1 $\pm$ 9.10E-3	1.79E-1 $\pm$ 4.50E-3	<b>4.58E-1</b> $\pm$ <b>7.40E-3</b>	<b>150</b>
RFS-LSSVR	<b>6.31E-1</b> $\pm$ <b>1.02E-2</b>	1.82E-1 $\pm$ 4.60E-3	4.73E-1 $\pm$ 4.41E-3	<b>150</b>
R <sup>2</sup> FS-LSSVR	6.41E-1 $\pm$ 8.20E-3	<b>1.78E-1</b> $\pm$ <b>2.79E-3</b>	4.78E-1 $\pm$ 5.90E-3	<b>150</b>

From Table 12, we can read the obtained RMSE values achieved in an outlier-contaminated scenario, where each training output ( $y^{(1)}$ ,  $y^{(2)}$  and  $y^{(3)}$ ) was contaminated with 5% of outliers. It is possible to observe that practically all the models achieved similar performances with respect to the  $y^{(3)}$  output. With regard to the  $y^{(1)}$  and  $y^{(2)}$  outputs, the proposed robust approaches, especially the R<sup>2</sup>FS-LSSVR model, outperformed the original FS-LSSVR and all of the dual models, using only 10% ( $M = 60$  samples) of the training data as prototype vectors. As a general conclusion for this set of experiments, we also observed a good performance for the RFS-LSSVR and R<sup>2</sup>FS-LSSVR models in nonlinear MIMO system identification tasks in the presence of outliers.

#### 5.4 Concluding Remarks

This chapter presented the first part of the contributions of this thesis, which corresponds to three outlier-robust strategies to solve the LSSVR optimization problem based on

the M-estimation framework. The proposed approaches, called the Recursive Least M-estimate SVR (RLM-SVR), Robust FS-LSSVR (RFS-LSSVR) and Reweighted Robust FS-LSSVR ( $R^2$ FS-LSSVR) models, produce robust solutions in nonlinear regression problems, especially in dynamical system identification tasks, when the estimate outputs are contaminated with non-Gaussian noise or outliers.

The RLM-SVR approach initially starts by obtaining the linear system of equations, equivalent to the dual formulation of the standard (unweighted) LSSVR model. Then, the next step consists in applying the linear and robust RLM algorithm, which is based on M-estimation and derived from the classical RLS algorithm. Therefore, the RLM-SVR solution is obtained in a recursive procedure, where each input sample possibly contaminated is individually treated at each iteration. By doing this, one can improve the performance of the model solution in the presence of outliers.

The resulting RLM-SVR model was successfully evaluated in nonlinear dynamical system identification tasks with benchmarking synthetic and real-world datasets, in  $k$ -step ahead prediction scenarios ( $k = 3$  or  $k \rightarrow +\infty$ ). For both synthetic and real-world datasets, our proposed approach outperformed, in terms of RMSE values, the LSSVR model and the robust W-LSSVR and IR-LSSVR models in almost all scenarios of outliers contamination. This was an evidence that the recursive procedure proposed by the RLM-SVR model could, actually, improve the model performance in the presence of outliers.

In a different context, the proposed RFS-LSSVR and  $R^2$ FS-LSSVR approaches were presented as robust versions of the FS-LSSVR model, based on the M-estimators and the weighted least squares algorithm. As in the FS-LSSVR model, they solve the LSSVR optimization problem in primal space by means of an approximation for the nonlinear map  $\phi$  using the Nyström method. Because of that, the RFS-LSSVR and  $R^2$ FS-LSSVR solutions are also sparse since they rely on a selected subsample of PVs.

In addition, the RFS-LSSVR and  $R^2$ FS-LSSVR proposals keep the same computational complexity of the FS-LSSVR model, do not require extra parameters for tuning and can also handle nonlinear system identification problems with large-scale data. In general, one can say that they are equivalent approaches in primal space of the W-LSSVR and IR-LSSVR models, respectively. Therefore, our RFS-LSSVR and  $R^2$ FS-LSSVR models produce solutions that are sparse and, at the same time, minimize the effect of the presence of outliers in the estimation samples. To the best of our knowledge, this was the first time that robust and sparse models were



Table 12 – RMSE values for test samples for Industrial Dryer dataset (MIMO system) in scenario with 5% f outliers.

Models	5% of outliers			#PV
	RMSE - $\hat{y}^{(1)}$	RMSE - $\hat{y}^{(2)}$	RMSE - $\hat{y}^{(3)}$	
LSSVR	1.08E0 ± 0.00E0	4.95E-1 ± 0.00E0	<b>4.99E-1 ± 0.00E0</b>	600
W-LSSVR	<b>9.19E-1 ± 0.00E0</b>	3.35E-1 ± 0.00E0	5.02E-1 ± 0.00E0	600
IR-LSSVR	9.24E-1 ± 0.00E0	<b>3.33E-1 ± 0.00E0</b>	5.06E-1 ± 0.00E0	600
FS-LSSVR	1.05E0 ± 3.67E-2	5.59E-1 ± 2.30E-2	5.02E-1 ± 5.34E-3	<b>60</b>
RFS-LSSVR	8.54E-1 ± 1.39E-2	3.32E-1 ± 8.68E-3	<b>5.01E-1 ± 4.92E-3</b>	<b>60</b>
R <sup>2</sup> FS-LSSVR	<b>8.25E-1 ± 1.21E-2</b>	<b>3.10E-1 ± 1.20E-2</b>	5.07E-1 ± 3.47E-3	<b>60</b>

derived from the solution of the LSSVR optimization problem in primal space.

The RFS-LSSVR and R<sup>2</sup>FS-LSSVR models were successfully evaluated in system identification tasks with benchmarking synthetic and real-world (including a large-scale dataset) datasets corresponding to SISO and MIMO systems. For the synthetic datasets and using infinite-steps ahead prediction, our proposals presented better performance, in terms of RMSE values, than the standard LSSVR, W-LSSVR and IR-LSSVR models in almost all scenarios. The superior performances of the proposed models were even more evident as the amount of outliers was increased. For the real-world datasets using 3-steps ahead prediction (Actuator dataset) and free simulation (Dryer dataset), at least one of the proposed approaches achieved better performances than all of the dual models in the scenarios with outliers. In general, the primal and dual models achieved performances quite close each other in the scenarios with no outliers.

In the experiments with a large-scale dataset (Silverbox dataset) in free simulation, the RFS-LSSVR and R<sup>2</sup>FS-LSSVR models significantly outperformed the FS-LSSVR model in the presence of outliers and presented similar performance in free-outlier case. For the Industrial Dryer dataset (corresponding to a MIMO system) in 3-steps ahead prediction scenario, in general, all the primal and dual models obtained performances quite close each other for each predicted output, in the scenario without outliers. However, in the case with 5% of contamination of outliers, the RFS-LSSVR and R<sup>2</sup>FS-LSSVR approaches outperformed the original FS-LSSVR and all the dual robust models for each evaluated output. One should highlight that, for the best of our knowledge, this was the first time that a robust system identification problem for a MIMO system was addressed.

Finally, we believe it is worth emphasizing the high sparsity level of the proposed RFS-LSSVR and R<sup>2</sup>FS-LSSVR models. In all of the aforementioned experiments, with synthetic,

real and large-scale datasets, the proposed approaches used much less training instances as prototype vectors, compared to the LSSVR, W-LSSVR and IR-LSSVR models. Such a high sparsity level of the solutions provided by the proposed models is of great relevance for memory savings in large-scale data modeling tasks, a common requirement in big data applications.

The next chapter will present the second part of the contributions of this thesis, which corresponds to novel online and/or robust kernel-based approaches derived from the KRLS model.

## 6 NOVEL ONLINE KERNEL-BASED MODELS

“Science is the acceptance of what works and the rejection of what does not.

That needs more courage than we might think.”

(Jacob Bronowski)

In this chapter, we develop the second part of the contributions of this thesis, which comprises two kernel-based filtering methods derived from the original KRLS model. Initially, Section 6.1 contextualizes the framework of kernel-based models associated with linear filters algorithms in order to solve nonlinear regression problems. Then, Section 6.2 presents a robust approach based on the  $M$ -estimation theory, called the ROB-KRLS model, which can be understood as a robust version to outliers of the KRLS model. The obtained results of computational experiments with the ROB-KRLS model are also presented and discussed in Section 6.2. In sequence, Section 6.3 develops the proposed OS-LSSVR model, which is based on the standard LSSVR formulation and is solved according to the KRLS online learning strategy. This section also encompasses the results of computational experiments with this proposal. Finally, the chapter is concluded in Section 6.4.

### 6.1 Introduction

As mentioned before, a further characteristic common to the SVR and LSSVR kernel-based models is the need of a batch learning process, since all the evaluated instances are used in their training phase to build the kernel matrix. Thus, despite these methods being powerful techniques in nonlinear regression and having a certain simplicity in selecting the kernel functions, for which the single Gaussian kernel performs very well on most problems (FALCK *et al.*, 2012), their kernel matrices scales quadratically with the number of training patterns and, consequently, this limits the amount of data that can be processed directly.

In this context, there are some challenges in using kernel models based on batch learning strategy. One of them consists in handling large-scale datasets, where the amount of data is too large to apply batch algorithms and, therefore, the model solution has to be updated sequentially to account for all processed data. Moreover, in many signal processing applications, it is necessary to have an online updating of the model solution. The reason might be that the knowledge about the signal or system properties accumulates as more observations become available and that a model based on current information is necessary to make an online

decision (LJUNG, 2002). Then, although the SVR/LSSVR models can be retrained from the scratch when the training set is modified, it is cumbersome and computationally inefficient. Thus, it is necessary to develop recursive and online algorithms to update the model efficiently.

In essence, the field of KAF models have attracted a great deal of attention from adaptive filtering and system identification communities (SAIDE *et al.*, 2015; VAERENBERGH *et al.*, 2012; TAOUALI *et al.*, 2012; FALCK *et al.*, 2012; ZHU *et al.*, 2012; LIU *et al.*, 2010; RICHARD *et al.*, 2009; TANG *et al.*, 2006; LIU *et al.*, 2008; VAERENBERGH *et al.*, 2007; ENGEL *et al.*, 2004). A successful example of this field is the KRLS model (ENGEL *et al.*, 2004), which is a kernelized version of the classical RLS algorithm. In this particular case, the support vectors are reduced to a sparse dictionary of input samples and a new sample is only added if it can not be represented by a combination of other patterns that are already present in the dictionary.

Despite its widespread use, the performance of the RLS algorithm (and hence, the KRLS model) drastically degrades or even completely breaks down when the estimation data is corrupted with outlier-like samples, e.g. impulsive disturbances (PAPAGEORGIU *et al.*, 2015; ZOUBIR *et al.*, 2012). This happens because the RLS is optimal (i.e., unbiased estimator with minimal variance) only for normally distributed errors, an assumption that leads to objective functions which are built upon the SSE. Such functions assign equal weights to all error terms, but outliers tend to generate large error terms eventually biasing the whole parameter estimation process and, hence, deteriorating the predictor's performance.

Regarding the discussion above, we offer an alternative approach by following the same line of reasoning that gave rise to the KRLS algorithm. In this sense, we present a theoretical development which eventually leads to a robust version of this algorithm, henceforth called the ROB-KRLS model. This approach is built upon concepts derived from the  $M$ -estimation framework, in which the SSE criterion is a special case. One should highlight that the proposed ROB-KRLS model keeps the same computational complexity of the standard KRLS and does not require extra parameters for tuning performance.

In another line of thinking, and aware of the limitations of the LSSVR model with regards to the aspects of nonsparsity of its solution and the required batch mode learning, it is highly desirable to devise online learning strategies for the LSSVR model that also culminate in a sparse solution vector. Then, we posit that a model building framework can be achieved if we simply incorporate the ALD strategy into the standard LSSVR formulation. This procedure

gives rise to the *Online Sparse LSSVR* (OS-LSSVR) model, which, as a consequence, can learn from data incrementally and achieve sparsity also in an adaptive manner. Furthermore, since the LSSVR theory is developed based on a regularized loss function formulated in the dual space, the resulting OS-LSSVR model is also regularized.

## 6.2 Proposed Robust KAF Model

The original KRLS model, which was properly discussed in Chapter 4, is briefly reviewed here for the sake of better understanding our approach. For this purpose, let us assume a stream of training examples  $\mathcal{D}_t = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_t, y_t)\}$ , where  $(\mathbf{x}_t, y_t) \in \mathbb{R}^d \times \mathbb{R}$  denotes the current input-output pair.

One should remember that the KRLS algorithm (assuming a functional form  $f(\mathbf{x}_i) = \boldsymbol{\phi}^\top(\mathbf{x}_i)\mathbf{w}$  minimizes, at each time step  $t$ , the following cost function:

$$J(\mathbf{w}) = \sum_{i=1}^t (y_i - f(\mathbf{x}_i))^2 = \|\mathbf{y}_t - \boldsymbol{\Phi}_t^\top \mathbf{w}\|^2, \quad (6.1)$$

where  $\boldsymbol{\phi}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^{d_h}$ ,  $\mathbf{w} \in \mathbb{R}^{d_h}$ ,  $\boldsymbol{\Phi}_t = [\boldsymbol{\phi}(\mathbf{x}_1), \dots, \boldsymbol{\phi}(\mathbf{x}_t)] \in \mathbb{R}^{d_h \times t}$  is a matrix storing all the projected vectors and  $\mathbf{y}_t = (y_1, \dots, y_t)^\top$  is the vector of outputs.

According to Engel *et al.* (2004), the optimal vector  $\mathbf{w}_t$  can be expressed as

$$\mathbf{w}_t = \sum_{i=1}^t \alpha_i \boldsymbol{\phi}(\mathbf{x}_i) = \boldsymbol{\Phi}_t \boldsymbol{\alpha}_t, \quad (6.2)$$

where  $\boldsymbol{\alpha}_t = (\alpha_1, \dots, \alpha_t)^\top$ . Then, the functional in Eq. (6.1) can be rewritten as

$$J(\boldsymbol{\alpha}_t) = \|\mathbf{y}_t - \mathbf{K}_t \boldsymbol{\alpha}_t\|^2, \quad (6.3)$$

where  $\mathbf{K}_t = \boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t$  is the kernel matrix at instant  $t$ .

Remembering the ALD sparsification criterion, as previously discussed in Section 4.3.2, let us assume that at time step  $t$  ( $2 \leq t \leq N$ ) we have collected a dictionary of SVs comprised of a subset of relevant training inputs  $\mathcal{D}_{t-1}^{\text{sv}} = \{\tilde{\mathbf{x}}_j\}_{j=1}^{m_{t-1}}$ . Then, when a new incoming sample  $\mathbf{x}_t$  is available, one must verify if  $\boldsymbol{\phi}(\mathbf{x}_t)$  is approximately linearly dependent on the dictionary vectors. If the test reveals that  $\boldsymbol{\phi}(\mathbf{x}_t)$  is independent on the dictionary vectors,  $\mathbf{x}_t$  must be added to the dictionary.

Therefore, to test if a training vector  $\mathbf{x}_t$  should be added or not to the dictionary, it is necessary to estimate the vector  $\mathbf{a} = (a_1, \dots, a_{m_{t-1}})^\top$  satisfying the following expression:

$$\delta_t \stackrel{\text{def}}{=} \min_{\mathbf{a}} \left\| \sum_{m=1}^{m_{t-1}} a_m \boldsymbol{\phi}(\tilde{\mathbf{x}}_m) - \boldsymbol{\phi}(\mathbf{x}_t) \right\|^2 \leq \nu, \quad (6.4)$$

where  $\nu$  is the sparsity level parameter. Developing the minimization in Eq. (6.4), we can write

$$\delta_t = \min_{\mathbf{a}} \{ \mathbf{a}^\top \tilde{\mathbf{K}}_{t-1} \mathbf{a} - 2 \mathbf{a}^\top \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) + k_{tt} \}, \quad (6.5)$$

whose solution is given by

$$\mathbf{a}_t = \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t), \quad (6.6)$$

for which we have

$$\delta_t = k_{tt} - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \mathbf{a}_t \leq \nu. \quad (6.7)$$

If otherwise  $\delta_t > \nu$ , the current dictionary must be expanded adding  $\mathbf{x}_t$ . Thus,  $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \cup \{\mathbf{x}_t\}$  and  $m_t = m_{t-1} + 1$ .

Therefore, according to Engel *et al.* (2004), it is possible to rewrite the problem in Eq. (6.3) as

$$J(\tilde{\boldsymbol{\alpha}}_t) = \|\mathbf{y}_t - \mathbf{A}_t \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t\|^2, \quad (6.8)$$

where  $\mathbf{A}_t = [\mathbf{a}_1 \ \mathbf{a}_2 \ \cdots \ \mathbf{a}_t]^\top \in \mathbb{R}^{t \times m_t}$ ,  $\tilde{\boldsymbol{\alpha}}_t \in \mathbb{R}^{m_t}$  is the reduced vector of  $m_t$  coefficients and  $\tilde{\mathbf{K}}_t$  is the kernel matrix built with the dictionary input samples. Then, the minimization of Eq. (6.8) yields the following solution:

$$\tilde{\boldsymbol{\alpha}}_t = \tilde{\mathbf{K}}_t^{-1} (\mathbf{A}_t^\top \mathbf{A}_t)^{-1} \mathbf{A}_t^\top \mathbf{y}_t. \quad (6.9)$$

By defining a matrix  $\mathbf{P}_t$  as

$$\mathbf{P}_t = (\mathbf{A}_t^\top \mathbf{A}_t)^{-1}, \quad (6.10)$$

one gets

$$\tilde{\boldsymbol{\alpha}}_t = \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t \mathbf{A}_t^\top \mathbf{y}_t. \quad (6.11)$$

In the next section we develop a robust approach for computing the matrix  $\mathbf{P}_t$  in Eq. (6.10) to estimate the vector  $\tilde{\boldsymbol{\alpha}}_t$  iteratively in outlier-contaminated scenarios.

### 6.2.1 The ROB-KRLS Model

Our goal now is to present a robust KRLS predictor model for system identification scenarios where the time series data are contaminated with heavy-tail (i.e. non-Gaussian) noise

or outliers. The proposed approach, called the ROB-KRLS model, is based on the extension of the  $M$ -estimators concepts to the KRLS predictor model, aiming at turning it robust to outliers.

However, instead of following a heuristic approach as done by Zou *et al.* (2000) for the RLS-based linear predictor, we adopt a principled approach for the nonlinear predictor where the parameter updating rule is derived from a suitable cost function. This means that, instead of minimizing the standard squared norm of the error vector as in Eq. (6.8), a robust KRLS predictor model is designed to minimize the following cost function:

$$J(\tilde{\boldsymbol{\alpha}}_t^r) = \rho(\mathbf{y}_t - \mathbf{A}_t \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t^r) = \rho(\mathbf{e}_t), \quad (6.12)$$

where  $\tilde{\boldsymbol{\alpha}}_t^r$  is the robust version of  $\tilde{\boldsymbol{\alpha}}_t$ ,  $\mathbf{e}_t = [e_1, \dots, e_i, \dots, e_t]^\top$  is the vector of prediction errors up to the time step  $t$  and  $\rho(\cdot)$  is an  $M$ -estimate function that computes the contribution of each error  $e_i = y_i - \mathbf{a}_i^\top \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t^r$  to the cost function in Eq. (6.12). Note that the standard least squares estimator is achieved if we set  $\rho(\cdot) = \|\cdot\|^2$ .

In order to minimize the cost function in Eq. (6.12), an optimal solution vector  $\boldsymbol{\alpha}_t^r$  is searched by computing the gradient vector  $\nabla_{\tilde{\boldsymbol{\alpha}}_t^r} J(\tilde{\boldsymbol{\alpha}}_t^r) = \partial J(\tilde{\boldsymbol{\alpha}}_t^r) / \partial \tilde{\boldsymbol{\alpha}}_t^r$  and equaling it to zero, which is given by

$$\frac{\partial \rho(\mathbf{e}_t)}{\partial \tilde{\boldsymbol{\alpha}}_t^r} \mathbf{A}_t \tilde{\mathbf{K}}_t = \mathbf{0}. \quad (6.13)$$

If we define a weight function for the prediction error vector  $\mathbf{e}_t$  as

$$v(\mathbf{e}_t) = \frac{1}{\mathbf{e}_t} \frac{\partial \rho(\mathbf{e}_t)}{\partial \tilde{\boldsymbol{\alpha}}_t^r}, \quad (6.14)$$

then, the problem in Eq. (6.13) may be written as

$$\mathbf{e}_t \mathbf{V}_t \mathbf{A}_t \tilde{\mathbf{K}}_t = (\mathbf{y}_t - \mathbf{A}_t \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t^r) \mathbf{V}_t \mathbf{A}_t \tilde{\mathbf{K}}_t = \mathbf{0}, \quad (6.15)$$

with  $\mathbf{V}_t = \text{diag}\{v_i\}_{i=1}^t \in \mathbb{R}^{t \times t}$  being a diagonal matrix, where each diagonal element is the weight  $v_i$  associated to the respective prediction error  $e_i$ .

Note that the expression in Eq. (6.15) defines a weighted least-squares problem. However, the weights  $v_i$ 's depend upon the errors  $e_i$ 's, these errors depend upon the estimated vectors  $\tilde{\boldsymbol{\alpha}}_i^r$ 's, and these estimated vectors depend upon the weights (FOX, 2002). Therefore, an iterative procedure is required based on iteratively reweighted least squares (IRLS) algorithm (WEDDERBURN, 1974), which approximated solution at step  $t$  is given by

$$\tilde{\boldsymbol{\alpha}}_t^r = \tilde{\mathbf{K}}_t^{-1} (\mathbf{A}_t^\top \mathbf{V}_t \mathbf{A}_t)^{-1} \mathbf{A}_t^\top \mathbf{V}_t \mathbf{y}_t. \quad (6.16)$$

Then, as done for the KRLS predictor, we define a matrix  $\mathbf{P}_t^r$  as

$$\mathbf{P}_t^r = (\mathbf{A}_t^\top \mathbf{V}_t \mathbf{A}_t)^{-1}, \quad (6.17)$$

and

$$\tilde{\boldsymbol{\alpha}}_t^r = \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t^r \mathbf{A}_t^\top \mathbf{V}_t \mathbf{y}_t. \quad (6.18)$$

Next, we adopt the same procedure developed for the original KRLS algorithm to iteratively solve Eq. (6.18). For this purpose, we have two possible situations, which are better described below. Furthermore, the general formulation of the ROB-KRLS model, including all the mathematical derivations in detail, is presented in Appendix A.

### 6.2.1.1 Case 1 - Unchanged Dictionary

In this case, we get  $\delta_t \leq \nu$ , i.e. the projection of the current input vector can be approximately written as a linear combination of the projections of the support vectors in the dictionary. In other words,  $\boldsymbol{\phi}(\mathbf{x}_t)$  is *approximately* linearly dependent on the projections of the dictionary vectors. Hence,  $\mathbf{x}_t$  is not added to the dictionary and the kernel matrix is not changed. In math terms,  $\mathcal{D}_t^{\text{sv}} = \mathcal{D}_{t-1}^{\text{sv}}$  and  $\tilde{\mathbf{K}}_t = \tilde{\mathbf{K}}_{t-1}$ .

Since  $\mathbf{a}_t$  needs to be computed by Eq. (6.6) to determine  $\delta_t$ , the matrix  $\mathbf{A}_t$  is built iteratively by the inclusion of  $\mathbf{a}_t$ , i.e.  $\mathbf{A}_t = [\mathbf{A}_{t-1}^\top \ \mathbf{a}_t]^\top$ . Then, let us define the matrix  $\mathbf{B}_t^r$  as

$$\begin{aligned} \mathbf{B}_t^r &= \mathbf{A}_t^\top \mathbf{V}_t \mathbf{A}_t \\ &= \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{A}_{t-1} + v_t \mathbf{a}_t \mathbf{a}_t^\top \\ &= \mathbf{B}_{t-1}^r + v_t \mathbf{a}_t \mathbf{a}_t^\top. \end{aligned} \quad (6.19)$$

This way, using a procedure similar to the one used for the standard RLS algorithm, which is based on the matrix inversion lemma (see Lemma 3.1), we can compute recursively the matrix  $\mathbf{P}_t^r$  as follows:

$$\mathbf{P}_t^r = (\mathbf{B}_t^r)^{-1} = \mathbf{P}_{t-1}^r - \frac{\mathbf{P}_{t-1}^r v_t \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1}^r}{1 + v_t \mathbf{a}_t^\top \mathbf{P}_{t-1}^r \mathbf{a}_t}. \quad (6.20)$$

We define a robust gain vector  $\mathbf{q}_t^r$  as

$$\mathbf{q}_t^r = \frac{v_t \mathbf{P}_{t-1}^r \mathbf{a}_t}{1 + v_t \mathbf{a}_t^\top \mathbf{P}_{t-1}^r \mathbf{a}_t}, \quad (6.21)$$

and consequently

$$\mathbf{P}_t^r = \mathbf{P}_{t-1}^r - \mathbf{q}_t^r \mathbf{a}_t^\top \mathbf{P}_{t-1}^r. \quad (6.22)$$



Finally, using the fact that  $\mathbf{A}_t^\top \mathbf{V}_t \mathbf{y}_t = \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{y}_{t-1} + \mathbf{a}_t v_t \mathbf{y}_t$ , the ROB-KRLS update rule for  $\tilde{\boldsymbol{\alpha}}_t^r$  can be written as

$$\begin{aligned} \tilde{\boldsymbol{\alpha}}_t^r &= \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t^r \mathbf{A}_t^\top \mathbf{V}_t \mathbf{y}_t, \\ &= \tilde{\boldsymbol{\alpha}}_{t-1}^r + \tilde{\mathbf{K}}_t^{-1} \mathbf{q}_t^r (y_t - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}^r), \end{aligned} \quad (6.23)$$

where the last equalities are based on  $\mathbf{q}_t^r = v_t \mathbf{P}_t^r \mathbf{a}_t$  and  $\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) = \tilde{\mathbf{K}}_t \mathbf{a}_t$ . The complete development to obtain the expression in Eq. (6.23) is presented in Eq. (A.22) of Appendix A.

### 6.2.1.2 Case 2 - Updating the Dictionary

In this case, we get  $\delta_t > v$ , 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  $\mathbf{x}_t$  must be added to the dictionary, i.e.  $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \cup \{\mathbf{x}_t\}$  and  $m_t = m_{t-1} + 1$ . As a consequence of this inclusion, the kernel matrix must be updated accordingly.

The challenge here is to compute  $\tilde{\mathbf{K}}_t$  (and hence  $\tilde{\mathbf{K}}_t^{-1}$ ) recursively using  $\tilde{\mathbf{K}}_{t-1}$  and the information provided by the new sample. For this purpose, based on Golub and VAN LOAN (2013), we compute the matrices  $\tilde{\mathbf{K}}_t$  and  $\tilde{\mathbf{K}}_t^{-1}$  as

$$\tilde{\mathbf{K}}_t = \begin{bmatrix} \tilde{\mathbf{K}}_{t-1} & \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \\ \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top & k_{tt} \end{bmatrix}, \quad (6.24)$$

and

$$\tilde{\mathbf{K}}_t^{-1} = \frac{1}{\delta_t} \begin{bmatrix} \delta_t \tilde{\mathbf{K}}_{t-1}^{-1} + \mathbf{a}_t \mathbf{a}_t^\top & -\mathbf{a}_t \\ -\mathbf{a}_t^\top & 1 \end{bmatrix}. \quad (6.25)$$

One should recall that for the Case 2, not only the dimension of the kernel matrix  $\tilde{\mathbf{K}}_t$  increases due to the inclusion of sample  $\mathbf{x}_t$  into the dictionary, but also the dimensions of the matrices  $\mathbf{A}_t$  and  $\mathbf{V}_t$ . Hence, we can write

$$\mathbf{A}_t^\top \mathbf{V}_t \mathbf{A}_t = \begin{bmatrix} \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{A}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad (6.26)$$

$$\mathbf{A}_t^\top \mathbf{V}_t = \begin{bmatrix} \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad (6.27)$$

and

$$\mathbf{P}_t^r = (\mathbf{A}_t^\top \mathbf{V}_t \mathbf{A}_t)^{-1} = \begin{bmatrix} \mathbf{P}_{t-1}^r & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad (6.28)$$

where  $\mathbf{0}$  is a zero vector of appropriate size. Then, we use Eqs. (6.25)-(6.28) to calculate  $\tilde{\boldsymbol{\alpha}}_t^r$  as

$$\begin{aligned}\tilde{\boldsymbol{\alpha}}_t^r &= \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t^r \mathbf{A}_t^\top \mathbf{V}_t \mathbf{y}_t, \\ &= \tilde{\mathbf{K}}_t^{-1} \begin{bmatrix} (\mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{A}_{t-1})^{-1} \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{y}_{t-1} \\ y_t \end{bmatrix}, \\ &= \begin{bmatrix} \tilde{\boldsymbol{\alpha}}_{t-1}^r - \frac{a_t}{\delta_t} (y_t - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}^r) \\ \frac{1}{\delta_t} (y_t - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}^r) \end{bmatrix},\end{aligned}\quad (6.29)$$

where for the final equality we use  $\mathbf{a}_t^\top \tilde{\mathbf{K}}_{t-1} = \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top$ . The complete development to obtain Eq. (6.29) is presented in Eqs. (A.32) to (A.35) of Appendix A.

It is worth noting that the updating expression that we obtained in Eq. (6.29) for the Case 2 does not depend on the weight  $v_t$ . In other words, this part of the ROB-KRLS algorithm is exactly the same as the one obtained for the KRLS algorithm by Engel *et al.* (2004). This is quite a remarkable theoretical result. One plausible explanation is that, since both models (KRLS and ROB-KRLS) are sparse, the majority of the training samples will cause the algorithm to use the ‘‘Case 1 scenario’’, a situation where a mechanism to constrain the influence of outliers is actually very welcome. Outlying samples can occasionally force the algorithm to enter into ‘‘Case 2’’, but it will be very rare (as we verified in preliminary experiments) if we compare to the number of situations where ‘‘Case 1 scenario’’ is used.

Since the parameter vector  $\tilde{\boldsymbol{\alpha}}_t^r$  has been updated (either for Case 1, or Case 2), the sparse and robust solution for the ROB-KRLS nonlinear predictor is given by

$$\hat{y}_t = \hat{f}(\mathbf{x}) = \sum_{m=1}^{m_t} \tilde{\alpha}_m^r k(\mathbf{x}, \mathbf{x}_m) = \tilde{\mathbf{k}}_{m_t}(\mathbf{x})^\top \tilde{\boldsymbol{\alpha}}_t^r. \quad (6.30)$$

Finally, we can infer that the ROB-KRLS model keeps the same general structure of the original KRLS, since it only adds the M-estimation framework based on weighted least squares algorithm to the KRLS formulation. Therefore, regarding the ROB-KRLS model, its computational complexity is  $\mathcal{O}(m_t^2)$  and its memory demand is  $\mathcal{O}(Nm_t)$ , as in the KRLS case.

### 6.2.1.3 On the Choice of the Weight Function

As an important issue of the proposed nonlinear prediction model, we make a few comments about the choice of the M-estimation function  $\rho(\cdot)$  and the corresponding weight function  $v(\cdot)$ . In this regard, many weight functions have been proposed in the literature, especially for linear regression (ROUSSEEUW; LEROY, 1987). Some of them, such as Huber’s,

---

**Algorithm 13:** - Pseudo-code for the ROB-KRLS model.

---

**Require:**  $\nu, \sigma$  (for Gaussian kernel);  
**Set:**  $\tilde{\mathbf{K}}_1 = k_{11}; \tilde{\mathbf{K}}_1^{-1} = 1/\tilde{\mathbf{K}}_1; \mathbf{P}_1^r = 1; \tilde{\boldsymbol{\alpha}}_1^r = y_1/k_{11}; \mathbf{A}_1 = 1; m_1 = 1;$   
**for**  $t = 2 : N$ , **do**  
  Get new sample  $(\mathbf{x}_t, y_t)$  and compute  $\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t);$   
   $\mathbf{a}_t = \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t);$   
   $\delta_t = k_{tt} - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \mathbf{a}_t;$   
   $e_t = y_t - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}^r;$   
  **if**  $\delta_t > \nu$  **then**  
     $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \cup \{\mathbf{x}_t\};$  % add  $\mathbf{x}_t$  to the dictionary  
    Compute  $\tilde{\mathbf{K}}_t^{-1}$  from Eq. (6.25);  
    Compute  $\mathbf{P}_t^r$  from Eq. (6.28);  
    Compute  $\tilde{\boldsymbol{\alpha}}_t^r$  from Eq. (6.29);  
     $m_t = m_{t-1} + 1;$   
  **else**  
     $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv};$  % unchanged dictionary  
    Compute  $v_t$  from Eq. (6.32);  
    Compute  $\mathbf{q}_t^r$  from Eq. (6.21);  
    Compute  $\mathbf{P}_t^r$  from Eq. (6.22);  
    Compute  $\tilde{\boldsymbol{\alpha}}_t^r$  from Eq. (6.23);  
  **end if**  
**end for**  
Output  $\tilde{\boldsymbol{\alpha}}_t^r, \mathcal{D}_t^{sv}.$

---

Hampel's and logistic functions are common choices in regression problems (DE-BRABANTER *et al.*, 2009b; DEBRUYNE *et al.*, 2008; DE-BRABANTER *et al.*, 2012).

For the ROB-KRLS prediction model, we decide to use the logistic function, which is defined as

$$\rho(e_t) = e_t \tanh(e_t), \quad (6.31)$$

and the correspondent weight function is

$$v_t = v(e_t) = \frac{\tanh(e_t)}{e_t}, \quad (6.32)$$

where  $e_t = y_t - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}^r$  is the a priori estimation error at the step time  $t$ .

One reason of our choice is due the simplicity of the logistic function since, unlike Huber's and Hampel's functions, it has no extra parameters to be tuned. A second reason is because reweighting using Hampel's function (as done by Zou *et al.* (2000), for instance) results in assigning zero weight when an error  $e_t$  is too large, which may occur with outliers and with their neighboring input samples (DEBRUYNE *et al.*, 2008). This means that if a weight  $v_t$  is zero, the gain vector  $\mathbf{q}_t^r$  in Eq. (6.21) is also null and, consequently, the vector  $\tilde{\boldsymbol{\alpha}}_t^r$  in Eq. (6.23)

is not updated for all of those input samples around the outliers. Thus, the convergence of the predictor model is slowed down, since training is performed online. By using the logistic function instead, the neighboring samples of outliers receive small weights and the outliers get even smaller ones. However, all of those weights are nonzero. Therefore, in some applications the logistic function can achieve better performance than the Hampel's function (DEBRUYNE *et al.*, 2008). The resulting ROB-KRLS model is summarized in Algorithm 13.

### 6.2.2 Computational Experiments

In this section, we report and discuss the results of comprehensive computer simulations comparing the proposed ROB-KRLS approach to the original KRLS model and other state-of-the-art robust KAF methods, namely, the Kernel Maximum Correntropy (KMC) (ZHAO *et al.*, 2011) and Kernel Recursive Maximum Correntropy (KRMC) (WU *et al.*, 2015) models. Next, we give a general explanation about the configuration of the performed experiments.

1. Evaluated Datasets - For the experiments in nonlinear system identification tasks, we use two synthetic datasets (Synthetic-1 and Synthetic-2) and two real-world datasets (Actuator and Silverbox), where the latter is a large-scale dataset. All of them were properly described in Chapter 5.
2. Outlier Contamination - We use a similar outlier generation methodology to the one performed by Mattos *et al.* (2016) and previously discussed in Chapter 5. Thus, in addition to the standard Gaussian noise added to estimation samples, we also contaminate the outputs of the estimation data (for Synthetic-1 and Synthetic-2 datasets) by replacing them randomly with outliers covering 0%, 5%, 10%, 15%, 20%, 25% and 30% of the total of estimation samples. Furthermore, we use the same procedure to contaminate with 5% (for Silverbox dataset) and 10% (for Actuator dataset) of outliers the estimation outputs of the real-world datasets.
3. Methodology - For the synthetic and Actuator datasets, we performed 5-fold cross validation strategy to set the hyperparameters  $\nu$  (used in the ALD test criterion),  $\sigma$  (used by the Gaussian kernel),  $\sigma_1$  (used by the KMC and KRMC models to compute the error weights), the learning rate  $\eta$  (for the KMC model), the regularization factor  $\gamma_2$  (for the KRMC model) and  $\delta_1, \delta_2$ , used by the novelty criterion sparsification method (PLATT, 1991) for the KRMC model, in the search for their optimal values. For the Silverbox dataset, we follow the same experimental setup described in Section 5.3.5 in order to search the

Table 13 – Important features of the evaluated datasets and models for the computational experiments.

<b>Important Features of the Evaluated Datasets</b>					
Dataset	Prediction Scenario	Training ( $N$ )	Test ( $N'$ )	$\hat{L}_u$	$\hat{L}_y$
Synthetic-1	Free Simulation	150	150	1	1
Synthetic-2	Free Simulation	300	100	1	1
Actuator	3-step ahead	512	512	4	4
Silverbox	Free Simulation	91,072	40,000	10	10

optimal values of the above hyperparameters.

Regarding the input and output regression orders ( $\hat{L}_u$  and  $\hat{L}_y$ ), they were set according to their largest delays from Eqs. (5.16) and (5.17) for the synthetic datasets. For the real-world datasets, we use  $\hat{L}_u = \hat{L}_y = 4$  for the Actuator dataset and  $\hat{L}_u = \hat{L}_y = 10$  for the Silverbox dataset, based on the same reasons discussed earlier in Chapter 5. Some features and corresponding parameters of the evaluated datasets are summarized in Table 13.

Finally, the figure of merit for evaluating the numerical performance of the ROB-KRLS is the RMSE distribution computed for test samples over 20 independent runs.

### 6.2.2.1 Results and Discussion

The computer experiments for evaluating the model performances correspond to the task of iterative  $k$ -step ahead prediction. For the two synthetic datasets (Synthetic-1 and Synthetic-2) and for the Silverbox dataset, we set  $k \rightarrow +\infty$  (free simulation scenario). For the Actuator dataset, we set  $k = 3$  based on obtained performance in exhaustive previous experiments.

#### 6.2.2.1.1 Experiments with Synthetic Datasets

In Fig. (22), we report the RMSE error bars for the synthetic datasets for increasing outlier contamination scenarios. The RMSE error bars produced by the evaluated models for Synthetic-1 dataset are shown in Fig. (22)a, where one can observe that up to 15% of outliers, almost all the models achieved, in general, equivalent performances. The exception was the KMC model, that achieved significantly lower performance compared to the other models, in all scenarios. This was somewhat expected, because the KMC model presents a slower convergence comparing to the RLS derived algorithms (KRLS, ROB-KRLS and KRMC), since it is a robust and kernelized version of the LMS algorithm. For higher contamination levels, the ROB-KRLS

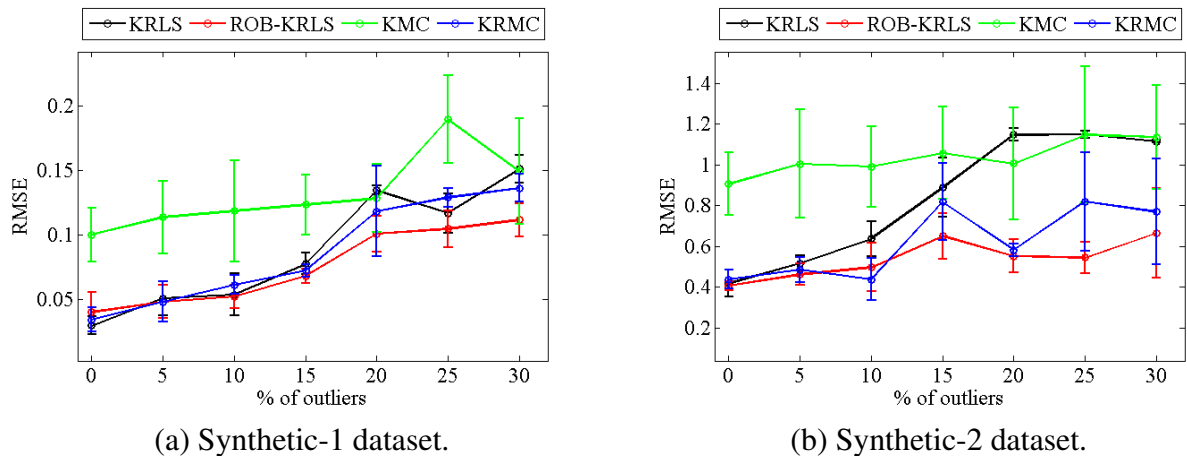


Figure 22 – RMSE values for the test samples over 20 independent runs.

Table 14 – Average number of training input vectors chosen as SVs for the synthetic datasets.

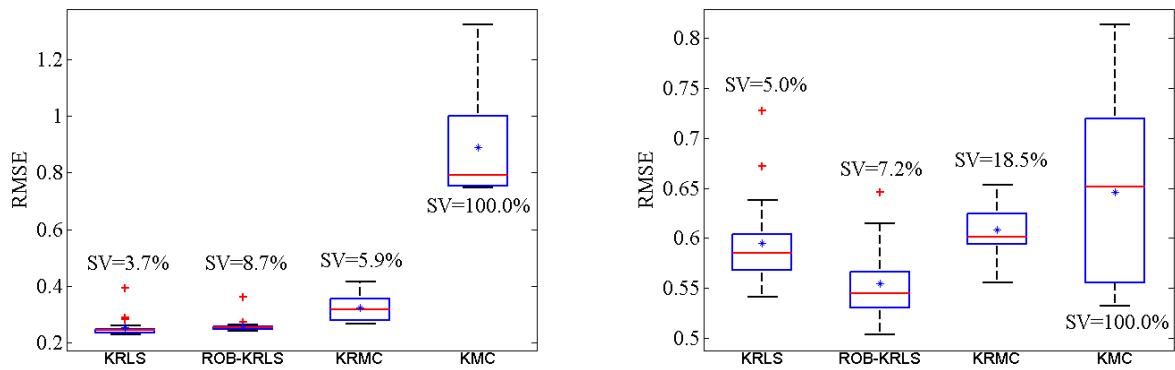
	% of SVs						
	0%	5%	10%	15%	20%	25%	30%
Synthetic-1							
KRLS	8.4	7.6	7.7	7.6	4.7	6.9	3.9
ROB-KRLS	7.9	8.3	10.4	7.4	6.9	7.1	6.8
KRMC	42.9	47.8	30.5	44.8	31.2	21.2	30.3
	% of SV						
Synthetic-2							
KRLS	4.1	6.4	8.9	6.0	1.2	2.8	1.3
ROB-KRLS	4.8	6.2	4.4	5.9	5.0	5.1	1.2
KRMC	14.9	20.6	46.2	19.3	73.8	19.4	39.4

predictor achieved the lowest values of RMSE among the evaluated models.

The RMSE error bars for the Synthetic-2 dataset are shown in Fig. (22)b. It can be seen that for almost all the contamination scenarios, except for the scenarios up to 10% of outliers, the proposed approach presented lower values of RMSE in comparison to the KRLS, KMC and KRMC models. An important issue to be highlighted in this experiment is the high resilience to outliers obtained by the ROB-KRLS model since its performance was just slightly affected, even with the inclusion of increasing amount of outliers. Again, the performance of the KMC model was very poor in all the contamination scenarios.

In order to check the sparsity for each model, in Table 14 we report the average number of training samples that were selected to be included in the dictionary as support vectors along the 20 runs of each model. The absence of results with the KMC model in this table is because its standard formulation, as proposed by Liu *et al.* (2011), is non-sparse.

Firstly, it is possible to note in Table 14 that the ALD sparsity criterion (for the



(a) Outlier-free scenario.

(b) Scenario with 10% of outliers.

Figure 23 – Boxplots for the RMSE values of the test samples after 20 independent runs - Actuator dataset.

KRLS and ROB-KRLS models) seems to be more aggressive than the NC criterion (for the KRMC model), because the KRLS and ROB-KRLS solutions use a significant lower amount of SVs for both artificial datasets comparing to the KRMC model. Additionally, we also observe that, despite the presence of outliers, the ROB-KRLS model successfully achieved very sparse solutions, sometimes even providing more sparse solutions than the KRLS model (specially for the Synthetic-2 dataset).

Elaborating a bit more on the issue of sparsity, in Table 14 we can observe two different trends. While the average number of SVs in the dictionary for the ROB-KRLS model remains approximately the same for all contamination scenarios (for both synthetic datasets), the corresponding numbers for the standard KRLS model tend to decrease as the contamination level increases. This is particularly evident for the Synthetic-2 dataset. It seems that the presence of outliers affected so much the parameter estimation process of the standard KRLS model to the point that it selected just a few training samples to be part of the dictionary. In other words, just a few estimation samples produced  $\delta_t > \nu$  for the ALD test criterion. Most of the estimation samples (including the outlying samples) produced  $\delta_t < \nu$ ; thus, letting the performance of the standard KRLS model to be highly influenced (i.e. biased) by the outliers and, hence, deteriorating its prediction ability.

From this behavior, we can infer that the inclusion of the  $M$ -estimation framework into the KRLS model, which gave rise to the ROB-KRLS model, had a beneficial stabilizing effect on the estimation process, improving its prediction performance eventually.

### 6.2.2.1.2 Experiments with Real-World Datasets

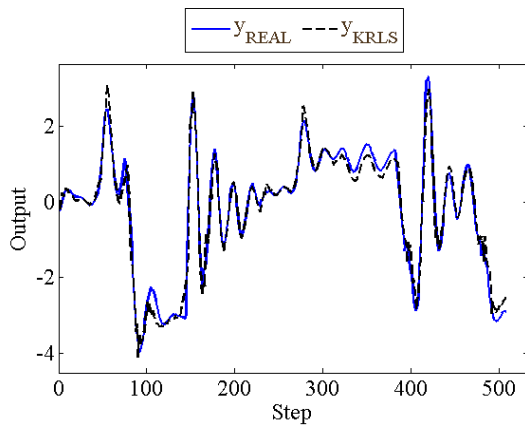
Next, we assess the achieved results for the Actuator dataset considering two different cases: (i) outlier-free scenario, and (ii) scenario with 10% of outliers. In Figs. (23a) and (23b) we report the boxplots of the RMSE values for each model in these scenarios. The corresponding average number of SVs are close to each boxplot. One can note in Fig. (23a) that in the outlier-free scenario, the standard KRLS and ROB-KRLS model achieved smaller RMSE values compared to the KMC and KRMC models, allowing a significantly small number of SVs into the dictionary. An analysis of Fig. (23b) reveals that the performances of all models have been degraded with the insertion of outliers, as expected. However, the ROB-KRLS model was much less affected, since its average RMSE value (the asterisk mark) was smaller than the minimum RMSE values achieved by the other models, a result that corroborates the robustness of the proposed approach.

Finally, we assess the effect of outliers in the prediction performance of the best models (KRLS and ROB-KRLS) according to results in Figs. (23a) and (23b), for both scenarios. In Fig. (24) we report the predicted outputs for the 3-step ahead prediction task in the worst case (those predicted outputs that led to the highest RMSE values among the independent runs) for the outlier-free scenario. In Fig. (25) we report the predicted outputs in the worst case for the scenario with 10% of outliers.

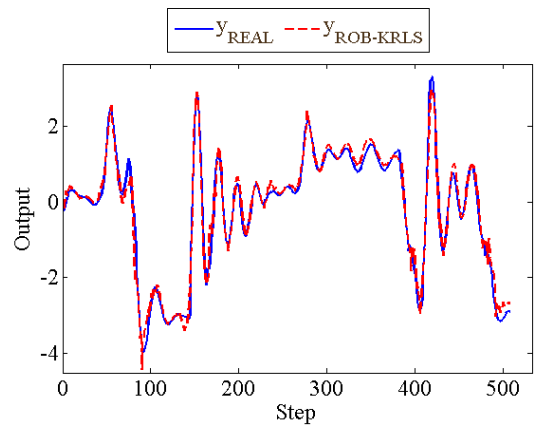
For the outlier-free scenario in Fig. (24), we can easily see that both models produced equivalent satisfactory predictions, being visually indistinguishable in the graphics since the predicted dynamical behaviors are quite close to the actual one. However, for the outlier-corrupted scenario in Fig. (25), it becomes evident the superiority of the proposed approach. The ROB-KRLS model performed much better than the standard KRLS algorithm, following the actual dynamics of the Actuator more closely along the whole duration of the signal. In particular, within the time steps 50 to 250 the ROB-KRLS clearly outperforms the KRLS algorithm. The ROB-KRLS is also better than the KRLS in reaching the peaks (i.e. points of high curvature) of the signal.

Regarding the ROB-KRLS performance for applications in big data, we report in Table 15 the obtained RMSE results (average value  $\pm$  standard deviation) in free simulation, and the amount of support vectors used by the KRLS, ROB-KRLS, KRMC and KMC models in two different scenarios for Silverbox dataset: (i) without outliers and (ii) with 5% of outliers contamination. One can see in this table that, in the scenario without contamination, the KRLS



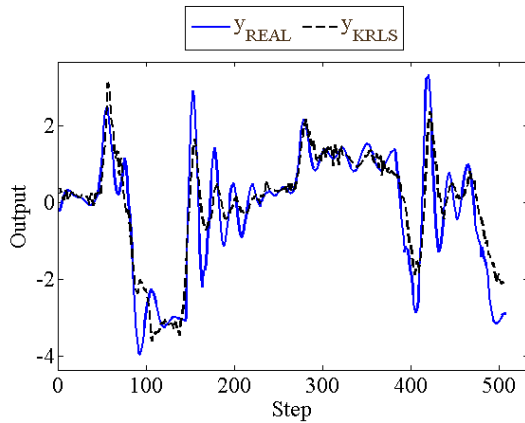


(a) KRLS predicted output.

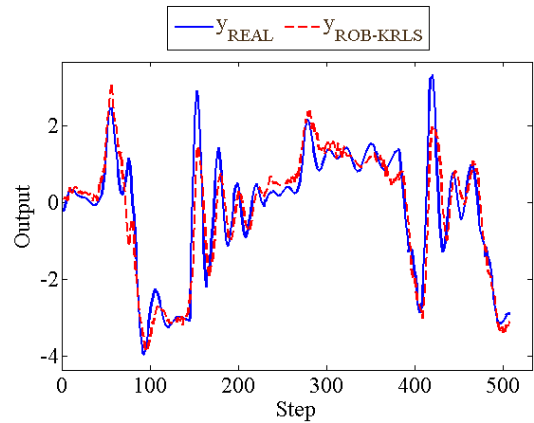


(b) ROB-KRLS predicted output.

Figure 24 – Predicted outputs by the model with worst performance in RMSE along the 20 runs for the outlier-free scenario - Actuator dataset.



(a) KRLS predicted output.



(b) ROB-KRLS predicted output.

Figure 25 – Predicted outputs by the model with worst performance in RMSE along the 20 runs for the scenario with 10% of outliers - Actuator dataset.

and ROB-KRLS models achieved RMSE values an order of magnitude lower than those obtained by the KRMC and KMC models. Furthermore, regarding solution sparsity, our approach ( $\#SV=305$ ) and the KRLS model ( $\#SV=302$ ) used smaller amounts of SV in comparison to the other models. For example, the number of support vectors used by the ROB-KRLS model represents only about 0.33% of the total training set.

In the scenario with 5% of contamination, it is possible to observe in Table 15 that the performances of all the models have been degraded with the insertion of outliers, as somewhat expected. However, the proposed robust approach was considerably less affected, since their RMSE values were smaller than those ones obtained by all the other models, using only 807 training instances as support vectors.

Table 15 – RMSE values for test samples and number of SVs for each evaluated model - Silverbox dataset.

Models	0% of outliers		5% of outliers	
	RMSE	#SV	RMSE	#SV
KRLS	<b>3.50E-3 ± 1.71E-4</b>	302	1.08E-2 ± 6.83E-4	811
ROB-KRLS	<b>3.50E-3 ± 1.68E-4</b>	305	<b>1.02E-2 ± 4.66E-4</b>	807
KRMC	1.18E-2 ± 2.89E-4	733	1.78E-2 ± 2.36E-4	5,949
KMC	4.67E-2 ± 4.23E-5	91,072	8.11E-2 ± 1.11E-5	91,072

### 6.2.2.2 Correlation Analysis of the Residuals

In order to further validate the proposed approach, we apply a set of statistical correlation tests for the residuals produced by the evaluated nonlinear models (BILLINGS; VOON, 1986; ZHANG *et al.*, 2013; AGUIRRE, 2007). For this purpose, the residuals are computed as  $\xi_t = y_t - \hat{y}_t$ , where  $\hat{y}_t$  denotes the  $t$ -th predicted value using the one-step-ahead (OSA) prediction scheme. The objective is to verify if the residuals produced by the model under evaluation on the training dataset are unpredictable (BILLINGS, 2013); in other words, if they do not possess any linear or nonlinear serial correlation (i.e. temporal dependency) with current and previous inputs  $u_t$ , or with previous residuals  $\xi_t$ . Presence of serial correlation may indicate unmodelled dynamics.

Thus, one autocorrelation function and four cross-correlation functions should be computed:

$$\begin{aligned}
r_{\xi\xi}(\tau) &= E\{\xi_{t-\tau}\xi_t\} = \delta_\tau, \\
r_{u\xi}(\tau) &= E\{u_{t-\tau}\xi_t\} = 0, \forall \tau, \\
r_{u^2\xi}(\tau) &= E\{(u_{t-\tau}^2 - \bar{u}_t^2)\xi_t\} = 0, \forall \tau, \\
r_{u^2\xi^2}(\tau) &= E\{(u_{t-\tau}^2 - \bar{u}_t^2)\xi_t^2\} = 0, \forall \tau, \\
r_{\xi(\xi u)}(\tau) &= E\{\xi_t(\xi_{t-1-\tau}u_{t-1-\tau})\} = 0, \tau \geq 0,
\end{aligned} \tag{6.33}$$

where  $E\{\cdot\}$  is the expected value operator,  $\delta(\cdot)$  is the Dirac delta function and  $\bar{u}_t$  denotes the mean value of the control input  $u_t$ . The first two tests in Eqs. (6.33) aims at detecting linear correlations, while the last three are useful for detecting unmodelled nonlinear effects (BILLINGS, 2013). Commonly, the 95% confidence bands are used to decide if the tests are satisfied and the model is validated.

We apply the aforementioned tests to the two best performing models for each dataset,

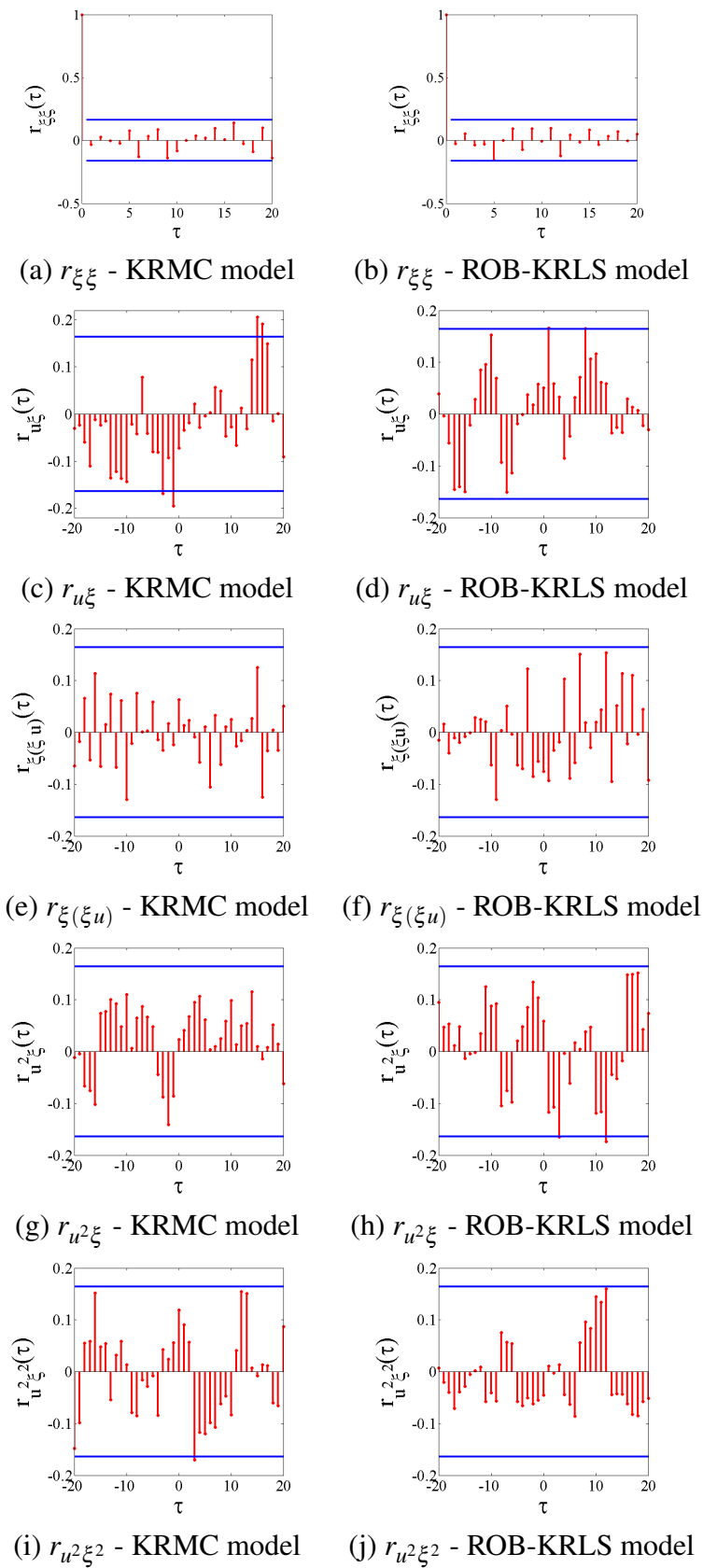


Figure 26 – Correlation tests for input-output models - Synthetic-1 dataset with 30% of outliers.

based on the RMSE results reported in Subsection 6.2.2.1 for the test data. For each tested model, the sequence of residuals used in the correlation analysis is the one which produced the best

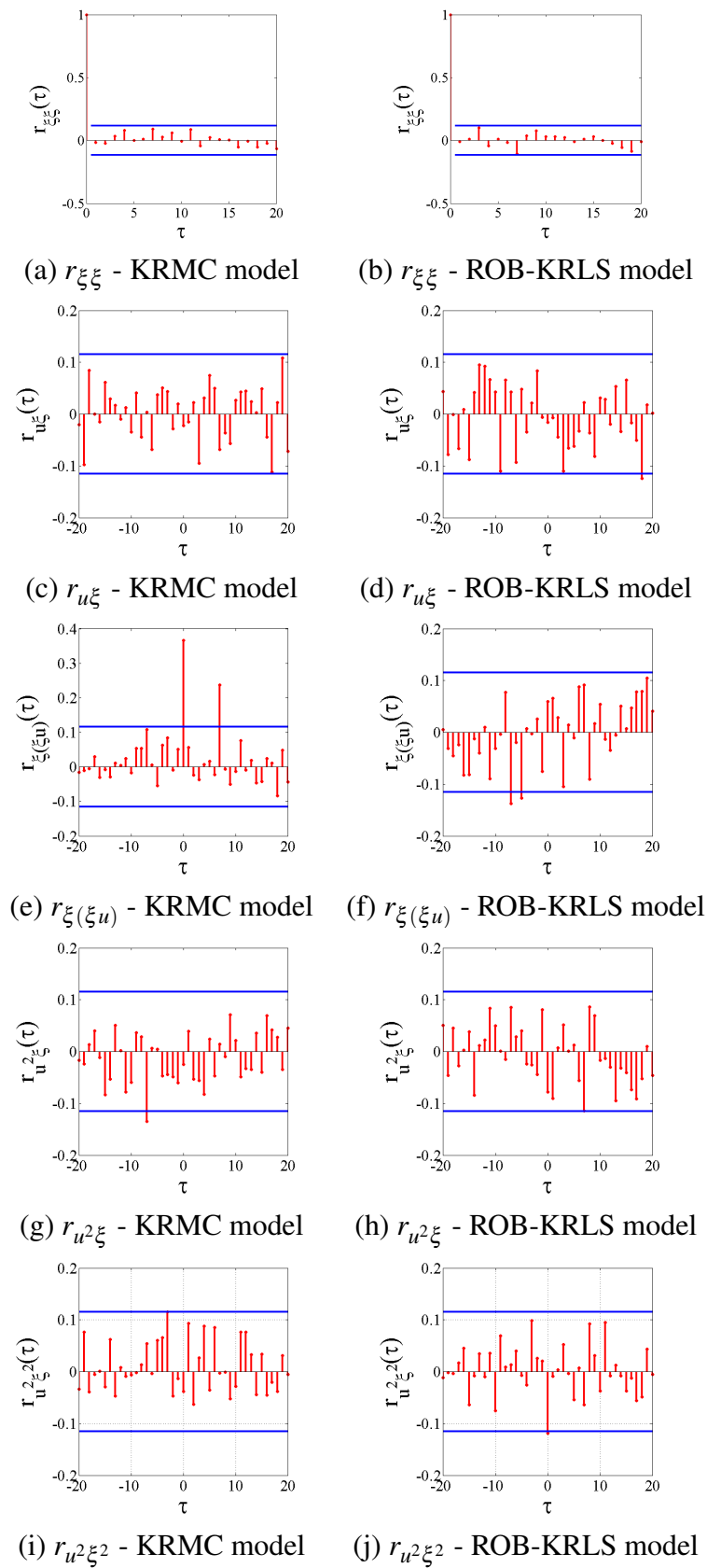


Figure 27 – Correlation tests for input-output models - Synthetic-2 dataset with 30% of outliers.

result (i.e. the smallest RMSE for the test data) among the 20 independent runs, considering the scenario of greater contamination of outliers for each evaluated dataset.

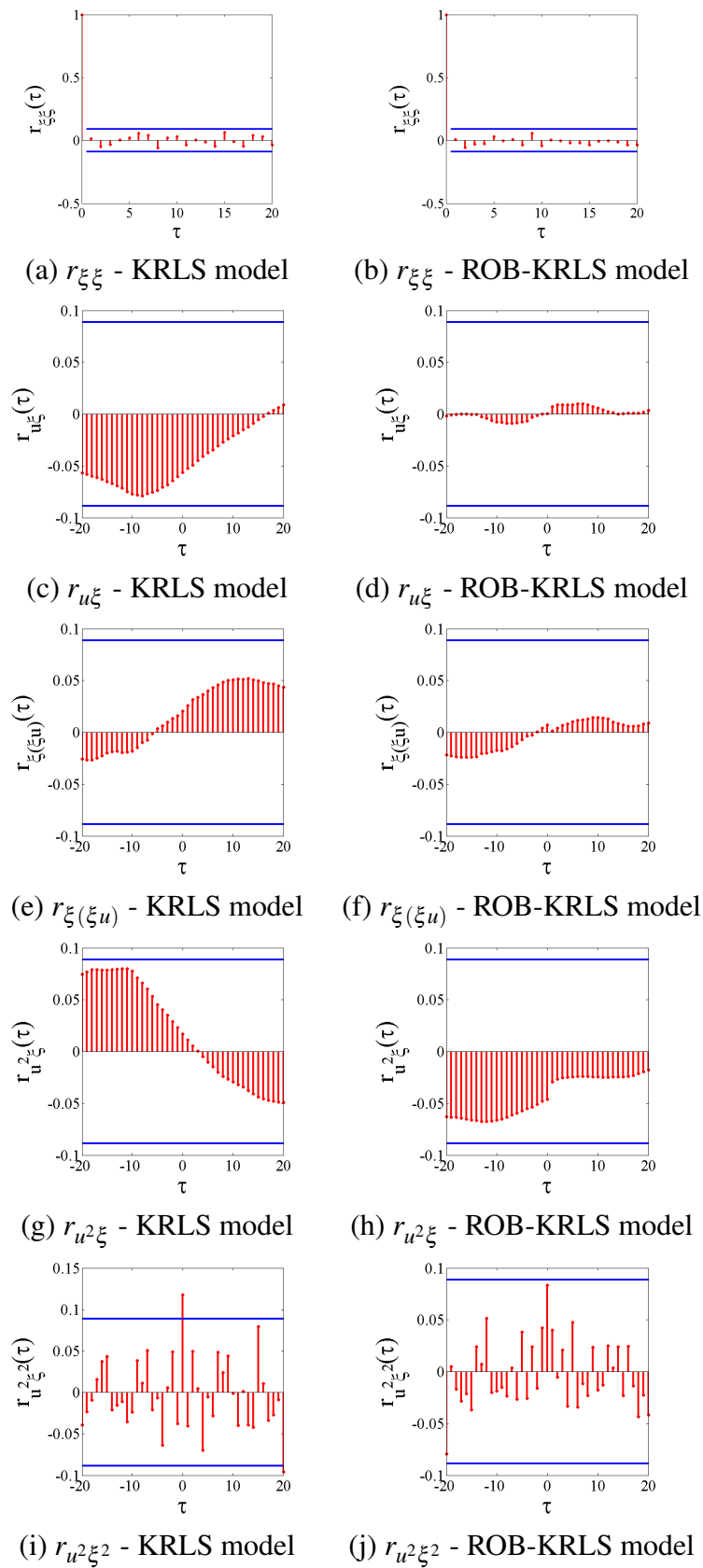


Figure 28 – Correlation tests for input-output models - Actuator dataset with 10% of outliers.

For the Synthetic-1 and Synthetic-2 datasets, the two best performing models are the KRMC and the ROB-KRLS. The corresponding results of the correlation tests are shown

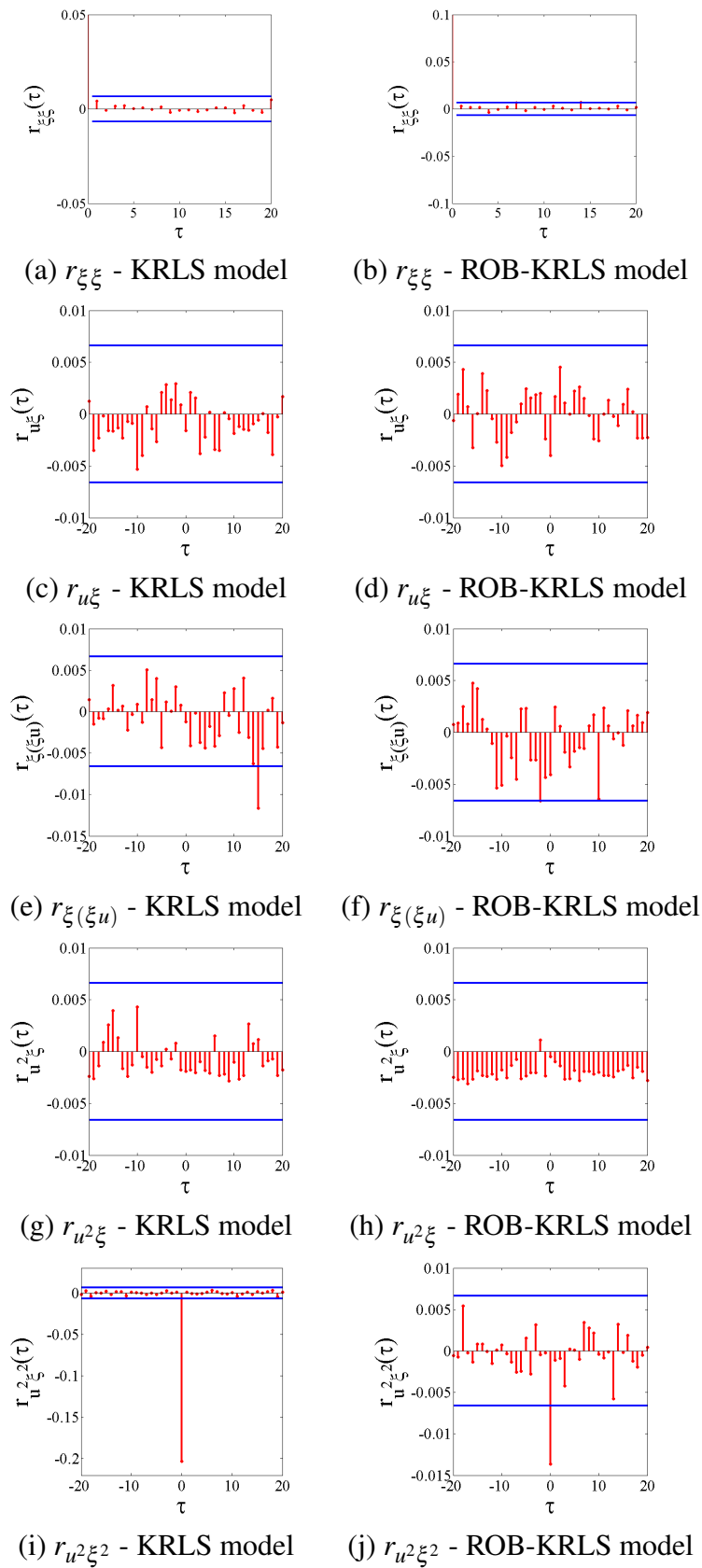


Figure 29 – Correlation tests for input-output models - Silverbox dataset with 5% of outliers.

in Figs. (26) and (27), respectively. A closer analysis of these results indicate that both robust models performed quite well, with a slight advantage to the proposed ROB-KRLS model. The

KRMC model failed to pass 2 (out of 5) the correlation tests for the Synthetic-1 dataset. For the Synthetic-2 dataset, the models achieved similar results, since both fail to pass the  $r_{\xi(\xi_u)}(\tau)$  test, but with the KRMC model violating this test in a more intense degree.

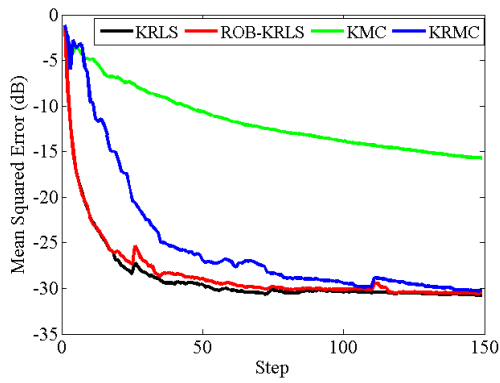
For the two real-world datasets, the Actuator and the Silverbox, the two best performing models are the KRLS and the ROB-KRLS. The corresponding results of the correlation tests are shown in Figs. (28) and (29), respectively. A closer look at these results reveals that the proposed ROB-KRLS clearly outperformed the KRLS model. For the Actuator dataset, the proposed ROB-KRLS model successfully passed all the correlation tests, while the KRLS model failed to pass the  $r_{u^2\xi^2}(\tau)$  test. For the Silverbox dataset, the KRLS model failed to pass 2 (out of 5) the correlation tests, while the proposed ROB-KRLS fail to pass the  $r_{u^2\xi^2}(\tau)$  test only.

As a final remark on the correlation analysis, it is worth mentioning that they were performed under the assumption that the system dynamics follows a NARX model. We also fixed the input and output memory orders  $\hat{L}_u$  and  $\hat{L}_y$  for each dataset. This may not be the best approach for each dataset, and better correlations could be achieved if the best model and memory delays for each dataset were found. It should be noted, however, that our major goal is not to find the best model for a given dataset, but rather to evaluate the performance of the parameter estimation techniques for kernel-based models under the presence of outliers in the estimation data. To facilitate our performance comparison on several benchmarking datasets, we fixed the dynamical model to be common to all datasets and to be of NARX-type. Even doing that, the results of the correlation analysis can be considered satisfactory.

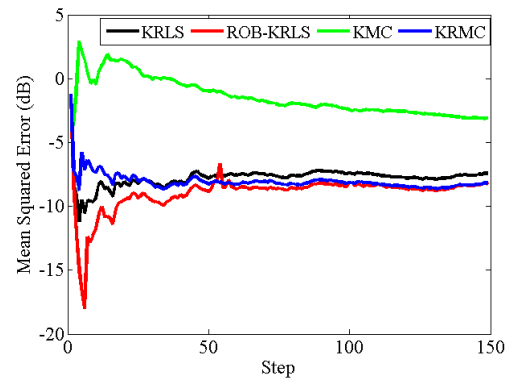
### 6.2.2.3 Convergence Analysis

In order to empirically analyze the convergence behavior of the KRLS, ROB-KRLS, KMC and KRMC models, we have included the ensemble-average learning curves for the evaluated datasets in Fig. (30). In these curves, one plots the mean squared error (MSE) values for each model during the training phase, averaged over the 20 independent runs, versus the number of iterations (which exactly corresponds to the number of training samples).

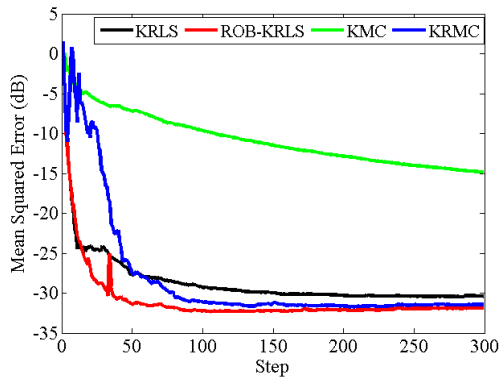
The learning curves were built for each evaluated dataset considering two different scenarios. On the left-hand side of Fig. (30), we report the results for the outlier-free scenario for all evaluated datasets. On the right side of this figure, we report the results for the following outlier-contaminated scenarios: 30% for the Synthetic-1 and Synthetic-2 datasets, 10% for the Actuator dataset and 5% of outliers for the Silverbox dataset.



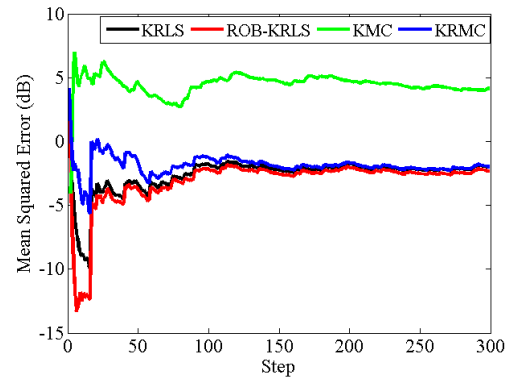
(a) Synthetic-1 without outliers.



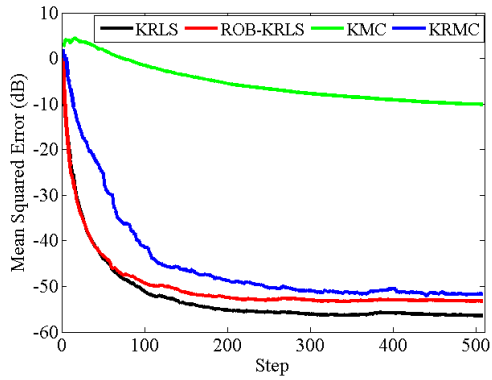
(b) Synthetic-1 with 30% of outliers.



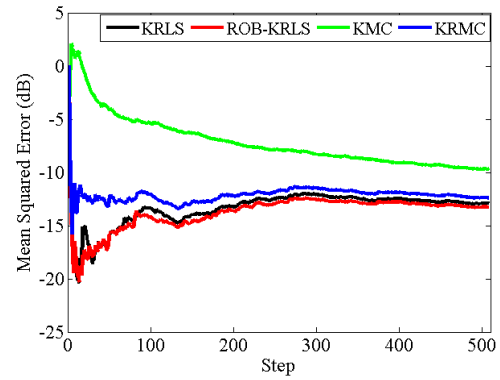
(c) Synthetic-2 without outliers.



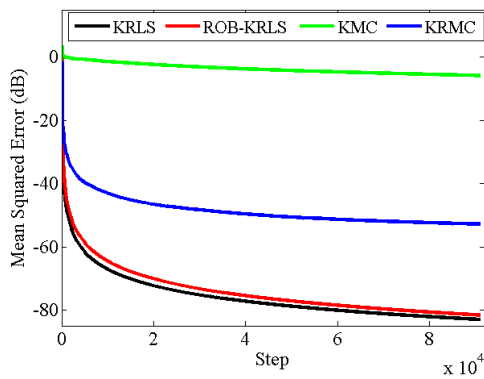
(d) Synthetic-2 with 30% of outliers.



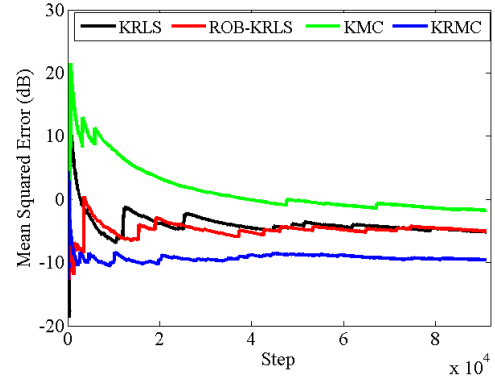
(e) Actuator dataset without outliers.



(f) Actuator dataset with 10% of outliers.



(g) Silverbox dataset without outliers.



(h) Silverbox dataset with 5% of outliers.

Figure 30 – Convergence curves for the evaluated online models.



One can easily see in Fig. (30) that the KMC presented the poorest performance, especially in scenarios without outliers, in terms of MSE values, comparing to the other models. Somehow, this was expected, since it derives from the KLMS algorithm, whose rate of convergence is typically an order of magnitude slower than the KRLS algorithm (LIU *et al.*, 2011) and, consequently, than the ROB-KRLS and KRMC algorithms.

We also can see in Fig. (30) that the KRLS and ROB-KRLS models presented basically the same rate of decay, a property which was expected since the ROB-KRLS model uses the KRLS for the sake of online learning. Despite this fact, the proposed approach ended up with lower MSE value compared to the other robust models for almost all datasets, except with the Silverbox dataset with 5% of outliers in Fig. 30(h), wherein the KRMC model achieved a lower MSE value among the evaluated algorithms.

However, it should be pointed out that the learning curves shown in Fig. 30(g) and 30(h) are to be analyzed together with the results shown in Table 15. Indeed, for the scenario with 5% of outliers, the KRMC model converged to an RMSE value lower than those achieved by the KRLS and the ROB-KRLS models; but, this only occurred because the KRMC ended with approximately 7 times more SVs than the KRLS and ROB-KRLS models (as shown in the last column of Table 15). For the testing phase, the ROB-KRLS model achieved an RMSE value lower than that achieved by the KRMC model, indicating that the former is more sparse and less sensitive to outliers than the latter.

Next section presents the second proposed approach in this chapter, which is based on the standard LSSVR optimization problem and is also derived from the KRLS model.

### 6.3 Proposed Online LSSVR Model

It is already known that the kernel-based methods constitute an appealing approach to the nonlinear regression problem of the form  $f(\mathbf{x}_i) = \boldsymbol{\phi}^\top(\mathbf{x}_i)\mathbf{w}$ , since their nonparametric solutions are typically given as (SCHÖLKOPF; SMOLA, 2002; VAPNIK, 1998)

$$f(\mathbf{x}) = \sum_{n=1}^N \alpha_n k(\mathbf{x}, \mathbf{x}_n), \quad (6.34)$$

where  $\alpha_n \in \mathbb{R}$  is the coefficient for the training sample  $\mathbf{x}_n$ . For simplicity and without loss of generality, the bias is absent in Eq. (6.34).

The Representer Theorem (KIMELDORF; WAHBA, 1971) assures that the solution in Eq. (6.34) may be expressed solely in terms of the kernel function over the training set

$\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$ . Furthermore, if the kernel is a *Mercer kernel*, then for any finite set of samples  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ , the matrix whose entries are  $\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$  is positive definite (VAPNIK, 1998).

As discussed in detail in Chapter 2, the parameter estimation problem for the LSSVR model is related to the cost function

$$J(\mathbf{w}) = \sum_{n=1}^N (f(\mathbf{x}_n) - y_n)^2 + \gamma \|\mathbf{w}\|^2, \quad (6.35)$$

where  $\gamma > 0$  is the regularization parameter. From the optimality conditions of the *Lagrangian* applied to the functional in Eq. (6.35), the parameter vector  $\mathbf{w}$  is determined as

$$\mathbf{w} = \sum_{n=1}^N \alpha_n \boldsymbol{\phi}(\mathbf{x}_n) = \boldsymbol{\Phi} \boldsymbol{\alpha}, \quad (6.36)$$

where  $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1), \dots, \boldsymbol{\phi}(\mathbf{x}_N)] \in \mathbb{R}^{d_h \times N}$  and  $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_N)^\top$  is the vector of Lagrange multipliers. Then, using  $f(\mathbf{x}_i) = \boldsymbol{\phi}^\top(\mathbf{x}_i) \mathbf{w}$  and Eq. (6.36), we can rewrite the functional in Eq. (6.35) as

$$J(\boldsymbol{\alpha}) = \|\mathbf{K} \boldsymbol{\alpha} - \mathbf{y}\|^2 + \gamma \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}, \quad (6.37)$$

where  $\mathbf{K} = \boldsymbol{\Phi}^\top \boldsymbol{\Phi}$  and  $\mathbf{y} = (y_1, \dots, y_N)^\top$ . Therefore, the optimal  $\boldsymbol{\alpha}$  can be estimated by finding the gradient vector  $\nabla_{\boldsymbol{\alpha}} J(\boldsymbol{\alpha}) = \partial J(\boldsymbol{\alpha}) / \partial \boldsymbol{\alpha}$  and equaling it to zero. Then, the vector  $\boldsymbol{\alpha}$  can be computed by solving the following linear system:

$$(\mathbf{K} + \gamma \mathbf{I}) \boldsymbol{\alpha} = \mathbf{y}, \quad (6.38)$$

whose ordinary solution is given by

$$\boldsymbol{\alpha} = (\mathbf{K} + \gamma \mathbf{I})^{-1} \mathbf{y}. \quad (6.39)$$

One should remember that the computational complexity of the LSSVR model, in solving Eq. (6.38), is  $\mathcal{O}(N^3)$ . The overbearing computational burden stems from the fact that every training sample will contribute one parameter to the resulting model.

In this context, we decide to apply the online and sparse strategy of the KRLS model to solve iteratively the standard dual LSSVR optimization problem in Eq. (6.37).

### 6.3.1 The OS-LSSVR Model

From now on, our goal is to maintain the original formulation of the LSSVR model in dual space and compute the vector of Lagrange multipliers of Eq. (6.39) in an online fashion

(i.e. pattern-by-pattern). In other words, we do not want to store all the training data, then compute the kernel matrix and finally solve the system in Eq. (6.38). In this regard, our main contribution is to use the sparsification method behind the KRLS model to develop an online version of the LSSVR model, called Online Sparse LSSVR (OS-LSSVR) model, that could learn from data incrementally. Since the LSSVR theory is developed from the beginning based on the regularized loss function in Eq. (6.35), the resulting online and sparse LSSVR model is also regularized.

We begin the development of an online sparse variant of the LSSVR problem by casting its original loss function in Eq. (6.37) into a sequential learning scenario. Thus, at the time step  $t$ , we get

$$J(\boldsymbol{\alpha}_t) = \|\mathbf{K}_t \boldsymbol{\alpha}_t - \mathbf{y}_t\|^2 + \gamma \boldsymbol{\alpha}_t^\top \mathbf{K}_t \boldsymbol{\alpha}_t, \quad (6.40)$$

which can also be written as

$$J(\boldsymbol{\alpha}_t) = \|\Phi_t^\top \Phi_t \boldsymbol{\alpha}_t - \mathbf{y}_t\|^2 + \gamma \boldsymbol{\alpha}_t^\top \Phi_t^\top \Phi_t \boldsymbol{\alpha}_t. \quad (6.41)$$

From Engel *et al.* (2004), we get the following equalities:

$$\mathbf{w}_t = \Phi_t \boldsymbol{\alpha}_t \approx \tilde{\Phi}_t \mathbf{A}_t^\top \boldsymbol{\alpha}_t = \tilde{\Phi}_t \tilde{\boldsymbol{\alpha}}_t, \quad (6.42)$$

where  $\mathbf{A}_t = [\mathbf{a}_1 \ \mathbf{a}_2 \ \cdots \ \mathbf{a}_t]^\top \in \mathbb{R}^{t \times m_t}$ . Then, Eq. (6.41) becomes

$$\begin{aligned} J(\tilde{\boldsymbol{\alpha}}_t) &= \|\Phi_t^\top \tilde{\Phi}_t \tilde{\boldsymbol{\alpha}}_t - \mathbf{y}_t\|^2 + \gamma \boldsymbol{\alpha}_t^\top \Phi_t^\top \tilde{\Phi}_t \tilde{\boldsymbol{\alpha}}_t, \\ &= \|\mathbf{A}_t \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t - \mathbf{y}_t\|^2 + \gamma \tilde{\boldsymbol{\alpha}}_t^\top \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t, \end{aligned} \quad (6.43)$$

where  $\tilde{\boldsymbol{\alpha}}_t \in \mathbb{R}^{m_t}$  is the reduced vector of  $m_t$  coefficients. Then, the minimization of Eq. (6.43) yields the following solution:

$$\begin{aligned} \tilde{\boldsymbol{\alpha}}_t &= (\tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{A}_t \tilde{\mathbf{K}}_t + \gamma \tilde{\mathbf{K}}_t)^{-1} \tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{y}_t, \\ &= [\tilde{\mathbf{K}}_t (\mathbf{A}_t^\top \mathbf{A}_t + \gamma \tilde{\mathbf{K}}_t^{-1})]^{-1} \mathbf{A}_t^\top \mathbf{y}_t, \\ &= \tilde{\mathbf{K}}_t^{-1} (\mathbf{A}_t^\top \mathbf{A}_t + \gamma \tilde{\mathbf{K}}_t^{-1})^{-1} \mathbf{A}_t^\top \mathbf{y}_t. \end{aligned} \quad (6.44)$$

By defining a matrix  $\mathbf{P}_t$  as

$$\mathbf{P}_t = (\mathbf{A}_t^\top \mathbf{A}_t + \gamma \tilde{\mathbf{K}}_t^{-1})^{-1}, \quad (6.45)$$

we finally get

$$\tilde{\boldsymbol{\alpha}}_t = \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t \mathbf{A}_t^\top \mathbf{y}_t. \quad (6.46)$$

The next step requires the computation of the inverse matrices in Eqs. (6.45) and (6.46) iteratively using the RLS algorithm. For this purpose, we have two possible situations following the same recursive procedure of the KRLS algorithm, which are described below. In addition, the general formulation of the OS-LSSVR model, including all the mathematical derivations, is presented in Appendix B.

### 6.3.1.1 Case 1 - Unchanged Dictionary

In this case,  $\delta_t \leq \nu$ , meaning that  $\phi(\mathbf{x}_t)$  is approximately linearly dependent on the dictionary vectors. Hence,  $\mathbf{x}_t$  is not added to the dictionary and the kernel matrix is not changed. Mathematically,  $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv}$  and  $\tilde{\mathbf{K}}_t = \tilde{\mathbf{K}}_{t-1}$ .

Since  $\mathbf{a}_t$  needs to be computed by Eq. (6.6) to determine  $\delta_t$ , the matrix  $\mathbf{A}_t$  is built iteratively by the inclusion of  $\mathbf{a}_t$ , i.e.  $\mathbf{A}_t = [\mathbf{A}_{t-1}^\top \mathbf{a}_t]^\top$ . Thus, let us define the matrix  $\mathbf{B}_t$  as

$$\begin{aligned} \mathbf{B}_t &= \mathbf{A}_t^\top \mathbf{A}_t + \gamma \tilde{\mathbf{K}}_t^{-1} \\ &= \mathbf{A}_{t-1}^\top \mathbf{A}_{t-1} + \gamma \tilde{\mathbf{K}}_{t-1}^{-1} + \mathbf{a}_t \mathbf{a}_t^\top \\ &= \mathbf{B}_{t-1} + \mathbf{a}_t \mathbf{a}_t^\top, \end{aligned} \quad (6.47)$$

where  $\mathbf{A}_t^\top \mathbf{A}_t = \mathbf{A}_{t-1}^\top \mathbf{A}_{t-1} + \mathbf{a}_t \mathbf{a}_t^\top$ . Again, we can use the Lemma 3.1 to recursively compute the matrix  $\mathbf{P}_t$  as

$$\mathbf{P}_t = \mathbf{B}_t^{-1} = \mathbf{P}_{t-1} - \frac{\mathbf{P}_{t-1} \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1}}{1 + \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t}. \quad (6.48)$$

The gain vector  $\mathbf{q}_t$  is defined as

$$\mathbf{q}_t = \frac{\mathbf{P}_{t-1} \mathbf{a}_t}{1 + \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t}, \quad (6.49)$$

and consequently

$$\mathbf{P}_t = \mathbf{P}_{t-1} - \mathbf{q}_t \mathbf{a}_t^\top \mathbf{P}_{t-1}. \quad (6.50)$$

Finally, using the fact that  $\mathbf{A}_t^\top \mathbf{y}_t = \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} + \mathbf{a}_t \mathbf{y}_t$ , the OS-LSSVR update rule for  $\tilde{\boldsymbol{\alpha}}_t$  can be written by

$$\begin{aligned} \tilde{\boldsymbol{\alpha}}_t &= \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t \mathbf{A}_t^\top \mathbf{y}_t, \\ &= \tilde{\mathbf{K}}_t^{-1} (\mathbf{P}_{t-1} - \mathbf{q}_t \mathbf{a}_t^\top \mathbf{P}_{t-1}) (\mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} + \mathbf{a}_t \mathbf{y}_t), \\ &= \tilde{\boldsymbol{\alpha}}_{t-1} + \tilde{\mathbf{K}}_t^{-1} (\mathbf{P}_t \mathbf{a}_t \mathbf{y}_t - \mathbf{q}_t \mathbf{a}_t^\top \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_{t-1}), \\ &= \tilde{\boldsymbol{\alpha}}_{t-1} + \tilde{\mathbf{K}}_t^{-1} (\mathbf{q}_t \mathbf{y}_t - \mathbf{q}_t \mathbf{a}_t^\top \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_{t-1}), \\ &= \tilde{\boldsymbol{\alpha}}_{t-1} + \tilde{\mathbf{K}}_t^{-1} \mathbf{q}_t (\mathbf{y}_t - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}), \end{aligned} \quad (6.51)$$

where the last equalities are based on  $\mathbf{q}_t = \mathbf{P}_t \mathbf{a}_t$  and  $\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) = \tilde{\mathbf{K}}_t \mathbf{a}_t$  (ENGEL *et al.*, 2004). The complete development to obtain the expression in Eq. (6.51) is presented in Eq. (B.20) of Appendix B.

One should note that, if we set  $\gamma = 0$  in Eq. (6.45), the proposed approach reduces to the KRLS model. In this sense, the OS-LSSVR proposal can be thought of as being a generalized (and regularized) version of the KRLS model.

### 6.3.1.2 Case 2 - Updating the Dictionary

In this case, one gets  $\delta_t > \nu$ , implying that  $\mathbf{x}_t$  must be added to the dictionary, i.e.  $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \cup \{\mathbf{x}_t\}$  and  $m_t = m_{t-1} + 1$ . Hence, the kernel matrix must be updated accordingly.

In order to compute  $\tilde{\mathbf{K}}_t$  (and hence  $\tilde{\mathbf{K}}_t^{-1}$ ) recursively using  $\tilde{\mathbf{K}}_{t-1}$  and the information provided by the new sample, we get

$$\tilde{\mathbf{K}}_t = \begin{bmatrix} \tilde{\mathbf{K}}_{t-1} & \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \\ \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top & k_{tt} \end{bmatrix} \quad (6.52)$$

and

$$\tilde{\mathbf{K}}_t^{-1} = \frac{1}{\delta_t} \begin{bmatrix} \delta_t \tilde{\mathbf{K}}_{t-1}^{-1} + \mathbf{a}_t \mathbf{a}_t^\top & -\mathbf{a}_t \\ -\mathbf{a}_t^\top & 1 \end{bmatrix}. \quad (6.53)$$

One should recall that for Case 2, while the dimension of the kernel matrix  $\tilde{\mathbf{K}}_t$  increases due to the inclusion of sample  $\mathbf{x}_t$  into the dictionary, the dimension of matrix  $\mathbf{A}_t$  does not. However, as required in Eq. (6.45), we need to compute the matrix  $\mathbf{A}_t^\top \mathbf{A}_t$  in order to determine  $\mathbf{P}_t$ . To avoid a mismatch in the sum  $\mathbf{A}_t^\top \mathbf{A}_t + \gamma \tilde{\mathbf{K}}_t^{-1}$ , we deliberately increase the dimension of matrix  $\mathbf{A}_t^\top \mathbf{A}_t$  as

$$\mathbf{A}_t^\top \mathbf{A}_t = \begin{bmatrix} \mathbf{A}_{t-1}^\top \mathbf{A}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}. \quad (6.54)$$

Then, we use Eqs. (6.53) and (6.54) to write down the following expression for matrix  $\mathbf{P}_t$ :

$$\mathbf{P}_t = \begin{bmatrix} \mathbf{A}_{t-1}^\top \mathbf{A}_{t-1} + \gamma \tilde{\mathbf{K}}_{t-1}^{-1} + \frac{\gamma}{\delta_t} \mathbf{a}_t \mathbf{a}_t^\top & -\frac{\gamma}{\delta_t} \mathbf{a}_t \\ -\frac{\gamma}{\delta_t} \mathbf{a}_t^\top & \frac{\gamma}{\delta_t} \end{bmatrix}^{-1}. \quad (6.55)$$

The next step is applying the same recursive procedure used in Eq. (6.53), for the calculation of  $\tilde{\mathbf{K}}_t^{-1}$ , to compute  $\mathbf{P}_t$  as

$$\mathbf{P}_t = \begin{bmatrix} \mathbf{P}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{1}{\Delta_b} \begin{bmatrix} -\mathbf{P}_{t-1} \mathbf{b} \\ 1 \end{bmatrix} \cdot \begin{bmatrix} -\mathbf{P}_{t-1} \mathbf{b} \\ 1 \end{bmatrix}^\top, \quad (6.56)$$

where  $\Delta_b = b^* - \mathbf{b}^\top \mathbf{B}_{t-1}^{-1} \mathbf{b}$ ,  $b^* = 1 + \frac{\gamma}{\delta_t}$  and  $\mathbf{b} = -\frac{\gamma}{\delta_t} \mathbf{a}_t$ . Defining a constant  $c = \gamma/\delta_t$ , we get

$$\Delta_b = 1 + c - c^2 \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t, \quad (6.57)$$

and the matrix  $\mathbf{P}_t$  can be rewritten as

$$\mathbf{P}_t = \frac{1}{\Delta_b} \begin{bmatrix} \Delta_b \mathbf{P}_{t-1} + c^2 \mathbf{P}_{t-1} \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1} & c \mathbf{P}_{t-1} \mathbf{a}_t \\ c \mathbf{a}_t^\top \mathbf{P}_{t-1} & 1 \end{bmatrix} \quad (6.58)$$

Finally, we use Eq. (6.46) to estimate  $\tilde{\boldsymbol{\alpha}}_t$  as

$$\tilde{\boldsymbol{\alpha}}_t = \tilde{\mathbf{K}}_t^{-1} \frac{1}{\Delta_b} \begin{bmatrix} \Delta_b \mathbf{P}_{t-1} + c^2 \mathbf{P}_{t-1} \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1} & c \mathbf{P}_{t-1} \mathbf{a}_t \\ c \mathbf{a}_t^\top \mathbf{P}_{t-1} & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} \\ y_t \end{bmatrix}. \quad (6.59)$$

After some simplifications in Eq. (6.59), we obtain

$$\tilde{\boldsymbol{\alpha}}_t = \frac{1}{\Delta_b \delta_t} \begin{bmatrix} \tilde{\boldsymbol{\alpha}}_t^{(1)} \\ \tilde{\alpha}_t^* \end{bmatrix}, \quad (6.60)$$

so that

$$\begin{aligned} \tilde{\boldsymbol{\alpha}}_t^{(1)} &= \Delta_b \delta_t \tilde{\boldsymbol{\alpha}}_{t-1} - \mathbf{a}_t (y_t - (\Delta_b - c) \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}) + \\ &\quad (y_t + c \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}) (\gamma \tilde{\mathbf{K}}_{t-1}^{-1} + c \mathbf{a}_t \mathbf{a}_t^\top) \mathbf{P}_{t-1} \mathbf{a}_t, \end{aligned} \quad (6.61)$$

and

$$\tilde{\alpha}_t^* = y_t - (\Delta_b - c) \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} - c \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t (y_t + c \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}). \quad (6.62)$$

The whole development to obtain Eqs. (6.61) and (6.62) is presented in Eqs. (B.35) and (B.36) of Appendix B, respectively.

Again, if we set  $\gamma = 0$ , we get  $c = 0$  and  $\Delta_b = 1$ . Then, the expressions in Eqs. (6.61) and (6.62) reduce to the ones reported by Engel *et al.* (2004) for the KRLS algorithm. Thus, based on the estimation equations developed for Case 1 ( $\delta_t \leq \nu$ ) and Case 2 ( $\delta_t > \nu$ ), we can state that the KRLS model is a particular case of the OS-LSSVR proposed approach. The OS-LSSVR pseudo-code is summarized in Algorithm 14.

Before proceeding to the computer experiments, we believe that some remarks are necessary in order to clarify the rationale behind the choices that eventually led to the proposal of the OS-LSSVR model.

**Remark 1** - In the KRLS model (and in the OS-LSSVR by extension), the sparsity is achieved “on the fly” or by *construction* (a term borrowed from Engel *et al.* (2004)), instead of by *pruning*

---

**Algorithm 14:** - Pseudo-code for the OS-LSSVR model.

---

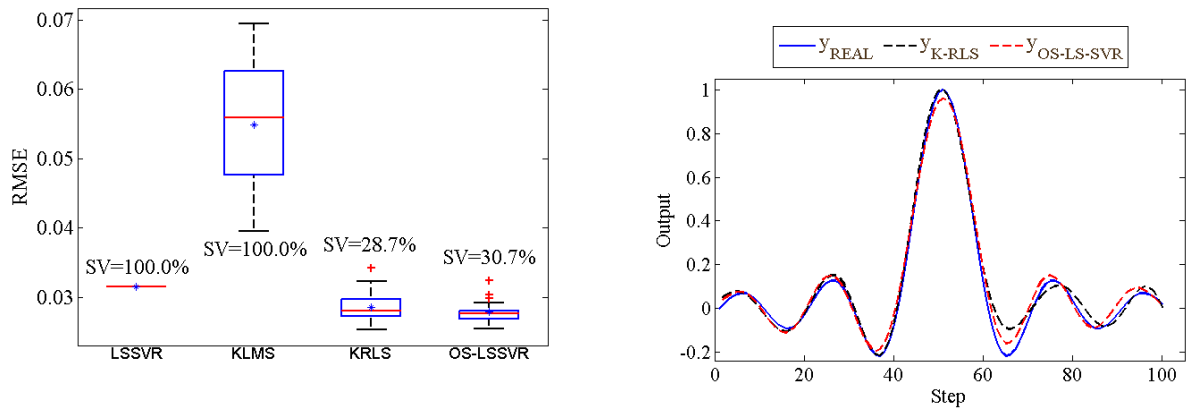
**Require:**  $\nu, \gamma, \sigma$  (for Gaussian kernel);  
**Set:**  $\tilde{\mathbf{K}}_1 = k_{11}; \tilde{\mathbf{K}}_1^{-1} = 1/\tilde{\mathbf{K}}_1; \mathbf{P}_1 = 1/(1 + \gamma\tilde{\mathbf{K}}_1^{-1}); \tilde{\mathbf{a}}_1 = y_1\mathbf{P}_1/\tilde{\mathbf{K}}_1; \mathbf{A}_1 = 1; m_1 = 1;$   
**for**  $t = 2 : N$ , **do**  
  Get new sample  $(\mathbf{x}_t, y_t)$  and compute  $\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t);$   
   $\mathbf{a}_t = \tilde{\mathbf{K}}_{t-1}^{-1}\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t);$   
   $\delta_t = k_{tt} - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \mathbf{a}_t;$   
  **if**  $\delta_t > \nu$  **then**  
     $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \cup \{\mathbf{x}_t\};$  % add  $\mathbf{x}_t$  to the dictionary  
    Compute  $\tilde{\mathbf{a}}_t$  from Eqs. (6.60), (6.61) and (6.62);  
    Compute  $\tilde{\mathbf{K}}_t$  and  $\tilde{\mathbf{K}}_t^{-1}$  from Eqs. (6.52) and (6.53), respectively;  
    Compute  $\mathbf{P}_t$  from Eq. (6.58);  
     $m_t = m_{t-1} + 1;$   
  **else**  
     $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv};$  % unchanged dictionary  
    Compute  $\mathbf{q}_t$  from Eq. (6.49);  
    Compute  $\mathbf{P}_t$  from Eq. (6.50);  
    Compute  $\tilde{\mathbf{a}}_t$  from Eq. (6.51);  
  **end if**  
**end for**  
Output  $\tilde{\mathbf{a}}_t, \mathcal{D}_t^{sv}.$

---

or by using a *penalty term* in the loss function. In fact, the use of a sparsity penalty term, such as the  $L_1$ -norm (PELCKMANS *et al.*, 2005) or  $L_0$ -norm (LÁZARO *et al.*, 2011; HUANG *et al.*, 2010) in the loss function could solve the lack of sparsity issue of the original LSSVR by inducing a sparse LSSVR model. However, this approach does not solve our “online learning” requirement, since the system in Eq. (6.38) would still be solved in batch mode.

**Remark 2** - Regarding the LSSVR model, Jung and Polani (2006) introduced an approach which is similar to the KRLS model, but differs from it in two important ways. First, their approach is supervised, since the selection of the relevant basis functions takes into account the error incurred from approximating the kernel as well as the reduction of the cost in the original learning task. Second, previously admitted SVs can also be deleted (i.e. pruned) in order to have even tighter control over the degree of sparsity. On the other hand, the model building framework of our OS-LSSVR model is conceptually simpler, since it can be achieved if we simply incorporate the ALD strategy into standard LSSVR formulation.

**Remark 3** - The proposal of the OS-LSSVR model has been strongly motivated by applications in recursive system identification (CHEN, 2009). For this purpose, we sinergetically amalgamated the KRLS ability of generating online *AND* sparse solutions, with the regularized formulation



(a) RMSE values for the test samples.

(b) Predicted outputs.

Figure 31 – Obtained RMSE values and predicted outputs for the sinc function.

of the LSSVR dual problem. As we will show from the next section onwards, the OS-LSSVR model consistently outperforms the KRLS and other related models.

### 6.3.2 Computational Experiments

In this section, we report the results of comprehensive computer simulations, comparing the proposed OS-LSSVR model with the plain LSSVR and KRLS models. For the sake of completeness, we also report results on the performance of the KLMS model, which was discussed in Chapter 4. The experiments are performed in the following regression tasks: function approximation, time series prediction and system identification, including experiments with a large-scale dataset. Next, we give some important details about the configuration of experiments and the evaluated datasets.

1. Evaluated Datasets - For the experiments in nonlinear system identification, we use two synthetic datasets (Synthetic-1 and Synthetic-2) and the large-scale (and real-world) Silverbox dataset, which have already been described in Chapter 5. For the experiments in function approximation and in time series prediction scenarios, we use the sinc function and the chaotic laser dataset, respectively, which are described below.
2. Methodology - In all experiments we use 5-fold cross validation to choose the hyper-parameters  $\nu$  (for the ALD criterion),  $\gamma$  (the regularization parameter),  $\sigma$  (for Gaussian kernel) and  $\eta$  (learning step for the KLMS model), in the search for their optimal values. Numerical performances are evaluated using root mean square errors (RMSE) computed for test samples over 20 independent runs.

It should be noted that the training data samples are presented only once, i.e. for only



Table 16 – Important features of the evaluated datasets for the computational experiments.

Important Features of the Selected Datasets						
Dataset	Task	Prediction Scenario	Training ( $N$ )	Test ( $N'$ )	$\hat{L}_u$	$\hat{L}_y$
Sinc function	Function approximation	-	50	100	-	-
Chaotic laser series	Time series prediction	Free Simulation	1,000	100/500	-	50
Synthetic-1	System identification	Free Simulation	150	150	1	1
Synthetic-2	System identification	Free Simulation	300	100	1	1
Silverbox	System identification	Free Simulation	91,072	40,000	10	10

one training epoch, for all evaluated models. However, unlike the common sparsity of the KRLS and OS-LSSVR, the KLMS model creates a growing radial-basis function network with the arrival of each input pattern, using all of the training samples and the respective a priori errors to make new predictions.

### 6.3.2.1 Results and Discussion

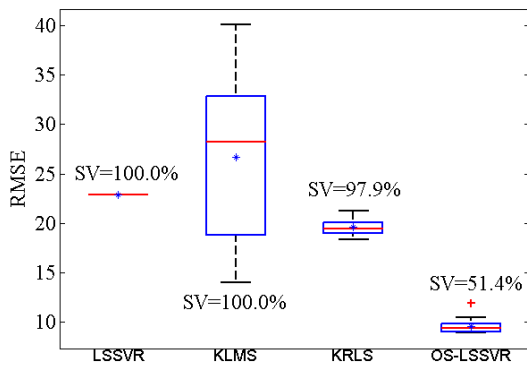
As mentioned before, the computer experiments involving the evaluated datasets correspond to their applications in function approximation, time series prediction and system identification tasks. Table 16 summarizes some important features of the computational experiments with each evaluated dataset.

#### 6.3.2.1.1 Function Approximation

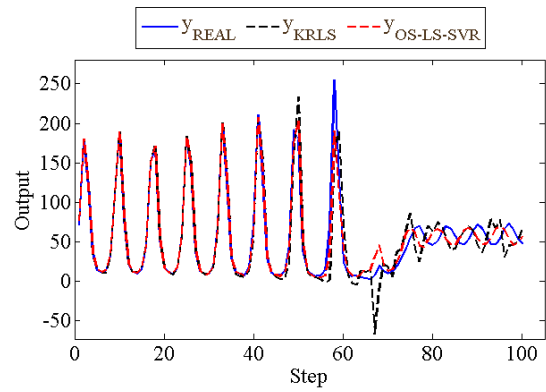
We first use the proposed OS-LSSVR model for learning the 1-dimensional sinc function  $y = \sin(\pi x)/\pi x$ , defined on the interval  $[-5, 5]$ . The training phase was performed on a random set of 50 samples, whose outputs were corrupted by additive Gaussian noise with zero-mean and a standard deviation of 0.05. The test was performed on an independent random set of 100 noise-free points.

The boxplots of the RMSE values for each model are shown in Fig. (31a), with the respective number of SVs relative to the total training set. One can observe that, despite providing a slightly higher number of SVs than the KRLS algorithm, the proposed OS-LSSVR model presented a lower average RMSE and a smaller dispersion. This result was somewhat expected since the OS-LSSVR model is regularized, while the KRLS is not. In this case, the KLMS algorithm presented the highest RMSE values among the evaluated models, even using all the training samples as support vectors.

One interesting fact is that the average RMSE values of the KRLS and OS-LSSVR models were much lower than the one achieved by the standard batch LSSVR, suggesting that

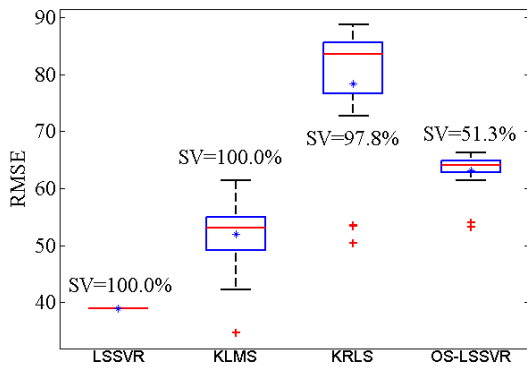


(a) RMSE values for test samples.

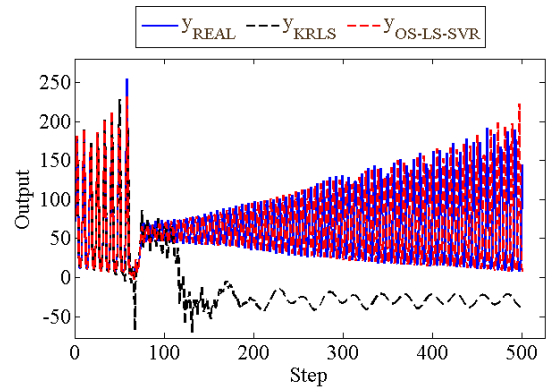


(b) Predicted outputs.

Figure 32 – Prediction of the laser time-series with 100 test samples.

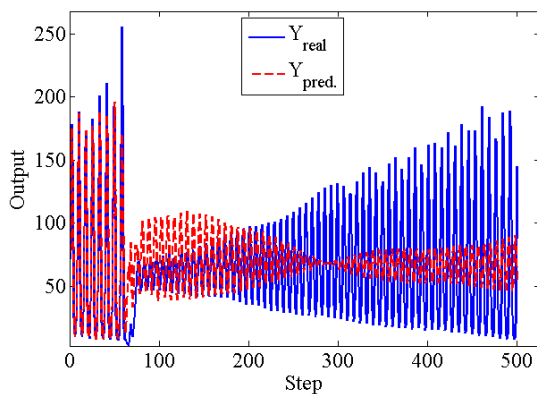


(a) RMSE values for the test samples.

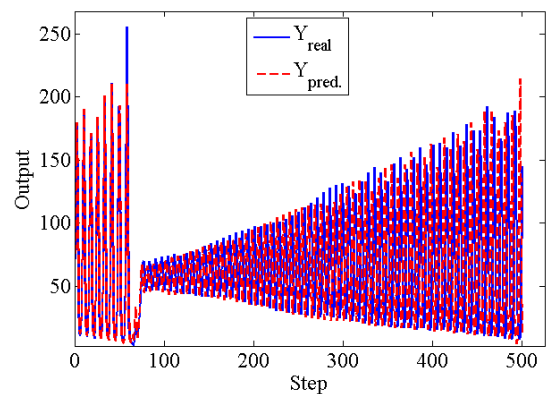


(b) Predicted outputs.

Figure 33 – Prediction of the laser time-series with 500 test samples.



(a) Standard LSSVR predicted output.



(b) OS-LSSVR predicted output.

Figure 34 – Prediction of the laser time-series with 500 test samples for LSSVR and OS-LSSVR models.

sparse online training is indeed a very good alternative to standard batch training of the LSSVR model.

The predicted outputs in the worst case, i.e. the predictions that obtained the highest

RMSE values over the 20 runs, for the sparse KRLS and OS-LSSVR models are illustrated in Fig. (31b). In general, the predictions of both models were close to each other until step 60, after which, they started to present visible differences in performance.

#### 6.3.2.1.2 Long-Term Time Series Prediction

Our second experiment involves a benchmarking time series prediction (TSP) task. For this purpose, we use the chaotic laser time series<sup>1</sup>(GERSHENFELD; WEIGEND, 1993) with 1,000 training samples and two different testing scenarios. In the first scenario, the models have to predict the next 100 samples; while for the second scenario they have to predict the next 500 samples. Moreover, we set the output order  $\hat{L}_y = 50$  for all the evaluated algorithms, as suggested by Suykens *et al.* (2002b).

In Fig. (32a) we report the RMSE values achieved by each model in the TSP task for the Scenario 1 (prediction with 100 test samples). We can see that the proposed OS-LSSVR model achieved significantly lower values of RMSE when compared to the KRLS, KLMS and standard LSSVR models. This was achieved using approximately half of the number of support vectors used by all the other models. The worst-case results obtained from the sparse models KRLS and OS-LSSVR for free simulation predictions are shown in Fig. (32b). One can observe that the OS-LSSVR model (in red) follows the actual time series (in blue) more closely than the KRLS algorithm (in black). The KRLS algorithm is also prone to produce higher peaks than the OS-LSSVR model.

In Fig. (33a) we report the RMSE values for the Scenario 2 (prediction with 500 test samples). Again, one can note that the OS-LSSVR model achieved significantly lower values and dispersion of RMSE compared to the KRLS model. One can also observe in Fig. (33b) the deterioration of the KRLS predicted outputs as the prediction horizon increases beyond step 100.

Based solely on the boxplots shown in Fig. (33a), one can erroneously be led to infer that the KLMS and, mainly, the standard LSSVR model achieved the best performance when compared to the proposed OS-LSSVR because they produced smaller RMSE values. It should be recalled, however, that the KLMS and standard LSSVR models use all training samples as support vectors, a feature that eventually led to overfitting. As we can see in Fig. (34a), the LSSVR predictions successfully followed the dynamic behavior of the real output until around step 70, when a steep collapse in laser intensity occurs. After that, the performance of the

<sup>1</sup>Available for download at [www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html](http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html)

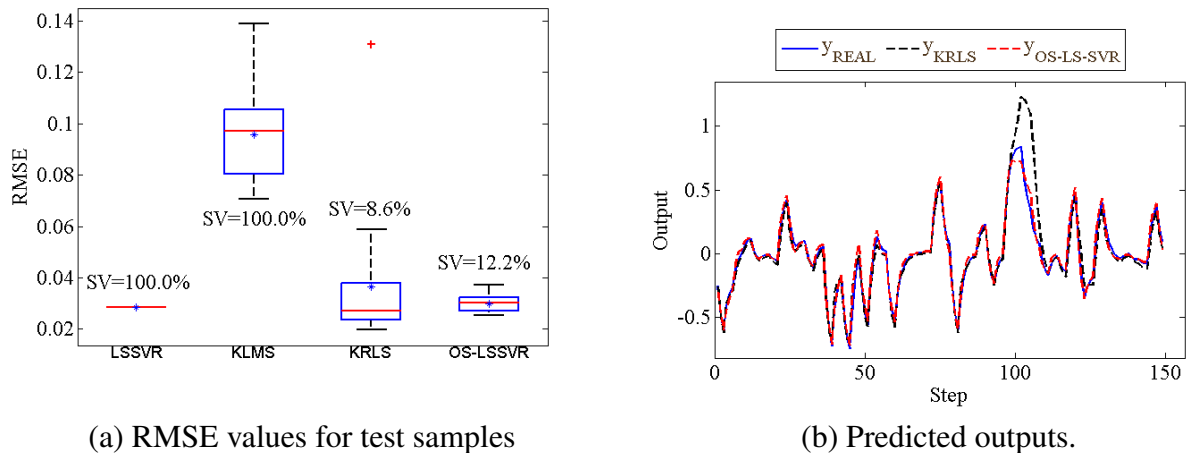


Figure 35 – Obtained RMSE values and predicted outputs for the Synthetic-1 dataset - system identification task.

standard batch LSSVR model becomes noticeably worse. In other words, despite the fact that the RMSE for the standard LSSVR has achieved the smallest values, its dynamical performance was very poor. This is a clear example of the difficulties in dealing with free-simulation in nonlinear prediction tasks, for which the predictor model should not be judged only by the RMSE performance. Although not shown in figure, we verified that, as in the case of the LSSVR model, the dynamical performance of the KLMS predictions was also very poor.

The OS-LSSVR performance is shown in Fig. (34b), from where it is possible to note a small time shift between the predicted and the actual outputs. This is a common effect in recursive chaotic time series prediction for long-term horizons. In such problems, the user may be interested not in predicting the next samples accurately, but rather in capturing the underlying chaotic dynamics of the system under investigation (HAYKIN; PRÍNCIPE, 1998; MENEZES-JÚNIOR; BARRETO, 2008). Since the dynamics of the system is chaotic, the predicted time series starts to diverge from the observed one at a certain point in the future onwards. This does not mean at all that the performance of the OS-LSSVR model is poor, as could be erroneously inferred from the RMSE values shown in Fig. (33a). As we report in Fig. (34b) the proposed model was able to track correctly the challenging chaotic dynamics of the laser system even in the worst case. This is a remarkable result, being comparable to those reported by MENEZES-JÚNIOR and Barreto (2008) for a two-hidden-layered recurrent neural network.

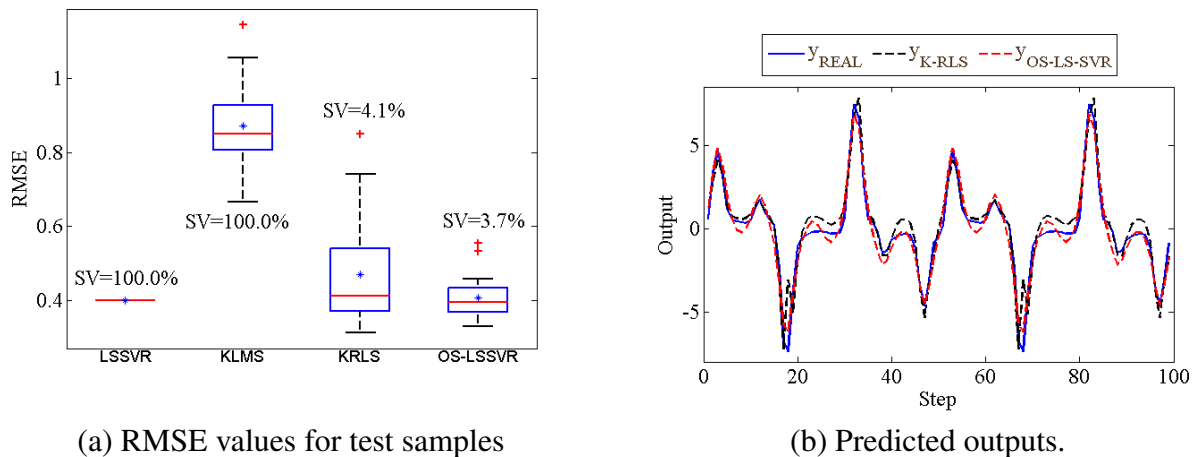


Figure 36 – Obtained RMSE values and predicted outputs for the Synthetic-2 dataset - system identification task.

### 6.3.2.1.3 Free-Simulation System Identification

In this experiment, we assess the performance of the proposed OS-LSSVR model in nonlinear system identification with Synthetic-1 and Synthetic-2 datasets. The RMSE values produced by each evaluated model for the Synthetic-1 dataset are shown in Fig. (35a), while the worst-case results for KRLS and OS-LSSVR models are shown in Fig. (35b). It can be seen in Fig. (35a) that the KRLS model ended up using less SVs than the OS-LSSVR at the expense of a higher average RMSE value. In this sense, when compared to the KRLS model, the OS-LSSVR model achieved a better tradeoff between the number of SVs and accuracy. In addition, the performance of the OS-LSSVR model was similar to the performance of the batch standard LSSVR.

Clearly, another notable aspect to be highlighted was, again, the considerable low dispersion of RMSE values produced by the OS-LSSVR model as compared to the ones obtained by the KLMS and KRLS models. This can be explained by the regularized nature of the OS-LSSVR, inherited from the standard LSSVR formulation. As we can infer from Fig. (35b), the KRLS and OS-LSSVR models were able to capture well the dynamic behavior of the system, except around step 100, where the predictions of the KRLS model are more distorted.

For the obtained results with Synthetic-2 dataset, an analysis of the boxplots in Fig. (36a) reveals that the RMSE distribution of the OS-LSSVR model achieved the lowest values as compared to the KLMS and KRLS models. Moreover, for the predictions, the OS-LSSVR model used only about 11 training instances (3.7% of training dataset) as support vectors. In Fig. (36b), we can see that, due to regularization, even the worst-case result for the OS-LSSVR

Table 17 – RMSE values for the test samples and number of SVs for each evaluated model - Silverbox dataset.

Models	Obtained results	
	RMSE	#SVs
KLMS	$7.10\text{E-}3 \pm 2.68\text{E-}4$	91,072
KRLS	$3.50\text{E-}3 \pm 1.71\text{E-}4$	302
OS-LSSVR	$2.00\text{E-}3 \pm 9.03\text{E-}5$	565

model is smoother than the one achieved by the KRLS model.

The OS-LSSVR model achieved an average value of RMSE comparable to the one obtained by the standard LSSVR, but the latter presented a much higher number of support vectors. An #SV=100% means that all training vectors were used as support vectors. For the Synthetic-2 dataset, this means that 298 data vectors were stored and used as support vectors.

#### 6.3.2.1.4 Performance on a Large-Scale Dataset

Since the OS-LSSVR model has inherited the sparsity property of the KRLS algorithm, we also decided to evaluate its performance in system identification on the Silverbox dataset. Thus, we report in Table 17 the obtained RMSE results (average value  $\pm$  standard deviation) and the number of SVs used by the KLMS, KRLS and OS-LSSVR models. The absence of results from the plain LSSVR model is explained by the practical impossibility of solving the linear system in Eq. (6.38) for a large-scale dataset using batch-mode estimation techniques.

It is possible to see in Table 17 that the KLMS model achieved RMSE values approximately twice as high as those achieved by the KRLS model, even using all the 91,072 training instances to make new predictions. Additionally, the proposed OS-LSSVR model presented significantly lower average RMSE value and lower dispersion when compared to the KRLS model, despite using a higher amount of support vectors (#SV=565) in comparison to the KRLS model (#SV=301). The number of support vectors used by the OS-LSSVR model represents only about 0.62% of the total training set. This supports the notion that our proposed approach achieved a suitable trade-off between sparsity and generalization accuracy. In contrast, the KRLS model is sparser than the OS-LSSVR model, but the former is not as accurate as the latter.

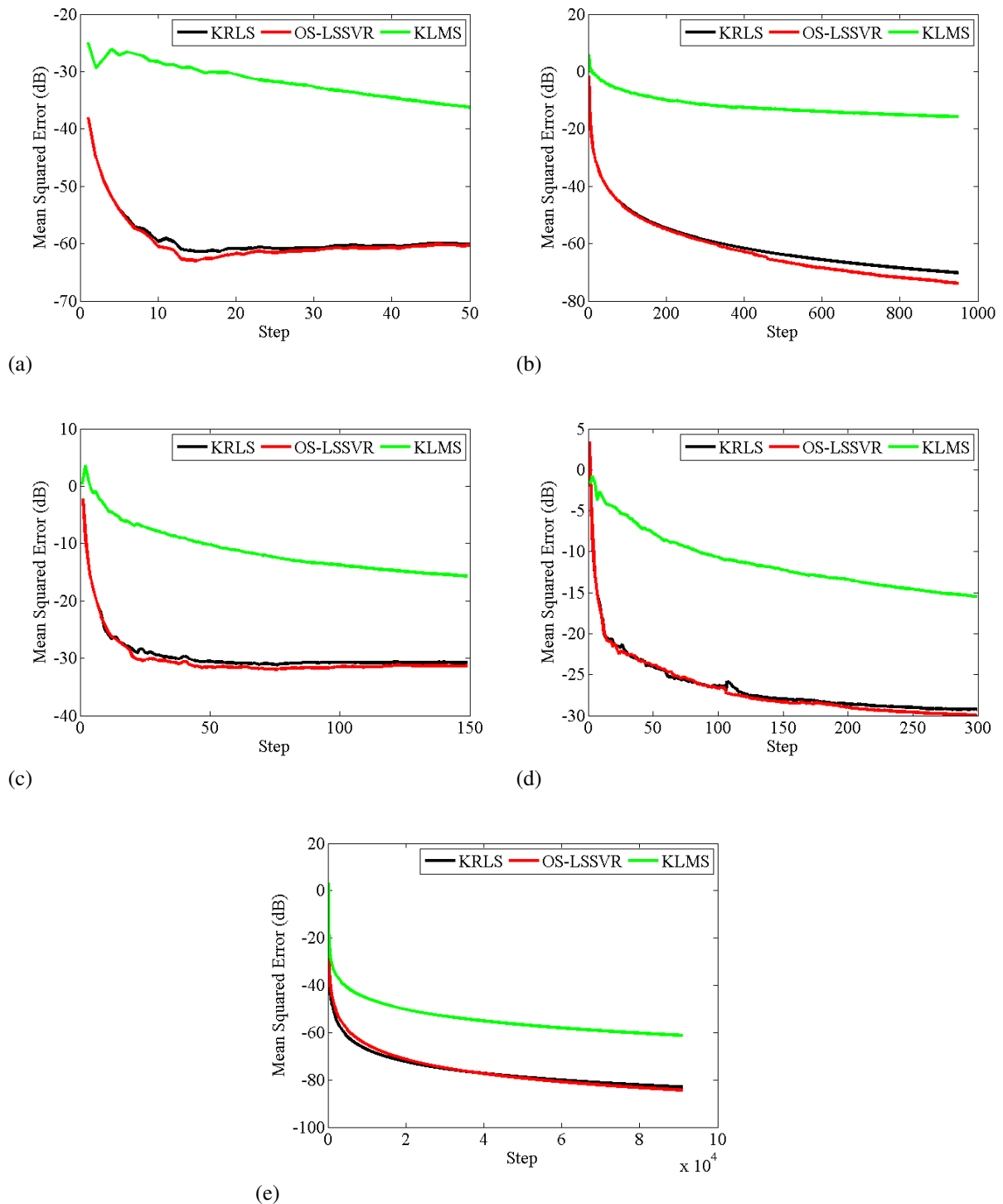


Figure 37 – Convergence curves for all the datasets: Fig. (37a) - Sinc function. Fig. (37b) - Laser time series. Fig. (37c) - Artificial 1. Fig. (37d) - Artificial 2. Fig. (37e) - Silverbox.

### 6.3.2.2 Convergence Analysis

A detailed theoretical analysis on the generalization error bounds of the OS-LSSVR models is presented in Appendix C. However, for the sake of completeness, we include empirical studies on its convergence. In order to empirically analyze the convergence behavior of the

Table 18 – AIC and BIC information criteria for all the evaluated models.

Dataset	AIC / BIC			
	LSSVR	KRLS	OS-LSSVR	KLMS
Sinc function	1.1E+2 / 2.0E+2	3.3E+1 / 6.2E+1	<b>3.2E+1 / 6.1E+1</b>	1.0E+2 / 1.9E+2
Laser time series	1.9E+3 / 6.5E+3	1.8E+3 / 6.4E+3	<b>9.7E+2 / 3.3E+3</b>	1.9E+3 / 6.5E+3
Synthetic-1	3.0E+2 / 7.5E+2	<b>2.3E+1 / 6.2E+1</b>	3.3E+1 / 8.7E+1	2.9E+2 / 7.4E+2
Synthetic-2	6.0E+2 / 1.7E+3	2.0E+1 / 6.4E+1	<b>1.8E+1 / 5.9E+1</b>	5.9E+2 / 1.7E+3
Silverbox	- / -	<b>5.9E+2 / 3.4E+3</b>	1.1E+3 / 6.4E+3	8.0E+4 / 4.2E+5

KLMS, KRLS and OS-LSSVR models, we have included the ensemble-average learning curves for each evaluated dataset in Fig. (37). In these curves, one plots the mean squared error (MSE) values for each evaluated model, during the training phase and over the 20 independent runs, versus the number of iterations (which exactly corresponds to the number of training samples).

One can note in Fig. (37) that the KLMS presented the poorest performance in terms of MSE comparing to the other models. Somehow, this was expected, since its rate of convergence is typically an order of magnitude slower than the KRLS algorithm (LIU *et al.*, 2011) and, consequently, than the OS-LSSVR algorithm.

For the learning curves in Fig. (37), the KRLS and OS-LSSVR models presented basically the same rate of decay, a property which was expected since the OS-LSSVR model uses the KRLS for the sake of online learning. Despite this fact, the proposed approach ended up with a slightly lower MSE value compared to the KRLS model for all datasets.

### 6.3.2.3 Model Efficiency

An adaptive filtering algorithm is considered efficient if it minimizes the usage of data while maximizing the quality of the solution, i.e. achieving parameter adjustments close to optimum (WIDROW; KAMENETSKY, 2003). The optimum in the linear case is usually provided by the well-known (non-adaptive) Wiener filter. However, minimizing the amount of used data and, at the same time, maximizing the quality of the solution are somewhat antagonistic. On the one hand, minimizing data usage corresponds to fast convergence. On the other hand, fast convergence could provide a poor quality of solution. This trade-off is present in all nonlinear learning systems. An analytical analysis of the efficiency for online kernel (i.e. nonlinear) models, mainly for the LMS- and RLS-based algorithms, is still an open issue in the signal processing community, with just a few initiatives available (see Constantin *et al.* (2005), for example).

In order to assess empirically the trade-off between the solution quality (given by



Table 19 – Average norm of the parameter vector  $\boldsymbol{\alpha}$  for the LSSVR model and  $\tilde{\boldsymbol{\alpha}}_t$  for the KRLS and OS-LSSVR models.

Dataset	$\ \boldsymbol{\alpha}\ , \ \tilde{\boldsymbol{\alpha}}_t\ $		
	LSSVR	KRLS	OS-LSSVR
Sinc function	5.1E0	2.3E+1	6.6E+1
Laser time series	3.3E+2	3.8E+4	8.6E+1
Synthetic-1	1.6E+2	1.2E+4	7.1E+2
Synthetic-2	3.6E+6	1.5E+4	1.2E+6
Silverbox	-	5.4E+2	2.4E+3

the sum of squared errors) and the number of parameters for each model, we decide to use as performance indices the well-known Akaike's information criterion (AIC) (AKAIKE, 1974), given by

$$AIC(n_p) = N \ln(\hat{\sigma}_e^2) + 2n_p, \quad (6.63)$$

and the Bayesian information criterion (BIC) (KASHYAP *et al.*, 1977; CRUTCHFIELD; MC-NAMARA, 1987):

$$BIC(n_p) = N \ln(\hat{\sigma}_e^2) + n_p \ln(N), \quad (6.64)$$

where  $\hat{\sigma}_e^2$  is the maximum likelihood estimate of the variance of the residuals,  $N$  is the number of data samples and  $n_p$  is the number of parameters of the model. The lower the AIC/BIC values, the better.

The rationale for this approach is that in the realm of the kernel-based models (such as the LSSVR and KRLS), the number of data samples used to build the model is equal to the dimension of the solution vector  $\boldsymbol{\alpha}_t$ . Since the LSSVR is a non-sparse model, the dimension of  $\boldsymbol{\alpha}_t$  is equal to the number of training data samples. For the KRLS and OS-LSSVR models, the dimension of  $\boldsymbol{\alpha}_t$  is the number of support vectors in the dictionary (i.e.  $n_p = m_t$ ). For the original (non-sparse) LSSVR model, we have  $n_p = N$ .

We report the values of AIC and BIC information criteria in Table 18, where it should be highlighted that all the models have the same order for each evaluated dataset. Therefore, we can see that the lowest AIC/BIC values were achieved by the KRLS and OS-LSSVR models for all datasets. This is an indication that these approaches presented a better compromise involving quality of solution and the number of used samples (and, hence parameters) compared to the standard LSSVR and KLMS. In addition, it is possible to note that the proposed approach achieved lower AIC/BIC values compared to the KRLS in most cases (Sinc function, Laser

time series and Artificial 2). Even for the Artificial 1 and Silverbox datasets, for which the KRLS achieved the lowest AIC/BIC values among all models, the OS-LSSVR presented better performance in terms of RMSE values with the test samples and convergence analysis, as already seen in Figs. (35a), (37c), (37e) and Table 17.

#### 6.3.2.4 A Note on the Norm of the Solution Vector

As a final experiment, we analyzed the influence of the regularization parameter  $\gamma$  in the model solutions. In Table 19 we report the average norm of the Lagrange multipliers vector  $\boldsymbol{\alpha}$  for the standard LSSVR and  $\tilde{\boldsymbol{\alpha}}_t$  for the KRLS and OS-LSSVR models, for all previously used datasets. It is possible to infer that, in general, the Lagrange multipliers vectors for LSSVR and OS-LSSVR have magnitude orders close to each other, which was expected since both models are regularized. However, for the KRLS model, the effects of its lack of regularization were more visible for the Synthetic-1 dataset and, in particular, for the Laser time series. Even for the Synthetic-2 and Silverbox datasets, wherein the regularized models obtained higher norms for  $\boldsymbol{\alpha}$  and  $\tilde{\boldsymbol{\alpha}}_t$ , their performances overcame the KRLS algorithm. Thus, we conclude by stating that the OS-LSSVR model is, indeed, a very efficient regularized online sparse variant of the LSSVR model.

## 6.4 Concluding Remarks

In this chapter, we presented the second part of the contributions of this thesis, which correspond to KAF proposals derived from the original KRLS model. As in the KRLS case, the proposed approaches, called the Robust KRLS (ROB-KRLS) and Online Sparse LSSVR (OS-LSSVR) models, apply a constructive sparsity procedure based on the ALD criterion in order to produce sparse solutions for nonlinear and online regression tasks.

First, we derived an outlier-robust version of the KRLS model, based on concepts from the  $M$ -estimation framework. The ROB-KRLS proposal produces solutions that minimize the effects of the presence of outliers in the estimation samples. To the best of our knowledge, the ROB-KRLS is the first online kernel-based model that includes, simultaneously, features of sparsity and robustness based on the  $M$ -estimators.

The ROB-KRLS model was successfully evaluated in nonlinear dynamical system identification tasks with artificial and real-world datasets (including a large-scale dataset) in

the presence of outliers. With the artificial datasets and using infinite-steps ahead prediction, the ROB-KRLS proposal presented better performance, in terms of RMSE values, than the original KRLS model and the robust KMC and KRMC approaches, in almost all scenarios. This superiority of our model was even more evident as the amount of outliers was increased. With the real-world datasets using 3-step ahead predictions (Actuator dataset) and free simulation (Silverbox dataset), again our approach achieved, in general, better performance among the evaluated models in all the scenarios, without and with outliers.

Still inspired by the KRLS model, we developed an online updating scheme for building regularized sparse LSSVR models, as in its standard dual formulation, based on the ALD criterion. The proposed framework of our OS-LSSVR model is comprehensive enough to encompass the KRLS algorithm as a non-regularized special case.

The OS-LSSVR model was successfully evaluated in tasks of increasing complexity, such as function approximation, time series prediction and nonlinear dynamical system identification, including tests with a large-scale dataset. Based on the results reported in this chapter, we can state that the proposed OS-LSSVR model consistently outperformed the state-of-the-art in KAF algorithms (KRLS and KLMS) in terms of accuracy on test samples, convergence, sparsity and statistical efficiency.

Next chapter will synthesize the main conclusions drawn throughout the development of this thesis.

## 7 CONCLUSIONS AND FUTURE DIRECTIONS

“I hadn’t been aware that there were doors closed to me  
until I started knocking on them.”

(Gertrude B. Elion)

In this chapter, we present a summarized overview of the contributions of this thesis, the respective conclusions and we point out some important topics for further research.

### 7.1 Conclusions and Discussion

In this thesis, we have investigated kernel-based methods to solve nonlinear regression problems, especially those from dynamical system identification tasks. As Machine Learning techniques, these kernel regression methods are universal approximators by nature, and their solid mathematical foundation guarantees that the resulting models come from convex optimization problems. Thus, the proposed approaches throughout this work are related through their foundation in kernel regression, inspired by the Vapnik’s support vectors theory.

Because of its simplicity, compared to the SVR quadratic programming problem, the LSSVR model was an important focus in kernel-based regression for the development of our proposals in this thesis. Since it originally works in the dual space formulation, its model solution only lies in solving a linear system of equations by using the OLS procedure (using Moore-Penrose inverse matrix) in batch mode. However, the training phase in batch mode, allied to the natural non-sparsity of its solution, can be a limiting factor in using the standard LSSVR for applications that require a large amount of data, for example. A feasible strategy to overcome this is solve the LSSVR optimization problem in the primal space with the aid of the Nyström method to find an approximation for the nonlinear map  $\phi$ . This resulting approach, which is called FS-LSSVR model and is discussed throughout this thesis, produces sparse solutions on a selected subsample of prototype vectors (PVs).

Despite the attractive features of the LSSVR model in the original dual space or in the primal space (FS-LSSVR), either by the computational simplicity or by the sparsity of the solution (only in the FS-LSSVR case), both formulations are based on SSE cost functions whose optimality is guaranteed only for normally distributed errors. Thus, their performances can be considerably compromised when the estimation data is corrupted with non-Gaussian noise or

outliers, which are very common in real-world datasets of nonlinear processing applications such as in dynamical system identification.

That being said, our **first** proposed approach in this thesis was a robust version of the standard LSSVR method, the **RLM-SVR** model, that combines the dual (and unweighted) LSSVR formulation with the linear robust RLM algorithm, which is based on the  $M$ -estimators and derived from the RLS algorithm. In essence, the RLM-SVR model solution is obtained in a recursive procedure where each sample, that is possibly contaminated with outliers, is individually treated at each iteration. As in the LSSVR case, the RLM-SVR model is non-sparse and, moreover, its robustness is achieved after a relatively large number of iterations.

A comprehensive performance evaluation of the RLM-SVR model has been carried out on synthetic and real-world datasets contaminated with outliers, in system identification scenarios. The performances were compared to the ones obtained by the standard LSSVR and two of its robust variants, namely the W-LSSVR and IR-LSSVR models. Despite showing dispersion in the obtained results, since it is sensitive to the initial conditions, the RLM-SVR model outperformed the other methods in almost all the scenarios with outlier contamination.

Our **second** and **third** contributions, referred to as the **RFS-LSSVR** and **R<sup>2</sup>FS-LSSVR** models, represent two robust models based on  $M$ -estimators and the weighted least squares algorithm, and are derived from the primal formulation of the FS-LSSVR model. Thus, these novel methodologies produce sparse solutions that minimize the effect of the presence of outliers in the estimation samples. The theoretical development of the RFS-LSSVR and R<sup>2</sup>FS-LSSVR proposals corresponds to equivalent robust strategies in the primal space of the W-LSSVR and IR-LSSVR models in the dual space, respectively. Notably, to the best of our knowledge, this was the first time that robust and sparse models were derived from the LSSVR formulation in the primal space.

The RFS-LSSVR and R<sup>2</sup>FS-LSSVR proposals were successfully evaluated in nonlinear dynamical system identification tasks in different scenarios: using benchmarking synthetic and real-world datasets (including a large-scale dataset), in SISO and MIMO systems and using 3-steps ahead prediction and free simulation. In general, the proposed models presented better performance in almost all scenarios, in terms of RMSE values and sparsity of the solution, than with the LSSVR, W-LSSVR, IR-LSSVR and FS-LSSVR models. The superior performances of the RFS-LSSVR and R<sup>2</sup>FS-LSSVR models were even more evident as the number of outliers was increased. Finally, to the best of our knowledge, this was the first time that a robust system

identification problem was treated using a MIMO system.

Despite their solid mathematical foundations and commendable achieved performances in a wide range of nonlinear regression applications, most kernel-based models (including those mentioned above), be them sparse, as with FS-LSSVR, or non-sparse as with LSSVR, have a learning process normally carried out in batch mode. However, there are several application domains, such as time series prediction, control, system identification and channel equalization, that require online learning strategies, where the model solution is updated with the arrival of each new input sample. Therefore, the field of KAF methods, which applies the kernelization strategy to linear adaptive filters as in the KLMS, KRLS and KAPA models, has aroused interest in the community of Machine Learning for solving nonlinear signal processing problems, such as the ones that demand big data applications or data nonstationarity.

Regarding the discussion above, we have also developed an outlier-robust version of the KRLS algorithm, called the **ROB-KRLS** model, which corresponds to the **fourth** contribution of this work. The proposed ROB-KRLS model is also derived from the  $M$ -estimators and, consequently, produces solutions that minimize the effects of the presence of outliers. Moreover, the ROB-KRLS model still preserves the sparsity and the computational complexity of the original KRLS model. The ROB-KRLS model is the first online kernel-based method that includes, simultaneously, features of sparsity and robustness based on  $M$ -estimation theory.

The performance of the ROB-KRLS model was evaluated in system identification tasks using artificial and real-world datasets, one of which was a large-scale dataset, contaminated with outliers. The computer experiments were carried out in scenarios of 3-step ahead prediction and free simulation. Based on the evaluated criteria of the achieved RMSE values, number of support vectors, correlation analysis of the residuals and empirical convergence behavior over the training phase, the ROB-KRLS model outperformed the original KRLS and two robust KAF approaches, namely the KMC and KRMC models.

As the **fifth** and last contribution in this work, we presented an online updating strategy to solve the LSSVR dual optimization problem. This proposed framework, called the **OS-LSSVR** model, incorporates the ALD criterion of the KRLS algorithm to achieve sparsity and online learning for the standard LSSVR model. One should note that our OS-LSSVR proposal is comprehensive enough to encompass the KRLS model as a special case.

The OS-LSSVR model was successfully evaluated in different tasks of nonlinear regression, such as function approximation, time series prediction and system identification,

Table 20 – Summary of the main characteristics of each proposed model.

Feature	Proposed Kernel-Based Models			
	RLM-SVR	RFS/R <sup>2</sup> FS-LSSVR	ROB-KRLS	OS-LSSVR
Primal/dual problem	dual	primal	dual	dual
Regularized problem	YES	YES	NO	YES
Batch/online learning	batch	batch	online	online
Sparsity solution	NO	YES	YES	YES
Robustness	YES	YES	YES	NO
Computational complexity	$\mathcal{O}(N^3)$	$\mathcal{O}(NM^2)$	$\mathcal{O}(m_t^2)$	$\mathcal{O}(m_t^2)$
Memory demand	$\mathcal{O}(N^2)$	$\mathcal{O}(NM)$	$\mathcal{O}(Nm_t)$	$\mathcal{O}(Nm_t)$

including tests with a big dataset. Based on results obtained in terms of RMSE values, number of support vectors, convergence analysis and statistical efficiency, the OS-LSSVR approach consistently outperformed the KLMS and KRLS kernel adaptive filtering models. Finally, Table 20 summarizes the main characteristics of each proposed approach in this thesis.

In what follows, we discuss the main topics that we wish to pursue from now on.

## 7.2 Future Directions

In this thesis, we have presented some novel proposals in kernel-based models for nonlinear regression. However, these contributions are merely a snapshot of work in a limited time. There are certainly additional leads and challenges that need further investigation. We sum up a few possibilities that could be undertaken.

An interesting aspect that deserves to be investigated is the application of the  $M$ -estimators framework, as done for the ROB-KRLS proposal, to the OS-LSSVR model. It may be a suitable trade-off between sparsity and generalization accuracy with the robustness of  $M$ -estimators against the effect of outliers.

For all the performed experiments in system identification in this thesis, we adopted the assumption that the system dynamics follow a NARX model. However, we intend to investigate the behavior of the proposed RLM-SVR, RFS-LSSVR, R<sup>2</sup>FS-LSSVR, ROB-KRLS and OS-LSSVR approaches in strategies based on different philosophies, as NARMAX (*nonlinear autoregressive moving average model with exogenous inputs*) model (BILLINGS, 2013), for instance.

Some kernel adaptive filtering methods can be formulated by using a general non-linear state model in the input space, i.e. they can implement a linear state model in RKHS. The Extended KRLS is a typical example in this line of work, whose model solution is guided by a kernel regressor using a Kalman filter with an explicit state transition process. Therefore, a prominent research topic is the extension of the ROB-KRLS and OS-LSSVR proposals in a framework of state space models, where they can be used for applications in time series prediction, dynamic positioning, tracking channel, etc.

Although our RFS-LSSVR and  $R^2$ FS-LSSVR proposed approaches present solutions that are sparsity and robust to outliers, they still need a batch mode procedure in their learning process. One important point to be investigated is the feasibility of applying an online learning procedure to the primal RFS-LSSVR and  $R^2$ FS-LSSVR models with the help of the strategy developed for the KRLS model, also based on the ALD sparsification criterion.

Finally, although all the proposed models (RLM-SVR, RFS-LSSVR,  $R^2$ FS-LSSVR, ROB-KRLS and OS-LSSVR) have been developed to solve regression problems, it would be possible to extend their formulations for applications in pattern classification. In principle, we believe that there is no impediment for our proposals to have satisfactory performances in online and/or robust classification tasks.



## BIBLIOGRAPHY

- AGUIRRE, L. A. **Introdução à Identificação de Sistemas Técnicas Lineares e Não-Lineares Aplicadas a Sistemas Reais**. 3th. ed. Belo Horizonte, Brazil: Editora UFMG, 2007.
- AKAIKE, H. A new look at the statistical model identification. **Automatic Control, IEEE Transactions on**, v. 19, n. 6, p. 716–723, 1974.
- ARCE, G. R. **Nonlinear signal processing: a statistical approach**. Hoboken, NJ, USA: John Wiley & Sons, 2005.
- BAKER, C. T. **The numerical treatment of integral equations**. Oxford, England: Clarendon Press, 1977.
- BAMNETT, V.; LEWIS, T. Outliers in statistical data. JSTOR, 1994.
- BAO, Y.; XIONG, T.; HU, Z. Multi-step-ahead time series prediction using multiple-output support vector regression. **Neurocomputing**, v. 129, p. 482–493, 2014.
- BARROS, A. L. B.; BARRETO, G. A. Building a robust extreme learning machine for classification in the presence of outliers. In: **Proceedings of the 8th International Conference on Hybrid Artificial Intelligence Systems (HAIS)**. Salamanca, Spain: Springer, 2013. p. 588–597.
- BARROS, A. L. B. P. **Revisitando o Problema de Classificação de Padrões na Presença de Outliers Usando Técnicas de Regressão Robusta**. PhD Thesis — Federal University of Ceará, Fortaleza, Ceará, Brazil, 2013.
- BASAK, D.; PAL, S.; PATRANABIS, D. C. Support vector regression. **Neural Information Processing-Letters and Reviews**, v. 11, n. 10, p. 203–224, 2007.
- BELLANGER, M. **Adaptive digital filters**. Boca Raton, Florida, USA: CRC Press, 2001.
- BEN-GAL, I. Outlier detection. In: **Data Mining and Knowledge Discovery Handbook**. New York, NY, USA: Springer, 2005. p. 131–146.
- BEN-ISRAEL, A.; GREVILLE, T. N. E. **Generalized inverses: theory and applications**. Berlin, Germany: Springer Science & Business Media, 2003.
- BILLINGS, S. A. **Nonlinear System Identification NARMAX Methods in the Time, Frequency and Spatio-Temporal Domains**. 1st. ed. Hoboken, NJ, USA: John Wiley & Sons, 2013.
- BILLINGS, S. A.; VOON, W. S. F. Correlation based model validity tests for non-linear models. **International Journal of Control**, v. 44, n. 1, p. 235–244, 1986.
- BISHOP, C. M. **Pattern Recognition and Machine Learning**. 3rd. ed. New York, NY, USA: Springer, 2006.
- BJÖRCK, A. **Numerical methods for least squares problems**. 1st. ed. Philadelphia, Pennsylvania, USA: SIAM, 1996.
- BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. In: **Proceedings of the 5th Annual Workshop on Computational Learning Theory**. Pittsburgh, PA, USA: ACM, 1992. p. 144–152.

- BURGES, C. J. C. A tutorial on support vector machines for pattern recognition. **Data Mining and Knowledge Discovery**, v. 2, n. 2, p. 121–167, 1998.
- BURGES, C. J. C.; SCHÖLKOPF, B. Improving the accuracy and speed of support vector machines. **Advances in Neural Information Processing Systems**, p. 375–381, 1997.
- CAI, Y.; WANG, H.; YE, X.; FAN, Q. A multiple-kernel LSSVR method for separable nonlinear system identification. **Journal of Control Theory and Applications**, v. 11, n. 4, p. 651–655, 2013.
- CASTRO, R.; MEHRKANOON, S.; MARCONATO, A.; SCHOUKENS, J.; SUYKENS, J. A. K. SVD truncation schemes for fixed-size kernel models. In: **Proceedings of the International Joint Conference on Neural Networks (IJCNN)**. Beijing, China: IEEE, 2014. p. 3922–3929.
- CHAPELLE, O. Training a support vector machine in the primal. **Neural Computation**, v. 19, n. 5, p. 1155–1178, 2007.
- CHAUDHARY, N. I.; RAJA, M. A. Z. Identification of Hammerstein nonlinear ARMAX systems using nonlinear adaptive algorithms. **Nonlinear Dynamics**, v. 79, n. 2, p. 1385–1397, 2015.
- CHEN, C.; YAN, C.; LI, Y. A robust weighted least squares support vector regression based on least trimmed squares. **Neurocomputing**, v. 168, p. 941–946, 2015.
- CHEN, H.-F. Recursive system identification. **Acta Mathematica Scientia**, v. 29, n. 3, p. 650–672, 2009.
- CHEN, K.; YU, J. Short-term wind speed prediction using an unscented kalman filter based state-space support vector regression approach. **Applied Energy**, v. 113, p. 690–705, 2014.
- CHEN, T.-T.; LEE, S.-J. A weighted LS-SVM based learning system for time series forecasting. **Information Sciences**, v. 299, p. 99–116, 2015.
- CHEN, X.; YANG, J.; LIANG, J.; YE, Q. Recursive robust least squares support vector regression based on maximum correntropy criterion. **Neurocomputing**, v. 97, p. 63–73, 2012.
- CHERKASSKY, V.; MULIER, F. M. **Learning from data: concepts, theory, and methods**. 2nd. ed. Hoboken, NJ, USA: John Wiley & Sons, 2007.
- CHOU, C. T.; MACIEJOWSKI, J. M. System identification using balanced parametrizations. **Automatic Control, IEEE Transactions on**, v. 42, n. 7, p. 956–974, 1997.
- CHRISTMANN, A.; STEINWART, I. Consistency and robustness of kernel-based regression in convex risk minimization. **Bernoulli**, p. 799–819, 2007.
- CHUANG, C.-C.; LEE, Z.-J. Hybrid robust support vector machines for regression with outliers. **Applied Soft Computing**, v. 11, n. 1, p. 64–72, 2011.
- CONSTANTIN, I.; RICHARD, C.; LENGELLE, R.; SOUFFLET, L. Regularized kernel-based Wiener filtering. application to magnetoencephalographic signals denoising. In: **Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)**. Philadelphia, PA, USA: IEEE, 2005. p. 289–292.

CRUTCHFIELD, J. P.; MCNAMARA, B. S. Equations of motion from a data series. **Complex Systems**, v. 1, n. 417-452, p. 121, 1987.

CUI, W.; YAN, X. Adaptive weighted least square support vector machine regression integrated with outlier detection and its application in QSAR. **Chemometrics and Intelligent Laboratory Systems**, v. 98, n. 2, p. 130–135, 2009.

DAVID, H. A. Early sample measures of variability. **Statistical Science**, v. 13, n. 4, p. 368–377, 1998.

DE-BRABANTER, K. **Least squares support vector regression with applications to large-scale data: a statistical approach**. PhD Thesis — Faculty of Engineering, Katholieke Universiteit Leuven, Leuven, Belgium, 2011.

DE-BRABANTER, K.; DE-BRABANTER, J.; SUYKENS, J. A. K.; MOOR, B. D. Optimized fixed-size kernel models for large data sets. **Computational Statistics & Data Analysis**, v. 54, n. 6, p. 1484–1504, 2010.

DE-BRABANTER, K.; DE-BRABANTER, J.; SUYKENS, J. A. K.; VANDEWALLE, J.; DE-MOOR, B. Robustness of kernel based regression: Influence and weight functions. In: **Proceedings of the International Joint Conference on Neural Networks (IJCNN)**. Brisbane, Australia: IEEE, 2012. p. 1–8.

DE-BRABANTER, K.; DREESEN, P.; KARSMAKERS, P.; PELCKMANS, K.; DE-BRABANTER, J.; SUYKENS, J. A. K.; MOOR, B. D.; SINT-LIEVEN, H. K. Fixed-size LS-SVM applied to the Wiener-Hammerstein benchmark. In: **Proceedings of the 15th IFAC Symposium on System Identification (SYSID)**. Palais du Grand Large, Saint-Malo, France: IEEE, 2009. p. 826–831.

DE-BRABANTER, K.; PELCKMANS, K.; DE-BRABANTER, J.; DEBRUYNE, M.; SUYKENS, J. A. K.; HUBERT, M.; MOOR, B. D. Robustness of kernel based regression: a comparison of iterative weighting schemes. In: **Proceedings of the 19th International Conference of Artificial Neural Networks (ICANN)**. Limassol, Cyprus: Springer, 2009. p. 100–110.

DEBRUYNE ANDREAS CHRISTMANN, M. H. M.; SUYKENS, J. A. K. Robustness of reweighted least squares kernel based regression. **Journal of Multivariate Analysis**, v. 101, p. 447–463, 2010.

DEBRUYNE, M.; CHRISTMANN, A.; HUBERT, M.; SUYKENS, J. A. K. **Robustness and stability of reweighted kernel based regression**. Leuven, Belgium, 2008.

DEBRUYNE, M.; CHRISTMANN, A.; HUBERT, M.; SUYKENS, J. A. Robustness of reweighted least squares kernel based regression. **Journal of Multivariate Analysis**, Elsevier, v. 101, n. 2, p. 447–463, 2010.

DINIZ, P. S. R. **Adaptive filtering: algorithms and practical implementation**. 3rd. ed. New York, NY, USA: Springer, 2008.

DITZLER, G.; ROVERI, M.; ALIPPI, C.; POLIKAR, R. Learning in nonstationary environments: A survey. **Computational Intelligence Magazine, IEEE**, v. 10, n. 4, p. 12–25, 2015.

DOWNES, T.; GATES, K. E.; MASTERS, A. Exact simplification of support vector solutions. **Journal of Machine Learning Research**, v. 2, p. 293–297, 2001.

ENGEL, Y.; MANNOR, S.; MEIR, R. Sparse online greedy support vector regression. In: **Proceedings of the 13th European Conference on Machine Learning (ECML)**. Helsinki, Finland: T. Elomaa and H. Mannila and H. Toivonen, 2002. v. 1, p. 84–96.

ENGEL, Y.; MANNOR, S.; MEIR, R. **The kernel recursive least squares algorithm**. Technion Institute of Technology, Israel, 2003.

ENGEL, Y.; MANNOR, S.; MEIR, R. The kernel recursive least-squares algorithm. **Signal Processing, IEEE Transactions on**, v. 52, n. 8, p. 2275–2285, 2004.

ESPINOZA, M.; PELCKMANS, K.; HOEGAERTS, L.; SUYKENS, J. A. K.; MOOR, B. D. A comparative study of LS-SVMs applied to the Silverbox identification problem. In: **Proceedings of the 6th Symposium on Nonlinear Control Systems (NOLCOS)**. Stuttgart, Germany: IFAC, 2004.

ESPINOZA, M.; SUYKENS, J. A.; MOOR, B. D. Fixed-size least squares support vector machines: A large scale application in electrical load forecasting. **Computational Management Science**, v. 3, n. 2, p. 113–129, 2006.

ESPINOZA, M.; SUYKENS, J. A. K.; MOOR, B. D. Load forecasting using fixed-size least squares support vector machines. In: **Proceedings of the 8th International Work-Conference on Artificial Neural Networks (IWANN)**. Barcelona, Spain: Springer, 2005. p. 1018–1026.

FALCK, T.; DREESEN, P.; DE-BRABANTER, K.; PELCKMANS, K.; MOOR, B. D.; SUYKENS, J. A. K. Least-squares support vector machines for the identification of Wiener–Hammerstein systems. **Control Engineering Practice**, v. 20, n. 11, p. 1165–1174, 2012.

FALCK, T.; SUYKENS, J. A. K.; DE-MOOR, B. Robustness analysis for least squares kernel based regression: an optimization approach. In: **Proceedings of the 48th IEEE Conference on Decision and Control (CDC)**. Shanghai, China: IEEE, 2009. p. 6774–6779.

FAN, H.; SONG, Q. A sparse kernel algorithm for online time series data prediction. **Expert Systems with Applications**, v. 40, p. 2174–2181, 2013.

FARHANG-BOROUJENY, B. **Adaptive filters: theory and applications**. 1st. ed. Hoboken, NJ, USA: John Wiley & Sons, 2013.

FENG, Y.; HUANG, X.; SHI, L.; YANG, Y.; SUYKENS, J. A. Learning with the maximum correntropy criterion induced losses for regression. **Journal of Machine Learning Research**, v. 16, p. 993–1034, 2015.

FLETCHER, R. **Practical methods of optimization**. 2nd. ed. Hoboken, NJ, USA: John Wiley & Sons, 2013.

FOX, J. **Applied regression analysis, linear models, and related methods**. 1st. ed. London, UK: Sage Publications, 1997.

FOX, J. Robust regression: Appendix to an R and S-PLUS companion to applied regression. 2002.

- FREIRE, A. L. **On the efficient design of extreme learning machines using intrinsic plasticity and evolutionary computation approaches**. PhD Thesis — Federal University of Ceará, Fortaleza, Ceará, Brazil, 2015.
- FRIESS, T.-T.; HARRISON, R. F. A kernel based adaline. In: **Proceedings of the 7th European Symposium on Artificial Neural Networks (ESANN)**. Bruges, Belgium: [s.n.], 1999. p. 245–250.
- GAUSS, C. F. **Theoria motus corporum coelestium in sectionibus conicis solem ambientium auctore Carolo Friderico Gauss**. [S.l.]: sumtibus Frid. Perthes et IH Besser, 1809.
- GERSHENFELD, N. A.; WEIGEND, A. S. **The future of time series**. CU-CS-670-93, Computer Science Technical Reports, University of Colorado, USA, 1993.
- GESTEL, T. V.; SUYKENS, J. A. K.; BAESTAENS, D.-E.; LAMBRECHTS, A.; LANCKRIET, G.; VANDAELE, B.; MOOR, B. D.; VANDEWALLE, J. Financial time series prediction using least squares support vector machines within the evidence framework. **Neural Networks, IEEE Transactions on**, v. 12, n. 4, p. 809–821, 2001.
- GIROLAMI, M. Orthogonal series density estimation and the kernel eigenvalue problem. **Neural Computation**, v. 14, n. 3, p. 669–688, 2002.
- 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**. [S.l.]: JHU Press, 2012.
- GOLUB, G. H.; VAN LOAN, C. F. **Matrix computations**. 4th. ed. Baltimore, Maryland, USA: JHU Press, 2013.
- GRETTON, A.; DOUCET, A.; HERBRICH, R.; RAYNER, P. J.; SCHOLKOPF, B. Support vector regression for black-box system identification. In: **Proceedings of the 11th IEEE Signal Processing Workshop on Statistical Signal Processing**. Singapore, Singapore: IEEE, 2001. p. 341–344.
- HAGER, W. W. Updating the inverse of a matrix. **SIAM Rev.**, v. 31, n. 2, p. 221–239, 1989.
- HAMPEL, F. R. A general definition of qualitative robustness. **Annals of Mathematical Statistics**, v. 42, p. 1887–1896, 1971.
- HAMPEL, F. R.; RONCHETTI, E. M.; ROUSSEEUW, P. J.; STAHEL, W. A. **Robust statistics: the approach based on influence functions**. Hoboken, NJ, USA: John Wiley & Sons, 2011.
- HAYKIN, S. O. **Adaptive filter theory**. 5th. ed. Upper Saddle River, NJ, USA: Prentice Hall, 2013.
- HAYKIN, S. O.; PRÍNCIPE, J. C. S. C. Making sense of a complex world. **Signal Processing Magazine, IEEE**, v. 15, n. 3, p. 66–81, 1998.
- HODGE, V. J.; AUSTIN, J. A survey of outlier detection methodologies. **Artificial Intelligence Review**, v. 22, n. 2, p. 85–126, 2004.

- HOFMANN, T.; SCHÖLKOPF, B.; SMOLA, A. J. Kernel methods in machine learning. **Annals of Statistics**, p. 1171–1220, 2008.
- HOI, S. C. H.; WANG, J.; ZHAO, P. LIBOL: A library for online learning algorithms. **Journal of Machine Learning Research**, v. 15, p. 495–499, 2014.
- HUANG, G.-B. An insight into extreme learning machines: random neurons, random features and kernels. **Cognitive Computation**, v. 6, n. 3, p. 376–390, 2014.
- HUANG, K.; ZHENG, D.; SUN, J.; HOTTA, Y.; FUJIMOTO, K.; NAOI, S. Sparse learning for support vector classification. **Pattern Recognition Letters**, v. 31, n. 13, p. 1944–1951, 2010.
- HUBER, P. J. *et al.* Robust estimation of a location parameter. **Annals of Mathematical Statistics**, v. 35, n. 1, p. 73–101, 1964.
- HUBER, P. J.; RONCHETTI, E. M. **Robust Statistics**. 2nd. ed. Hoboken, NJ, USA: John Wiley & Sons, 2009.
- JANG, J.-S. ANFIS: adaptive-network-based fuzzy inference system. **Systems, Man and Cybernetics, IEEE Transactions on**, v. 23, n. 3, p. 665–685, 1993.
- JUNG, T.; POLANI, D. Sequential learning with LS-SVM for large-scale data sets. In: **Proceedings of the 16th International Conference on Artificial Neural Networks (ICANN)**. Athens, Greece: S. Kollias and A. Stafylopatis and W. Duch and E. Oja, 2006. p. 381–390.
- KANDEL, R. S. **Our changing climate**. 1st. ed. New York, NY, USA: McGraw-Hill, 1992.
- KARTHIK, C.; RAMALAKSHMI, A.; VALARMATHI, K. Support vector regression for Hammerstein-Wiener model identification. In: **Proceedings of the International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE)**. Tamil Nadu, India: IEEE, 2016. p. 1–5.
- KARUSH, W. **Minima of functions of several variables with inequalities as side constraints**. Dissertação (Mestrado) — Department of Mathematics, University of Chicago, Chicago, IL, USA, 1939.
- KASHYAP, R. L.; SUBAS, S.; YAO, S. B. Analysis of the multiple-attribute-tree data-base organization. **Software Engineering, IEEE Transactions on**, n. 6, p. 451–467, 1977.
- KAZEM, A.; SHARIFI, E.; HUSSAIN, F. K.; SABERI, M.; HUSSAIN, O. K. Support vector regression with chaos-based firefly algorithm for stock market price forecasting. **Applied Soft Computing**, v. 13, n. 2, p. 947–958, 2013.
- KHALIL, H. M.; EL-BARDINI, M. Implementation of speed controller for rotary hydraulic motor based on LS-SVM. **Expert Syst. Appl.**, v. 38, n. 11, p. 14249–14256, 2011.
- KIMELDORF, G.; WAHBA, G. Some results on tchebycheffian spline functions. **Journal of Mathematical Analysis and Applications**, v. 33, n. 1, p. 82–95, 1971.
- KIVINEN, J.; SMOLA, A. J.; WILLIAMSON, R. C. Online learning with kernels. **Signal Processing, IEEE Transactions on**, v. 52, n. 8, p. 2165–2176, 2004.
- KOCIJAN, J.; GIRARD, A.; BANKO, B.; MURRAY-SMITH, R. Dynamic systems identification with gaussian processes. **Mathematical and Computer Modelling of Dynamical Systems**, v. 11, n. 4, p. 411–424, 2005.

KUHN, H. W.; TUCKER, A. W. Nonlinear programming. In: **Proceedings of the 2nd Berkeley Symposium**. Berkeley, CA, USA: Jerzy Neyman, 1951. p. 481–492.

LAURAIN, V.; TÓTH, R.; PIGA, D.; ZHENG, W. X. An instrumental least squares support vector machine for nonlinear system identification. **Automatica**, v. 54, p. 340–347, 2015.

LÁZARO, J. L.; De Brabanter, K.; DORRONSORO, J. R.; SUYKENS, J. A. K. Sparse LS-SVMs with  $L_0$ -norm minimization. In: **Proceedings of the 19th European Symposium on Artificial Neural Networks (ESANN)**. Bruges, Belgium: [s.n.], 2011. p. 189–194.

LE, V. L.; BLOCH, G.; LAUER, F. Reduced-size kernel models for nonlinear hybrid system identification. **Neural Networks, IEEE Transactions on**, v. 22, n. 12, p. 2398–2405, 2011.

LEGENDRE, A. M. **Nouvelles méthodes pour la détermination des orbites des comètes**. [S.l.]: F. Didot, 1805.

LI, G.; WEN, C.; LI, Z. G.; ZHANG, A.; YANG, F.; MAO, K. Model-based online learning with kernels. **Neural Networks and Learning Systems, IEEE Transactions on**, v. 24, n. 3, p. 356–369, 2013.

LI, X.; ZHOU, L.; SHENG, J.; DING, R. Recursive least squares parameter estimation algorithm for dual-rate sampled-data nonlinear systems. **Nonlinear Dynamics**, v. 76, n. 2, p. 1327–1334, 2014.

LIU, B.; WANG, Z. On system identification based on online least squares support vector machine. In: **Proceedings of the International Conference on Intelligent Systems and Knowledge Engineering (ISKE)**. Chengdu, China: Atlantis Press, 2007.

LIU, W.; PARK, I.; PRÍNCIPE, J. C. S. C. An information theoretic approach of designing sparse kernel adaptive filters. **Neural Networks, IEEE Transactions on**, v. 20, n. 12, p. 1950–1961, 2009.

LIU, W.; PARK, I.; WANG, Y.; PRÍNCIPE, J. C. S. C. Extended kernel recursive least squares algorithm. **Signal Processing, IEEE Transactions on**, v. 57, n. 10, p. 3801–3804, 2009.

LIU, W.; POKHAREL, P.; PRÍNCIPE, J. C. S. C. Recursively adapted radial basis function networks and its relationship to resource allocating networks and online kernel learning. In: **Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)**. Thessaloniki, Greece: IEEE, 2007. p. 245–250.

LIU, W.; POKHAREL, P.; PRÍNCIPE, J. C. S. C. The kernel least mean square algorithm. **Signal Processing, IEEE Transactions on**, v. 56, n. 2, p. 543–554, 2008.

LIU, W.; PRÍNCIPE, J. C. S. C. Kernel affine projection algorithms. **Journal on Advances in Signal Processing**, n. 1, p. 1–12, 2008.

LIU, W.; PRÍNCIPE, J. C. S. C.; HAYKIN, S. **Kernel adaptive filtering: a comprehensive introduction**. 1st. ed. Hoboken, NJ, USA: John Wiley & Sons, 2011.

LIU, Y.; CHEN, J. Correntropy-based kernel learning for nonlinear system identification with unknown noise: an industrial case study. In: **Proceedings of the 10th International Symposium on Dynamics and Control of Process Systems (DYCOPS)**. Mumbai, India: [s.n.], 2013. p. 361–366.

LIU, Y.; CHEN, J. Correntropy kernel learning for nonlinear system identification with outliers. **Industrial and Engineering Chemistry Research**, p. 1–13, 2013.

LIU, Y.; WANG, H.; YU, J.; LI, P. Selective recursive kernel learning for online identification of nonlinear systems with narx form. **Journal of Process Control**, v. 20, n. 2, p. 181–194, 2010.

LJUNG, L. **System identification: theory for the user**. [S.l.]: Prentice Hall, 1987.

LJUNG, L. **System identification: theory for the User**. 2nd. ed. Upper Saddle River, NJ, USA: Prentice Hall, 1999.

LJUNG, L. Recursive identification algorithms. **Circuits, Systems and Signal Processing**, v. 21, n. 1, p. 57–68, 2002.

MACIEJOWSKI, J. M. **Identification, adaptation, learning: the science of learning models from data**. 1st. ed. New York, NY, USA: Springer, 1996.

MALL, R.; SUYKENS, J. A. Very sparse LSSVM reductions for large-scale data. **Neural Networks and Learning Systems, IEEE Transactions on**, v. 26, n. 5, p. 1086–1097, 2015.

MALL, R.; SUYKENS, J. A. K. Sparse reductions for fixed-size least squares support vector machines on large scale data. In: J. TSENG V. S., C. L. M. H. X. G. P. (Ed.). **Advances in Knowledge Discovery and Data Mining**. Berlin, Germany: Springer, 2013. p. 161–173.

MALLAT, S. G.; ZHANG, Z. Matching pursuits with time-frequency dictionaries. **Signal Processing, IEEE Transactions on**, v. 41, n. 12, p. 3397–3415, 1993.

MARONNA, R. D. M. R. A.; YOHAI, V. J. **Robust statistics: theory and methods**. 1st. ed. Hoboken, NJ, USA: John Wiley & Sons, 2006.

MATTOS, C. L. C.; DAMIANOU, A.; BARRETO, G. A.; LAWRENCE, N. D. Latent autoregressive Gaussian process models for robust system identification. In: **Proceedings of the 11th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems (DYCOPS-CAB)**. Trondheim, Norway: IFAC Publisher, 2016. p. 1121–1126.

MEIR, R.; ZHANG, T. Generalization error bounds for bayesian mixture algorithms. **Journal of Machine Learning Research**, v. 4, p. 839–860, 2003.

MELLIT, A.; PAVAN, A. M.; BENGHANEM, M. Least squares support vector machine for short-term prediction of meteorological time series. **Theoretical and Applied Climatology**, v. 111, n. 1-2, p. 297–307, 2013.

MENEZES-JÚNIOR, J. M.; BARRETO, G. A. Long-term time series prediction with the NARX network: an empirical evaluation. **Neurocomputing**, v. 71, n. 16, p. 3335–3343, 2008.

MERCER, J. Functions of positive and negative type, and their connection with the theory of integral equations. **Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character**, JSTOR, v. 209, p. 415–446, 1909.

MODAGHEGH, H.; KHOSRAVI, H.; MANESH, S. A.; YAZDI, H. S. A new modeling algorithm-normalized kernel least mean square. In: **Proceedings of the International Conference on Innovations in Information Technology (IIT)**. Al-Ain, United Arab Emirates: IEEE, 2009. p. 120–124.



NARENDRA, K. S.; PARTHASARATHY, K. Identification and control of dynamical systems using neural networks. **Neural Networks, IEEE Transactions on**, v. 1, n. 1, p. 4–27, 1990.

NELLES, O. **Nonlinear system identification: from classical approaches to neural networks and fuzzy models**. 1st. ed. New York, NY, USA: Springer, 2001.

ORMÁNDI, R. Variance minimization least squares support vector machines for time series analysis. In: **Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)**. Pisa, Italy: IEEE, 2008. p. 965–970.

PAPAGEORGIOU, G.; BOUBOULIS, P.; THEODORIDIS, S. Robust linear regression analysis - a greedy approach. **Signal Processing, IEEE Transactions on**, v. 63, n. 15, p. 3872–3887, 2015.

PELCKMANS, K.; SUYKENS, J. A. K.; DE MOOR, B. Building sparse representations and structure determination on LS-SVM substrates. **Neurocomputing**, v. 64, p. 137–159, 2005.

PINTELON, R.; SCHOUKENS, J. **System identification: a frequency domain approach**. Hoboken, NJ, USA: John Wiley & Sons, 2012.

PLATT, J. A resource-allocating network for function interpolation. **Neural Computation**, v. 3, n. 2, p. 213–225, 1991.

POKHAREL, P. P.; LIU, W.; PRÍNCIPE, J. C. S. C. Kernel LMS. In: **Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. Honolulu, HI, USA: IEEE, 2007. v. 3, p. III–1421.

PRÍNCIPE, J. C. S. C. **Information theoretic learning: Renyi's entropy and kernel perspectives**. New York, NY, USA: Springer Science & Business Media, 2010.

QUAN, T.; LIU, X.; LIU, Q. Weighted least squares support vector machine local region method for nonlinear time series prediction. **Applied Soft Computing**, v. 10, n. 2, p. 562–566, 2010.

RASMUSSEN, C.; WILLIAMS, C. **Gaussian Processes for Machine Learning**. 1. ed. Cambridge, MA, USA: MIT Press, 2006.

RASMUSSEN, C. E. **Evaluation of Gaussian processes and other methods for non-linear regression**. PhD Thesis — University of Toronto, Toronto, Canada, 1996.

RICHARD, C.; BERMUDEZ, J. C. M.; HONEINE, P. Online prediction of time series data with kernels. **Signal Processing, IEEE Transactions on**, v. 57, n. 3, p. 1058–1066, 2009.

ROJO-ÁLVAREZ, J. L.; MARTINEZ-RAMON, M.; MUNOZ-MARIC, J.; CAMPS-VALLS, G. A unified SVM framework for signal estimation. **Digital Signal Processing**, v. 26, p. 1–20, 2014.

ROJO-ÁLVAREZ, J. L.; MARTÍNEZ-RAMÓN, M.; PRADO-CUMPLIDO, M. de; ARTÉS-RODRÍGUEZ, A.; FIGUEIRAS-VIDAL, A. R. Support vector method for robust arma system identification. **Signal Processing, IEEE Transactions on**, v. 52, n. 1, p. 155–164, 2004.

ROUSSEEUW, P. J.; LEROY, A. M. **Robust Regression and Outlier Detection**. 1st. ed. Hoboken, NJ, USA: John Wiley & Sons, 1987.

- ROUSSEEUW, P. J.; LEROY, A. M. **Robust regression and outlier detection**. [S.l.]: John Wiley & Sons, 2005.
- SAIDE, C.; LENGELLÉ, R.; HONEINE, P.; ACHKAR, R. Online kernel adaptive algorithms with dictionary adaptation for MIMO models. **Signal Processing Letters, IEEE**, 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, n. 11, p. 1391–1410, 2015.
- SANTOS, J. D. A.; MATTOS, C. L. C.; BARRETO, G. A. Performance evaluation of least squares SVR in robust dynamical system identification. In: **Proceedings of the 13th International Work-Conference on Artificial Neural Networks (IWANN)**. Mallorca-Palma, Spain: IEEE, 2015. p. 422–435.
- SAUNDERS, C.; GAMMERMAN, A.; VOVK, V. Ridge regression learning algorithm in dual variables. In: **Proceedings of the 15th International Conference on Machine Learning (ICML)**. Madison, WI, USA: [s.n.], 1998. p. 515–521.
- SAYED, A. H. **Fundamentals of adaptive filtering**. Hoboken, NJ, USA: John Wiley & Sons, 2003.
- SCHÖLKOPF, B.; SMOLA, A. J. **Learning with kernels: Support vector machines, regularization, optimization, and beyond**. 1st. ed. Cambridge, MA, USA: MIT press, 2002.
- SCHOUKENS, J.; NÉMETH, J. G.; CRAMA, P.; ROLAIN, Y.; PINTELON, R. Fast approximate identification of nonlinear systems. **Automatica**, v. 39, n. 7, p. 1267–1274, 2003.
- SJÖBERG, J.; ZHANG, Q.; LJUNG, L.; BENVENISTE, A.; DELYON, B.; GLORENNEC, P.-Y.; HJALMARSSON, H.; JUDITSKY, A. Nonlinear black-box modeling in system identification: a unified overview. **Automatica**, v. 31, n. 12, p. 1691–1724, 1995.
- SLAVAKIS, K.; KIM, S.-J.; MATEOS, G.; GIANNAKIS, G. B. Stochastic approximation vis-à-vis online learning for big data analytics. **Signal Processing Magazine, IEEE**, v. 31, n. 6, p. 124–129, 2014.
- SLAVAKIS, K.; THEODORIDIS, S. Sliding window generalized kernel affine projection algorithm using projection mappings. **Journal on Advances in Signal Processing**, v. 2008, n. 1, p. 1–16, 2008.
- SMOLA, A.; VAPNIK, V. Support vector regression machines. **Advances in Neural Information Processing Systems**, v. 9, p. 155–161, 1997.
- SMOLA, A. J. **Learning With Kernels**. PhD Thesis — Technischen Universitätsat, Berlin, Germany, 1998.
- SMOLA, A. J.; SCHÖLKOPF, B. A tutorial on support vector regression. **Statistics and Computing**, v. 14, n. 3, p. 199–222, 2004.
- SÖDERSTRÖM, T. S.; STOICA, P. G. **System identification**. 1st. ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1989.

SOUZA-JÚNIOR, A. H. **Regional Models and Minimal Learning Machines for Nonlinear Dynamic System Identification**. PhD Thesis — Federal University of Ceará, Fortaleza, Ceará, Brazil, 2014.

STEVENS, J. P. Outliers and influential data points in regression analysis. **Psychological Bulletin**, v. 95, n. 2, p. 334, 1984.

SUYKENS, J. A. K.; De Brabanter, J.; LUKAS, L.; VANDEWALLE, J. Weighted least squares support vector machines: robustness and sparse approximation. **Neurocomputing**, v. 48, n. 1, p. 85–105, 2002.

SUYKENS, J. A. K.; GESTEL, T. V.; BRABANTER, J. D.; MOOR, B. D.; VANDEWALLE, J. **Least Squares Support Vector Machines**. 1st. ed. Hackensack, NJ, USA: World Scientific, 2002.

SUYKENS, J. A. K.; VANDEWALLE, J.; MOOR, B. D. Optimal control by least squares support vector machines. **Neural Networks**, v. 14, n. 1, p. 23–35, 2001.

TAKAGI, T.; SUGENO, M. Fuzzy identification of systems and its application to modeling and control. **Systems, Man and Cybernetics, IEEE Transactions on**, v. 15, n. 1, p. 116–132, 1985.

TANG, H.-S.; XUE, S.-T.; CHEN, R.; SATO, T. Online weighted LS-SVM for hysteretic structural system identification. **Engineering Structures**, v. 28, n. 12, p. 1728–1735, 2006.

TAOUALI, O.; ELAISSI, E.; MESSAOUD, H. Design and comparative study of online kernel methods identification of nonlinear system in RKHS space. **Artificial Intelligence Review**, v. 37, n. 4, p. 289–300, 2012.

TUKEY, J. W. A survey of sampling from contaminated distributions. **Contributions to Probability and Statistics**, v. 2, p. 448–485, 1960.

VAART, A. W. V. D.; WELLNER, J. A. Weak convergence and empirical processes with applications to statistics. In: **Springer Series in Statistics**. New York, NY, USA: Springer, 1996. p. 16–28.

VAERENBERGH, S. V.; LÁZARO-GREDILLA, M.; SANTAMARÍA, I. Kernel recursive least-squares tracker for time-varying regression. **Neural Networks and Learning Systems, IEEE Transactions on**, v. 23, n. 8, p. 1313–1326, 2012.

VAERENBERGH, S. V.; VIA, J.; SANTAMARÍA, I. A sliding-window kernel RLS algorithm and its application to nonlinear channel identification. In: **Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)**. Toulouse, France: IEEE, 2006. v. 5.

VAERENBERGH, S. V.; VÍA, J.; SANTAMARÍA, I. Nonlinear system identification using a new sliding-window kernel RLS algorithm. **Journal of Communications**, v. 2, n. 3, p. 1–8, 2007.

VALYON, J.; HORVÁTH, G. A robust LS-SVM regression. **International Journal of Computational Intelligence**, v. 3, p. 243–248, 2006.

VAPNIK, V. **The nature of statistical learning theory**. 1st. ed. New York, NY, USA: Springer, 1995.

- VAPNIK, V. **Statistical learning theory**. 1st. ed. Hoboken, NJ, USA: John Wiley & Sons, 1998.
- VAPNIK, V.; CHERVONENKIS, A. A note on one class of perceptrons. **Automation and Remote Control**, v. 25, n. 1, 1964.
- VAPNIK, V.; GOLOWICH, S. E.; SMOLA, A. Support vector method for function approximation, regression estimation, and signal processing. In: **Proceedings of the Advances in Neural Information Processing Systems (NIPS)**. [S.l.]: Citeseer, 1996.
- VAPNIK, V. N. Pattern recognition using generalized portrait method. **Automation and Remote Control**, v. 24, p. 774–780, 1963.
- VAPNIK, V. N.; CHERVONENKIS, A. J. Theory of pattern recognition. Nauka, 1974.
- WANG, K.; ZHANG, J.; CHEN, Y.; ZHONG, P. Least absolute deviation support vector regression. **Mathematical Problems in Engineering**, v. 2014, 2014.
- WEDDERBURN, R. W. M. Quasi-likelihood functions, generalized linear models, and the Gauss—Newton method. **Biometrika**, v. 61, n. 3, p. 439–447, 1974.
- WEN, W.; HAO, Z.; YANG, X. A heuristic weight-setting strategy and iteratively updating algorithm for weighted least-squares support vector regression. **Neurocomputing**, v. 71, n. 16, p. 3096–3103, 2008.
- WEN, W.; HAO, Z.; YANG, X. Robust least squares support vector machine based on recursive outlier elimination. **Soft Computing**, v. 14, n. 11, p. 1241–1251, 2010.
- WIDROW, B.; HOFF, M. E. Adaptive switching circuits. In: NEW YORK. **IRE WESCON convention record**. [S.l.], 1960. v. 4, n. 1, p. 96–104.
- WIDROW, B.; KAMENETSKY, M. Statistical efficiency of adaptive algorithms. **Neural Networks**, v. 16, n. 5-6, p. 735–744, 2003.
- WILLIAMS, C.; SEEGER, M. Using the Nyström method to speed up kernel machines. In: **Proceedings of the 14th Annual Conference on Neural Information Processing Systems (NIPS)**. [S.l.]: MIT Press, 2001. p. 682–688.
- WILLS, A.; SCHÖN, T. B.; LJUNG, L.; NINNESS, B. Identification of Hammerstein-Wiener models. **Automatica**, v. 49, n. 1, p. 70–81, 2013.
- WU, Z.; SHI, J.; ZHANG, X.; MA, W.; CHEN, B.; MEMBER, S. Kernel recursive maximum correntropy. **Signal Processing**, v. 117, p. 11–16, 2015.
- YANG, X.; TAN, L.; HE, L. A robust least squares support vector machine for regression and classification with noise. **Neurocomputing**, v. 140, p. 41–52, 2014.
- YUILLE, A. L.; RANGARAJAN, A. The concave-convex procedure. **Neural Computation**, v. 15, n. 4, p. 915–936, 2003.
- ZHANG, B.; BILLINGS, S. Identification of continuous-time nonlinear systems: The nonlinear difference equation with moving average noise (NDEMA) framework. **Mechanical Systems and Signal Processing**, v. 60, p. 810–835, 2015.

ZHANG, B.; BILLINGS, S. A. Volterra series truncation and kernel estimation of nonlinear systems in the frequency domain. **Mechanical Systems and Signal Processing**, v. 84-A, p. 39–57, 2017.

ZHANG, K.; FEI, M.; WU, J.; ZHANG, P. Fast prediction model based big data system identification. In: **Proceedings of the Chinese Automation Congress (CAC)**. Changsha, Hunan, China: IEEE, 2013. p. 465–469.

ZHANG, Q.; LJUNG, L. **Multiple steps prediction with nonlinear ARX models**. Linköping, Sweden, 2007.

ZHANG, Z. Parameter estimation techniques: A tutorial with application to conic fitting. **Image and Vision Computing**, v. 15, n. 1, p. 59–76, 1997.

ZHANG, Z.; WANG, H.; LIU, Y.; LIU, D.; YANG, S. Self-tuning PID control based on LSSVM for ship steering system. In: **Proceedings of the International Conference on Computer Technologies in Physical and Engineering Applications (ICCTPEA)**. Saint-Petersburg, Russia: IEEE, 2014. p. 215–216.

ZHAO, S.; CHEN, B.; PRÍNCIPE, J. C. S. C. Kernel adaptive filtering with maximum correntropy criterion. In: **Proceedings of the International Joint Conference on Neural Networks (IJCNN)**. San Jose, CA, USA: IEEE, 2011. p. 2012–2017.

ZHU, P.-P.; CHEN, B.; PRÍNCIPE, J. C. S. C. A novel extended kernel recursive least squares algorithm. **Neural Networks**, v. 32, p. 349–357, 2012.

ZOU, Y.; CHAN, S.; NG, T. A recursive least M-estimate (RLM) adaptive filter for robust filtering in impulse noise. **Signal Processing Letters, IEEE**, v. 7, n. 11, p. 324–326, 2000.

ZOUBIR, A. M.; KOIVUNEN, V.; CHAKHCHOUKH, Y.; MUMA, M. Robust estimation in signal processing: A tutorial-style treatment of fundamental concepts. **Signal Processing Magazine, IEEE**, v. 29, n. 4, p. 61–80, 2012.

## APPENDIX A – THE ROB-KRLS MODEL

This appendix presents a general formulation of the ROB-KRLS model, including all the mathematical derivations carried out up to obtain the model solution. For this purpose, it is worth highlighting that this appendix follows a similar procedure to that presented in Section 6.2 and also adopts the same mathematical notation used throughout this thesis.

### A.1 Complete Formulation

Initially, consider a given stream of training examples

$$\mathcal{D}_t = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_t, y_t)\}, \quad (\text{A.1})$$

where  $(\mathbf{x}_t, y_t) \in \mathbb{R}^d \times \mathbb{R}$  denotes the current input-output pair. Then, the KRLS algorithm assumes a functional form, e.g.  $f(\mathbf{x}_i) = \boldsymbol{\phi}^\top(\mathbf{x}_i)\mathbf{w}$ , and minimizes in a sequential procedure the following cost function:

$$J(\mathbf{w}) = \sum_{i=1}^t (y_i - f(\mathbf{x}_i))^2 = \|\mathbf{y}_t - \boldsymbol{\Phi}_t^\top \mathbf{w}\|^2, \quad (\text{A.2})$$

where  $\boldsymbol{\phi}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^{d_h}$  is the nonlinear mapping,  $\mathbf{w} \in \mathbb{R}^{d_h}$  is the vector of parameters,  $\boldsymbol{\Phi}_t = [\boldsymbol{\phi}(\mathbf{x}_1), \dots, \boldsymbol{\phi}(\mathbf{x}_t)] \in \mathbb{R}^{d_h \times t}$  is a matrix storing all the projected vectors and  $\mathbf{y}_t = (y_1, \dots, y_t)^\top$  is the vector of outputs.

The optimal vector  $\mathbf{w}_t$  in Eq. (A.2) can be expressed as,

$$\mathbf{w}_t = \sum_{i=1}^t \alpha_i \boldsymbol{\phi}(\mathbf{x}_i) = \boldsymbol{\Phi}_t \boldsymbol{\alpha}_t, \quad (\text{A.3})$$

where  $\boldsymbol{\alpha}_t = (\alpha_1, \dots, \alpha_t)^\top$  is the vector of coefficients. Therefore, the problem in Eq. (A.2) can be rewritten as

$$J(\boldsymbol{\alpha}_t) = \|\mathbf{y}_t - \mathbf{K}_t \boldsymbol{\alpha}_t\|^2, \quad (\text{A.4})$$

where  $\mathbf{K}_t = \boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t$ . From Engel *et al.* (2004), it is possible to write the following equalities:

$$\mathbf{w}_t = \boldsymbol{\Phi}_t \boldsymbol{\alpha}_t \approx \tilde{\boldsymbol{\Phi}}_t \mathbf{A}_t^\top \boldsymbol{\alpha}_t = \tilde{\boldsymbol{\Phi}}_t \tilde{\boldsymbol{\alpha}}_t, \quad (\text{A.5})$$

where  $\mathbf{A}_t = [\mathbf{a}_1 \ \mathbf{a}_2 \ \dots \ \mathbf{a}_t]^\top \in \mathbb{R}^{t \times m_t}$ . Then, Eq. (A.4) becomes

$$\begin{aligned} J(\tilde{\boldsymbol{\alpha}}_t) &= \|\mathbf{y}_t - \boldsymbol{\Phi}_t^\top \tilde{\boldsymbol{\Phi}}_t \tilde{\boldsymbol{\alpha}}_t\|^2, \\ &= \|\boldsymbol{\Phi}_t^\top \tilde{\boldsymbol{\Phi}}_t \tilde{\boldsymbol{\alpha}}_t - \mathbf{y}_t\|^2, \\ &= \|\mathbf{A}_t \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t - \mathbf{y}_t\|^2, \end{aligned} \quad (\text{A.6})$$

where  $\tilde{\boldsymbol{\alpha}}_t \in \mathbb{R}^{m_t}$  is the reduced vector of  $m_t$  coefficients. The next step is to take the gradient vector  $\nabla_{\tilde{\boldsymbol{\alpha}}_t} J(\tilde{\boldsymbol{\alpha}}_t) = \partial J(\tilde{\boldsymbol{\alpha}}_t) / \partial \tilde{\boldsymbol{\alpha}}_t$  of Eq. (A.6) and equaling it to zero, such as

$$\begin{aligned} \frac{\partial J(\tilde{\boldsymbol{\alpha}}_t)}{\partial \tilde{\boldsymbol{\alpha}}_t} &= 2(\mathbf{A}_t \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t - \mathbf{y}_t) \mathbf{A}_t \tilde{\mathbf{K}}_t, \\ &= \tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{A}_t \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t - \tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{y}_t = 0, \\ \Rightarrow \tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{A}_t \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t &= \tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{y}_t. \end{aligned} \quad (\text{A.7})$$

The expression in Eq. (A.7) corresponds to the minimization of the functional in Eq. (A.6), which yields the following solution:

$$\begin{aligned} \tilde{\boldsymbol{\alpha}}_t &= (\tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{A}_t \tilde{\mathbf{K}}_t)^{-1} \tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{y}_t, \\ &= \tilde{\mathbf{K}}_t^{-1} (\mathbf{A}_t^\top \mathbf{A}_t)^{-1} \mathbf{A}_t^\top \mathbf{y}_t. \end{aligned} \quad (\text{A.8})$$

Finally, by defining a matrix  $\mathbf{P}_t$  as

$$\mathbf{P}_t = (\mathbf{A}_t^\top \mathbf{A}_t)^{-1}, \quad (\text{A.9})$$

one can write the expression for  $\tilde{\boldsymbol{\alpha}}_t$  in Eq. (A.8) as

$$\tilde{\boldsymbol{\alpha}}_t = \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t \mathbf{A}_t^\top \mathbf{y}_t. \quad (\text{A.10})$$

From now on, instead of minimizing the standard squared norm of the error vector as in Eq. (A.6), a robust KRLS predictor model is designed to minimize the following cost function:

$$J(\tilde{\boldsymbol{\alpha}}_t^r) = \rho(\mathbf{y}_t - \mathbf{A}_t \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t^r) = \rho(\mathbf{e}_t), \quad (\text{A.11})$$

where  $\tilde{\boldsymbol{\alpha}}_t^r$  is the robust version of  $\tilde{\boldsymbol{\alpha}}_t$  and  $\mathbf{e}_t = [e_1, \dots, e_i, \dots, e_t]^\top$  is the vector of prediction errors up to the time step  $t$ .

In order to minimize the cost function in Eq. (A.11), an optimal solution vector  $\boldsymbol{\alpha}_t^r$  is searched by computing the gradient vector  $\nabla_{\tilde{\boldsymbol{\alpha}}_t^r} J(\tilde{\boldsymbol{\alpha}}_t^r) = \partial J(\tilde{\boldsymbol{\alpha}}_t^r) / \partial \tilde{\boldsymbol{\alpha}}_t^r$  and equaling it to zero, which is given by

$$\frac{\partial \rho(\mathbf{e}_t)}{\partial \tilde{\boldsymbol{\alpha}}_t^r} \mathbf{A}_t \tilde{\mathbf{K}}_t = \mathbf{0}. \quad (\text{A.12})$$

Defining a weight function for the prediction error vector  $\mathbf{e}_t$  as

$$v(\mathbf{e}_t) = \frac{1}{\mathbf{e}_t} \frac{\partial \rho(\mathbf{e}_t)}{\partial \tilde{\boldsymbol{\alpha}}_t^r}, \quad (\text{A.13})$$

the problem in Eq. (A.12) can be rewritten as

$$\begin{aligned}
\mathbf{e}_t \mathbf{V}_t \mathbf{A}_t \tilde{\mathbf{K}}_t &= (\mathbf{y}_t - \mathbf{A}_t \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t^r) \mathbf{V}_t \mathbf{A}_t \tilde{\mathbf{K}}_t, \\
&= \tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{V}_t \mathbf{y}_t - \tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{V}_t \mathbf{A}_t \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t^r = \mathbf{0}, \\
\Rightarrow \tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{V}_t \mathbf{A}_t \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t^r &= \tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{V}_t \mathbf{y}_t,
\end{aligned} \tag{A.14}$$

where  $\mathbf{V}_t = \text{diag}\{v_i\}_{i=1}^t \in \mathbb{R}^{t \times t}$ . The expression in Eq. (A.14) corresponds to the minimization of the functional in Eq. (A.11), which yields the following solution:

$$\begin{aligned}
\tilde{\boldsymbol{\alpha}}_t^r &= (\tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{V}_t \mathbf{A}_t \tilde{\mathbf{K}}_t)^{-1} \tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{V}_t \mathbf{y}_t, \\
&= \tilde{\mathbf{K}}_t^{-1} (\mathbf{A}_t^\top \mathbf{V}_t \mathbf{A}_t)^{-1} \mathbf{A}_t^\top \mathbf{V}_t \mathbf{y}_t,
\end{aligned} \tag{A.15}$$

Then, as done for the KRLS predictor, one defines a matrix  $\mathbf{P}_t^r$  as

$$\mathbf{P}_t^r = (\mathbf{A}_t^\top \mathbf{V}_t \mathbf{A}_t)^{-1}, \tag{A.16}$$

and, therefore,

$$\tilde{\boldsymbol{\alpha}}_t^r = \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t^r \mathbf{A}_t^\top \mathbf{V}_t \mathbf{y}_t. \tag{A.17}$$

Next, we adopt the same procedure developed for the original KRLS algorithm to iteratively solve Eq. (A.17).

### A.1.1 Case 1 - Unchanged Dictionary

In this case,  $\delta_t \leq \nu$ , meaning that  $\boldsymbol{\phi}(\mathbf{x}_t)$  is approximately linearly dependent on the dictionary vectors. Hence,  $\mathbf{x}_t$  is not added to the dictionary ( $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv}$ ) and, consequently, the kernel matrix is not changed ( $\tilde{\mathbf{K}}_t = \tilde{\mathbf{K}}_{t-1}$ ).

Since  $\mathbf{a}_t$  needs to be computed by Eq. (6.6) to determine  $\delta_t$ , the matrix  $\mathbf{A}_t$  is built iteratively by the inclusion of  $\mathbf{a}_t$ , i.e.  $\mathbf{A}_t = [\mathbf{A}_{t-1}^\top \ \mathbf{a}_t]^\top$ . Thus, by defining a matrix  $\mathbf{B}_t^r$  as

$$\begin{aligned}
\mathbf{B}_t^r &= \mathbf{A}_t^\top \mathbf{V}_t \mathbf{A}_t \\
&= \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{A}_{t-1} + v_t \mathbf{a}_t \mathbf{a}_t^\top \\
&= \mathbf{B}_{t-1}^r + v_t \mathbf{a}_t \mathbf{a}_t^\top.
\end{aligned} \tag{A.18}$$

where  $\mathbf{A}_t^\top \mathbf{V}_t \mathbf{A}_t = \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{A}_{t-1} + v_t \mathbf{a}_t \mathbf{a}_t^\top$ , one can apply the matrix inversion lemma (GOLUB; LOAN, 2012; HAGER, 1989) to recursively compute the matrix  $\mathbf{P}_t^r$  as

$$\mathbf{P}_t^r = (\mathbf{B}_t^r)^{-1} = \mathbf{P}_{t-1}^r - \frac{\mathbf{P}_{t-1}^r v_t \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1}^r}{1 + v_t \mathbf{a}_t^\top \mathbf{P}_{t-1}^r \mathbf{a}_t}. \tag{A.19}$$



Also defining a robust gain vector  $\mathbf{q}_t^r$  as

$$\mathbf{q}_t^r = \frac{v_t \mathbf{P}_{t-1}^r \mathbf{a}_t}{1 + v_t \mathbf{a}_t^\top \mathbf{P}_{t-1}^r \mathbf{a}_t}, \quad (\text{A.20})$$

one gets

$$\mathbf{P}_t^r = \mathbf{P}_{t-1}^r - \mathbf{q}_t^r \mathbf{a}_t^\top \mathbf{P}_{t-1}^r. \quad (\text{A.21})$$

Finally, using the fact that  $\mathbf{A}_t^\top \mathbf{V}_t \mathbf{y}_t = \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{y}_{t-1} + \mathbf{a}_t v_t \mathbf{y}_t$ , the ROB-KRLS update rule for  $\tilde{\boldsymbol{\alpha}}_t^r$  can be written as

$$\begin{aligned} \tilde{\boldsymbol{\alpha}}_t^r &= \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t^r \mathbf{A}_t^\top \mathbf{V}_t \mathbf{y}_t, \\ &= \tilde{\mathbf{K}}_t^{-1} (\mathbf{P}_{t-1}^r - \mathbf{q}_t^r \mathbf{a}_t^\top \mathbf{P}_{t-1}^r) (\mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{y}_{t-1} + \mathbf{a}_t v_t \mathbf{y}_t), \\ &= (\tilde{\mathbf{K}}_t^{-1} \mathbf{P}_{t-1}^r - \tilde{\mathbf{K}}_t^{-1} \mathbf{q}_t^r \mathbf{a}_t^\top \mathbf{P}_{t-1}^r) (\mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{y}_{t-1} + \mathbf{a}_t v_t \mathbf{y}_t), \\ &= \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_{t-1}^r \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{y}_{t-1} + \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_{t-1}^r \mathbf{a}_t v_t \mathbf{y}_t - \tilde{\mathbf{K}}_t^{-1} \mathbf{q}_t^r \mathbf{a}_t^\top \mathbf{P}_{t-1}^r \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{y}_{t-1} \\ &\quad - \tilde{\mathbf{K}}_t^{-1} \mathbf{q}_t^r \mathbf{a}_t^\top \mathbf{P}_{t-1}^r \mathbf{a}_t v_t \mathbf{y}_t, \\ &= \tilde{\boldsymbol{\alpha}}_{t-1}^r + \tilde{\mathbf{K}}_t^{-1} \mathbf{a}_t v_t \mathbf{y}_t (\mathbf{P}_{t-1}^r - \mathbf{q}_t^r \mathbf{a}_t^\top \mathbf{P}_{t-1}^r) - \mathbf{q}_t^r \mathbf{a}_t^\top \tilde{\boldsymbol{\alpha}}_{t-1}^r, \\ &= \tilde{\boldsymbol{\alpha}}_{t-1}^r + \tilde{\mathbf{K}}_t^{-1} \mathbf{a}_t v_t \mathbf{y}_t \mathbf{P}_t^r - \mathbf{q}_t^r \mathbf{a}_t^\top \tilde{\boldsymbol{\alpha}}_{t-1}^r, \\ &= \tilde{\boldsymbol{\alpha}}_{t-1}^r + \tilde{\mathbf{K}}_t^{-1} (\mathbf{P}_t^r \mathbf{a}_t v_t \mathbf{y}_t - \mathbf{q}_t^r \mathbf{a}_t^\top \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_{t-1}^r), \\ &= \tilde{\boldsymbol{\alpha}}_{t-1}^r + \tilde{\mathbf{K}}_t^{-1} (\mathbf{q}_t^r \mathbf{y}_t - \mathbf{q}_t^r \mathbf{a}_t^\top \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_{t-1}^r), \\ &= \tilde{\boldsymbol{\alpha}}_{t-1}^r + \tilde{\mathbf{K}}_t^{-1} \mathbf{q}_t^r (\mathbf{y}_t - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}^r), \end{aligned} \quad (\text{A.22})$$

where the last equalities are based on  $\mathbf{q}_t^r = v_t \mathbf{P}_t^r \mathbf{a}_t$  and  $\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) = \tilde{\mathbf{K}}_t \mathbf{a}_t$ .

### A.1.2 Case 2 - Updating the Dictionary

In this case, one gets  $\delta_t > v$ , implying that  $\mathbf{x}_t$  must be added to the dictionary, i.e.  $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv} \cup \{\mathbf{x}_t\}$  and  $m_t = m_{t-1} + 1$ . Hence, the kernel matrix must be updated accordingly.

In order to compute  $\tilde{\mathbf{K}}_t^{-1}$  recursively, consider the matrix  $\tilde{\mathbf{K}}_{t-1}$  and the information provided by the new sample to build  $\tilde{\mathbf{K}}_t$  as

$$\tilde{\mathbf{K}}_t = \begin{bmatrix} \tilde{\mathbf{K}}_{t-1} & \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \\ \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top & k_{tt} \end{bmatrix}, \quad (\text{A.23})$$

and also consider the following matrix identity (GOLUB; LOAN, 2012):

$$\text{if } \mathbf{D}_t = \begin{bmatrix} \mathbf{D}_{t-1} & \mathbf{d} \\ \mathbf{d}^\top & d^* \end{bmatrix}, \quad (\text{A.24})$$

then

$$\mathbf{D}_t^{-1} = \begin{bmatrix} \mathbf{D}_{t-1}^{-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{1}{\Delta_d} \begin{bmatrix} -\mathbf{D}_{t-1}^{-1} \mathbf{d} \\ 1 \end{bmatrix} \begin{bmatrix} -\mathbf{D}_{t-1}^{-1} \mathbf{d} \\ 1 \end{bmatrix}^\top, \quad (\text{A.25})$$

where  $\mathbf{D}_t$  and  $\mathbf{d}_t$  are a square matrix and a vector, respectively, of appropriate sizes. Furthermore,  $d^*$  and  $\Delta_d$  are scalars such that  $\Delta_d = d^* - \mathbf{d}^\top \mathbf{D}_{t-1}^{-1} \mathbf{d}$ .

Then, assuming the equalities  $\mathbf{D}_{t-1} = \tilde{\mathbf{K}}_{t-1}$ ,  $\mathbf{d} = \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)$  and  $d^* = k_{tt}$  in Eq. (A.24), and applying them in Eq. (A.25), it is possible to recursively compute the inverse matrix  $\tilde{\mathbf{K}}_t^{-1}$  as

$$\begin{aligned} \tilde{\mathbf{K}}_t^{-1} &= \begin{bmatrix} \tilde{\mathbf{K}}_{t-1}^{-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{1}{\delta_t} \begin{bmatrix} -\tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \\ 1 \end{bmatrix} \begin{bmatrix} -\tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \\ 1 \end{bmatrix}^\top, \\ &= \begin{bmatrix} \tilde{\mathbf{K}}_{t-1}^{-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{1}{\delta_t} \begin{bmatrix} \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\mathbf{K}}_{t-1}^{-1} & -\tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \\ -\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\mathbf{K}}_{t-1}^{-1} & 1 \end{bmatrix}, \\ &= \begin{bmatrix} \tilde{\mathbf{K}}_{t-1}^{-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{1}{\delta_t} \begin{bmatrix} \mathbf{a}_t \mathbf{a}_t^\top & -\mathbf{a}_t \\ -\mathbf{a}_t^\top & 1 \end{bmatrix}, \end{aligned} \quad (\text{A.26})$$

where  $\delta_t = k_{tt} - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)$  and  $\mathbf{a}_t = \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)$ . The final expression for  $\tilde{\mathbf{K}}_t^{-1}$  is obtained by adding the matrices in Eq. (A.26), which is given by

$$\tilde{\mathbf{K}}_t^{-1} = \frac{1}{\delta_t} \begin{bmatrix} \delta_t \tilde{\mathbf{K}}_{t-1}^{-1} + \mathbf{a}_t \mathbf{a}_t^\top & -\mathbf{a}_t \\ -\mathbf{a}_t^\top & 1 \end{bmatrix}, \quad (\text{A.27})$$

One should recall that for the Case 2, not only the dimension of the kernel matrix  $\tilde{\mathbf{K}}_t$  increases due to the inclusion of sample  $\mathbf{x}_t$  into the dictionary, but also the dimensions of the matrices  $\mathbf{A}_t$  and  $\mathbf{V}_t$ . Hence, one can write

$$\mathbf{A}_t^\top \mathbf{V}_t \mathbf{A}_t = \begin{bmatrix} \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{A}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad (\text{A.28})$$

$$\mathbf{A}_t^\top \mathbf{V}_t = \begin{bmatrix} \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad (\text{A.29})$$

and

$$\mathbf{P}_t^r = (\mathbf{A}_t^\top \mathbf{V}_t \mathbf{A}_t)^{-1} = \begin{bmatrix} \mathbf{P}_{t-1}^r & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}. \quad (\text{A.30})$$

In order to compute the vector solution  $\tilde{\alpha}_t^r = \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t^r \mathbf{A}_t^\top \mathbf{V}_t \mathbf{y}_t$ , one can still represent the vector  $\mathbf{y}_t$  by

$$\mathbf{y}_t = \begin{bmatrix} \mathbf{y}_{t-1} \\ y_t \end{bmatrix}. \quad (\text{A.31})$$

Thus, substituting Eqs. (A.27), (A.29), (A.30) and (A.31) into Eq. (A.17), it is possible to develop the expression for  $\tilde{\alpha}_t^r$  as

$$\begin{aligned} \tilde{\alpha}_t^r &= \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t^r \mathbf{A}_t^\top \mathbf{V}_t \mathbf{y}_t, \\ &= \tilde{\mathbf{K}}_t^{-1} \begin{bmatrix} \mathbf{P}_{t-1}^r & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{y}_{t-1} \\ y_t \end{bmatrix}, \\ &= \tilde{\mathbf{K}}_t^{-1} \begin{bmatrix} \mathbf{P}_{t-1}^r \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{y}_{t-1} \\ y_t \end{bmatrix}, \\ &= \tilde{\mathbf{K}}_t^{-1} \begin{bmatrix} \mathbf{P}_{t-1}^r \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{y}_{t-1} \\ y_t \end{bmatrix}, \\ &= \frac{1}{\delta_t} \begin{bmatrix} \delta_t \tilde{\mathbf{K}}_{t-1}^{-1} + \mathbf{a}_t \mathbf{a}_t^\top & -\mathbf{a}_t \\ -\mathbf{a}_t^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_{t-1}^r \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{y}_{t-1} \\ y_t \end{bmatrix}, \\ &= \frac{1}{\delta_t} \begin{bmatrix} \delta_t \tilde{\mathbf{K}}_{t-1}^{-1} \mathbf{P}_{t-1}^r \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{y}_{t-1} + \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1}^r \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{y}_{t-1} - \mathbf{a}_t y_t \\ -\mathbf{a}_t^\top \mathbf{P}_{t-1}^r \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{y}_{t-1} + y_t \end{bmatrix}, \quad (\text{A.32}) \\ &= \begin{bmatrix} (\tilde{\alpha}_t^r)^{(1)} \\ (\tilde{\alpha}_t^r)^{(2)} \end{bmatrix}. \quad (\text{A.33}) \end{aligned}$$

Then, developing the upper and lower terms in Eq. (A.32) for compute  $(\tilde{\alpha}_t^r)^{(1)}$  and  $(\tilde{\alpha}_t^r)^{(2)}$  in Eq. (A.33), respectively, one gets

$$\begin{aligned} (\tilde{\alpha}_t^r)^{(1)} &= \frac{1}{\delta_t} (\delta_t \tilde{\mathbf{K}}_{t-1}^{-1} \mathbf{P}_{t-1}^r \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{y}_{t-1} + \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1}^r \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{y}_{t-1} - \mathbf{a}_t y_t), \\ &= \tilde{\alpha}_{t-1}^r - \frac{\mathbf{a}_t}{\delta_t} (y_t - \mathbf{a}_t^\top \mathbf{P}_{t-1}^r \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{y}_{t-1}), \\ &= \tilde{\alpha}_{t-1}^r - \frac{\mathbf{a}_t}{\delta_t} (y_t - \mathbf{a}_t^\top \tilde{\mathbf{K}}_{t-1} \tilde{\mathbf{K}}_{t-1}^{-1} \mathbf{P}_{t-1}^r \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{y}_{t-1}), \\ &= \tilde{\alpha}_{t-1}^r - \frac{\mathbf{a}_t}{\delta_t} (y_t - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\alpha}_{t-1}^r), \quad (\text{A.34}) \end{aligned}$$

and

$$\begin{aligned}
(\tilde{\boldsymbol{\alpha}}_t^r)^{(2)} &= \frac{1}{\delta_t} (y_t - \mathbf{a}_t^\top \mathbf{P}_{t-1}^r \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{y}_{t-1}), \\
&= \frac{1}{\delta_t} (y_t - \mathbf{a}_t^\top \tilde{\mathbf{K}}_{t-1} \tilde{\mathbf{K}}_{t-1}^{-1} \mathbf{P}_{t-1}^r \mathbf{A}_{t-1}^\top \mathbf{V}_{t-1} \mathbf{y}_{t-1}), \\
&= \frac{1}{\delta_t} (y_t - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}^r),
\end{aligned} \tag{A.35}$$

where, for the final equalities in Eqs. (A.34) and (A.35), one applies  $\mathbf{a}_t^\top \tilde{\mathbf{K}}_{t-1} = \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top$ .

Therefore, the resulting solution vector is given by

$$\tilde{\boldsymbol{\alpha}}_t^r = \begin{bmatrix} \tilde{\boldsymbol{\alpha}}_{t-1}^r - \frac{a_t}{\delta_t} (y_t - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}^r) \\ \frac{1}{\delta_t} (y_t - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}^r) \end{bmatrix}. \tag{A.36}$$

Finally, since the parameter vector  $\tilde{\boldsymbol{\alpha}}_t^r$  has been updated (either for Case 1, or Case 2), the sparse and robust solution for the ROB-KRLS nonlinear predictor is given by

$$\hat{y}_t = \hat{f}(\mathbf{x}) = \sum_{m=1}^{m_t} \tilde{\alpha}_m^r k(\mathbf{x}, \mathbf{x}_m) = \tilde{\mathbf{k}}_{m_t}(\mathbf{x})^\top \tilde{\boldsymbol{\alpha}}_t^r. \tag{A.37}$$

## APPENDIX B – THE OS-LSSVR MODEL

This appendix presents a general formulation of the OS-LSSVR model, including all the mathematical derivations carried out up to obtain the model solution. For this purpose, it is worth highlighting that this appendix follows a similar procedure to that presented in Section 6.3 and also adopts the same mathematical notation used throughout this thesis.

### B.1 Complete Formulation

In general, the kernel-based models are able to solve nonlinear regression problems of the form  $f(\mathbf{x}_i) = \boldsymbol{\phi}^\top(\mathbf{x}_i)\mathbf{w}$ ,  $\mathbf{w} \in \mathbb{R}^{d_h}$ . Their respective model solutions are typically given as

$$f(\mathbf{x}) = \sum_{n=1}^N \alpha_n k(\mathbf{x}, \mathbf{x}_n), \quad (\text{B.1})$$

computed over the entire training dataset  $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$ . The value  $\alpha_n \in \mathbb{R}$  is the coefficient associated with the training sample  $\mathbf{x}_n$ .

As already discussed in Chapter 2, the parameter estimation problem for the standard LSSVR formulation is expressed by

$$J(\mathbf{w}) = \sum_{n=1}^N (f(\mathbf{x}_n) - y_n)^2 + \gamma \|\mathbf{w}\|^2, \quad (\text{B.2})$$

where  $\gamma > 0$  is the regularization parameter. Then, applying the optimality conditions of the *Lagrangian* into the problem in Eq. (B.2), the parameter vector  $\mathbf{w}$  can be computed as

$$\mathbf{w} = \sum_{n=1}^N \alpha_n \boldsymbol{\phi}(\mathbf{x}_n) = \boldsymbol{\Phi} \boldsymbol{\alpha}, \quad (\text{B.3})$$

where  $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1), \dots, \boldsymbol{\phi}(\mathbf{x}_N)] \in \mathbb{R}^{d_h \times N}$  and  $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_N)^\top$  is the vector of Lagrange multipliers. Then, using  $f(\mathbf{x}_i) = \boldsymbol{\phi}^\top(\mathbf{x}_i)\mathbf{w}$  and Eq. (B.3), one can rewrite the functional in Eq. (B.2) as

$$J(\boldsymbol{\alpha}) = \|\mathbf{K}\boldsymbol{\alpha} - \mathbf{y}\|^2 + \gamma \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}, \quad (\text{B.4})$$

where  $\mathbf{K} = \boldsymbol{\Phi}^\top \boldsymbol{\Phi}$  and  $\mathbf{y} = (y_1, \dots, y_N)^\top$ . Therefore, the optimal  $\boldsymbol{\alpha}$  can be estimated by finding the gradient vector  $\nabla_{\boldsymbol{\alpha}} J(\boldsymbol{\alpha}) = \partial J(\boldsymbol{\alpha}) / \partial \boldsymbol{\alpha}$  and equating it to zero, which results in the following linear system:

$$(\mathbf{K} + \gamma \mathbf{I}) \boldsymbol{\alpha} = \mathbf{y}, \quad (\text{B.5})$$

whose solution is given by the OLS algorithm as

$$\boldsymbol{\alpha} = (\mathbf{K} + \gamma \mathbf{I})^{-1} \mathbf{y}. \quad (\text{B.6})$$

From now on, the OS-LSSVR model consists in applying the learning procedure of the KRLS model to solve iteratively the standard dual LSSVR optimization problem in Eq. (B.4). Thereby, consider the sequential training pairs of input-output

$$\mathcal{D}_t = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_t, y_t)\}, \quad (\text{B.7})$$

where the functional in Eq. (B.4) should be solved into a sequential learning scenario. Thus, one gets

$$J(\boldsymbol{\alpha}_t) = \|\mathbf{K}_t \boldsymbol{\alpha}_t - \mathbf{y}_t\|^2 + \gamma \boldsymbol{\alpha}_t^\top \mathbf{K}_t \boldsymbol{\alpha}_t, \quad (\text{B.8})$$

which, based on  $\mathbf{K}_t = \boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t$ , can also be written as

$$J(\boldsymbol{\alpha}_t) = \|\boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t \boldsymbol{\alpha}_t - \mathbf{y}_t\|^2 + \gamma \boldsymbol{\alpha}_t^\top \boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t \boldsymbol{\alpha}_t. \quad (\text{B.9})$$

From Engel *et al.* (2004), it is possible to write the following equalities:

$$\mathbf{w}_t = \boldsymbol{\Phi}_t \boldsymbol{\alpha}_t \approx \tilde{\boldsymbol{\Phi}}_t \mathbf{A}_t^\top \boldsymbol{\alpha}_t = \tilde{\boldsymbol{\Phi}}_t \tilde{\boldsymbol{\alpha}}_t, \quad (\text{B.10})$$

where  $\mathbf{A}_t = [\mathbf{a}_1 \ \mathbf{a}_2 \ \dots \ \mathbf{a}_t]^\top \in \mathbb{R}^{t \times m_t}$ . Then, Eq. (B.9) becomes

$$\begin{aligned} J(\tilde{\boldsymbol{\alpha}}_t) &= \|\tilde{\boldsymbol{\Phi}}_t^\top \tilde{\boldsymbol{\Phi}}_t \tilde{\boldsymbol{\alpha}}_t - \mathbf{y}_t\|^2 + \gamma \boldsymbol{\alpha}_t^\top \boldsymbol{\Phi}_t^\top \tilde{\boldsymbol{\Phi}}_t \tilde{\boldsymbol{\alpha}}_t, \\ &= \|\mathbf{A}_t \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t - \mathbf{y}_t\|^2 + \gamma \boldsymbol{\alpha}_t^\top \mathbf{A}_t \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t, \\ &= \|\mathbf{A}_t \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t - \mathbf{y}_t\|^2 + \gamma \tilde{\boldsymbol{\alpha}}_t^\top \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t, \end{aligned} \quad (\text{B.11})$$

where  $\tilde{\boldsymbol{\alpha}}_t \in \mathbb{R}^{m_t}$  is a reduced vector of  $m_t$  coefficients, and  $\tilde{\mathbf{K}}_t \in \mathbb{R}^{m_t \times m_t}$  is the corresponding reduced kernel matrix up instant  $t$ . The next step is to take the gradient vector  $\nabla_{\tilde{\boldsymbol{\alpha}}_t} J(\tilde{\boldsymbol{\alpha}}_t) = \partial J(\tilde{\boldsymbol{\alpha}}_t) / \partial \tilde{\boldsymbol{\alpha}}_t$  of Eq. (B.11) and equating it to zero, such as

$$\begin{aligned} \frac{\partial J(\tilde{\boldsymbol{\alpha}}_t)}{\partial \tilde{\boldsymbol{\alpha}}_t} &= 2(\mathbf{A}_t \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t - \mathbf{y}_t) \mathbf{A}_t \tilde{\mathbf{K}}_t + 2\gamma \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t, \\ &= \tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{A}_t \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t + \gamma \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_t - \tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{y}_t = 0, \\ \Rightarrow &(\tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{A}_t \tilde{\mathbf{K}}_t + \gamma \tilde{\mathbf{K}}_t) \tilde{\boldsymbol{\alpha}}_t = \tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{y}_t. \end{aligned} \quad (\text{B.12})$$

The expression in Eq. (B.12) corresponds to the minimization of the functional in Eq. (B.11), which yields the following solution:

$$\begin{aligned}
\tilde{\boldsymbol{\alpha}}_t &= (\tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{A}_t \tilde{\mathbf{K}}_t + \gamma \tilde{\mathbf{K}}_t)^{-1} \tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{y}_t, \\
&= [\tilde{\mathbf{K}}_t (\mathbf{A}_t^\top \mathbf{A}_t \tilde{\mathbf{K}}_t + \gamma \mathbf{I}_t)]^{-1} \tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{y}_t, \\
&= (\mathbf{A}_t^\top \mathbf{A}_t \tilde{\mathbf{K}}_t + \gamma \mathbf{I}_t)^{-1} \tilde{\mathbf{K}}_t^{-1} \tilde{\mathbf{K}}_t \mathbf{A}_t^\top \mathbf{y}_t, \\
&= (\mathbf{A}_t^\top \mathbf{A}_t \tilde{\mathbf{K}}_t + \gamma \mathbf{I}_t)^{-1} \mathbf{A}_t^\top \mathbf{y}_t, \\
&= [(\mathbf{A}_t^\top \mathbf{A}_t + \gamma \tilde{\mathbf{K}}_t^{-1}) \tilde{\mathbf{K}}_t]^{-1} \mathbf{A}_t^\top \mathbf{y}_t, \\
&= \tilde{\mathbf{K}}_t^{-1} (\mathbf{A}_t^\top \mathbf{A}_t + \gamma \tilde{\mathbf{K}}_t^{-1})^{-1} \mathbf{A}_t^\top \mathbf{y}_t.
\end{aligned} \tag{B.13}$$

Finally, by defining a matrix  $\mathbf{P}_t$  as

$$\mathbf{P}_t = (\mathbf{A}_t^\top \mathbf{A}_t + \gamma \tilde{\mathbf{K}}_t^{-1})^{-1}, \tag{B.14}$$

one can write the expression for  $\tilde{\boldsymbol{\alpha}}_t$  in Eq. (B.13) as

$$\tilde{\boldsymbol{\alpha}}_t = \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t \mathbf{A}_t^\top \mathbf{y}_t. \tag{B.15}$$

The next step requires the computation of the inverse matrices in Eqs. (B.14) and (B.15) iteratively using the RLS algorithm. For this purpose, we have two possible situations following the same recursive procedure of the KRLS algorithm, which are described below.

### B.1.1 Case 1 - Unchanged Dictionary

In this case,  $\delta_t \leq \nu$ , meaning that  $\boldsymbol{\phi}(\mathbf{x}_t)$  is approximately linearly dependent on the dictionary vectors. Hence,  $\mathbf{x}_t$  is not added to the dictionary ( $\mathcal{D}_t^{sv} = \mathcal{D}_{t-1}^{sv}$ ) and, consequently, the kernel matrix is not changed ( $\tilde{\mathbf{K}}_t = \tilde{\mathbf{K}}_{t-1}$ ).

Since  $\mathbf{a}_t$  needs to be computed (see Eq. (6.6)) to determine  $\delta_t$ , the matrix  $\mathbf{A}_t$  is built iteratively by the inclusion of  $\mathbf{a}_t$ , i.e.  $\mathbf{A}_t = [\mathbf{A}_{t-1}^\top \ \mathbf{a}_t]^\top$ . Thus, by defining a matrix  $\mathbf{B}_t$  as

$$\begin{aligned}
\mathbf{B}_t &= \mathbf{A}_t^\top \mathbf{A}_t + \gamma \tilde{\mathbf{K}}_t^{-1}, \\
&= \mathbf{A}_{t-1}^\top \mathbf{A}_{t-1} + \gamma \tilde{\mathbf{K}}_{t-1}^{-1} + \mathbf{a}_t \mathbf{a}_t^\top, \\
&= \mathbf{B}_{t-1} + \mathbf{a}_t \mathbf{a}_t^\top,
\end{aligned} \tag{B.16}$$

where  $\mathbf{A}_t^\top \mathbf{A}_t = \mathbf{A}_{t-1}^\top \mathbf{A}_{t-1} + \mathbf{a}_t \mathbf{a}_t^\top$ , one can apply the matrix inversion lemma to recursively compute the matrix  $\mathbf{P}_t$  as

$$\mathbf{P}_t = \mathbf{B}_t^{-1} = \mathbf{P}_{t-1} - \frac{\mathbf{P}_{t-1} \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1}}{1 + \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t}. \tag{B.17}$$

Also defining the gain vector  $\mathbf{q}_t$  as

$$\mathbf{q}_t = \frac{\mathbf{P}_{t-1}\mathbf{a}_t}{1 + \mathbf{a}_t^\top \mathbf{P}_{t-1}\mathbf{a}_t}, \quad (\text{B.18})$$

one gets

$$\mathbf{P}_t = \mathbf{P}_{t-1} - \mathbf{q}_t \mathbf{a}_t^\top \mathbf{P}_{t-1}. \quad (\text{B.19})$$

Finally, using the fact that  $\mathbf{A}_t^\top \mathbf{y}_t = \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} + \mathbf{a}_t \mathbf{y}_t$ , the OS-LS-SVR update rule for  $\tilde{\boldsymbol{\alpha}}_t$  can be written by

$$\begin{aligned} \tilde{\boldsymbol{\alpha}}_t &= \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t \mathbf{A}_t^\top \mathbf{y}_t, \\ &= \tilde{\mathbf{K}}_t^{-1} (\mathbf{P}_{t-1} - \mathbf{q}_t \mathbf{a}_t^\top \mathbf{P}_{t-1}) (\mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} + \mathbf{a}_t \mathbf{y}_t), \\ &= (\tilde{\mathbf{K}}_t^{-1} \mathbf{P}_{t-1} - \tilde{\mathbf{K}}_t^{-1} \mathbf{q}_t \mathbf{a}_t^\top \mathbf{P}_{t-1}) (\mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} + \mathbf{a}_t \mathbf{y}_t), \\ &= \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_{t-1} \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} + \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_{t-1} \mathbf{a}_t \mathbf{y}_t - \tilde{\mathbf{K}}_t^{-1} \mathbf{q}_t \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} - \tilde{\mathbf{K}}_t^{-1} \mathbf{q}_t \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t \mathbf{y}_t, \\ &= \tilde{\boldsymbol{\alpha}}_{t-1} + \tilde{\mathbf{K}}_t^{-1} \mathbf{a}_t \mathbf{y}_t (\mathbf{P}_{t-1} + \mathbf{q}_t \mathbf{a}_t^\top \mathbf{P}_{t-1}) - \mathbf{q}_t \mathbf{a}_t^\top \tilde{\boldsymbol{\alpha}}_{t-1}, \\ &= \tilde{\boldsymbol{\alpha}}_{t-1} + \tilde{\mathbf{K}}_t^{-1} \mathbf{a}_t \mathbf{y}_t \mathbf{P}_t - \mathbf{q}_t \mathbf{a}_t^\top \tilde{\boldsymbol{\alpha}}_{t-1}, \\ &= \tilde{\boldsymbol{\alpha}}_{t-1} + \tilde{\mathbf{K}}_t^{-1} (\mathbf{P}_t \mathbf{a}_t \mathbf{y}_t - \mathbf{q}_t \mathbf{a}_t^\top \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_{t-1}), \\ &= \tilde{\boldsymbol{\alpha}}_{t-1} + \tilde{\mathbf{K}}_t^{-1} (\mathbf{q}_t \mathbf{y}_t - \mathbf{q}_t \mathbf{a}_t^\top \tilde{\mathbf{K}}_t \tilde{\boldsymbol{\alpha}}_{t-1}), \\ &= \tilde{\boldsymbol{\alpha}}_{t-1} + \tilde{\mathbf{K}}_t^{-1} \mathbf{q}_t (\mathbf{y}_t - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}), \end{aligned} \quad (\text{B.20})$$

where the last equalities are based on  $\mathbf{q}_t = \mathbf{P}_t \mathbf{a}_t$  and  $\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) = \tilde{\mathbf{K}}_t \mathbf{a}_t$ .

### B.1.2 Case 2 - Updating the Dictionary

In this case, one gets  $\delta_t > \nu$ , implying that  $\mathbf{x}_t$  must be added to the dictionary, i.e.  $\mathcal{D}_t^{\text{sv}} = \mathcal{D}_{t-1}^{\text{sv}} \cup \{\mathbf{x}_t\}$  and  $m_t = m_{t-1} + 1$ . Hence, the kernel matrix must be updated accordingly.

In order to compute  $\tilde{\mathbf{K}}_t^{-1}$  recursively, consider the matrix  $\tilde{\mathbf{K}}_{t-1}$  and the information provided by the new sample to build  $\tilde{\mathbf{K}}_t$  as

$$\tilde{\mathbf{K}}_t = \begin{bmatrix} \tilde{\mathbf{K}}_{t-1} & \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \\ \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top & k_{tt} \end{bmatrix}, \quad (\text{B.21})$$

and also consider the following matrix identity:

$$\text{if } \mathbf{D}_t = \begin{bmatrix} \mathbf{D}_{t-1} & \mathbf{d} \\ \mathbf{d}^\top & d^* \end{bmatrix}, \quad (\text{B.22})$$



then

$$\mathbf{D}_t^{-1} = \begin{bmatrix} \mathbf{D}_{t-1}^{-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{1}{\Delta_d} \begin{bmatrix} -\mathbf{D}_{t-1}^{-1} \mathbf{d} \\ 1 \end{bmatrix} \begin{bmatrix} -\mathbf{D}_{t-1}^{-1} \mathbf{d} \\ 1 \end{bmatrix}^\top, \quad (\text{B.23})$$

where  $\mathbf{D}_t$  and  $\mathbf{d}_t$  are a square matrix and a vector, respectively, of appropriate sizes. Furthermore,  $d^*$  and  $\Delta_d$  are scalars such that  $\Delta_d = d^* - \mathbf{d}^\top \mathbf{D}_{t-1}^{-1} \mathbf{d}$ .

Then, assuming the equalities  $\mathbf{D}_{t-1} = \tilde{\mathbf{K}}_{t-1}$ ,  $\mathbf{d} = \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)$  and  $d^* = k_{tt}$  in Eq. (B.22), and applying them in Eq. (B.23), it is possible to recursively compute the inverse matrix  $\tilde{\mathbf{K}}_t^{-1}$  as

$$\begin{aligned} \tilde{\mathbf{K}}_t^{-1} &= \begin{bmatrix} \tilde{\mathbf{K}}_{t-1}^{-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{1}{\delta_t} \begin{bmatrix} -\tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \\ 1 \end{bmatrix} \begin{bmatrix} -\tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \\ 1 \end{bmatrix}^\top, \\ &= \begin{bmatrix} \tilde{\mathbf{K}}_{t-1}^{-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{1}{\delta_t} \begin{bmatrix} \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\mathbf{K}}_{t-1}^{-1} & -\tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \\ -\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\mathbf{K}}_{t-1}^{-1} & 1 \end{bmatrix}, \\ &= \begin{bmatrix} \tilde{\mathbf{K}}_{t-1}^{-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{1}{\delta_t} \begin{bmatrix} \mathbf{a}_t \mathbf{a}_t^\top & -\mathbf{a}_t \\ -\mathbf{a}_t^\top & 1 \end{bmatrix}, \end{aligned} \quad (\text{B.24})$$

where  $\delta_t = k_{tt} - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)$  and  $\mathbf{a}_t = \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)$ . The final expression for  $\tilde{\mathbf{K}}_t^{-1}$  is obtained adding the matrices in Eq. (B.24), which is given by

$$\tilde{\mathbf{K}}_t^{-1} = \frac{1}{\delta_t} \begin{bmatrix} \delta_t \tilde{\mathbf{K}}_{t-1}^{-1} + \mathbf{a}_t \mathbf{a}_t^\top & -\mathbf{a}_t \\ -\mathbf{a}_t^\top & 1 \end{bmatrix}. \quad (\text{B.25})$$

One can infer that the dimension of the matrix  $\tilde{\mathbf{K}}_t^{-1}$  increases due to the inclusion of sample  $\mathbf{x}_t$  into the dictionary. Thus, to avoid a mismatch in the calculation of  $\mathbf{P}_t$  in Eq. (B.14), the dimension of matrix  $\mathbf{A}_t^\top \mathbf{A}_t$  should be increased as follows:

$$\mathbf{A}_t^\top \mathbf{A}_t = \begin{bmatrix} \mathbf{A}_{t-1}^\top \mathbf{A}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}. \quad (\text{B.26})$$

From Eqs. (B.25), (B.26) and since the matrix  $\mathbf{P}_t$  is computed as  $\mathbf{P}_t = (\mathbf{A}_t^\top \mathbf{A}_t + \gamma \tilde{\mathbf{K}}_t^{-1})^{-1}$ , one can express the sum between parentheses as

$$\mathbf{A}_t^\top \mathbf{A}_t + \gamma \tilde{\mathbf{K}}_t^{-1} = \begin{bmatrix} \mathbf{A}_{t-1}^\top \mathbf{A}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} + \begin{bmatrix} \gamma \tilde{\mathbf{K}}_{t-1}^{-1} + \frac{\gamma}{\delta_t} \mathbf{a}_t \mathbf{a}_t^\top & -\frac{\gamma}{\delta_t} \mathbf{a}_t \\ -\frac{\gamma}{\delta_t} \mathbf{a}_t^\top & \frac{\gamma}{\delta_t} \end{bmatrix},$$

and, therefore,

$$\mathbf{P}_t = \begin{bmatrix} \mathbf{A}_{t-1}^\top \mathbf{A}_{t-1} + \gamma \tilde{\mathbf{K}}_{t-1}^{-1} + \frac{\gamma}{\delta_t} \mathbf{a}_t \mathbf{a}_t^\top & -\frac{\gamma}{\delta_t} \mathbf{a}_t \\ -\frac{\gamma}{\delta_t} \mathbf{a}_t^\top & \frac{\gamma}{\delta_t} \end{bmatrix}^{-1}. \quad (\text{B.27})$$

The next step is applying the same matrix identity of Eqs. (B.22) and (B.23), used for the calculation of  $\tilde{\mathbf{K}}_t^{-1}$ , to compute  $\mathbf{P}_t$  as

$$\mathbf{P}_t = \begin{bmatrix} \mathbf{P}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{1}{\Delta_b} \begin{bmatrix} -\mathbf{P}_{t-1}\mathbf{b} \\ 1 \end{bmatrix} \cdot \begin{bmatrix} -\mathbf{P}_{t-1}\mathbf{b} \\ 1 \end{bmatrix}^\top, \quad (\text{B.28})$$

where  $\Delta_b = b^* - \mathbf{b}^\top \mathbf{B}_{t-1}^{-1} \mathbf{b}$ ,  $b^* = 1 + \frac{\gamma}{\delta_t}$  and  $\mathbf{b} = -\frac{\gamma}{\delta_t} \mathbf{a}_t$ . Defining a constant  $c = \gamma/\delta_t$ , one gets

$$\Delta_b = 1 + \frac{\gamma}{\delta_t} - \left(-\frac{\gamma}{\delta_t} \mathbf{a}_t^\top\right) \mathbf{P}_{t-1} \left(-\frac{\gamma}{\delta_t} \mathbf{a}_t\right) = 1 + c - c^2 \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t, \quad (\text{B.29})$$

where  $c = \frac{\gamma}{\delta_t}$ . Then, the matrix  $\mathbf{P}_t$  can be computed by

$$\begin{aligned} \mathbf{P}_t &= \begin{bmatrix} \mathbf{P}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{1}{\Delta_b} \begin{bmatrix} c^2 \mathbf{P}_{t-1} \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1} & c \mathbf{P}_{t-1} \mathbf{a}_t \\ c \mathbf{a}_t^\top \mathbf{P}_{t-1} & 1 \end{bmatrix}, \\ &= \frac{1}{\Delta_b} \begin{bmatrix} \Delta_b \mathbf{P}_{t-1} + c^2 \mathbf{P}_{t-1} \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1} & c \mathbf{P}_{t-1} \mathbf{a}_t \\ c \mathbf{a}_t^\top \mathbf{P}_{t-1} & 1 \end{bmatrix}. \end{aligned} \quad (\text{B.30})$$

In order to compute the vector solution  $\tilde{\boldsymbol{\alpha}}_t = \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t \mathbf{A}_t^\top \mathbf{y}_t$ , one can still represent the product  $\mathbf{A}_t^\top \mathbf{y}_t$  by

$$\mathbf{A}_t^\top \mathbf{y}_t = \begin{bmatrix} \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} \\ y_t \end{bmatrix}. \quad (\text{B.31})$$

Thus, one uses Eqs. (B.30) and (B.31) to develop the expression for  $\tilde{\boldsymbol{\alpha}}_t$  as

$$\begin{aligned} \tilde{\boldsymbol{\alpha}}_t &= \tilde{\mathbf{K}}_t^{-1} \mathbf{P}_t \mathbf{A}_t^\top \mathbf{y}_t, \\ &= \tilde{\mathbf{K}}_t^{-1} \frac{1}{\Delta_b} \begin{bmatrix} \Delta_b \mathbf{P}_{t-1} + c^2 \mathbf{P}_{t-1} \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1} & c \mathbf{P}_{t-1} \mathbf{a}_t \\ c \mathbf{a}_t^\top \mathbf{P}_{t-1} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} \\ y_t \end{bmatrix}, \\ &= \tilde{\mathbf{K}}_t^{-1} \frac{1}{\Delta_b} \begin{bmatrix} \Delta_b \mathbf{P}_{t-1} \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} + c^2 \mathbf{P}_{t-1} \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} + c \mathbf{P}_{t-1} \mathbf{a}_t y_t \\ c \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} + y_t \end{bmatrix}, \\ &= \tilde{\mathbf{K}}_t^{-1} \frac{1}{\Delta_b} \begin{bmatrix} \Delta_b \mathbf{P}_{t-1} \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} + c^2 \mathbf{P}_{t-1} \mathbf{a}_t \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} + c \mathbf{P}_{t-1} \mathbf{a}_t y_t \\ y_t + c \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} \end{bmatrix}. \end{aligned} \quad (\text{B.32})$$

From Eq. (B.25), it is possible to rewrite Eq. (B.32) as

$$\tilde{\boldsymbol{\alpha}}_t = \frac{1}{\Delta_b \delta_t} \begin{bmatrix} \delta_t \tilde{\mathbf{K}}_{t-1}^{-1} + \mathbf{a}_t \mathbf{a}_t^\top & -\mathbf{a}_t \\ -\mathbf{a}_t^\top & 1 \end{bmatrix} \begin{bmatrix} \Delta_b \mathbf{P}_{t-1} \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} + c^2 \mathbf{P}_{t-1} \mathbf{a}_t \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} + c \mathbf{P}_{t-1} \mathbf{a}_t y_t \\ y_t + c \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} \end{bmatrix},$$

where the matrix product can be extended to

$$\begin{bmatrix} (\delta_t \tilde{\mathbf{K}}_{t-1}^{-1} + \mathbf{a}_t \mathbf{a}_t^\top) (\Delta_b \mathbf{P}_{t-1} \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} + c^2 \mathbf{P}_{t-1} \mathbf{a}_t \tilde{\mathbf{k}}_{t-1}^\top \tilde{\boldsymbol{\alpha}}_{t-1} + c \mathbf{P}_{t-1} \mathbf{a}_t y_t) - \mathbf{a}_t (y_t + c \tilde{\mathbf{k}}_{t-1}^\top \tilde{\boldsymbol{\alpha}}_{t-1}) \\ -\mathbf{a}_t^\top (\Delta_b \mathbf{P}_{t-1} \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} + c^2 \mathbf{P}_{t-1} \mathbf{a}_t \tilde{\mathbf{k}}_{t-1}^\top \tilde{\boldsymbol{\alpha}}_{t-1} + c \mathbf{P}_{t-1} \mathbf{a}_t y_t) + y_t + c \tilde{\mathbf{k}}_{t-1}^\top \tilde{\boldsymbol{\alpha}}_{t-1} \end{bmatrix} \quad (\text{B.33})$$

where, for the sake of space reduction, it was used the representation  $\tilde{\mathbf{k}}_{t-1}$  instead  $\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)$ . Finally, the OS-LSSVR model solution can be expressed by

$$\tilde{\boldsymbol{\alpha}}_t = \frac{1}{\Delta_b \delta_t} \begin{bmatrix} \tilde{\boldsymbol{\alpha}}_t^{(1)} \\ \tilde{\alpha}_t^* \end{bmatrix}, \quad (\text{B.34})$$

where the vector  $\tilde{\boldsymbol{\alpha}}_t^{(1)}$  and the scalar  $\tilde{\alpha}_t^*$  can be computed, respectively, by algebraically developing the upper and lower terms in Eq. (B.33), as

$$\begin{aligned} \tilde{\boldsymbol{\alpha}}_t^{(1)} &= \Delta_b \delta_t \tilde{\mathbf{K}}_{t-1}^{-1} \mathbf{P}_{t-1} \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} + \delta_t c^2 \tilde{\mathbf{K}}_{t-1}^{-1} \mathbf{P}_{t-1} \mathbf{a}_t \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} + \delta_t c \tilde{\mathbf{K}}_{t-1}^{-1} \mathbf{P}_{t-1} \mathbf{a}_t y_t \\ &+ \Delta_b \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} + c^2 \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} + c \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t y_t \\ &- \mathbf{a}_t (y_t + c \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}), \\ &= \Delta_b \delta_t \tilde{\boldsymbol{\alpha}}_{t-1} + \delta_t c^2 \tilde{\mathbf{K}}_{t-1}^{-1} \mathbf{P}_{t-1} \mathbf{a}_t \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} + \gamma \tilde{\mathbf{K}}_{t-1}^{-1} \mathbf{P}_{t-1} \mathbf{a}_t y_t \\ &+ \Delta_b \mathbf{a}_t \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} + c^2 \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} + c \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t y_t \\ &- \mathbf{a}_t y_t - c \mathbf{a}_t \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}, \\ &= \Delta_b \delta_t \tilde{\boldsymbol{\alpha}}_{t-1} + \gamma \tilde{\mathbf{K}}_{t-1}^{-1} \mathbf{P}_{t-1} \mathbf{a}_t (y_t + c \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}) \\ &+ \mathbf{a}_t (\Delta_b \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} - c \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} - y_t) + c \mathbf{a}_t \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t (y_t + c \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}), \\ &= \Delta_b \delta_t \tilde{\boldsymbol{\alpha}}_{t-1} + \mathbf{a}_t ((\Delta_b - c) \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} - y_t) \\ &+ \mathbf{P}_{t-1} \mathbf{a}_t (y_t + c \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}) (\gamma \tilde{\mathbf{K}}_{t-1}^{-1} + c \mathbf{a}_t \mathbf{a}_t^\top), \\ &= \Delta_b \delta_t \tilde{\boldsymbol{\alpha}}_{t-1} - \mathbf{a}_t (y_t - (\Delta_b - c) \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}) \\ &+ \mathbf{P}_{t-1} \mathbf{a}_t (y_t + c \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}) (\gamma \tilde{\mathbf{K}}_{t-1}^{-1} + c \mathbf{a}_t \mathbf{a}_t^\top), \end{aligned} \quad (\text{B.35})$$

and

$$\begin{aligned} \tilde{\alpha}_t^* &= -\Delta_b \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{A}_{t-1}^\top \mathbf{y}_{t-1} - c^2 \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} - c \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t y_t + y_t + c \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}, \\ &= -\Delta_b \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} + y_t + c \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} - c \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t (y_t + c \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}), \\ &= y_t + (c - \Delta_b) \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} - c \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t (y_t + c \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}), \\ &= y_t - (\Delta_b - c) \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1} - c \mathbf{a}_t^\top \mathbf{P}_{t-1} \mathbf{a}_t (y_t + c \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\boldsymbol{\alpha}}_{t-1}). \end{aligned} \quad (\text{B.36})$$

Thereby, once the parameter vector  $\tilde{\boldsymbol{\alpha}}_t$  in Eq. (B.34) has been updated, the sparse resulting solution for the OS-LSSVR model is given by

$$\hat{y}_t = f(\mathbf{x}) = \sum_{m=1}^{m_t} \tilde{\alpha}_m k(\mathbf{x}, \mathbf{x}_m) = \tilde{\mathbf{k}}_{m_t}(\mathbf{x})^\top \tilde{\boldsymbol{\alpha}}_t. \quad (\text{B.37})$$

## APPENDIX C – BOUND ON GENERALIZATION ERROR - OS-LSSVR MODEL

This appendix presents an analysis of the generalization error bound applicable to the proposed OS-LSSVR model. For this purpose, one follows a similar strategy adopted in Engel *et al.* (2003) for the KRLS model, extending it to the OS-LSSVR proposal.

### C.1 A Generalization Bound

Let  $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$  be a set of IID samples drawn from some distribution  $P(\mathbf{X}, \mathbf{Y})$ . In the OS-LSSVR case, the loss function is expressed by

$$\ell_f(\mathbf{x}, y) = \ell(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2 + \gamma \|\mathbf{w}\|^2, \quad (\text{C.1})$$

where  $f$  is given by  $f(\mathbf{x}) = \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}) \rangle$ . In addition, let

$$\mathcal{L}_{\mathcal{F}} = \{\ell_f(\mathbf{x}, y) : f \in \mathcal{F}\}, \quad (\text{C.2})$$

be a class of functions (defined on  $\mathcal{D}$ ) and let  $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_N)$  be a sequence of IID random variables assuming the values  $\{-1, +1\}$ , such as  $P(\sigma_n = 1) = P(\sigma_n = -1) = 1/2$ . According to Vaart and Wellner (1996), the empirical Rademacher complexity of  $\mathcal{F}$  is defined as

$$\hat{R}_N(\mathcal{F}) = \mathbf{E}_{\boldsymbol{\sigma}} \sup_{f \in \mathcal{F}} \left\{ \frac{1}{N} \sum_{n=1}^N \sigma_n f(\mathbf{x}_n) \right\}, \quad (\text{C.3})$$

where the Rademacher complexity is given by  $R_N(\mathcal{F}) = \mathbf{E} \hat{R}_N(\mathcal{F})$ .

As in the KRLS case, one should set  $L(f) = \mathbf{E}_{X,Y} \ell(Y, f(X))$  and  $\hat{L}(f) = \hat{\mathbf{E}}_N \ell(Y, f(X))$ . Then, the bound for the expected loss can be obtained by means of the following theorem (MEIR; ZHANG, 2003; ENGEL *et al.*, 2003):

**Theorem C.1** *Let  $F$  be a class of functions mapping from a domain  $\mathcal{X}$  to  $\mathbb{R}$ , and let  $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ ,  $\mathbf{x}_n \in \mathcal{X}$ ,  $y_n \in \mathbb{R}$ , be independently selected according to a probability measure  $P$ . Assume there exists a positive real number  $M$  such that for all positive  $\lambda$*

$$\log \mathbf{E}_{X,Y} \sup_{f \in \mathcal{F}} \cosh(2\lambda \ell(Y, f(X))) \leq \lambda^2 M^2 / 2. \quad (\text{C.4})$$

*Then, with probability at least  $1 - \delta$  over the samples of length  $N$ , every  $f \in \mathcal{F}$  satisfies*

$$L(f) \leq \hat{L}(f) + 2R_N(\mathcal{L}_{\mathcal{F}}) + M \sqrt{\frac{2 \log(1/\delta)}{N}}. \quad (\text{C.5})$$

In addition, the empirical Rademacher complexity  $\hat{R}_N$  is limited according to the following result (MEIR; ZHANG, 2003; ENGEL *et al.*, 2003):

**Lemma C.2** Consider the class of functions  $\mathcal{F}_A = \{f(\mathbf{x}) = \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}) \rangle : (1/2)\|\mathbf{w}\|^2 \leq A\}$ . Then

$$\hat{R}_N(\mathcal{F}_A) \leq \sqrt{\frac{2A}{N} \left( \frac{1}{N} \sum_{n=1}^N \|\boldsymbol{\phi}(\mathbf{x}_n)\|^2 \right)}$$

where one assumes initially that  $(1/2)\|\mathbf{w}\|^2 \leq A$  (we use  $\mathbf{w} \in \Omega_A$  to indicate this inequality).

Then, assuming that  $k(\mathbf{x}, \mathbf{x}') \leq B$  for all  $\mathbf{x}$  and  $\mathbf{x}'$ , and the expectation with respect to the product distribution  $P^N$  over the samples  $S$  by  $\mathbf{E}_S$ , it is possible to write

$$R_N(\mathcal{L}_{\mathcal{F}_{sd}}) = \mathbf{E}_S \mathbf{E}_\sigma \sup_{\mathbf{w} \in \Omega_A} \frac{1}{N} \sum_{n=1}^N \sigma_n [(y_n - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_n) \rangle)^2 + \gamma \|\mathbf{w}\|^2], \quad (\text{C.6})$$

$$= \mathbf{E}_S \mathbf{E}_\sigma \sup_{\mathbf{w} \in \Omega_A} \frac{1}{N} \sum_{n=1}^N \sigma_n [\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_n) \rangle^2 - 2\langle \mathbf{w}, y_n \boldsymbol{\phi}(\mathbf{x}_n) \rangle + \gamma \langle \mathbf{w}, \mathbf{w} \rangle], \quad (\text{C.7})$$

$$\leq \mathbf{E}_S \mathbf{E}_\sigma \sup_{\mathbf{w} \in \Omega_A} \frac{1}{N} \sum_{n=1}^N \sigma_n (\sqrt{2AB} + 2|y_n|) \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_n) \rangle + \sigma_n 2\gamma A. \quad (\text{C.8})$$

Based on the fact that  $\mathbf{E} \sigma_n = 0$  for any  $n$ , the last term in Eq. (C.8) is null, since

$$\mathbf{E}_S \mathbf{E}_\sigma \sup_{\mathbf{w} \in \Omega_A} \frac{1}{N} \sum_{n=1}^N \sigma_n 2\gamma A = 2\gamma A \mathbf{E}_S \mathbf{E}_\sigma \frac{1}{N} \sum_{n=1}^N \sigma_n = 0. \quad (\text{C.9})$$

Based on the procedure introduced in Engel *et al.* (2003)(see e.g. page 20) for the KRLS algorithm, the derived bound on  $R_N$  for the proposed OS-LSSVR model is given by

$$R_N(\mathcal{L}_{\mathcal{F}_{sd}}) \leq \frac{2\sqrt{2AB} + 4\sqrt{AB\mathbf{E}[\mathbf{Y}^2]}}{\sqrt{N}}, \quad (\text{C.10})$$

expressed in terms of the parameters  $A$  and  $B$ . In order to get rid of the dependence on the parameter  $A$ , one proceeds with the same mathematical manipulations as described in Meir and Zhang (2003), Engel *et al.* (2003) to establish the following bound of the expected loss for the OS-LSSVR model:

$$\mathbf{E}((Y - f_w(X))^2 + \gamma \|\mathbf{w}\|^2) \leq \frac{1}{N} \sum_{n=1}^N (y_n - f_w(\mathbf{x}_n))^2 + \gamma \tilde{g}(\mathbf{w})$$

$$\begin{aligned}
& + \frac{4\sqrt{2B}\tilde{g}(\mathbf{w}) + 8\sqrt{B\tilde{g}(\mathbf{w})\mathbf{E}[Y^2]}}{\sqrt{N}} \\
& + M\sqrt{\frac{4\log\log_2(2\tilde{g}(\mathbf{w})/g_0) + 2\log(1/\delta)}{N}}, \tag{C.11}
\end{aligned}$$

with

$$\tilde{g}(\mathbf{w}) = 2\max((1/2)\|\mathbf{w}\|^2, g_0), \tag{C.12}$$

where  $g_0$  is a constant.

The first two terms on the right-hand side of Eq. (C.11) denote the empirical loss over the training data. The third term corresponds to the Rademacher complexity, which was demonstrated in Eqs. (C.6)-(C.10) to be the same one obtained for the KRLS algorithm. Finally, the last term is common to both KRLS and OS-LSSVR models, since it does not depend on the considered loss function  $\ell(y, f(\mathbf{x}))$ .

Therefore, one should note that the expression in Eq. (C.11) is similar to that one reported in Engel *et al.* (2003) (Theorem 4.3), but with a regularization term in both sides of the inequality. Then, the generalization error bound is controlled by the parameter  $\gamma$ . If we set  $\gamma = 0$  (i.e. no regularization term) in Eq. (C.11), one gets the original form as derived for the KRLS algorithm.