Kernel-Based Adaptive Estimation: Multidimensional and State-Space Approaches

by
Felipe A. Tobar

A Thesis submitted in fulfilment of requirements for the degree of
Doctor of Philosophy of Imperial College London

Communication and Signal Processing Group
Department of Electrical and Electronic Engineering
Imperial College London
2014

Dedicated to my parents

Abstract

Several disciplines, from engineering to social sciences, critically depend on adaptive signal estimation to either remove observation noise (filtering), or to approximate quantities before they become available (prediction). When an optimal estimator cannot be expressed in closed form, e.g. due to model uncertainty or complexity, machine learning algorithms have proven to successfully *learn* a model which captures rich relationships from large datasets. This thesis proposes two novel approaches to signal estimation based on support vector regression (SVR): high-dimensional kernel learning (HDKL) and kernel-based state-spaces modelling (KSSM).

In real-world applications, signal dynamics usually depend on both time and the value of the signal itself. The HDKL concept extends the standard, single-kernel, SVR estimation approach by considering a feature space constructed as an ensemble of real-valued feature spaces; the resulting feature space provides highly-localised estimation by averaging the *subkernels* estimates and is well-suited for multichannel signals, as it captures interchannel data-dependency. This thesis then provides a rigorous account for the existence of such higher-dimensional RKHS and their corresponding kernels by considering the complex-, quaternion- and vector-valued cases.

Current kernel adaptive filters employ nonlinear autoregressive models and express the current value of the signal as a function of past values with added noise. The motivation for the second main contribution of this thesis is to depart from this class of models and propose a state-space model designed using kernels (KSSM), whereby the signal of interest is a latent state and the observations are noisy measurements of the hidden process. This formulation allows for jointly estimating the signal (state) and the parameters, and is robust to observation noise. The posterior density of the kernel mixing parameters is then found in an unsupervised fashion using Markov chain Monte Carlo and particle filters, and both the offline and online cases are addressed.

The capabilities of the proposed algorithms are initially illustrated by simulation examples using synthetic data in a controlled environment. Finally, both the HDKL and the KSSM approaches are validated in the estimation of real-world signals including body-motion trajectories, bivariate wind speed, point-of-gaze location, and national grid frequency.

Acknowledgments

I have been very fortunate to be part of the Signal Processing and Communications Group (CSP) at Imperial College London for almost four years. I am deeply grateful to my supervisor, Prof. Danilo P. Mandic, for encouraging rigour in both content and presentation of my work, for always taking the time to discuss research ideas and read my manuscripts, and for the financial support he provided. Danilo has not only been my mentor in the fascinating field of Adaptive Signal Processing, he has also shared his expertise with me on more worldly arts such as bicycle and motorcycle riding, and life in London.

I am indebted to Prof. Sun-Yuan Kung, Princeton University, for his advice on multikernel learning, and to Prof. Petar M. Djurić, Stony Brook University, for guidance on sequential Monte Carlo methods; the collaborations with Profs Kung and Djurić have greatly benefitted this thesis and its related publications. I am also thankful to my examiners, Prof. Jose C. Principe, University of Florida, and Prof. Anthony G. Constantinides, Imperial College London, for enlightening comments and encouraging feedback.

From Imperial, I would also like to thank to my fellow lab mates in CSP, David Looney and Jon Oñativia, for their friendship and company in academic matters, such as the quest for elegance when producing research material and the infinite proofreading, as well as in not-so-academic ones, such as the *sobre mesa* coffees and some other types of beverages around South Kensington. Special thanks to my Chilean friends: Luis Pizarro, Javier Correa and Diego Oyarzún (at Imperial), and Diego Muñoz-Carpintero (at Oxford); they have made the stay away from home much easier.

From Universidad de Chile, I want to express my sincere gratitude to Dr. Marcos E. Orchard for his guidance prior to my PhD studies and Prof. Manuel A. Duarte before that, they encouraged my initiation into research and remain a source of guidance on academic life. I am also grateful to Chile's National Commission for Scientific and Technological Research (CONICYT) for funding my PhD studies.

Finally, I want to thank my parents, for their infinite love and support in every possible manner; my sister, for providing wise pieces of advice on life matters; and Pía, for always being by my side.

Felipe A. Tobar.
London, August 2014.

Contents

Abstract 3

Acknowledgments 4

Contents 5

List of Figures 9

List of Tables 13

Statement of Originality 14

Copyright Declaration 15

Publications 16

Abbreviations and Symbols 17

Chapter 1. Introduction 18
1.1 Scope of the Thesis: Kernels and Signal Estimation 19
1.2 Historical Background 21
1.3 Contributions of the Thesis 23
1.4 Organisation 25

Chapter 2. Kernel Learning 27
2.1 Support Vector Machines 27
2.2 Support Vector Regression 29
2.3 Scalar-Valued Kernels 32
2.3.1 Radial Basis Functions (RBF) 32
2.3.2 Polynomial Kernels 34

I High-Dimensional Kernel Adaptive Filters 36

Chapter 3. Linear and Kernel Adaptive Filtering: A Unified View 37

3.1 Problem Formulation . . . 37
3.2 Linear Adaptive Filters . . . 38
3.2.1 The Wiener Filter . . . 38
3.2.2 Least Squares and Ridge Regression . . . 40
3.2.3 The Least Mean Square Algorithm . . . 42
3.2.4 Recursive Least Squares . . . 44
3.2.5 Example: Adaptive Filters for System Identification . . . 45
3.3 Kernel Adaptive Filtering . . . 46
3.3.1 Sparsification Critera . . . 48
3.3.2 Kernel Ridge Regression . . . 50
3.3.3 Kernel Least Mean Square . . . 52
3.3.4 Kernel Recursive Least Squares . . . 55

Chapter 4. Hypercomplex Kernels **56**
4.1 Complex-Valued Kernels . . . 56
4.1.1 The Complex-Valued Gaussian Kernel . . . 57
4.1.2 Complexification of Real-Valued Kernels . . . 58
4.2 Quaternion-Valued Kernels . . . 60
4.2.1 Background on Quaternion Vector Spaces . . . 61
4.2.2 Quaternion Reproducing Kernel Hilbert Spaces . . . 64
4.2.3 Design of Quaternion-Valued Mercer Kernels . . . 68
4.3 Examples . . . 72
4.3.1 Systems with Correlated and Uncorrelated Noise: Quaternion Linear Kernel . . . 72
4.3.2 Nonlinear Channel Equalisation: Quaternion Gaussian Kernel . . . 73
4.4 Discussion . . . 77

Chapter 5. Vector-Valued Kernels **78**
5.1 A Hilbert Space of Vector-Valued Functions with a Reproducing Property 78
5.2 Multikernel Regression . . . 81
5.2.1 Multikernel Ridge Regression . . . 81
5.2.2 Multikernel Least Mean Square . . . 83
5.3 Implementation of Multikernel Algorithms . . . 86
5.3.1 Convergence Properties of Multikernel Methods . . . 86
5.3.2 Computational Complexity of Proposed Algorithms . . . 87
5.4 Examples . . . 88
5.4.1 Nonlinear Function Approximation: Multikernel Ridge Regression 88
5.4.2 Prediction of Nonlinear Signals: Multikernel Least Mean Square . 89
5.5 Discussion . . . 92

II Kernel-Based State-Space Models 94

Chapter 6. Bayesian Filtering and Monte Carlo Methods 95
6.1 Bayesian Filtering . . . 96
6.1.1 Filtering Equations . . . 96
6.1.2 Learning the Dynamical Model . . . 97
6.2 Monte Carlo Sampling . . . 98
6.2.1 Importance Sampling . . . 99
6.2.2 Markov Chain Monte Carlo (MCMC) . . . 100
6.3 Sequential Monte Carlo Methods . . . 101

Chapter 7. Unsupervised State-Space Modelling 104
7.1 Kernel State Space Models . . . 105
7.1.1 Offline Learning of the State-Transition Function . . . 106
7.1.2 Choice of Support Vectors and Kernel Width . . . 107
7.2 Online Update of the KSSM Model . . . 109
7.2.1 Model Design by Artificial Evolution of Parameters . . . 109
7.2.2 Recursive Sampling From $p(\mathbf{a}_t|y_{1:t+1})$ using MCMC and SMC . . . 110
7.2.3 Online Sparsification and Choice of the Prior $p(\mathbf{a}_t|\mathbf{a}_{t-1})$ . . . 111
7.2.4 Multivariate KSSM for Autoregressive Modelling . . . 112
7.3 Examples . . . 113
7.3.1 Offline Estimation of a Nonlinear State-Transition Function . . . 113
7.3.2 Prediction of a Nonlinear Prediction: KSSM and Artificial Evolution 115
7.3.3 Identification of a Time-Varying State-Transition Function: KSSM and MCMC . . . 117
7.4 Discussion . . . 118

III Real-World Simulation Examples and Concluding Remarks 122

Chapter 8. Experimental Validation 123
8.1 Bivariate Wind Speed . . . 123
8.1.1 One-Step Prediction using Complex-Valued Kernels and KLMS . . 124
8.1.2 Long-term Prediction using Multikernel LMS and Presence-Based Sparsification . . . 125
8.2 Bodysensor Signals . . . 128
8.2.1 Data Acquisition and Preprocessing . . . 129
8.2.2 Signal Tracking using Quaternion Cubic Kernels and Ridge Regression . . . 130
8.2.3 Signal Tracking using Multikernel Ridge Regression . . . 133
8.3 Prediction of Bivariate Point-of-Gaze using KSSM and Artificial Evolution 134

8.4 Prediction of Power-Grid Frequency using KSSM and MCMC 136

Chapter 9. Conclusions **140**
9.1 Hypercomplex Kernels 140
9.2 Multikernel Learning 141
9.3 Learning and Prediction using Kernel State-Space Models (KSSM) 141
9.4 Experimental Validation of the Proposed Methods 142
9.5 Future Research Directions 142

Bibliography **144**

Appendix A. Additional Material **153**
A.1 Algebraic Identities for the Trace Operator 153
A.2 Basis of Cubic Polynomials in $\langle \mathbf{x}, \mathbf{y} \rangle_{\Re}$ 154
A.3 Quaternion Widely-Linear Ridge Regression 155
A.4 Proof of Lemma 7 156

List of Figures

2.1 Classification of linearly separable data. The left plot shows how an infinite number of planes classify the data samples, while the right plot shows the (unique) maximum margin separating plane and the support vectors (with red shadows). 28
2.2 Classification of non-linearly-separable data. The left plot shows the original samples in $\mathbb{R}^2$ corresponding to classes red and blue, while the right plot shows both original and feature samples in $\mathbb{R}^3$ with the corresponding separating plane. 29
2.3 RBF kernels defined on $\mathbb{R}^2$. The triangular and Epanechnikov kernels are rotation invariant and have finite support, the Gaussian kernel is rotation-invariant and has infinite support, and the Mahalanobis kernel is rotation-sensitive and has infinite support. 34

3.1 Estimation of the process $\mathbf{d}_t$ as a projection onto the space of linear estimators $\mathbf{y}_t = \theta \mathbf{x}_t, \theta \in \Theta$. 39
3.2 Performance of adaptive filters for system identification: Norm of the misalignment (parametric error) as a function of iteration number. 46
3.3 Diagram of an adaptive filter that is nonlinear in the input ($\mathbf{x}$) and linear in the parameters (A). This filter is constructed by applying a nonlinear transformation on the input and then performing adaptive filtering using the transformed sample. 47
3.4 Kernel adaptive filter. The input is mapped to an infinite-dimensional feature space, and then linear adaptive filtering is performed on the features. This requires the update of both mixing parameters and support vectors. . 48

4.1 Contour plot of real and imaginary parts of the complex Gaussian kernel $K_{\mathbb{C}G}$ in eq. (4.1) and $\sigma^2 = 10^3$ 58
4.2 Independent complex kernel in eq. (4.1) generated from a real-valued Gaussian with kernel width $\sigma^2 = 10^3$. 60

4.3 Real (left) and i-imaginary (right) parts of K_{QP}. The colourmap is dark blue for $-13 \cdot 10^3$, white for the interval $[-10, 10]$, and red for $13 \cdot 10^3$ with a logarithmic RGB interpolation. 70
4.4 Real (left) and i-imaginary (right) parts of K_{QG}. The colourmap is dark blue for $-7 \cdot 10^{-4}$, white for 0, and red for $7 \cdot 10^4$, with a logarithmic RGB interpolation. 72
4.5 MSE $\pm$ 0.5 standard deviations for kernel algorithms as a function of the number of support vectors in the estimation of both uncorrelated (top, $A = 0.6808, B = 0.1157$) and correlated (bottom, $A = 0.6808 + i0.07321 + j0.6222 - k0.2157, B = 0.1157 + i0.1208 + j0.8425 - k0.5121$) AR(1) processes. 73
4.6 Choice of kernel parameters (red) A_R and A_Q based on the second moment of the kernel evaluations and histograms corresponding to the chosen parameters. 75
4.7 Training MSE of ridge regression algorithms for channel equalisation. . . 76
4.8 Validation MSE of ridge regression algorithms for channel equalisation. . 76

5.2 Training error norm for different values of the kernel width. The global minimum has the value of 3.11 and is reached for $\sigma = 1.48$. 88
5.1 A nonlinear function and support vectors. 88
5.3 Nonlinear function estimation using mono- and multi-kernel ridge regression algorithms. 89
5.4 Trivariate original signal and KLMS estimates. 90
5.5 Time varying magnitude of the weight matrix for all the three implemented kernel algorithms. 91
5.6 Averaged MSE and dictionary size (in number of samples) over 30 trials for the kernel algorithms. 92

7.1 Observed process for the system in eq. (7.24). 114
7.2 Original state-transition function, state samples and kernel-based approximation. 115
7.3 Value of the posterior $p(\mathbf{a}|y_{1:40})$ for the supervised solution and the samples generated by the proposed method. 116
7.4 Prediction using the proposed KSSM. 116
7.5 The state-transition function of the system in (7.26) is time-varying between $t = 30$ and $t = 60$, and constant for $t < 30$ (system stable) and for $t > 60$ (system unstable). 118
7.6 KSSM estimate (mean and standard deviation) of the time-varying state-transition function in (7.26) for $t = 30$ and $t = 90$. The true state samples are plotted with red borders for the regions $[1, 30]$ (green fill) and $[61, 90]$ (blue fill). 119

7.7 Filtered state signal using the KSSM and SIR particle filter. The original state is shown in blue and the posterior mean in red, with a one-standard-deviation confidence interval in light red. 120

8.1 Original north-south wind speed component and KLMS estimates. 124
8.2 Input sample deviation for the low, medium, and high dynamics wind regime. 125
8.3 Original bivariate wind signal and its KLMS estimates for the low dynamics region. 126
8.4 Weight magnitudes for the kernels within the MKLMS algorithm, evaluated for mixed regime wind, using the proposed adaptive sparsification criteria. 126
8.5 Dictionary size for all kernel algorithms for the prediction of mixed-regime wind using the proposed adaptive sparsification criteria. 127
8.6 Averaged MSE and dictionary size over 30 trials of combined low, medium and high dynamics regions. 128
8.7 Inertial body sensor setting. **[Left]** Fixed coordinate system (red), sensor coordinate system (blue) and Euler angles (green). **[Right]** A 3D inertial body sensor at the right wrist. 129
8.8 Raw angle measurements and considered features. **[Top]** Original discontinuous angle recording and **[Bottom]** the corresponding continuous sine and cosine mapping. Observe that in the right (left) circle only cosine (sine) preserves the dynamics of the angle signal accurately. 130
8.9 Body sensor signal tracking: Angle features $(\sin\theta, \cos\theta)$ and KRR estimates. 131
8.10 Performance of KRR algorithms for body sensor signal tracking as a function of the number of support vectors. 132
8.11 Computation time of KRR algorithms for body sensor signal tracking. . . 132
8.12 Average running time and standard deviation for kernel algorithms as a function of the number of support vectors. 133
8.13 MSE and standard deviation for kernel algorithms as a function of number of support vectors. 134
8.14 Performances of kernel algorithms for gaze prediction. 135
8.15 Kernel algorithms prediction and original gaze signal for KNLMS, KRLS and KSSM. 135
8.16 Original signal and kernel SSM densities for the prediction of gaze signals. 136
8.17 Online estimation of the state-transition function for $t \in [2, 7]$. The hidden state samples are shown in red. 137
8.18 Kernel predictions of the UK National Grid frequency. The original signal is shown in red, the KLMS prediction in red, the KSSM prediction in blue, and the one-standard-deviation confidence interval in light blue. 138

8.19 Prediction performance of the considered algorithms. 139

List of Tables

5.1 Computational complexity of weight computation for KRR algorithms . . 87
5.2 Estimation error for mono- and multi-kernel ridge regression algorithms. 89
5.3 Averaged prediction gains and running times for the Gaussian KLMS, triangular KLMS and MKLMS. 91

7.1 Prediction performance of kernel algorithms 117

8.1 $10\log_{10}$(MSE) for wind prediction (last 500 samples). 124
8.2 Averaged prediction gains for the low-fitted KLMS (LF), medium-fitted KLMS (MF), high-fitted KLMS (HF), and MKLMS for all three dynamic regions 127
8.3 Averaged running time (in seconds) for the low-fitted KLMS (LF), medium-fitted KLMS (MF), high-fitted KLMS (HF), and MKLMS for all three dynamic regions 128

Statement of Originality

I declare that this is an original thesis and it is entirely based on my own work. I acknowledge the sources in every instance where I used the ideas of other writers. This thesis was not and will not be submitted to any other university or institution for fulfilling the requirements of a degree.

Copyright Declaration

Publications

The following publications support the material given in this thesis.

Submitted

- **F. Tobar and D. Mandic**, "High-Dimensional Kernel Regression: A Guide for Practitioners," Book chapter, Festschrift.
- **F. Tobar, P. Djurić and D. Mandic**, "Unsupervised state-space modelling using reproducing kernels," *IEEE Transactions on Signal Processing*, 2014.

Peer-reviewed journals

- **F. Tobar, S.-Y. Kung, and D. Mandic**, "Multikernel least mean square algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 2, pp. 265-277, 2014.
- **F. Tobar and D. Mandic**, "Quaternion reproducing kernel Hilbert spaces: Existence and uniqueness conditions," *IEEE Transactions on Information Theory*, vol. 60, no. 9, pp. 5736-5749, 2014.

Peer-reviewed conferences

- **F. Tobar, A. Kuh, and D. Mandic**, "A novel augmented complex valued kernel LMS," in *Proc. of the 7th IEEE Sensor Array and Multichannel Signal Processing Workshop*, 2012, pp. 473-476.
- **F. Tobar and D. Mandic**, "Multikernel least squares estimation," in *Proc. of the Sensor Signal Processing for Defence Conference*, 2012, pp. 1-5.
- **F. Tobar and D. Mandic**, "The quaternion kernel least squares," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6128-6132.
- **F. Tobar and D. Mandic**, "A particle filtering based kernel HMM predictor," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014, pp. 8019-8023.

Abbreviations and Symbols

RKHS	Reproducing kernel Hilbert Space
KLMS	Kernel least mean square
KRLS	Kernel recursive least squares
KRR	Kernel ridge regression
KSSM	Kernel state space model
MSE	Mean square error
MMSE	Minimum mean square estimate
QRKHS	Quaternion Reproducing kernel Hilbert Space
RBF	Radial basis function
SSM	State space model
SVM	Support vector machine
SVR	Support vector regression
$\{\mathbf{A}\}_i$	i^{th} entry of vector $\mathbf{A}$
$\mathbf{A}_{1:t} = \{\mathbf{A}_1, \ldots, \mathbf{A}_t\}$	Collection of samples $\mathbf{A}_1, \ldots, \mathbf{A}_t$
$\mathcal{D} = \{\mathbf{s}^1, \ldots, \mathbf{s}^N\}$	Dictionary (set of support vectors)
N	Number of support vectors
N_p	Number of particles
$\mathbb{R}, \mathbb{C}, \mathbb{H}$	Real field, complex field and quaternion ring respectively.
$\Re\{a\}$ and $\Im\{a\}$	Real and imaginary parts of the complex/quaternion a respectively.
$T_N = \{(d_i, \mathbf{x}_i)\}_{i=1:N}$	Training set
$\text{Tr}\{\cdot\}$	Trace operator
X_t	Latent state of the state-space
Y_t	Observed process of the state-space model
a_i	Kernel mixing parameters
$\mathbf{d}_t$	Target (desired) process
$\mathbf{s}^i$	i^{th} support vector
$w^{(j)}$	Weight of the j^{th} particle or sample
$\mathbf{x}^{(j)}$	j^{th} particle or sample
$\mathbf{x}_t$	Regressor process
$\mathbf{y}_t$	Estimate of $\mathbf{d}_t$ using $\mathbf{x}_t$

Chapter 1

Introduction

I believe that learning has just started, because whatever we did before, it was some sort of a classical setting known to classical statistics as well. Now we come to the moment where we are trying to develop a new philosophy which goes beyond classical models.

-Vladimir Vapnik.
(In an interview for `learningtheory.org`, 2008.)

According to Thomas Kuhn's *The Structure of Scientific Revolutions* [Kuhn, 1962], the progress in science is not only to be viewed as "development-by-accumulation", but rather as a revolutionary process whereby old paradigms are abandoned in favour of new ones. This so-called paradigm shift is triggered by the inconsistency between existing theories and observed phenomena. From its very beginnings, experimental science has relied on mathematics to interpret measurements and express models that support scientific hypotheses. As a consequence, our understanding of nature has been historically limited by our ability to produce mathematical models that are consistent with available evidence. The design of such models can be achieved by combining existing (theoretical) knowledge and collected measurements; however, when the theoretical description of the process of interest is scarce, one is left with the question: Can we learn merely from data? Or more specifically: How can we extract knowledge from data?

Since the dawn of the Information Age in the early 1990s, we have been exposed to ever-increasing volumes of data arising in disciplines such as social networks, distributed sensing and finance. These complex systems comprise several variables and sources of uncertainty, and we are in general unable to model them using first principles[1]; we thus aim to produce empirical models from the data they generate. Although learning from data is rooted in the foundations of science, processing large volumes of information has only become possible in the last three decades owing to the developments in com-

[1] A calculation is said to be from first principles, or *ab initio*, if it is based on established laws of nature without additional assumptions or special models.

putational resources. This allows us to *harvest* knowledge from data, while at the same time requires us to develop learning strategies that benefit from available computational power and the diversity of data sources.

Learning, from both practical and epistemological perspectives, has been at the core of the human endeavour. In this sense, inspired by the very way in which we, humans, process information, we can exploit computational resources and empirical evidence to design intelligent machines capable of replicating and enhancing our understanding of nature. We then say that these machines *learn* from the available data, and refer to the design of such machines as *Machine Learning*. Whether this is, according to Kuhn, a paradigm shift, is only to be confirmed by future generations.

1.1 Scope of the Thesis: Kernels and Signal Estimation

Machine learning is an emerging discipline that draws on tools from a number of well-established fields such as linear algebra, functional analysis, optimisation and statistics, as well as newer concepts from artificial intelligence and biologically-inspired systems. Depending on the nature of the task, machine learning can be categorised in three sub-disciplines: i) supervised learning, where input and output (label) data is available and the algorithm is trained to replicate the input-output relationship, ii) unsupervised learning, where the output is unlabelled and therefore direct minimisation of the estimation error is not possible, and iii) reinforcement learning, where the learning takes place by maximising a reward function. Across all these three divisions of machine learning, we find a rich set of techniques that address a wide range of applications in scientific and industrial disciplines. Our focus is on the class of kernel methods and its application to signal estimation.

During the last two decades, kernel learning [Cristianini and Shawe-Taylor, 2000, Schölkopf and Smola, 2001] has been a fundamental resource within machine learning, and a vast amount of research has been generated in both applied and theoretical directions. Kernel algorithms for regression are referred to as support vector regression (SVR), a generalisation of support vector machines (SVM). The concept underpinning both SVM and SVR is that of learning on high-dimensional feature spaces: these methods perform nonlinear estimation by first mapping the input data to a feature space and then performing linear estimation in such a space. At first, it may not be clear why this procedure replaces direct nonlinear estimation by the execution of operations in a high-, or even infinite-dimensional space, as one could argue that the complexity of the overall estimation is still prohibitively large. However, it turns out that when the feature space has a set of desired properties, more specifically, those of a reproducing kernel Hilbert space (RKHS), the high-dimensional operations can be replaced by rather simple computations in the input space, thus allowing for the design of nonlinear estimation algorithms at the

price of a linear increase in computation.

By combining the feature space concept of kernel learning with existing linear estimation approaches, kernel counterparts of classic estimation algorithms can be devised. This procedure is referred to as *kernelisation* and it allows for any algorithm with a linear stage to operate on a RKHS, provided that the operations performed by the algorithm can be reproduced in the feature space. In this sense, kernel methods provide a fertile ground for signal processing algorithms, as learning from data at a low computational cost, where kernel methods excel, is at the core of filtering and prediction tasks. Kernel methods for signal estimation is a fast-growing discipline, and a number of algorithms have been already proposed and validated by the academic community. These include kernel principal component analysis [Schölkopf et al., 1998], kernel ridge regression [Saunders et al., 1998], kernel least mean square [Liu et al., 2008], and kernel affine projection and kernel normalised LMS [Richard et al., 2009], to name but a few.

This thesis proposes further contributions to the field of kernel methods for signal estimation. We firmly believe that the field of kernel adaptive filtering (KAF) [Liu et al., 2010] will continue to benefit from proven concepts and ideas established in standard kernel learning, such as multiclass classification using multiple kernels, and—more importantly—from novel methods specifically designed to address KAF issues, this includes the design of novel kernels and the incorporation of Bayesian inference.

Due to the universal approximation property of reproducing kernels, the choice of the kernel, or more specifically, of its parameters, is usually neglected in KAF applications. This is justified by the fact that the performance of SVR algorithms is known to be robust with respect to the choice of kernel parameters when the kernel is *universal* [Steinwart et al., 2006]. Adaptive filtering deals with varying signal dynamics, either in time (nonstationary signals) or in space (nonhomogeneous signals), and therefore requires algorithms that have the ability to learn different nonlinear behaviours. To this end, we consider high-dimensional kernels (HDK), that is, kernel functions the codomain[2] of which has a dimension greater than unity, to estimate signals that exhibit multiple and/or coupled nonlinear patterns. The use of HDK in this scenario arises naturally, just as one would aim to improve the learning performance by incorporating more neurons to a neural network, or by adding multiple covariance functions to a Gaussian process. We propose different alternatives to designing HDK, first focusing on complex-valued kernels, for which the underlying RKHS is readily developed and therefore is primarily a kernel-design problem. We then study quaternion-valued kernels, where the aims are to give a rigorous account of the concept of quaternion-valued RKHS and also to design quaternion kernel functions. Finally, we address vector-valued kernels, including both the design of the kernels and the analysis of their corresponding vector-RKHS. Through synthetic and real-world examples, we illustrate the appeal of the HDK

[2] Also known as range or image.

approach in adaptive filtering, not only because of the higher degrees of freedom associated with HDK, but also due to the ability of HDK to learn different nonlinear behaviours in a localised fashion.

Another discipline that naturally benefits from the enhanced modelling ability and simplicity of SVR is Bayesian filtering [Candy, 2011], a well-established field that spans a wide variety of applications from engineering and finance to sociology and biology. Bayesian filtering focuses on the probabilistic estimation of a signal from noisy observations, where numerical methods—such as those based on Monte Carlo simulations—allow us to approximate the optimal filter, even when the underlying dynamical model is nonlinear and the driving noises are non-Gaussian. With the development of numerical methods for filtering and the increasing computational power, dynamical models need not be constrained in order to satisfy stringent requirements of the filters; this opens completely new possibilities for the design of dynamical models using kernels. We focus on learning the state-transition function within state space models (SSM) using kernels, this requires unsupervised learning of the mixing parameters and is achieved using Monte Carlo methods to sample from the posterior density of the parameters. The resulting algorithms allow for joint system identification and state estimation, and are particularly suited for large observation noise.

1.2 Historical Background

The estimation problem can be traced back to ancient Greeks and Egyptians, who relied on the use of the mode and the midrange to approximate unknown quantities from a set of measurements [Harter, 1974]. Modern estimation builds on the optimal least squares approach, first published by Adrien-Marie Legendre (1805), although Carl Friedrich Gauss claimed in 1809 that he had previously used it in 1795 while estimating the orbit of an asteroid. In 1886, Sir Francis Galton studied the hereditary properties of height, where he concluded that offsprings of individuals with extreme heights (i.e. tall or short) had heights closer to the mean, thus stating that the the offspring's heights *regress towards the mean* and then coining the term *regression* [Galton, 1886]. More than half a century later, Andrey Kolmogorov introduced the modern axiomatic foundations of probability theory [Kolmogorov, 1931] and then analysed the prediction problem for discrete-time stationary processes [Kolmogorov, 1941], while Norbert Wiener (1942) was the first to address the problem of optimal estimation of continuous-time process in the presence of noise and to give an explicit form for the filter [Wiener, 1949, Kailath, 1974]. The following twenty years saw many contributions to filtering theory that culminated with the publication of the optimal linear filter by Rudolf E. Kalman [Kalman, 1960] and the solution to the general nonlinear filtering problem by Ruslan Stratonovich and Harold J. Kushner in the mid-1960s [Kushner, 1964]. During the 1960s, the linear fil-

ter became extremely popular due to its simplicity and accuracy, and was even used in the Apollo missions [Bain and Crisan, 2009]; however, the use of more expressive dynamical models was not possible at the time, as the closed-form solutions to the general case (Kushner-Stratonovich) could not be devised. During the following 30 years, a number of algorithms based on approximated solutions were proposed, but it was only in 1993 when Neil J. Gordon and his collaborators proposed what we today know as Particle Filters, a method that uses Monte Carlo sampling to implement the Bayesian filtering recursions and does not rely on any (e.g. linear) approximation [Gordon et al., 1993]. This coincided with the so-called *revolution* of Markov chain Monte Carlo methods (MCMC) [Robert et al., 2011], which although being developed since the early 1950s [Metropolis et al., 1953] was only fully accepted by the mainstream academic community with the seminal work in [Gelfand and Smith, 1990].

Besides the introduction of the Kalman filter, the year 1960 can be considered a landmark for the community on the overlap between learning systems and stochastic, adaptive, and nonlinear approaches to filtering. The motivation for using biologically-inspired concepts in filtering can be traced back to the work by Denis Gabor, who, in his words, *"[proposed] to take a short cut through mathematical difficulties by constructing a filter which optimizes its response as do animals and men—by learning"* [Gabor et al., 1960]. In the same year, Bernard Widrow and Marcian Hoff [Widrow and Hoff, 1960] proposed the least mean square (LMS) algorithm, which not only is a *de facto* standard in adaptive filtering, but also an important contribution to the field of neural networks: it gave birth to the adaptive linear neuron (ADALINE). Although the idea of an artificial neural network had existed since as early as Alfred Smee[3] and his research in biology and electricity [Smee, 1850], it was only after the work by [McCulloch and Pitts, 1943] and the invention of the perceptron [Rosenblatt, 1958] that the field of neural networks became an emerging discipline, and reached widespread popularity with the introduction of the backpropagation rule [Rumelhart et al., 1986], the basis of which is the LMS algorithm. As a consequence, by considering signal estimation as a particular case of the much general problem of function approximation [Principe, 2001], a plethora of estimation algorithms based on neural networks were developed in the following years.

Early concepts of machine intelligence emerged with the first fully-electronic computer, the ENIAC (1946), and with the introduction of the Turing test in 1950. Computer gaming pioneer Arthur Samuel created the first learning program in 1952 at IBM, a computer implementation of draughts, and first referred to machine learning as the *"field of study that gives computers the ability to learn without being explicitly programmed"* [Samuel, 1959]. At the time, the approach to learning was rather algorithmic and lacked theoretical foundations, it was only with the visionary work

[3]We thank Prof. Jose Principe, University of Florida, for bringing to our attention the work of a pioneer electro-biologist Alfred Smee.

by Vladimir Vapnik and Alexey Chervonenkis that the formal statistical learning theory began in 1974 [Vapnik, 1982]. However, according to Vapnik himself, *"until the 1990s [statistical learning theory] was a purely theoretical analysis of the problem of function estimation"* [Vapnik, 1999] and it was only in 1992 when the proposed theory allowed for the breakthrough learning algorithms: support vector machines (SVM) [Boser et al., 1992]. The SVM is a nonlinear extension of the generalised portrait algorithm [Vapnik and Lerner, 1963] that builds on the statistical foundations of the Vapnik-Chervonenkis theory, functional analysis results from [Mercer, 1909, Aronszajn, 1950], and what we know today as the kernel trick [Aizerman et al., 1964]. This novel approach to learning using high-dimensional feature spaces allowed for the design of learning machines by incorporating proven optimisation techniques, and thus resulted in powerful regression algorithms [Vapnik, 1995, Drucker et al., 1996, Saunders et al., 1998, Cristianini and Shawe-Taylor, 2000, Schölkopf and Smola, 2001].

In the last two decades, the popularity of Bayesian methods has not been exclusive to filtering applications. Machine learning has also benefitted from the ability of Bayesian inference to combine both prior *belief* of unknown quantities and available evidence [MacKay, 1992]. Novel Bayesian-based machine learning methods include training of neural networks [Neal, 1995] and Gaussian processes [Rasmussen and Williams, 2005].

1.3 Contributions of the Thesis

This thesis presents background theory of kernel learning and filtering, novel approaches to signal estimation in the field of kernel-based signal estimation, and experimental validation of the proposed methods. The description of the original contributions of the thesis is summarised as follows.

- **A unified account of linear and kernel adaptive filters for multivariate signals.** A practical presentation of both linear and kernel adaptive filtering is given with an emphasis on their common features and advantages. Additionally, a novel interpretation of the existence of the kernel ridge regression solutions is provided—as introduced in [Tobar and Mandic, 2012]. This overview is given in Chapter 3.

- **Adaptive sparsification criteria.** Current adaptive sparsification criteria operate on the basis of accepting samples according to sparsity or performance improvement, and do not eliminate samples. We address this issue by proposing adaptive sparsification criteria with a sample-elimination stage, in this way, support vectors that are not representative of the current operation region, and therefore do not contribute to the estimate, are removed from the dictionary. This contribution is elaborated in Chapter 3 and was introduced in [Tobar et al., 2014b].

- **Design of complex kernels using real kernels.** The standard reproducing kernel Hilbert space (RKHS) theory admits the use of complex-valued kernels, however, these are rarely used in kernel learning applications. To address the role of complex-valued kernels within practical estimators, we propose a framework for designing complex-valued kernels from real-valued ones through a process called *complexification*, whereby physical interpretation based on the properties of the original real kernel is provided. The proposed procedure is presented in Chapter 4 and has been published in [Tobar et al., 2012].

- **Quaternion RKHS and quaternion kernels.** Following on from the concept of learning on infinite-dimensional RKHS and the derivation of complex-kernels, we provide a rigorous account of quaternion-valued RKHS (QRKHS) in order to incorporate even higher-dimensional features that allow for enhanced learning, while still performing scalar (rather than vector) algebraic operations. We study the quaternion-valued Gaussian kernel and discuss the choice of its parameter, together with proposing a quaternion version of the cubic kernel. The QRKHS and the accompanying examples have been published in [Tobar and Mandic, 2013, Tobar and Mandic, 2014b] and are presented in Chapter 4.

- **Vector RKHS and multikernel learning.** Motivated by the enhanced estimation ability of hypercomplex kernels, we propose a vector-valued RKHS (VRKHS), where the features can be of an arbitrary dimension. We provide a detailed construction of the proposed VRKHS from a set of standard RKHSs, and then show how standard SVR methods can also benefit from the VRKHS approach. This is supported by simulation examples and a discussion about the properties of the VRKHS, as well as its relationship with the QRKHS approach. This concept is presented in Chapter 5 and has been published in [Tobar and Mandic, 2012, Tobar et al., 2014b].

- **Design of state-space models using kernels.** Monte Carlo methods for Bayesian filtering are general and do not assume linearity, thus admitting accurate albeit complex dynamical models. In Chapter 7, we propose a framework to design state-space models using kernels, whereby the kernel mixing parameters are learnt in a Bayesian fashion using Markov chain Monte Carlo (MCMC). The method is flexible and can be combined with different approaches to Monte Carlo sampling and particle filters (PF), as evidenced in [Tobar et al., 2014a].

- **Implementation of kernel state space models for system identification, filtering and prediction.** The proposed kernel SSM approach has been implemented for the aforementioned tasks, this required us to study different numerical approaches for parameter estimation including MCMC, PF and artificial evolution. Both off-line and online versions were tested over a number of synthetic examples given in

Chapter 7. Some of these results can be found in [Tobar and Mandic, 2014a] and [Tobar et al., 2014a].

- **Experimental validation of the proposed algorithms using real-world signals.** All the proposed concepts were validated and compared against standard kernel-based methods using real-world signals of different natures, these include 2D wind speed, inertial bodymotion trajectories, 2D point-of-gaze signals and power grid frequency. This allowed us to validate the ability of kernel-methods in both adaptive and Bayesian filtering, and to assess the performance of the proposed kernel algorithms in quantitative terms. These experiments are compiled in Chapter 8.

1.4 Organisation

This thesis is organised in three parts and nine chapters. After the Introduction in Chapter 1 and the background on kernel learning in Chapter 2, the theoretical contributions and simulation examples are organised into **Part I: High-Dimensional Kernel Adaptive Filters** and **Part II: Kernel-Based State-Space Models**, which include their own background material. **Part III: Real-World Simulations and Concluding Remarks** presents experimental results and conclusions for the methods introduced in the preceding parts.

A detailed summary of each chapter is given as follows:

- **Chapter 2: Kernel Learning** gives a brief presentation of the classification problem using graphical examples and then presents the kernel regression paradigm in a feature-space manner. Radial-basis-function and polynomial kernels are also introduced.

Part I: High-Dimensional Kernel Adaptive Filters

- **Chapter 3 Linear and Kernel Adaptive Filtering: A Unified View** provides an overview of linear and kernel adaptive filters, and how they relate to one another, with emphasis on vector-valued outputs.

- **Chapter 4 Hypercomplex Kernels** first presents a method for designing complex-valued kernels from real-valued ones and then proceeds to give a rigorous account of quaternion-valued RKHS. Practical design of quaternion kernels and their usefulness are elucidated over illustrative examples.

- **Chapter 5 Vector-Valued Kernels** addresses the design of vector-valued RKHS and vector-kernels, introduces the multikernel ridge regression and multikernel least mean square, and provides simulation examples. An interpretation and comparison with the quaternion RKHS is also presented.

Part II: Kernel-Based State-Space Models

- **Chapter 6 Bayesian Filtering and Monte Carlo Methods** gives a brief overview of the Bayesian approach to filtering and then presents Monte Carlo methods for both batch sampling (importance sampling and Markov chain Monte Carlo) and sequential sampling (particle filters).
- **Chapter 7 Unsupervised State-Space Modelling** proposes a novel approach to modelling dynamical systems using kernels. A (parametric) kernel-based state-space model is introduced to then study different ways of estimating its kernel mixing parameters. Illustrative examples for the offline and online cases are also presented.

Part III: Real-World Simulations and Concluding Remarks

- **Chapter 8 Experimental Validation** verifies and validates the proposed algorithms using real-world signals against existing kernel estimation algorithms. Practical experiments include prediction and tracking of wind speed, bodysensor trajectories, point-of-gaze signals and power grid frequency estimation, set within both adaptive and Bayesian filtering frameworks.
- **Chapter 9 Conclusions** provides the concluding remarks of the thesis and also suggest future research directions.

Chapter 2

Kernel Learning

It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts.

-Sir Arthur Conan Doyle
("A Scandal in Bohemia," 1891.)

This chapter presents the theoretical background on kernel estimation required for the thesis. A brief overview of support vector machines is provided in a graphical manner to then focus on support vector regression from a feature-space perspective, with examples given for standard reproducing kernels used in support vector estimation. The existing literature on kernel methods is vast, with several outstanding textbooks available [Cristianini and Shawe-Taylor, 2000, Schölkopf and Smola, 2001, Steinwart and Christmann, 2008]; our aim is to present the background material in a concise fashion to aid the reading of the following chapters.

2.1 Support Vector Machines

In the supervised setting, classification refers to the categorisation of a set of measurements according to historical labelled data. The classification problem is at the core of machine learning and also commonly encountered in a wide variety of disciplines.

A toy example is presented in fig. 2.1, where data samples $\mathbf{x}_i \in \mathbb{R}^n$ of categories $d_i \in \{-1, +1\}$, are linearly separable. In this case, the classification problem boils down to finding the hyperplane $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$, or more specifically, the parameters $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$, such that the class of the sample $\mathbf{x}_i$ is expressed as

$$d_i = \text{sgn}\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right). \tag{2.1}$$

The system defined in eq. (2.1) for a collection of N_{sam} samples $\{\mathbf{x}_i\}_{i=1:N_{sam}}$ is overdeter-

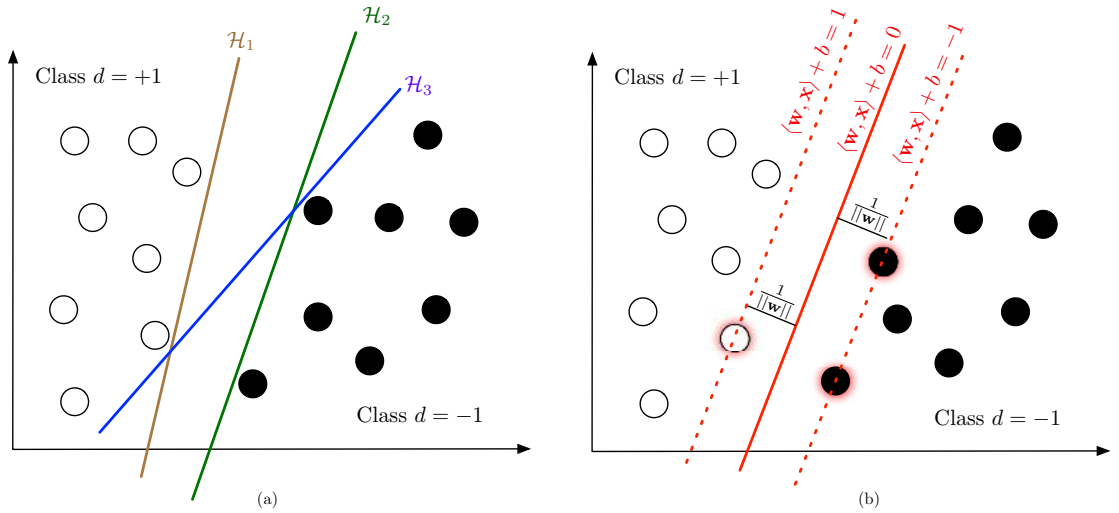

Figure 2.1: Classification of linearly separable data. The left plot shows how an infinite number of planes classify the data samples, while the right plot shows the (unique) maximum margin separating plane and the support vectors (with red shadows).

mined, meaning that there are multiple hyperplanes that satisfy such condition—see fig. 2.1 (a). This allows us to introduce additional constrains that induce desired properties on the solution. A common practice is to find the *maximum-margin separating plane*, this is achieved by maximising the distance between the plane and the nearest data samples subject to the plane being a separating plane (see fig. 2.1 (b)). As the margin is equal to $\frac{1}{||\mathbf{w}||}$, this optimisation problem can be posed as

$$\underset{\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}}{\text{minimise}} \ \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.2}$$

$$\text{subject to } d_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) \geq 1, \forall i = 1, \ldots, N_{sam} \tag{2.3}$$

where the constraint "greater or equal than one" indicates that the closest points to the plane satisfy $|\langle \mathbf{w}, \mathbf{x} \rangle + b| = 1$. For a detailed explanation of the linear classifier, see [Schölkopf and Smola, 2001].

The classification problem becomes much more challenging in real-world applications, where the raw high-dimensional data rarely exhibits linear separability. Consider the example in fig. 2.2, where samples $(x_1, x_2) \in [-5, 5] \times [-5, 5]$ corresponding to classes 'red' and 'blue' need to be classified. Although in this case the data samples can be clearly separated by an ellipse, we proceed by *mapping the samples* into a space where they can be separated by a plane; this way, we can address the general classification problem using existing tools for linear classification. We thus transform the data from $\mathbb{R}^2$ to $\mathbb{R}^3$ through

$$(x_1, x_2) \longmapsto (x_1, x_2, 40 + 4x_1^2 + 4x_2^2). \tag{2.4}$$

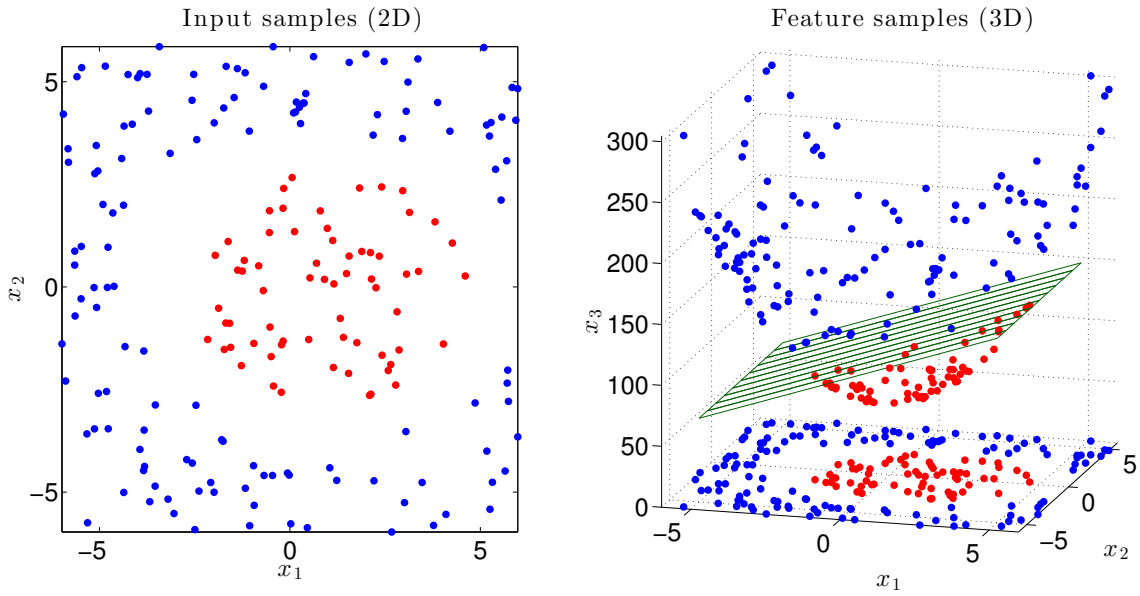

Figure 2.2: Classification of non-linearly-separable data. The left plot shows the original samples in $\mathbb{R}^2$ corresponding to classes red and blue, while the right plot shows both original and feature samples in $\mathbb{R}^3$ with the corresponding separating plane.

The space of transformed samples, in this case $\mathbb{R}^3$, is referred to as the *feature space*.

As the feature samples are linearly separable, a linear classifier can now be implemented in the **feature space**, thus making use of the many readily available methods for linear classification.

Unlike the example in fig. 2.2, finding a feature transformation that results in linearly-separable features is challenging in real-world classification tasks. This can be surmounted by developing a classification algorithm that: (i) uses feature spaces of a sufficiently high dimension for the feature data to become linearly separable, and (ii) has a computational complexity that does not increase with the dimension of the feature space. This concept is known as support vector classification and it will be addressed from a regression point of view in the next section.

2.2 Support Vector Regression

We now turn to the more general regression problem, where the function to be learnt is no longer binary but continuous. We focus on scalar-valued outputs, since the vector-valued case can be understood as an ensemble of scalar estimators, and denote the vector-valued input by $\mathbf{x} \in X \subset \mathbb{R}^n$ and the scalar-valued output by $d \in D \subset \mathbb{R}$.

Consider the set of observed input-output pairs $T_N = \{(d_i, \mathbf{x}_i)\}_{i=1:N} \subset D \times X$ referred to as the *training set*. The aim of the regression problem is to find an estimate of

the nonlinear function

$$f : X \longrightarrow D \tag{2.5}$$

$$\mathbf{x}_i \longmapsto f(\mathbf{x}_i) = d_i. \tag{2.6}$$

To approximate the nonlinear function f, we proceed as in the example of fig. 2.2, that is, by mapping the input samples $\mathbf{x}_i$ through an arbitrary function ϕ into the feature space H. In general, the function ϕ

$$\phi : X \longrightarrow H \tag{2.7}$$

$$\mathbf{x}_i \longmapsto \phi_{\mathbf{x}_i} \tag{2.8}$$

is chosen to be nonlinear with an infinite-dimensional codomain H. Consequently, each feature element $\phi_{\mathbf{x}_i}$ is a scalar-valued function that depends on $\mathbf{x}_i$ and can be evaluated on X, that is,

$$\phi_{\mathbf{x}_i} : X \longrightarrow \mathcal{F} \tag{2.9}$$

$$\mathbf{x} \longmapsto \phi_{\mathbf{x}_i}(\mathbf{x}) \tag{2.10}$$

where by $\mathcal{F}$ we denote either the real field, $\mathbb{R}$, or the complex field, $\mathbb{C}$.

We then approximate the function f by a linear estimator, operating on the feature space H, given by[1]

$$f_{\mathbf{A}}(x) = \langle \mathbf{A}, \phi_x \rangle \tag{2.11}$$

where the coefficients $\mathbf{A} \in H$ and $\langle \cdot, \cdot \rangle$ is the inner product in H. The notation $f_{\mathbf{A}}$ is used to emphasise the dependency of the estimator on $\mathbf{A}$. The optimal weight $\mathbf{A}^*$ can then be found as

$$\mathbf{A}^* = \underset{\mathbf{A} \in H}{\operatorname{argmin}} \left\{ J\left((\mathbf{x}_1, d_1, f_{\mathbf{A}}(\mathbf{x}_1)), ..., (\mathbf{x}_N, d_N, f_{\mathbf{A}}(\mathbf{x}_N)) \right) + \rho\left(\|f_{\mathbf{A}}\| \right) \right\} \tag{2.12}$$

where J is an arbitrary cost function and ρ is a non-decreasing real function.

The representer theorem [Schölkopf et al., 2001] states that the optimal weight given by eq. (2.12) resides on the span of $\{\phi_{\mathbf{x}_i}\}_{i=1:N}$. This space, which has a smaller dimension than the original space H, is termed *empirical feature space* [Kung, 2014] and is defined by[2]

$$\mathcal{H} = \left\{ \phi \in H, \text{ s.t. } \phi = \sum_{i=1}^{N} c_i \phi_{\mathbf{x}_i}, c_i \in \mathbb{R}, \mathbf{x}_i \in T_N \right\}. \tag{2.13}$$

[1] The linear estimate in this case collapses into an inner product, however, in the case of a vector-valued output the linear estimator is a tensor product, since the vector-space is infinite dimensional.

[2] For ease of presentation, we have slightly abused of the notation when writing $\mathbf{x}_i \in T_N$, since $T_N = \{(d_i, \mathbf{x}_i)\}_{i=1:N} \subset D \times X$.

The representer theorem then allows us to write $\mathbf{A} = \sum_{i=1}^{N} a_i \phi_{\mathbf{x}_i}$ for some coefficients $\mathbf{a} = [a_i, \ldots, a_N]^T \in \mathbb{R}^N$, and write the desired estimate as

$$f_{\mathbf{a}}(\mathbf{x}) = \left\langle \sum_{i=1}^{N} a_i \phi_{\mathbf{x}_i}, \phi_{\mathbf{x}} \right\rangle = \sum_{i=1}^{N} a_i \left\langle \phi_{\mathbf{x}_i}, \phi_{\mathbf{x}} \right\rangle \tag{2.14}$$

where we use $f_{\mathbf{a}}$ instead of $f_{\mathbf{A}}$ to express the explicit dependency of the estimate on the $\mathbb{R}^N$-vector $\mathbf{a}$ rather than the infinite-dimensional weight $\mathbf{A} \in H$.

Observe that the inner product in eq. (2.14) depends on the mapping ϕ and the samples $\mathbf{x}_i, \mathbf{x}$ only; we then denote $K(\mathbf{x}_i, \mathbf{x}) = \langle \phi_{\mathbf{x}_i}, \phi_{\mathbf{x}} \rangle$ and write the estimate as

$$f_{\mathbf{a}}(\mathbf{x}) = \sum_{i=1}^{N} a_i K(\mathbf{x}_i, \mathbf{x}). \tag{2.15}$$

The function K is known as *reproducing kernel* and the substitution $K(\mathbf{x}_i, \mathbf{x}) = \langle \phi_{\mathbf{x}_i}, \phi_{\mathbf{x}} \rangle$, referred to as the *kernel trick* [Aizerman et al., 1964], allows us to replace inner products on H by kernel evaluations on X.

The resulting estimator in eq. (2.15) has desired properties regarding tractability of parameter identification, low-complexity implementation, and enhanced capability for nonlinear regression. Firstly, by converting the problem of finding the infinite-dimensional weight $\mathbf{A}$ into that of finding the vector $\mathbf{a}$, the parameter identification becomes much simpler and can be addressed with standard optimisation techniques. Secondly, via the kernel trick, the evaluation of the inner product in H is converted into the evaluation of a function in X, therefore reducing the computational complexity and the burden of finding the mapping ϕ, since only the kernel function is required. Thirdly, since for every mapping ϕ the kernel K is guaranteed to exist[3], the nonlinear regression capability is retained even when the mapping ϕ is unknown and the estimator is designed based on the kernel K only. This last point raises the following question: when focusing solely on the choice of kernel K (rather than the mapping ϕ), what kernels can be considered so that a mapping ϕ exists? We address this question by first stating the following definition and theorem.

Definition 1 (Reproducing kernel [Mercer, 1909])**.** *A reproducing kernel over the set X is a continuous, symmetric, positive-definite function $K : X \times X \longrightarrow \mathbb{R}$.*

Theorem 1 (Moore-Aronszajn [Aronszajn, 1950])**.** *For any reproducing kernel K over a set X, there is a unique Hilbert space of functions on H for which K is a reproducing kernel.*

As a consequence, for a chosen kernel K there exists a corresponding RKHS if and only if K is symmetric and positive definite. This means that kernel estimators of the form in eq. (2.15) are guaranteed to have an underlying nonlinear mapping ϕ, even if

[3]Indeed, for any ϕ, the corresponding kernel is given by $K(\mathbf{x}_1, \mathbf{x}_2) = \langle \phi_{\mathbf{x}_1}, \phi_{\mathbf{x}_2} \rangle$.

it is not known explicitly, thus making possible regression on high-dimensional spaces spanned by the features $\{\phi_{\mathbf{x}_i}\}$.

The kernel trick plays a pivotal role in the development of nonlinear classification and regression by extending linear estimation algorithms through a procedure called *kernelisation*. We shall review some kernel extension of adaptive filtering algorithms in Section 3.3.

2.3 Scalar-Valued Kernels

We now review standard real-valued, symmetric, and positive-definite kernels used in classification and regression settings.

2.3.1 Radial Basis Functions (RBF)

The RBF kernels are functions of the distance of its input arguments, that is,

$$K(\mathbf{x}_i, \mathbf{x}_j) = K(d(\mathbf{x}_i, \mathbf{x}_j)) \tag{2.16}$$

where $d(\cdot, \cdot)$ is a metric on X. Observe that although $d(\cdot, \cdot)$ is usually chosen to be the Euclidean distance, any metric can be considered.

The advantage of RBF kernels is that they can be defined over metric spaces, that is, a set for which distances between members are defined, and do not require additional algebraic properties such as those present in vector or inner product spaces; this allows to define kernels over non-vector spaces such as the sphere $\{\mathbf{x} \in \mathbb{R}^n, \text{s.t. } \|\mathbf{x}\| = 1\}$ or strings (words). RBF kernels also provide a measure of similarity between an input sample and the set of support vectors, this is because (due to their positive definiteness) they reach its maximum when the input samples are equal and vanish for samples are too dissimilar.

The RBF kernels of compact support include the triangular and Epanechnikov kernels, respectively given by

$$\text{Triangular kernel:} \quad K_T(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{\Delta}\left(\Delta - \|\mathbf{x}_i - \mathbf{x}_j\|\right) \mathbf{1}_{\{\|\mathbf{x}_i - \mathbf{x}_j\| \leq \Delta\}} \tag{2.17}$$

$$\text{Epanechnikov kernel:} \quad K_E(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{\Delta}\left(\Delta - \|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \mathbf{1}_{\{\|\mathbf{x}_i - \mathbf{x}_j\| \leq \Delta\}} \tag{2.18}$$

where $\mathbf{1}_A$ denotes the indicator function and Δ is a positive parameter known as kernel *threshold*. The triangular kernel has been used for similarity-based modelling [Tobar et al., 2011] and in classification tasks where its scaling properties have been analysed [Fleuret and Sahbi, 2003], whereas the Epanechnikov kernel is used for kernel density estimation as it is optimal in the minimum variance sense [Epanechnikov, 1969].

These kernels are shown in Fig. 2.3 a) and b).

Another RBF kernel widely-used in both classification and regression is the Gaussian kernel[4]

$$K_G(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right) \tag{2.19}$$

where the parameter $\sigma > 0$ is known as the kernel width. Observe that the Gaussian kernel has infinite support.

The properties of the RKHS induced by the Gaussian kernel have been studied in [Steinwart et al., 2006]. In particular, the Gaussian kernel is proven to be an *universal kernel* [Steinwart, 2001, Example 1], meaning that its induced RKHS is dense in the space of continuous functions. Fig. 2.3 c) shows the Gaussian kernel for $\sigma = 0.5$.

Observe that for vector-valued input samples, the Gaussian kernel is insensitive to rotations, as it is a function of the Euclidean distance. One way to address this issue is to replace the Euclidean distance $d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|$ by the Mahalanobis distance [Mahalanobis, 1936] given by

$$d_M(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_i - \mathbf{x}_j)} \tag{2.20}$$

where $\boldsymbol{\Sigma}$ is a covariance matrix. The Mahalanobis distance $d_M(\mathbf{x}_i, \mathbf{x}_j)$ measures the numbers of standard deviations a sample $\mathbf{x}_j$ is away from the Gaussian distribution with mean $\mathbf{x}_i$ and covariance matrix $\boldsymbol{\Sigma}$ along each principal axis, and therefore allows for the design of orientation-sensitive kernels. The Mahalanobis Gaussian kernel is then expressed by

$$K_M(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-(\mathbf{x}_i - \mathbf{x}_j)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_i - \mathbf{x}_j)\right) \tag{2.21}$$

and is shown in fig. 2.3 d) for $\boldsymbol{\Sigma} = \begin{bmatrix} 0.4 & 0.2 \\ 0.2 & 1.2 \end{bmatrix}$.

[4]The Gaussian kernel is also referred to as square exponential kernel in the context of Gaussian processes [Rasmussen and Williams, 2005] to avoid confusion.

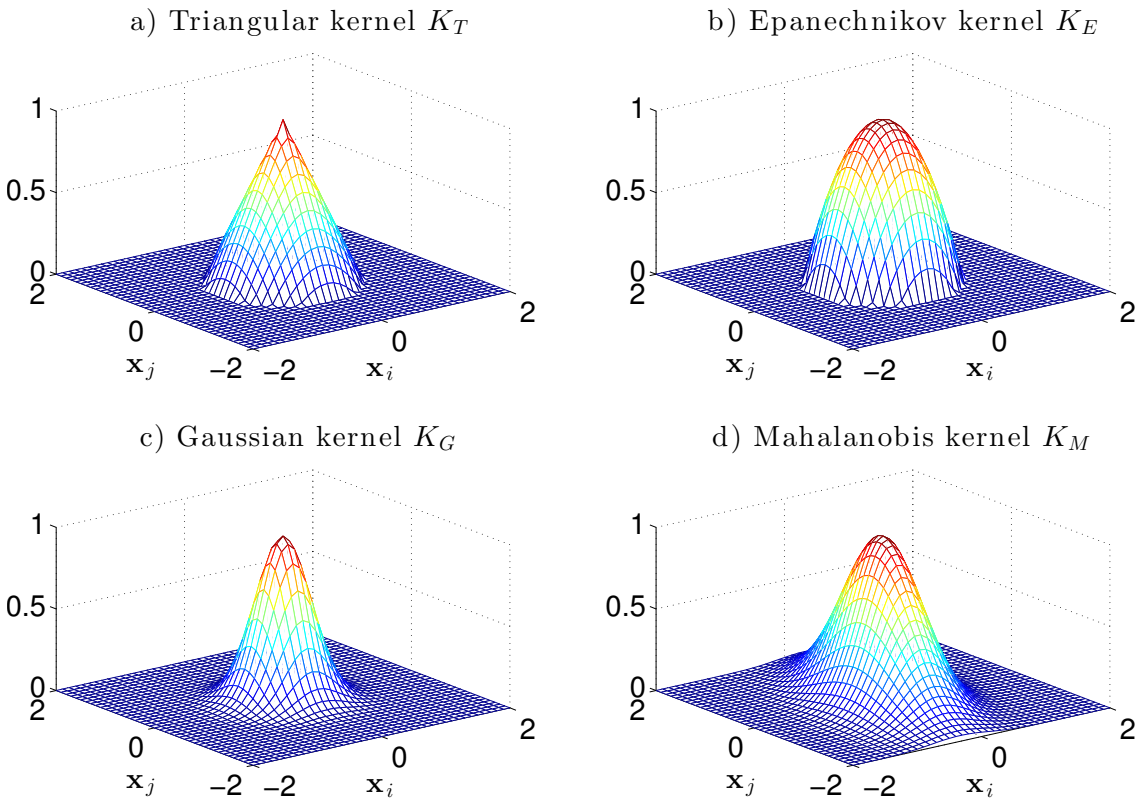

Figure 2.3: RBF kernels defined on $\mathbb{R}^2$. The triangular and Epanechnikov kernels are rotation invariant and have finite support, the Gaussian kernel is rotation-invariant and has infinite support, and the Mahalanobis kernel is rotation-sensitive and has infinite support.

2.3.2 Polynomial Kernels

Some real-world applications require highly-nonlinear classification; this can be achieved by extracting information contained in the monomials of the entries of the input vector $\mathbf{x} \in \mathbb{R}^n$, that is,

$$\{\mathbf{x}\}_{j_1} \cdot \{\mathbf{x}\}_{j_2} \cdots \{\mathbf{x}\}_{j_p} \tag{2.22}$$

where $\{\mathbf{x}\}_j \in \mathbb{R}$ is the j-th entry of the vector $\mathbf{x}$, the indices $j_1, \ldots, j_p \in \{1, \ldots, n\}$, and $p \in \mathbb{N}$ is referred to as the order of the monomial.

Polynomial classifiers [Schürmann, 1996] make use of these features by mapping the input data to the feature space of all monomials of order up to p. The resulting feature space is an RKHS, the reproducing kernel of which is the polynomial kernel[5]

$$K_P(\mathbf{x}, \mathbf{z}) = \big(\langle \mathbf{x}, \mathbf{z} \rangle + c\big)^p. \tag{2.23}$$

[5]When $c = 0$, the kernel $K_H(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x}, \mathbf{y} \rangle)^p$ is called *homogeneous* and its associated RKHS is the space of monomials of order p only, rather than *up to* p.

For the case $p = 2$, it can be shown that the monomials occur in the kernel evaluation by considering the expansion of $K_2(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x}, \mathbf{z} \rangle + c)^2$ given in [Cristianini and Shawe-Taylor, 2000] by

$$\begin{aligned} K_2(\mathbf{x}, \mathbf{z}) &= \left(\sum_{i=1}^{n} \{\mathbf{x}\}_i \{\mathbf{z}\}_i + c \right) \left(\sum_{j=1}^{n} \{\mathbf{x}\}_j \{\mathbf{z}\}_j + c \right) \\ &= \sum_{i,j=1}^{n} \{\mathbf{x}\}_i \{\mathbf{z}\}_i \{\mathbf{x}\}_j \{\mathbf{z}\}_j + 2c \sum_{i=1}^{n} \{\mathbf{x}\}_i \{\mathbf{z}\}_i + c^2 \\ &= \sum_{i,j=1}^{n} \left(\{\mathbf{x}\}_i \{\mathbf{x}\}_j \right) \left(\{\mathbf{z}\}_i \{\mathbf{z}\}_j \right) + \sum_{i=1}^{n} \left(\sqrt{2c} \{\mathbf{x}\}_i \right) \left(\sqrt{2c} \{\mathbf{z}\}_i \right) + c^2. \end{aligned} \tag{2.24}$$

By inspection of the parenthesis, we can see all the possible $\binom{n+2}{2}$ monomials of order zero, one and two, with relative weights controlled by the parameter c. In a general case, that is, for an arbitrary order $p \in \mathbb{N}$, all the possible $\binom{n+p}{p}$ monomials appear on the kernel evaluation, meaning that the kernel corresponding to the RKHS associated to such mapping is the polynomial kernel and is unique up to a scaling factor [Cristianini and Shawe-Taylor, 2000, Schölkopf and Smola, 2001].

Observe that polynomial kernels of high orders can be very difficult to implement in practice, since when the argument $1 + \langle \mathbf{x}, \mathbf{z} \rangle$ is less (cf. greater) that 1, the kernel evaluation rapidly converges to zero (cf. diverges). This feature will also be present in the complex- and quaternion-valued exponential kernels presented in Chapter 4, where the exponential growth of the kernel is effectively found to be beneficial in adaptive filtering applications.

Part I

High-Dimensional Kernel Adaptive Filters

Chapter 3

Linear and Kernel Adaptive Filtering: A Unified View

Happy families are all alike; every unhappy family is unhappy in its own way[1].

-Leo Tolstoi.
("Anna Karenina," 1878.)

This chapter provides an overview of linear adaptive filters and how they operate on RKHS to design kernel adaptive filters (KAF). The use of kernel learning is appealing within nonlinear filtering, since, unlike the linear case, each filter is *nonlinear in its own way*. We present the approaches to classic linear filtering and the more recent kernel adaptive filtering in an unconventional fashion, with emphasis on their properties and similarities. Furthermore, we consider the multivariate-output case throughout this chapter, this requires a careful treatment of notation and matrix-algebra operations, however, we trust this will aid the reading of the following chapters.

3.1 Problem Formulation

We consider the problem of estimating the *target process* $\{\mathbf{d}_t\}_{t\in\mathbb{N}} \subset \mathbb{R}^m$ from the *regressor process* $\{\mathbf{x}_t\}_{t\in\mathbb{N}} \subset \mathbb{R}^n$, and denote the *estimation process*d by $\{\mathbf{y}_t\}_{t\in\mathbb{N}} \subset \mathbb{R}^m$. Notice that the use of lowercase bold font implies that we assume the processes $\{\mathbf{x}_t\}_{t\in\mathbb{N}}$ and $\{\mathbf{d}_t\}_{t\in\mathbb{N}}$ are **observed** and therefore supervised learning strategies can be used. The aim of this formulation is to extract knowledge about $\mathbf{d}_t$ from the noisy measurements $\mathbf{x}_t$, or in other words, to *filter out* the noise from $\mathbf{x}_t$ to recover, or at least find the best estimate of, $\mathbf{d}_t$. We then refer to this estimation problem as *filtering* and identify the model that yields $\mathbf{y}_t$ (as a function of $\mathbf{x}_t$) as the *filter*.

[1]The author wishes to thank Prof. Danilo P. Mandic for this enlightening interpretation of the individualism of nonlinear models and the generality of linear ones.

A standard criterion to assess the performance of a filter is the mean-square error (MSE) $e_t = \mathbb{E}\left\{\|\mathbf{d}_t - \mathbf{y}_t\|^2 \middle| \mathbf{x}_{0:t}\right\}$, the minimisation of which leads to the minimum mean square estimator (MMSE) given—under weak regularity assumptions—by $\mathbf{y}_t = \mathbb{E}\{\mathbf{d}_t|\mathbf{x}_{0:t}\}$. In order to compute the MMSE in explicit form, it is therefore required to first assume a model relating the regressor $\mathbf{x}_t$ and the target process $\mathbf{d}_t$. This can be addressed by considering a broad class of models $\{\mathcal{M}_\theta\}_{\theta\in\Theta}$ expressing $\mathbf{d}_t$ as a function of the process $\mathbf{x}_{0:t}$ and parametrised by $\theta \in \Theta$. As for each model $\mathcal{M}_\theta$ there is an associated MSE, namely $e_t(\theta)$, the optimal model can be found by minimising $e_t(\theta)$ over the set of models, or equivalently, over the set of parameters Θ. In the following two sections we illustrate this procedure for the class of linear and kernel models.

3.2 Linear Adaptive Filters

The class of linear models [Haykin, 2001, Sayed, 2003] is considered in adaptive filtering not only when there is prior evidence of a linear relationship between the processes $\mathbf{x}_t$ and $\mathbf{d}_t$, but also when knowledge about the nonlinear data dependencies allows for preprocessing the data, e.g. using feature spaces, so that they relate to the output in a linear fashion. Consider the linear and Gaussian model[2]

$$\mathcal{M}_\theta : \ \mathbf{d}_t = \theta\mathbf{x}_t + \eta_t, \ \eta_t \sim \mathcal{N}(0,\sigma) \text{ and } \theta \in \mathbb{R}^{n\times m} \tag{3.1}$$

which leads to the MMSE given by $\mathbf{y}_t = \theta\mathbf{x}_t$. The optimal linear filter is then found by minimising the estimation error with respect to the parameter θ.

3.2.1 The Wiener Filter

We first focus on the stationary case, that is, when the statistics of both $\mathbf{d}_t$ and $\mathbf{x}_t$ are time-invariant. In order to find the optimal parameter θ (in the least squares sense), we first need to express the MSE associated with the choice of the model $\mathcal{M}_\theta$, that is[3],

$$\begin{aligned} e_t(\theta) &= \mathbb{E}\left\{\|\mathbf{d}_t - \mathbf{y}_t\|^2 \middle| \mathbf{x}_{0:t}, \mathcal{M}_\theta\right\} \\ &= \mathbb{E}\left\{\|\mathbf{d}_t\|^2 - 2\mathbf{d}_t^T\theta\mathbf{x}_t + \mathbf{x}_t^T\theta^T\theta\mathbf{x}_t\right\} \\ &= \mathbb{E}\left\{\|\mathbf{d}_t\|^2 - 2\text{Tr}\left\{\theta^T(\mathbf{d}_t\mathbf{x}_t^T)\right\} + \text{Tr}\left\{(\theta^T\theta)(\mathbf{x}_t\mathbf{x}_t^T)\right\}\right\} \\ &= \mathbb{E}\left\{\|\mathbf{d}_t\|^2\right\} - 2\text{Tr}\left\{\theta^T\mathbb{E}\left\{\mathbf{d}_t\mathbf{x}_t^T\right\}\right\} + \text{Tr}\left\{\theta^T\theta\mathbb{E}\left\{\mathbf{x}_t\mathbf{x}_t^T\right\}\right\} \\ &= \mathbb{E}\left\{\|\mathbf{d}_t\|^2\right\} - 2\text{Tr}\left\{\theta^T\mathbf{p}\right\} + \text{Tr}\left\{\theta^T\theta\mathbf{R}\right\} \end{aligned} \tag{3.2}$$

[2]We assume both $\mathbf{d}_t$ and $\mathbf{x}_t$ are zero-mean processes and use a linear, rather than affine, model.

[3]We use the linearity of the trace operator, denoted by $\text{Tr}\{\cdot\}$, and the identity $\text{Tr}\left\{\mathbf{M}^T(\mathbf{v}\mathbf{u}^T)\right\} = \mathbf{v}^T\mathbf{M}\mathbf{u}$ in Appendix A.1.

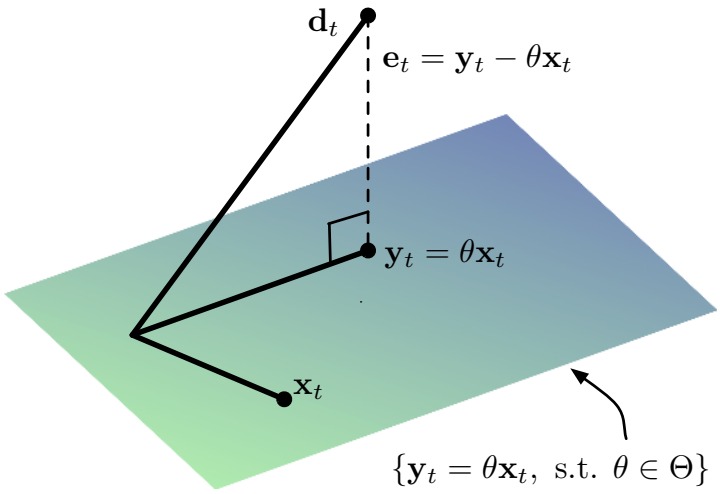

Figure 3.1: Estimation of the process $\mathbf{d}_t$ as a projection onto the space of linear estimators $\mathbf{y}_t = \theta\mathbf{x}_t, \theta \in \Theta$.

where we have identified the following second-order statistics: the covariance matrix of $\mathbf{x}_t$, $\mathbf{R} = \mathbb{E}\left\{\mathbf{x}_t\mathbf{x}_t^T\right\}$ and the covariance between $\mathbf{x}_t$ and $\mathbf{y}_t$, $\mathbf{p} = \mathbb{E}\left\{\mathbf{d}_t\mathbf{x}_t^T\right\}$. Recall that these statistics are time-invariant in the stationary case.

By rewriting the trace operator as a summation, we can now compute the partial derivatives $\frac{\partial e(\theta)}{\partial \theta_{i,j}}$ of each term in (3.2). The term $\mathbb{E}\left\{\|\mathbf{d}_t\|^2\right\}$, is independent of θ and its gradient is therefore zero, while for the remaining terms we have[4]

$$\text{Tr}\left\{\theta^T\mathbf{p}\right\} = \sum_{r=1}^{m}\sum_{s=1}^{n}\theta_{r,s}\mathbf{p}_{r,s};\ \text{therefore,}\ \frac{\partial \text{Tr}\left\{\theta^T\mathbf{p}\right\}}{\partial\theta_{i,j}} = \mathbf{p}_{i,j} \tag{3.3}$$

$$\text{Tr}\left\{\theta^T\theta\mathbf{R}\right\} = \sum_{q=1}^{m}\sum_{r=1}^{n}\sum_{s=1}^{n}\theta_{q,r}\theta_{q,s}\mathbf{R}_{r,s};\ \text{therefore,}\ \frac{\partial \text{Tr}\left\{\theta^T\theta\mathbf{R}\right\}}{\partial\theta_{i,j}} = 2\sum_{r=1}^{n}\theta_{i,r}\mathbf{R}_{r,j}. \tag{3.4}$$

In matrix form, the gradient takes the form $\nabla_\theta e = \mathbf{0} - 2\mathbf{p} + 2\theta\mathbf{R}$. By setting $\nabla_\theta e = \mathbf{0}$ we obtain the optimal parameters

$$\theta = \mathbf{p}\mathbf{R}^{-1}. \tag{3.5}$$

This result is known as the Wiener filter and was the first optimal filter derived in explicit form, originally developed for continuous-time series [Wiener, 1949]. Observe that we have arrived at $\theta = \mathbf{p}\mathbf{R}^{-1}$, which is the transpose of the standard expression $\mathbf{R}^{-1}\mathbf{p}$; this is due to the consideration of multichannel signals and the left-multiplication by θ in eq. (3.1).

[4]See Appendix A.1 for a derivation of eq. (3.4).

The Wiener filter can also be derived using the geometric interpretation of optimal estimation. Fig. 3.1 shows a vector representation of the process $\mathbf{d}_t$ and the vector space composed by linear transformations of as $\mathbf{x}_t$. The optimal estimator can therefore be thought of the element in the set $\{\mathbf{y}_t = \theta\mathbf{x}_t, \text{ s.t. } \theta \in \Theta\}$ that is *closest* to the true process $\mathbf{d}_t$. According to the orthogonality principle, this optimal estimator $\mathbf{y}_t$ must be orthogonal to the error $\mathbf{e}_t = \mathbf{d}_t - \mathbf{y}_t$. Considering that these quantities are random vectors, the orthogonality condition is given by the expectation

$$\mathbf{e}_t \perp \mathbf{y}_t \Leftrightarrow \mathbb{E}\{\mathbf{e}_t^T \mathbf{y}_t\} = 0, \ \theta \neq \mathbf{0}. \tag{3.6}$$

The expectation in the above equation can be expanded to give

$$\begin{aligned} \mathbb{E}\{\mathbf{e}_t^T \mathbf{y}_t\} &= \mathbb{E}\left\{\mathbf{d}_t^T \theta \mathbf{x}_t - \mathbf{x}_t^T \theta^T \theta \mathbf{x}_t\right\} \\ &= \mathbb{E}\left\{\text{Tr}\left\{\theta^T \mathbf{d}_t \mathbf{x}_t^T - \theta^T \theta \mathbf{x}_t \mathbf{x}_t^T\right\}\right\} \\ &= \text{Tr}\left\{\theta^T \mathbf{p} - \theta^T \theta \mathbf{R}\right\}, \end{aligned} \tag{3.7}$$

where it can be shown that the Wiener filter fulfils the orthogonality condition by replacing $\theta = \mathbf{p}\mathbf{R}^{-1}$ to give $\mathbb{E}\{\mathbf{e}_t^T \mathbf{y}_t\} = 0$.

Furthermore, upon replacing the optimal weight $\theta = \mathbf{p}\mathbf{R}^{-1}$ onto the MSE expression in eq. (3.2), we have

$$\begin{aligned} e_t(\theta) &= \mathbb{E}\left\{\|\mathbf{d}_t\|^2\right\} - 2\text{Tr}\left\{\theta^T \mathbf{p}\right\} + \text{Tr}\left\{\theta^T \left(\mathbf{p}\mathbf{R}^{-1}\right)\mathbf{R}\right\} \\ &= \mathbb{E}\left\{\|\mathbf{d}_t\|^2\right\} - \text{Tr}\left\{\theta^T \mathbf{p}\right\}. \end{aligned} \tag{3.8}$$

This expression for the discrepancy between the target and estimated signals reveals that the filter will remove power from the original signal provided that the cross-correlation between $\mathbf{d}_t$ and $\mathbf{y}_t$ is large enough, that is, the more correlated $\mathbf{d}_t$ and $\mathbf{y}_t$ are, the closer the terms in eq. (3.8) are and therefore smaller the error. In the limit, when the proceses $\mathbf{d}_t$ and $\mathbf{y}_t$ are uncorrelated (which implies independency in the Gaussian case), we have $\mathbf{p} = 0$, and therefore $\theta = \mathbf{0}$, meaning that the target process is perpendicular to the plane in fig. 3.1.

3.2.2 Least Squares and Ridge Regression

Implementing the Wiener filter requires exact knowledge of the second order statistics of the joint process $(\mathbf{x}_t, \mathbf{d}_t)$; however, in real-world applications these quantities are not known and can only be estimated. For ergodic processes, and when a collection of observations of the form $\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{N_{obs}}]^T$, $\mathbf{D} = [\mathbf{d}_0, \mathbf{d}_1, ..., \mathbf{d}_{N_{obs}}]^T$ is available, the statistics required by the Wiener filter, that is, $\mathbf{p}$ and $\mathbf{R}$, can be approximated according

to

$$\mathbb{E}\{\mathbf{d}_t\mathbf{x}_t^T\} = \mathbf{p} \approx \frac{1}{N_{obs}+1}\sum_{t=0}^{N_{obs}} \mathbf{d}_t\mathbf{x}_t^T = \frac{1}{N_{obs}+1}\mathbf{D}^T\mathbf{X} \tag{3.9}$$

$$\mathbb{E}\{\mathbf{x}_t\mathbf{x}_t^T\} = \mathbf{R} \approx \frac{1}{N_{obs}+1}\sum_{t=0}^{N_{obs}} \mathbf{x}_t\mathbf{x}_t^T = \frac{1}{N_{obs}+1}\mathbf{X}^T\mathbf{X}. \tag{3.10}$$

These empirical estimates of the second order statistics can then be used to compute the optimal parameters of the linear filter $\mathbf{y}_t = \theta\mathbf{x}_t$, this gives

$$\mathbf{y}_t = \mathbf{D}^T\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_t. \tag{3.11}$$

Eq. (3.11) is also the optimal least squares solution of the linear filter constrained to the available datasets $\mathbf{X}$ and $\mathbf{D}$; we therefore refer to this solution as the least squares filter (LS). A limitation of this data-driven approach, especially for multichannel signals, is that the estimate of the inverse covariance matrix $\mathbf{R}^{-1}$ is usually ill-conditioned. This can be because (i) the linear model is not able to explain the observations (overdetermined), or (ii) there is not enough data to produce an accurate estimate of the covariance matrix (underdetermined). A consequence of this ill-posed problem is that the optimal solution will have undesirable properties such as large norm or too dissimilar coefficients.

When the available data do not allow for an accurate reconstruction of the required statistics, the solution of the least squares filter degenerates because its design only minimises the estimation error and does not consider any particular class of solutions. One way to overcome the ill-posed nature of such problem is to include an additional restriction on the space of solutions, thus giving preference to *regular* solutions with respect to some criteria.

We then consider the problem of finding the optimal coefficient of the filter $\mathbf{y}_t = \theta\mathbf{x}_t$ with respect to the regularised cost function

$$J = \sum_{t=0}^{N_{obs}} \|\mathbf{d}_t - \mathbf{y}_t\|^2 + \rho\,\|\theta\|^2 \tag{3.12}$$

where $\rho > 0$ is known as the regularisation parameter. This cost function penalises both the discrepancy between the desired process and the estimate (through the summation of errors), and the regularity of the estimate (through norm of θ); this allows for a trade-off between model accuracy and physical meaning of the solutions.

The regularisation term can be chosen to be any norm or function of the parameter θ. We consider the L^2 norm in this case, for which the solution is given by

$$\mathbf{y}_t = \mathbf{D}^T\mathbf{X}\left(\mathbf{X}^T\mathbf{X} + \rho\mathbb{I}\right)^{-1}\mathbf{x}_t \tag{3.13}$$

where $\mathbb{I} \in \mathbb{R}^{n \times n}$ is the identity matrix. The estimate in eq. (3.13), known as ridge regression (RR), overcomes the ill-posed problem of computing the inverse of $\mathbf{X}^T\mathbf{X}$ by introducing the factor $\rho\mathbb{I}$, making the resulting matrix always invertible[5] and well-conditioned for a sufficiently large ρ. As a result, the estimation error over the training set will be larger than that of the non-regularised least squares filter, however, the regularised solution is expected to have better generalising properties.

3.2.3 The Least Mean Square Algorithm

The Wiener filter and its numerical approximations assume stationarity, this produces a time-invariant filter with coefficients estimated from observations (for ergodic processes). This condition is too stringent for real-world applications, where historical observations are not available to compute the required covariance matrices or when these statistics are time-varying. We now review the least mean square (LMS) algorithm, a widely-used algorithm in adaptive filtering due to its low computational complexity, well-known stability properties, and convergence to the Wiener solution.

Recall from the previous section that the optimal parameter of the linear filter $\mathbf{y}_t = \theta\mathbf{x}_t$ is obtained through the minimisation of the MSE and is given by $\theta = \mathbf{p}\mathbf{R}^{-1}$. A recursion for the parameter θ that converges to the optimal solution can be expressed in terms of the gradient of the error $e_t = \mathbb{E}\left\{\|\mathbf{d}_t - \mathbf{y}_t\|^2\right\}$ and a step size $\mu > 0$ in the form

$$\theta_{t+1} = \theta_t - \mu\nabla e_t \tag{3.14}$$

which, by replacing the gradient $\nabla e_t = -2\mathbf{p} + 2\theta_t\mathbf{R}$, depends explicitly[6] on the statistics $\mathbf{p}$ and $\mathbf{R}$, that is,

$$\theta_{t+1} = \theta_t + \mu\left(\mathbf{p} - \theta_t\mathbf{R}\right). \tag{3.15}$$

For nonstationary signals, the second order statistics can be replaced by instantaneous approximations, that is, sample statistics considering only a single observation of each process. We then consider the estimates

$$\widehat{\mathbf{R}}_t = \mathbf{x}_t\mathbf{x}_t^T \tag{3.16}$$

$$\widehat{\mathbf{p}}_t = \mathbf{d}_t\mathbf{x}_t^T \tag{3.17}$$

[5]Recall that diagonally-dominant matrices are invertible—see the Gershgorin circle theorem [Gershgorin, 1931].

[6]The factor 2 is absorbed by the step size μ and has been omitted for simplicity.

and upon replacing them in eq. (3.15) we obtain[7]

$$\begin{aligned}\theta_{t+1} &= \theta_t + \mu\left(\mathbf{d}_t\mathbf{x}_t^T - \theta_t\mathbf{x}_t\mathbf{x}_t^T\right) \\ &= \theta_t + \mu\left(\mathbf{d}_t - \theta_t\mathbf{x}_t\right)\mathbf{x}_t^T \\ &= \theta_t + \mu\mathbf{e}_t\mathbf{x}_t^T.\end{aligned} \tag{3.18}$$

The LMS filter can also be derived from direct minimisation of the instantaneous square error $e_t = \|\mathbf{d}_t - \theta_t\mathbf{x}_t\|^2 = \left(\mathbf{d}_t^T\mathbf{d}_t - 2\mathbf{d}_t^T\theta_t\mathbf{x}_t + \mathbf{x}_t^T\theta_t^T\theta_t\mathbf{x}_t\right)$. This is achieved by computing the the gradient of e_t

$$\begin{aligned}\nabla e_t &= -2\nabla\left(\mathbf{d}_t^T\theta_t\mathbf{x}_t\right) + \nabla\left(\mathbf{x}_t^T\theta_t^T\theta_t\mathbf{x}_t\right) \\ &= -2\mathbf{d}_t\mathbf{x}_t^T + 2\theta_t\mathbf{x}_t\mathbf{x}_t^T \\ &= -2\left(\mathbf{d}_t + 2\theta_t\mathbf{x}_t\right)\mathbf{x}_t^T \\ &= -2\mathbf{e}_t\mathbf{x}_t^T\end{aligned}$$

and replacing in eq. (3.14) to yield the LMS filter in eq. (3.18).

The convergence speed of the LMS algorithm depends on the choice of the step size μ, meaning that large step sizes result in short transient but large-variance steady state behaviour, whereas small step sizes exhibit less variability in steady state but at a price of a longer transient. Furthermore, the algorithm only converges for step sizes in the range $0 < \mu < \lambda_{\text{max}}$, where λ_{max} is the largest eigenvalue of $\mathbf{R}$; this is again a drawback for real-world applications where the covariance matrix $\mathbf{R}$ is unknown.

The normalised LMS (NLMS) addresses the sensitivity of the standard LMS to the scaling of the input by considering a variable step size that *normalises* the input signal. The step size is then set to be inversely proportional to the instantaneous (sample) covariance of the input, that is, $\mu_t \propto \left(\mathbf{x}_t^T\mathbf{x}_t\right)^{-1}$. The resulting filter is therefore given by

$$\theta_{t+1} = \theta_t + \overline{\mu}\frac{\mathbf{e}_t\mathbf{x}_t^T}{\mathbf{x}_t^T\mathbf{x}_t} \tag{3.19}$$

where $\overline{\mu}$ is the NLMS step size. Akin to the least squares algorithm, the factor $\left(\mathbf{x}_t^T\mathbf{x}_t\right)^{-1}$ needs to be regularised to avoid singularities when the signal is close to, or crosses, zero. A regularisation parameter $\epsilon > 0$ can then be introduced to the filter in the form

$$\theta_{t+1} = \theta_t + \overline{\mu}\frac{\mathbf{e}_t\mathbf{x}_t^T}{\epsilon + \mathbf{x}_t^T\mathbf{x}_t}. \tag{3.20}$$

and can be even made adaptive [Mandic, 2004].

Recall that the bound for the step size of the LMS algorithm is given by $\mu <$

[7]We emphasise that all vectors, i.e. $\mathbf{x}_t, \mathbf{d}_t, \mathbf{e}_t$, are **column vectors** and the parameter θ_t is a **matrix**.

$\text{Tr}\{\mathbf{R}\}^{-1}$. Therefore, a bound on the NLMS step size $\overline{\mu}$ can be found by identifying the NLMS as an instance of the LMS with an adaptive step size equal to $\mu = \overline{\mu}\left(\mathbf{x}_t^T\mathbf{x}_t\right)^{-1}$. The bound then becomes

$$\overline{\mu} < \frac{\mathbf{x}_t^T\mathbf{x}_t}{\text{Tr}\{\mathbf{R}\}} \tag{3.21}$$

and depends on the current value of the signal $\mathbf{x}_t$. For stationary signals, the expected value of this *random* upper bound is unity, since $\mathbb{E}\left\{\mathbf{x}_t^T\mathbf{x}_t\right\} = \text{Tr}\{\mathbf{R}\}$. This solves the problem of finding a suitable step size, since for stationary signals the NLMS step-size bound that ensures convergence is one—although a practical approach for nonstationary signals is to use a step size that is *less that one*.

3.2.4 Recursive Least Squares

The LMS computes the filter parameters by minimising the instantaneous estimation error. This may lead to noisy estimates that are not robust to observation noise or outliers, since the algorithm only targets the current estimate even when this results in poor estimates of past values. We now review the recursive least squares algorithm (RLS) which finds the optimal filter parameters in the least squares sense, by considering a cost function in the form of a weighted sum of square estimation errors

$$\xi_t = \sum_{i=0}^{t} \lambda^{t-i} \left\|\mathbf{d}_i - \theta_t\mathbf{x}_i\right\|^2. \tag{3.22}$$

There are three main differences between the RLS cost function in eq. (3.22) and that of the LMS algorithm. Firstly, the RLS minimises the error over a time window rather than just at one time instant. Secondly, the RLS considers a the signals $\mathbf{d}_t$ and $\mathbf{x}_t$ to be deterministic and defines the cost function as a sum rather than an expectation. Thirdly, the error terms in the RLS cost function are *a posteriori* errors ($\mathbf{d}_i - \theta_t\mathbf{x}_i$), rather than the *a priori* errors considered by the LMS ($\mathbf{d}_i - \theta_i\mathbf{x}_i$).

The gradient of ξ_t is given by

$$\begin{aligned}\nabla_\theta\xi_t &= -2\sum_{i=0}^{t} \lambda^{t-i}\left(\mathbf{d}_i - \theta_t\mathbf{x}_i\right)\mathbf{x}_i^T \\ &= -2\left(\sum_{i=0}^{t} \lambda^{t-i}\mathbf{d}_i\mathbf{x}_i^T - \theta_t\sum_{i=0}^{t} \lambda^{t-i}\mathbf{x}_i\mathbf{x}_i^T\right)\end{aligned}$$

which upon setting to zero gives

$$\theta_t = \sum_{i=0}^{t} \lambda^{t-i}\mathbf{d}_i\mathbf{x}_i^T\left(\sum_{i=0}^{t} \lambda^{t-i}\mathbf{x}_i\mathbf{x}_i^T\right)^{-1}. \tag{3.23}$$

By identifying the sample cross-correlation $\mathbf{p}_t = \sum_{i=0}^{t} \lambda^{t-i} \mathbf{d}_i \mathbf{x}_i^T$ and the sample covariance $\mathbf{R}_t = \left(\sum_{i=0}^{t} \lambda^{t-i} \mathbf{x}_i \mathbf{x}_i^T\right)^{-1}$, the RLS algorithm can be written as $\theta_t = \mathbf{p}_t \mathbf{R}_t^{-1}$, hence resembling a time-varying version of the Wiener filter.

The aim of the RLS algorithm is to provide an online mode of operation. However, eq. (3.23) makes the implementation of the RLS prohibitively expensive, since it requires the computation of both the inverse of $\mathbf{R}_t$, which has a computational complexity of $\mathcal{O}[n^3]$, and $\mathbf{p}_t$ in every iteration. This can be overcome by finding recursive expressions for both $\mathbf{R}_t^{-1}$ and $\mathbf{p}_t$. A recursive expression for $\mathbf{p}_t$ is straightforward and given by

$$\mathbf{p}_t = \sum_{i=0}^{t} \lambda^{t-i} \mathbf{d}_i \mathbf{x}_i^T = \lambda \sum_{i=0}^{t-1} \lambda^{(t-1)-i} \mathbf{d}_i \mathbf{x}_i^T + \mathbf{d}_t \mathbf{x}_t^T = \lambda \mathbf{p}_{t-1} + \mathbf{d}_t \mathbf{x}_t^T \tag{3.24}$$

whereas for $\mathbf{R}_t^{-1}$, a recursion can be found based on a recursion for $\mathbf{R}_t$

$$\mathbf{R}_t = \sum_{i=0}^{t} \lambda^{t-i} \mathbf{x}_i \mathbf{x}_i^T = \lambda \sum_{i=0}^{t-1} \lambda^{(t-1)-i} \mathbf{x}_i \mathbf{x}_i^T + \mathbf{x}_t \mathbf{x}_t^T = \lambda \mathbf{R}_{t-1} + \mathbf{x}_t \mathbf{x}_t^T, \tag{3.25}$$

and the matrix inversion lemma[8]

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + VA^{-1}U\right)^{-1} VA^{-1} \tag{3.26}$$

by identifying $A = \lambda \mathbf{R}_{t-1}$, $U = \mathbf{x}_t$, $V = \mathbf{x}_t^T$, and $C = 1$. Combining eqs. (3.25) and (3.26) then gives

$$\mathbf{R}_t^{-1} = \frac{1}{\lambda}\left(\mathbf{R}_{t-1}^{-1} - \frac{\mathbf{R}_{t-1}^{-1}\mathbf{x}_{t-1}\mathbf{x}_{t-1}^T\mathbf{R}_{t-1}^{-1}}{\lambda + \mathbf{x}_{t-1}^T\mathbf{R}_{t-1}^{-1}\mathbf{x}_{t-1}}\right). \tag{3.27}$$

These recursive expressions for $\mathbf{p}_t$ and $\mathbf{R}_t^{-1}$ in eqs. (3.24) and (3.27) respectively allow us to compute the RLS in a recursive and computationally-efficient manner.

3.2.5 Example: Adaptive Filters for System Identification

We now provide a comparison of the LS, LMS and RLS algorithms in the identification of a linear system. Consider the bivariate process defined by the following difference equation.

$$\mathbf{x}_{t+1} = \begin{bmatrix} 0.5 & -0.8 \\ 0.5 & 0.8 \end{bmatrix} \mathbf{x}_t + 50 \begin{bmatrix} 2 & 0.3 \\ 0.3 & 1 \end{bmatrix} \eta_t \tag{3.28}$$

[8]The identity in eq. (3.26) corresponds to the Woodbury matrix identity [Woodbury, 1950], although a simpler identity known as the Sherman-Morrison formula [Sherman and Morrison, 1950] can also be used.

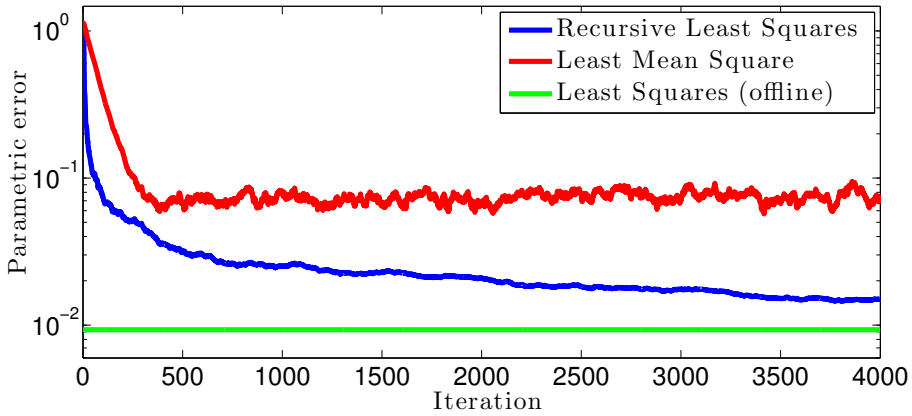

Figure 3.2: Performance of adaptive filters for system identification: Norm of the misalignment (parametric error) as a function of iteration number.

where $\eta \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_2)$ is a 2D zero-mean Gaussian noise process. The aim of this example is to perform system identification using the above-introduced algorithms.

We considered a realisation of $T = 4000$ samples and implemented both the LMS and RLS algorithms online; in addition, the nonregularised LS solution was also computed (offline) using all 4000 sample pairs. The LMS step size was set to $\mu = 5 \cdot 10^{-7}$ and the RLS forgetting factor to $\lambda = 0.9999$. The parameter error as a function of time is shown in fig. 3.2.

Even though both the LMS step size and the RLS forgetting factor allow for controlling the trade-off between convergence speed and steady-state behaviour, a comparison between the two is not particularly objective and requires analysis beyond these two parameters. The parameters μ and λ were chosen to achieve a short transient and low steady state noise; within this setup, the RLS reached a lower misalignment (i.e. parametric error) much faster that the LMS and at the same time was much more accurate in steady state. This confirms the theory behind the RLS, which not only penalises the current error but a weighted average of past errors. Fig. 3.2 also shows that the offline LS solution is the best of the three, since it includes all the samples.

3.3 Kernel Adaptive Filtering

Linear adaptive filters have had an impact in a number of branches of engineering and science by providing approximate solutions to real-world nonlinear problems at a reduced computational cost. With the ever-increasing computational power, it is now possible to construct more computationally-demanding algorithms that aim to better approximate the true nonlinear nature of real-world problems.

An approach to low-complexity nonlinear adaptive filtering is to consider algo-

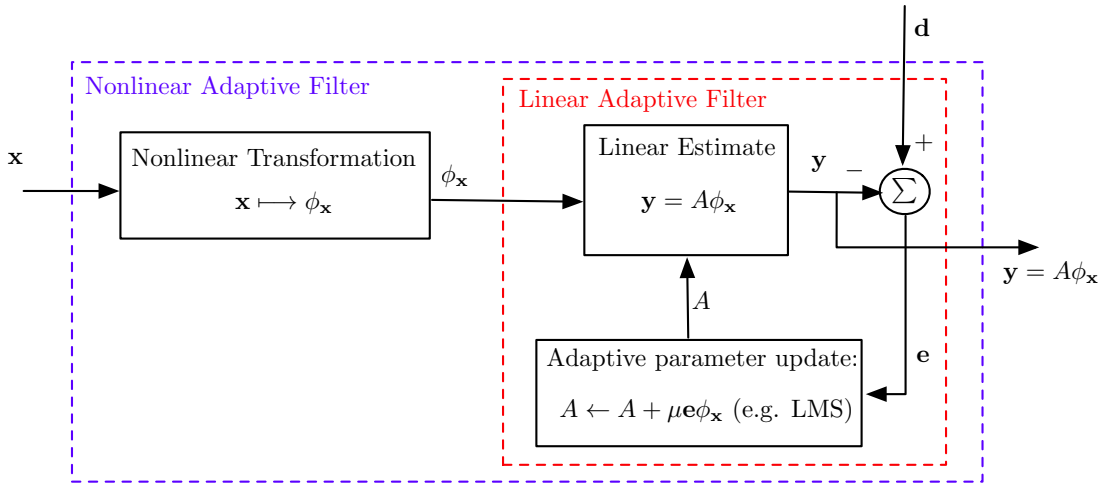

Figure 3.3: Diagram of an adaptive filter that is nonlinear in the input ($\mathbf{x}$) and linear in the parameters (A). This filter is constructed by applying a nonlinear transformation on the input and then performing adaptive filtering using the transformed sample.

rithms that are nonlinear in the input but linear in the parameters, this way, we obtain nonlinear estimation ability with straightforward parametric identification. As shown in fig. 3.3, this is achieved by transforming the input to the filter by a nonlinear function and then implement a linear filter operating on the transformed sample (known as feature sample). Well-known nonlinear filters of this type consider polynomial transformations, neural networks or Volterra series [Haykin, 1994].

Due to its linear-in-the-parameters nature, this basic yet flexible class of nonlinear filters allows for the estimation of nonlinear process at the price of linear computation. However, referring to the introductory quote of this chapter, nonlinear processes are all nonlinear in their own way and therefore the success of this naive approach to nonlinear filtering depends heavily on prior knowledge of the type of nonlinearity. This means that, unlike the linear case, where *the* linear filter can be found, each nonlinear case requires a filter with a particular nonlinearity. The approach to nonlinear adaptive filtering then needs a more general class of nonlinear transformations that do not rely on prior knowledge of the underlying nonlinearity of the signals. This can be addressed by replacing the parametric nonlinear transformation by an infinite-dimensional one, thus performing filtering in a feature space where the signals are related in a linear manner. We next study this approach to nonlinear filtering in a practical manner using support vector regression, see fig. 3.4, where we continue to use the notation $\mathbf{d}_t$ for the target process, $\mathbf{x}_t$ for the regressor and $\mathbf{y}_t$ for the estimate.

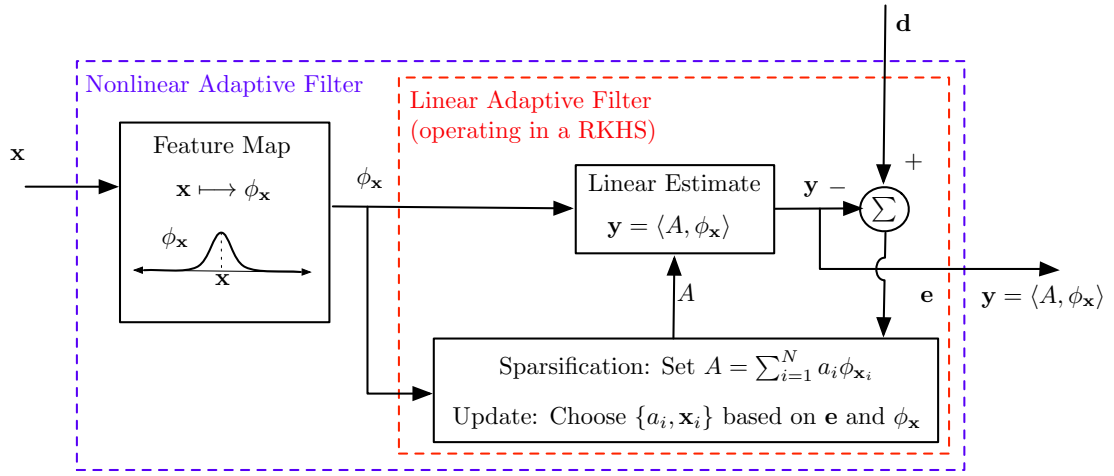

Figure 3.4: Kernel adaptive filter. The input is mapped to an infinite-dimensional feature space, and then linear adaptive filtering is performed on the features. This requires the update of both mixing parameters and support vectors.

3.3.1 Sparsification Critera

We refer to *sparsification* as the process of choosing an appropriate set of support vectors. As learning in RKHS is achieved by both updating weights and incorporating support vectors, sparsification criteria are crucial in the design of adaptive filters, since these provide a trade-off between the accuracy and complexity of the estimator. We next review three sparsification approaches for online kernel learning, these are designed to build a set of physically-meaningful support vectors in a recursive fashion. We start by presenting the concept of dictionary.

Definition 2 (Dictionary). *The set of support vectors is referred to as dictionary and denoted by* $\mathcal{D} = \{\mathbf{s}^i\}_{i=1:N}$.

Approximate Linear Dependence (ALD) [Engel et al., 2002]

In the context of kernel regression, the ALD sparsification criterion includes the observation $\mathbf{x}_t$ to the dictionary $\mathcal{D} = \{\mathbf{s}^i\}_{i=1:N}$ when its feature sample $\phi_{\mathbf{x}_t}$ does not fulfil the condition

$$\delta = \min_{\mathbf{b} \in \mathbb{R}^N} \left\| [\phi_{\mathbf{s}^1}, \ldots, \phi_{\mathbf{s}^N}]\, \mathbf{b} - \phi_{\mathbf{x}_t} \right\|^2 \leq \eta \tag{3.29}$$

for $\eta > 0$.

The optimal coefficients are calculated as $\mathbf{b}_{\text{opt}} = \mathbf{K}^{-1}\mathbf{h}(\mathbf{x}_t)$, where the entries of both the Gram matrix $\mathbf{K}$ and the kernel-evaluation vector $\mathbf{h}(\mathbf{x}_t)$ are given respectively by $\mathbf{K}_{p,q} = K(\mathbf{s}^p, \mathbf{s}^q)$ and $\mathbf{h}_p(\mathbf{x}_t) = K(\mathbf{x}_t, \mathbf{s}^p)$. Upon replacing $\mathbf{b} = \mathbf{b}_{\text{opt}}$ into (3.29), the ALD

condition becomes

$$\delta = K(\mathbf{x}_t, \mathbf{x}_t) - \mathbf{h}^T(\mathbf{x}_t)\mathbf{b}_{\text{opt}} \leq \eta. \tag{3.30}$$

The idea underpinning ALD can be summarised as follows: if a feature sample is *approximately linearly dependent* with respect to the current dictionary, its inclusion would be redundant and the associated computational cost would not justify the (marginal) increase in performance.

The Novelty and Coherence Criteria [Platt, 1991, Richard et al., 2009].

The novelty sparsification criterion includes a sample $\mathbf{x}_t$ only if: (i) the norm of the error between the desired value $\mathbf{d}_t$ and the estimate $\mathbf{y}_t$ is larger than a predefined threshold δ_e and (ii) its distance to the dictionary $d(\mathbf{x}_t, \mathcal{D}) = \min_{\mathbf{s}^i \in \mathcal{D}} \left\|\mathbf{s}^i - \mathbf{x}_t\right\|$ is greater than some threshold δ_d. The novelty criterion requirement is summarised in as

$$\|\mathbf{d}_t - \mathbf{y}_t\| \geq \delta_e \tag{3.31}$$

$$\min_{\mathbf{s}^i \in D} \left\|\mathbf{s}^i - \mathbf{x}_t\right\| \geq \delta_d. \tag{3.32}$$

A kernel-specific variant of the novelty criterion is the so-called coherence criterion proposed in [Richard et al., 2009], which gives the relationship (3.32) in terms of kernel evaluations.

Regarding the physical meaning of the sparsification rules (3.31)-(3.32), the parameter δ_e represents a measure of accuracy of the algorithm, and its choice is related to the desired steady-state error, whereas δ_d—regarded as the coherence of the dictionary—is a direct measure of sparsity. This way, the parameter δ_d represents a trade-off between learning by updating parameters only, or by adding more support vectors. The choice of δ_d also determines the maximum dictionary size, which is guaranteed to be finite for a compact set X [Richard et al., 2009].

Presence-Based Sparsification [Tobar et al., 2014b].

Together with the addition/rejection stage, a truly adaptive structure operating on non-stationary data should include an elimination step whereby dictionary samples no longer contributing to the estimation are eliminated. The elimination can be performed based on the *presence* of support vectors in the current neighbourhood of the state space: if the distance between a given support vector and the last received samples remains below a predefined bound, the support vector is retained, otherwise, it is considered to be representative of an invalid state space region, or an outlier due to noisy observations, and is therefore eliminated.

To assess the presence of a given support vector, consider the Gaussian kernel

$K_G(\cdot,\cdot)$ and define the *instantaneous presence* of the support vector $\mathbf{s}$ at time t as

$$p_t(\mathbf{s}) = K_G(\mathbf{s}, \mathbf{x}_t). \tag{3.33}$$

By adopting this definition, the larger p_i the closer $\mathbf{s}$ to the current input sample, and hence to the valid region of operation. To enhance robustness to outliers, we define the *presence* of $\mathbf{s}$ as a filtered version of $p_t(\mathbf{s})$, given by:

$$P_t(\mathbf{s}) = (1-\rho)P_{t-1}(\mathbf{s}) + \rho p_t(\mathbf{s}), \tag{3.34}$$

where $\rho \in]0,1]$ is a parameter controlling the smoothness of $P_t(\mathbf{s})$. For a support vector $\mathbf{s}$, $P_t(\mathbf{s})$ can then be understood as a measure of its contribution to the estimation of the output throughout the adaptation. Consequently, the elimination of samples with a presence P_t which is below a predefined threshold will lead to a dictionary that accurately represents the current operating region, and will yield an algorithm that requires fewer operations. In this way, the presence-based elimination rule can be stated as a dictionary update of the form

$$\mathcal{D}_t = \{\mathbf{s} \in \mathcal{D}_{t-1} : P_t(\mathbf{s}) \geq \delta_p\}, \tag{3.35}$$

where $\delta_p > 0$ is a parameter controlling the size of the operating region represented by the algorithm dictionary.

One pragmatic sparsification approach is to build the dictionary using either the ALD or coherence criteria, and then perform the elimination stage in eq. (3.35) at a lower rate than that of the addition/rejection stage.

3.3.2 Kernel Ridge Regression

With a dictionary $\mathcal{D} = \{\mathbf{s}^i\}_{i=1,\ldots,N}$ and a set of training pairs $T = \{\mathbf{x}_t, \mathbf{d}_t\}_{t=1,\ldots,M}$, we can now apply the SVR paradigm to ridge regression[9] [Drucker et al., 1996, Saunders et al., 1998]. Recall that the aim is to design an estimator $\mathbf{y}_t$, as a function of the regressor $\mathbf{x}_t$, that is as close as possible to the target signal $\mathbf{d}_t$.

As illustrated in in Chapter 2, eq. (2.15), the representer theorem allows us to write the SVR estimate in the form

$$\mathbf{y}_t = \sum_{i=1}^{N} \mathbf{a}_i K(\mathbf{s}^i, \mathbf{x}_t) \tag{3.36}$$

where $\mathbf{a}_i \in \mathbb{R}^m$.

By considering the kernel evaluations as regressors, the ridge regression estimate

[9] Observe that the support vector are not necessarily equal to the training samples.

is obtained via the minimisation of the regularised cost function as in Section 3.2.2

$$J = \frac{1}{2}\sum_{t=1}^{M}\left\|\mathbf{d}_t - \sum_{j=1}^{N}\mathbf{a}_j K(\mathbf{s}^j, \mathbf{x}_t)\right\|^2 + \frac{\rho}{2}\sum_{j=1}^{N}\|\mathbf{a}_j\|^2 \tag{3.37}$$

where ρ is the regularisation factor. By identifying the parameters $\mathbf{a}_j = [a_{1j}, a_{2j}, \ldots, a_{mj}]^T$ and writing the norm coordinate-wise[10], we can write the cost function as an explicit function of the parameters a_{ij}

$$J = \frac{1}{2}\sum_{t=1}^{M}\sum_{i=1}^{m}\left(\{\mathbf{d}_t\}_i - \sum_{j=1}^{N} a_{ij}K(\mathbf{s}^j, \mathbf{x}_t)\right)^2 + \frac{\rho}{2}\sum_{j=1}^{N}\sum_{i=1}^{m} a_{ij}^2 \tag{3.38}$$

and find the gradient

$$\begin{aligned}\frac{\partial J}{\partial a_{pq}} &= -\sum_{t=1}^{M}\left(\{\mathbf{d}_t\}_p - \sum_{j=1}^{N} a_{pj}K(\mathbf{s}^j, \mathbf{x}_t)\right)K(\mathbf{s}^q, \mathbf{x}_t) + \rho a_{pq} \\ &= -\sum_{t=1}^{M}\mathbf{E}_{pt}\mathbf{K}_{tq} + \rho a_{pq} \\ &= -\{\mathbf{EK}\}_{pq} + \rho a_{pq}\end{aligned} \tag{3.39}$$

where we have denoted $\mathbf{E}_{pt} = \{\mathbf{d}_t\}_p - \sum_{j=1}^{N} a_{pj}K(\mathbf{s}^j, \mathbf{x}_t)$ the p-th coordinate of the error at time t and $\mathbf{K}_{tq} = K(\mathbf{s}^q, \mathbf{x}_t)$ the kernel evaluation between support vector $\mathbf{s}^t$ and signal at time $\mathbf{x}_t$, that is,

$$\mathbf{K} = \begin{bmatrix} K(\mathbf{s}^1, \mathbf{x}_1) & \cdots & K(\mathbf{s}^N, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ K(\mathbf{s}^1, \mathbf{x}_M) & \cdots & K(\mathbf{s}^N, \mathbf{x}_M) \end{bmatrix}. \tag{3.40}$$

By denoting $\mathbf{a} = [\mathbf{a}_1, \ldots, \mathbf{a}_N]$, the gradient in eq. (3.39) can be expressed in matrix form as $\frac{\partial J}{\partial \mathbf{a}} = -\mathbf{EK} + \rho\mathbf{a}$ and the error matrix as $\mathbf{E} = \mathbf{D} - \mathbf{a}\mathbf{K}^T$, where $D = [\mathbf{d}_1, ..., \mathbf{d}_M]$ is the observation matrix. Therefore, the optimality condition can be written as $\rho\mathbf{a} = \left(\mathbf{D} - \mathbf{a}\mathbf{K}^T\right)\mathbf{K}$, which gives

$$\mathbf{a} = \mathbf{DK}\left(\rho\mathbb{I} + \mathbf{K}^T\mathbf{K}\right)^{-1}. \tag{3.41}$$

Since diagonally dominant matrices are invertible, the existence of the optimal solution in (3.41) can be ensured by controlling the invertibility of the matrix $\rho\mathbb{I} + \mathbf{K}^T\mathbf{K}$ through the regularisation parameter ρ. Furthermore, in order for this solution to be physically meaningful, a sufficient condition for the existence of the solution of the non-regularised problem (i.e. $\rho = 0$) is presented in the next lemma.

[10]Recall the the notation $\{\mathbf{A}\}_{ij}$ corresponds to the (i, j) entry of the array $\mathbf{A}$.

Lemma 1. *For an infinite-dimensional RKHS, the matrix* $\mathbf{K}^T\mathbf{K}$ *is invertible if*

(i) the number of training samples is greater than or equal to that of the support vectors ($M \geq N$*), and*

(ii) the support vectors are chosen according to the coherence sparsification criterion,[11]

thus guaranteeing the existence of the solution for the non-regularised problem.

Proof. According to the Mercer theorem [Mercer, 1909], the kernel K can be expanded as $K(\mathbf{s}, \mathbf{x}) = \phi(\mathbf{s})^H\phi(\mathbf{x})$; furthermore, denote $\Phi(\mathbf{s}) = [\phi(\mathbf{s}^1), ..., \phi(\mathbf{s}^N)]$ and $\Phi(\mathbf{x}) = [\phi(\mathbf{x}_1), ..., \phi(\mathbf{x}_M)]$. By choosing the support vectors according to the coherence criterion, Proposition 1 in [Richard et al., 2009] ensures linear independence of the kernel functions $\phi(\mathbf{x}_i), \phi(\mathbf{x}_j), \mathbf{x}_i \neq \mathbf{x}_j$, hence

$$\text{rank}\,(\Phi(\mathbf{s})) = N, \text{ and rank}\,(\Phi(\mathbf{x})) = M. \tag{3.42}$$

We now write the kernel evaluation matrix $\mathbf{K}$ in (3.40) in an inner product form by

$$\mathbf{K} = \Phi^T(\mathbf{x})\Phi(\mathbf{s}), \tag{3.43}$$

where $\text{rank}\left(\mathbf{K}^T\mathbf{K}\right) = \text{rank}\,(\mathbf{K}) = \min(N, M)$. As a consequence, by choosing $M \geq N$ the matrix $\mathbf{K}^T\mathbf{K} \in \mathbb{R}^{N\times N}$ is full rank ($\text{rank}\left(\mathbf{K}^T\mathbf{K}\right) = N$) and invertible, meaning that the optimal solution of the non-regularised problem exists and can be computed from (3.41) by setting $\rho = 0$. □

The kernel ridge regression (KRR) algorithm exploits the nonlinear approximation ability of reproducing kernels and has similar complexity to the standard ridge regression algorithm—except for the number of support vectors and kernel evaluations. Its derivation is also straightforward and follows naturally from incorporating the SVR paradigm in Section 2.2 into the ridge regression setting in Section 3.2.2.

3.3.3 Kernel Least Mean Square

The feature-space properties of SVR also make it possible to derive LMS-based nonlinear estimators operating on RKHSs that are well-suited for nonstationary environments, we refer to this approach as KLMS [Liu et al., 2008].

We proceed from a feature space standpoint and consider the infinite-dimensional

[11] See Section 3.3.1 for the coherence sparsification criterion.

map $\phi(\cdot)$ given by

$$\begin{aligned} \phi : \mathbb{R}^n &\longrightarrow \mathcal{H} \\ \mathbf{x} &\longmapsto \phi_{\mathbf{x}} \end{aligned} \tag{3.44}$$

where $\mathcal{H}$ is an RKHS, to produce the estimate[12]

$$\mathbf{y}_t = \mathbf{A}\phi_{\mathbf{x}_t} \tag{3.45}$$

where the weight $\mathbf{A}$ is an array composed of m transposed elements of $\mathcal{H}$ so that $\mathbf{y}_t = \mathbf{A}\phi_{\mathbf{x}_t} \in \mathbb{R}^m$. As the algorithm is linear on the feature samples, the optimal weight can be found on a gradient-based fashion input samples $\mathbf{x}_t$, or feature samples $\phi_{\mathbf{x}_t}$, become available, this yields the LMS update rule

$$\mathbf{A}_t = \mathbf{A}_{t-1} + \bar{\mu}\mathbf{e}_t\phi_{\mathbf{x}_t}^T, \tag{3.46}$$

where $\bar{\mu} > 0$ is a step size.

Observe that although the update law in (3.46) is theoretically correct, it cannot be implemented in practice due to the infinite dimensionality of the regressors $\phi_{\mathbf{x}_t}$. In order **not to perform** algebraic operations in the feature space, we set out to restate the algorithm in terms of inner products that can be replaced by kernel evaluations. Consider the update rule (3.46) in a non-recursive fashion

$$\mathbf{A}_t = \bar{\mu}\sum_{j=1}^{t}\mathbf{e}_j\phi_{\mathbf{x}_j}^T. \tag{3.47}$$

This allows us to write the estimate in eq. (3.45) as

$$\begin{aligned} \mathbf{y}_t &= \left(\bar{\mu}\sum_{j=1}^{t-1}\mathbf{e}_j\phi_{\mathbf{x}_j}^T\right)\phi_{\mathbf{x}_t} \\ &= \bar{\mu}\sum_{j=1}^{t-1}\mathbf{e}_j\phi_{\mathbf{x}_j}^T\phi_{\mathbf{x}_t}. \end{aligned} \tag{3.48}$$

Finally, as the mapping ϕ yields a RKHS, the product $\phi_{\mathbf{x}_j}^T\phi_{\mathbf{x}_t}$ can be computed through the kernel trick [Aizerman et al., 1964], this gives [Liu et al., 2008]

$$\mathbf{y}_t = \bar{\mu}\sum_{j=1}^{t-1}\mathbf{e}_j K(\mathbf{x}_j, \mathbf{x}_t), \tag{3.49}$$

[12]Observe we consider a tensor product instead of the inner product used by the standard SVR formulation, this is because we are addressing the case of vector-valued target process $\mathbf{d}_t \in \mathbb{R}^m$.

where K is the reproducing kernel of $\mathcal{H}$ [Aronszajn, 1950].

Observe that the estimator in (3.49) becomes more computationally demanding at each time step, and *learns* by adding terms rather than by updating its parameters. These issues can be addressed by introducing a sparsification criterion before computing the optimal coefficients, and, based on the representer theorem [Kimeldorf and Wahba, 1971, B. Schölkopf and Williamson, 2000], write

$$\mathbf{y}_t = \sum_{j=1}^{N} \mathbf{a}_j K(\mathbf{s}^j, \mathbf{x}_t). \tag{3.50}$$

This representation allows us to choose the weights $\mathbf{a}_j$ and support vectors $\mathbf{s}^j$ by minimising $\|\mathbf{d}_t - \mathbf{y}_t\|^2$ constrained to the coherence sparsification criterion [Richard et al., 2009].

The parameter update rule can be then stated according to whether a new sample $\mathbf{x}_t$ is included as support vector in the following manner:

$$\text{Sample not added:} \quad \mathbf{a} \leftarrow \mathbf{a} + \eta \left(\mathbf{d}_t - \mathbf{a}\mathbf{h}_t\right) \mathbf{h}_t^T \tag{3.51}$$

$$\text{Sample added:} \quad \mathbf{a} \leftarrow [\mathbf{a},\ 0] + \eta \left(\mathbf{d}_t - [\mathbf{a},\ 0]\, \mathbf{h}_t\right) \mathbf{h}_t^T. \tag{3.52}$$

where $\mathbf{h}_t = [K(s^1, \mathbf{x}_t), \ldots, K(s^N, \mathbf{x}_t)]^T$. Observe that for unit-norm kernels[13] such as the Gaussian kernel, when a new sample is added, eq. (3.52) assigns the new weight as the original KLMS algorithm, $\mathbf{a}_j = \eta \mathbf{e}_t$.

The KLMS can also be implemented in a normalised way, where the step size is normalised by the square norm of the regressor. In this case, the normalised KLMS is given by [Richard et al., 2009]

$$\text{Sample not added:} \quad \mathbf{a} \leftarrow \mathbf{a} + \frac{\eta}{\epsilon + \|\mathbf{h}_t\|^2} \left(\mathbf{d}_t - \mathbf{a}\mathbf{h}_t\right) \mathbf{h}_t^T \tag{3.53}$$

$$\text{Sample added:} \quad \mathbf{a} \leftarrow [\mathbf{a},\ 0] + \frac{\eta}{\epsilon + \|\mathbf{h}_t\|^2} \left(\mathbf{d}_t - [\mathbf{a},\ 0]\, \mathbf{h}_t\right) \mathbf{h}_t^T. \tag{3.54}$$

where $\epsilon > 0$ is a regularisation term.

The KLMS can be applied in a general class scenarios due to its feature-space nature, yet it is simple and straightforward to implement. Additionally, the KLMS solves the problem of learning sequentially from large datasets, where batch methods are prohibitively expensive [Liu et al., 2008].

[13] That is, $K(\mathbf{x}, \mathbf{x}) = \langle K_{\mathbf{x}}, K_{\mathbf{x}} \rangle = \|K_{\mathbf{x}}\|^2 = 1, \forall \mathbf{x} \in X$.

3.3.4 Kernel Recursive Least Squares

The kernel extension of the RLS algorithm (KRLS) proposed in [Engel et al., 2004] considers the growing-window version of the linear algorithm, that is, where the forgetting factor is set to $\lambda = 1$. The KRLS algorithm performs ALD sparsification directly in the feature space and then finds optimal weights of the estimator $\mathbf{y}_t = \sum_{i=1}^{N} \mathbf{a}_i K(\mathbf{s}^i, \mathbf{x}_t)$ in an RLS fashion.

Within KRLS, the weight update for both the parameters $\mathbf{a}$ and the covariance matrix $\mathbf{P}$ is based on whether a new sample is added to the dictionary according to

$$\mathbf{a} \leftarrow \left[\mathbf{a} + \delta^{-1}\mathbf{e}_{t+1}\mathbf{b}^T,\ \delta^{-1}\mathbf{e}_{t+1}\right], \quad \mathbf{P} \leftarrow \begin{bmatrix} \mathbf{P} & 0 \\ 0 & 1 \end{bmatrix} \tag{3.55}$$

or when the dictionary remains unchanged according to

$$\mathbf{a} \leftarrow \mathbf{a} + \frac{\mathbf{e}_{t+1}\mathbf{b}^T\mathbf{P}\mathbf{K}^{-1}}{1 + \mathbf{b}^T\mathbf{P}\mathbf{b}}, \quad \mathbf{P} \leftarrow \mathbf{P} - \frac{\mathbf{P}\mathbf{b}\mathbf{b}^T\mathbf{P}}{1 + \mathbf{b}^T\mathbf{P}\mathbf{b}} \tag{3.56}$$

where the error $\mathbf{e}_{t+1} = \mathbf{d}_{t+1} - \mathbf{y}_{t+1}$, and the quantities δ and $\mathbf{b}$ are calculated from approximate linear dependence sparsification [Engel et al., 2002]. See eqs. (3.29) and (3.30) in Section 3.3.1.

Due to the unsupervised nature of the ALD sparisifcation, the KRLS algorithm is robust to measurement noise and does not require to make assumptions that may not hold in real-world applications. Furthermore, observe that the KRLS can be implemented alongside any online sparsification criteria [Engel et al., 2004].

Chapter 4

Hypercomplex Kernels

I regard it as an inelegance, or imperfection, in quaternions, or rather in the state to which it has been hitherto unfolded, whenever it becomes or seems to become necessary to have recourse to x, y, z, etc.

-Sir William Rowan Hamilton.
(Quoted in a letter from Peter Tait to Arthur Cayley, 1894.)

The kernel learning paradigm relies on the generality of infinite-dimensional RKHS to extract rich data representations, we aim to further exploit this concept by adopting even higher-dimensional RKHS. This is achieved by considering RKHS comprising complex- and quaternion-valued elements, as they correspond to feature transformations comprising two and four standard real-valued transformation respectively. For the complex-valued case the extension is straightforward, since the standard RKHS theory admits complex-valued kernels and the analysis is towards kernel-design only. For quaternions, however, the extension is more challenging, as a sound theory for quaternion-valued RKHS has not yet been developed—this is a theoretical requirement for the design of quaternion kernels and their use in kernel adaptive filtering.

4.1 Complex-Valued Kernels

Complex-valued algorithms have become popular in adaptive signal processing due to the attractive properties that the complex domain offers for dealing with coupled variables and phase information [Mandic and Goh, 2009]. In nonstationary environments, the augmented complex LMS (ACLMS) [Javidi et al., 2008] is a *de facto* resource due to its generality and ease of implementation; however, further research to perform nonlinear estimation using complex-valued algorithms, thereby connecting universal function approximation and the ability to utilise full potential of the complex domain, is the next important step.

The standard RKHS theory [Mercer, 1909, Aronszajn, 1950] admits complex-valued feature spaces and therefore provides theoretical support for the implementation of complex-valued kernel algorithms. Nevertheless, only real-valued kernels are considered by most kernel-estimation algorithms; this is the case from the original classification applications to more recent signal processing ones, where usually the (real-valued) polynomials or Gaussian kernel are implemented. In addition to their additional degrees of freedom, complex-valued kernels are particularly well-suited in complex-valued signal processing, since they perform estimation on the feature space where data resides, that is, the complex field $\mathbb{C}$. In this way, full advantage inherent to the topology of complex spaces is taken of [Mandic and Goh, 2009]. Kernel-based algorithms are readily being developed for complex signals and have seen application in wind prediction [Kuh and Mandic, 2009] and channel equalisation [Bouboulis and Theodoridis, 2011]. Further research in this direction aims to both investigate the design of novel complex-valued kernels and to study the existence of augmented complex-valued kernel estimators; as these will provide a powerful and general framework exploiting noncircularity in complex-valued feature spaces.

4.1.1 The Complex-Valued Gaussian Kernel

A complex-valued kernel inspired by the real-valued Gaussian kernel in eq. (2.19) is given by [Steinwart et al., 2006]

$$K_{\mathbb{C}G}(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{(\mathbf{x}-\mathbf{y}^*)^T(\mathbf{x}-\mathbf{y}^*)}{\sigma^2}\right), \tag{4.1}$$

where $\sigma > 0$ is the kernel parameter. Although $K_{\mathbb{C}G}$ is Hermitian and positive definite, observe that it is not an RBF kernel and therefore does not give an intuitive measure of "similarity" as its real-valued counterpart in in eq. (2.19).

The complex-valued nature of $K_{\mathbb{C}G}$ allows for recognising rotation-sensitive deviations due to the extra degree of freedom, and follows from[1] the complex-valued product in its argument $(\mathbf{x}-\mathbf{y}^*)^T(\mathbf{x}-\mathbf{y}^*)$. By denoting $\mathbf{e}_r = \Re\{\mathbf{x}-\mathbf{y}^*\}, \mathbf{e}_i = \Im\{\mathbf{x}-\mathbf{y}^*\}$, where the operators $\Re$ and $\Im$ denote respectively the real and imaginary parts of a complex number, we can rewrite (4.1) as

$$\begin{aligned} K_{\mathbb{C}G}(\mathbf{x},\mathbf{y}) &= \exp\left(-\frac{\mathbf{e}_r^T\mathbf{e}_r - \mathbf{e}_i^T\mathbf{e}_i + j2\mathbf{e}_r^T\mathbf{e}_i}{\sigma^2}\right) \\ &= \exp\left(\frac{\|\mathbf{e}_i\|^2 - \|\mathbf{e}_r\|^2}{\sigma^2}\right)\left(\cos\left(\frac{2\mathbf{e}_r^T\mathbf{e}_i}{\sigma^2}\right) - j\sin\left(\frac{2\mathbf{e}_r^T\mathbf{e}_i}{\sigma^2}\right)\right). \end{aligned}$$

This reveals that the complex Gaussian kernel grows exponentially with $\|\mathbf{e}_i\|^2 - \|\mathbf{e}_r\|^2$.

[1] Recall that the complex exponential has the property $\exp(a+jb) = e^a(\cos b + j\sin b)$.

Observe that, as $K_{\mathbb{C}G}$ grows unbounded when $\|\mathbf{e}_i\|^2 >> \|\mathbf{e}_r\|^2$, the kernel estimate deviates considerably for inputs from regions that are not yet learnt, this can boost the learning in gradient-based learning algorithms—this similar instability property is also found in polynomial kernels, see Section 2.3.2. Furthermore, despite its enhanced estimation capability, it is rather difficult to find a physically-meaningful interpretation of $K_{\mathbb{C}G}(\mathbf{x}, \mathbf{y})$ in terms of the samples $\mathbf{x}, \mathbf{y}$ as it is the case with its real-valued counterpart K_G. Fig. 4.1 shows a contour plot for $K_{\mathbb{C}G}$ using $\sigma^2 = 10^3$.

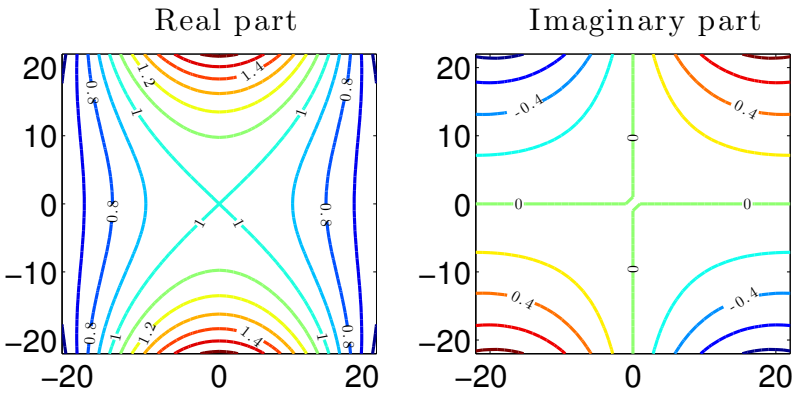

Figure 4.1: Contour plot of real and imaginary parts of the complex Gaussian kernel $K_{\mathbb{C}G}$ in eq. (4.1) and $\sigma^2 = 10^3$.

4.1.2 Complexification of Real-Valued Kernels

The main stumbling block for the dissemination of complex-valued kernel algorithms is the availability of complex-valued kernels. We address this issue by presenting a novel family of complex-valued kernels, termed *independent complex kernels*; this is achieved through the *complexification* of existing (real kernel) RKHSs.

Consider the real-valued kernel K defined over the set X. Through the Mercer theorem [Mercer, 1909], there is a basis $\{\phi\}$ for the RKHS such that $K(\mathbf{x}, \mathbf{y}) = \langle \phi_{\mathbf{x}}, \phi_{\mathbf{y}} \rangle$ and

$$\begin{aligned} \phi : X &\longrightarrow H \\ \mathbf{x} &\longmapsto \phi_{\mathbf{x}}. \end{aligned} \tag{4.2}$$

Based on the mapping ϕ, we define a complex-valued mapping over the complex set $\overline{X} = \{\mathbf{x}_r + j\mathbf{x}_i : \mathbf{x}_r, \mathbf{x}_i \in X\}$ by

$$\begin{aligned} \Phi : \overline{X} &\longrightarrow \overline{H} \\ \mathbf{x}_r + j\mathbf{x}_i &\longmapsto \phi_{\mathbf{x}_r} + j\phi_{\mathbf{x}_i}, \end{aligned} \tag{4.3}$$

where $\overline{H} = \{\phi_1 + j\phi_2, s.t.\ \phi_1, \phi_2 \in H\}$ is a space composed of complex-valued functions, the real and imaginary parts of which are elements of H.

Lemma 2 (Independent Complex RKHS)**.** *The space*

$$\overline{H} = \{\Phi_{\mathbf{x}_r+j\mathbf{x}_i} = \phi_{\mathbf{x}_r} + j\phi_{\mathbf{x}_i}, s.t.\ \mathbf{x}_r, \mathbf{x}_i \in X\}$$

is a RKHS of complex-valued functions defined on $\overline{X} = \{\mathbf{x}_r + j\mathbf{x}_i, s.t.\ \mathbf{x}_r, \mathbf{x}_i \in X\}$

Proof. We proceed by first proving that the so-defined space $\overline{H}$ is a Hilbert space and then showing that the evaluation functional is continuous.

By construction, $\overline{H}$ is a complete vector-space of complex-valued functions: The elements of the sequence $\{h_i\}_{i\in\mathbb{N}} \in \overline{H}$ can be written as $h_i = f_i + jg_i$, where the real-valued sequences $\{f_i\}_{i\in\mathbb{N}}, \{g_i\}_{i\in\mathbb{N}} \in H$ and their limits $f, g \in H$ (since H is complete), therefore, $\lim h_i = f + jg$ is by definition in $\overline{H}$, meaning that $\overline{H}$ is complete. Furthermore, by equipping $\overline{H}$ with the inner product $\langle h_1, h_2\rangle = \int_X h_1(\mathbf{x})h_2^*(\mathbf{x})\mathrm{d}\mathbf{x}$, it becomes a Hilbert space.

Consider now the evaluation functional on $\overline{H}$,

$$\begin{aligned} L_{\mathbf{x}_r+j\mathbf{x}_i}(\Phi) &= \Phi_{\mathbf{x}_r+j\mathbf{x}_i} \\ &= \phi_{\mathbf{x}_r} + j\phi_{\mathbf{x}_i} \\ &= L_{\mathbf{x}_r}(\phi) + jL_{\mathbf{x}_i}(\phi). \end{aligned}$$

This means that the evaluation functional in $\overline{H}$ can be expressed as the sum of two evaluation functionals in H. As H is a RKHS, its evaluation functional is linear and therefore so is the evaluation functional in $\overline{H}$. We have then shown that $\overline{H}$ is an RKHS with a continuous evaluation functional, that is, a RKHS. □

According to the Riesz representation theorem, there is a unique reproducing kernel for the so-constructed space $\overline{H}$. We denote this kernel as $K_{\mathbb{C}}$, and express it in explicit form by calculating the inner product between the elements $\Phi_{\mathbf{x}}$ and $\Phi_{\mathbf{y}}$, that is,

$$\begin{aligned} K_{\mathbb{C}}(\mathbf{x}, \mathbf{y}) &= \langle \Phi_{\mathbf{x}_r+j\mathbf{x}_i}, \Phi_{\mathbf{y}_r+j\mathbf{y}_i}\rangle \\ &= \langle \phi_{\mathbf{x}_r} + j\phi_{\mathbf{x}_i}, \phi_{\mathbf{y}_r} + j\phi_{\mathbf{y}_i}\rangle \\ &= \langle \phi_{\mathbf{x}_r}, \phi_{\mathbf{y}_r}\rangle + \langle \phi_{\mathbf{x}_i}, \phi_{\mathbf{y}_i}\rangle + j\left(\langle \phi_{\mathbf{x}_i}, \phi_{\mathbf{y}_r}\rangle - \langle \phi_{\mathbf{x}_r}, \phi_{\mathbf{y}_i}\rangle\right) \\ &= K(\mathbf{x}_r, \mathbf{y}_r) + K(\mathbf{x}_i, \mathbf{y}_i) + j\big(K(\mathbf{x}_i, \mathbf{y}_r) - K(\mathbf{x}_r, \mathbf{y}_i)\big), \end{aligned} \tag{4.4}$$

where K is the generating kernel associated to the RKHS H, that is, $K(\mathbf{x}, \mathbf{y}) = \langle \phi_{\mathbf{x}}, \phi_{\mathbf{y}}\rangle$.

Theorem 2. *For any real-valued kernel K defined on the set X of real-valued vectors, the kernel given by*

$$K_{\mathbb{C}}(\mathbf{x}, \mathbf{y}) = K(\mathbf{x}_r, \mathbf{y}_r) + K(\mathbf{x}_i, \mathbf{y}_i) + j\big(K(\mathbf{x}_i, \mathbf{y}_r) - K(\mathbf{x}_r, \mathbf{y}_i)\big) \tag{4.5}$$

where $\mathbf{x} = \mathbf{x}_r + j\mathbf{x}_i$ *and* $\mathbf{y} = \mathbf{y}_r + j\mathbf{y}_i$*, is a complex-valued positive-definite kernel defined on the set of complex-valued vectors* $\overline{X} = \{\mathbf{x}_r + j\mathbf{x}_i,\ s.t.\ \mathbf{x}_r, \mathbf{x}_i \in X\}$.

Proof. The positive definiteness of the kernel K follows form eq. (4.4), as inner product forms are positive definite. □

Observe that for any arbitrary real kernel $K(\mathbf{x}, \mathbf{y})$ which provides a measure of deviation (cf. similarity) of arguments $\mathbf{x}$ and $\mathbf{y}$, the independent complex kernel $K_{\mathbb{C}}$ in (4.5) inherits this property and can thus be considered a generic complex-valued extension of K. A particular case is obtained when the real kernel K is chosen to be the Gaussian kernel. In such case, the independent complex-valued kernel $K_{\mathbb{C}}$ in (4.5) has an associated physical meaning: its real part accounts for the magnitude of the deviation of the samples while its imaginary part conveys a notion of the phase of such deviation. Fig. 4.2 shows the contour plot of an independent complex kernel based on the real Gaussian kernel using $\sigma^2 = 10^3$.

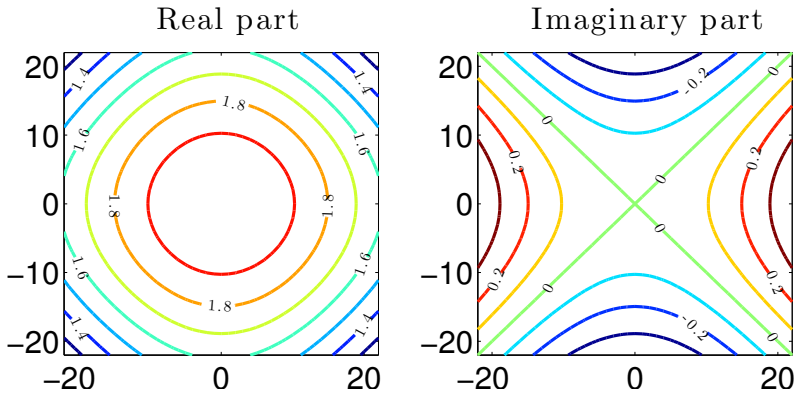

Figure 4.2: Independent complex kernel in eq. (4.1) generated from a real-valued Gaussian with kernel width $\sigma^2 = 10^3$.

4.2 Quaternion-Valued Kernels

A natural step to follow now is to consider quaternion-valued kernels. Quaternions [Hamilton, 1844] are a 4D noncommutative division algebra built on the real field and have already shown advantages over real-valued vectors within signal processing owing to their enhanced modelling of rotation, orientation, and cross-information between multichannel data. However, quaternion kernel estimation is in its infancy and quaternion RKHSs require a rigorous existence and uniqueness analysis, that enable kernel algorithms operating on quaternion-valued feature spaces.

We will then first revisit the quaternion ring and quaternion left Hilbert spaces in order to define the quaternion RKHS (QRKHS), to then present Quaternion versions of

the Riesz representation [Friedman, 1982] and Moore-Aronszajn [Aronszajn, 1950] theorems. This equips us with a theoretical basis for quaternion kernel estimation, whereby the feature space has a corresponding quaternion-valued reproducing kernel. We also give examples of two quaternion-valued kernels.

4.2.1 Background on Quaternion Vector Spaces

The Quaternion Division Ring

The quaternion set $\mathbb{H}$ is a four-dimensional vector space over the real field $\mathbb{R}$ spanned by the linearly independent basis $\{1, i, j, k\}$ [Hamilton, 1844]. Accordingly, any element $q \in \mathbb{H}$ can be written as a linear combination $q = a1 + bi + cj + dk$, where $a, b, c, d \in \mathbb{R}$.

The sum and the scalar multiplication are defined in an element-wise fashion as in $\mathbb{R}^4$, that is

$$\begin{pmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \end{pmatrix} + \begin{pmatrix} a_2 \\ b_2 \\ c_2 \\ d_2 \end{pmatrix} = \begin{pmatrix} a_1 + a_2 \\ b_1 + b_2 \\ c_1 + c_2 \\ d_1 + d_2 \end{pmatrix} \tag{4.6}$$

$$\alpha(a, b, c, d) = (\alpha a, \alpha b, \alpha c, \alpha d),\ \alpha \in \mathbb{R}$$

where the notation $(a, b, c, d) = (a, b, c, d)^T = a1 + bi + cj + dk \in \mathbb{H}$ is used for convenience of presentation.

Remark 1. *The pair $(\mathbb{H}, +)$ is an Abelian group [Friedman, 1982], for which the addition operation is defined in (4.6) and the additive identity is $0 = (0, 0, 0, 0) \in \mathbb{H}$.*

The *quaternion multiplication* (or *Hamilton product*) is a bilinear mapping $\mathbb{H} \times \mathbb{H} \to \mathbb{H}$, $(p, q) \mapsto pq$, defined by

$$pq = \begin{pmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \end{pmatrix} \begin{pmatrix} a_2 \\ b_2 \\ c_2 \\ d_2 \end{pmatrix} = \begin{pmatrix} a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2 \\ a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2 \\ a_1c_2 - b_1d_2 + c_1a_2 + d_1b_2 \\ a_1d_2 + b_1c_2 - c_1b_2 + d_1a_2 \end{pmatrix}. \tag{4.7}$$

Remark 2. *The quaternion product defined in (4.7) distributes over the sum, i.e. $\forall p, q, r \in \mathbb{H}$*

$$p(q + r) = pq + pr$$
$$(p + q)r = pr + qr.$$

It is also possible to express the quaternion multiplication using the basis expansion representation, that is, $(a_1 1 + b_1 i + c_1 j + d_1 k)(a_2 1 + b_2 i + c_2 j + d_2 k)$, and applying

the multiplication rule

$$i^2 = j^2 = k^2 = ijk = -1.$$

Note that the basis element $1 = (1, 0, 0, 0) \in \mathbb{H}$ is the multiplicative identity, meaning that $q1 = 1q = q, \forall q \in \mathbb{H}$, and is therefore omitted in the basis representation, $q = a+bi+cj+dk$. We refer to the factor of $(1, 0, 0, 0)$ as *real part of* q, denoted by $\Re\{q\} = a$, and to the remaining factors as the *imaginary part of* q, denoted by $\Im\{q\} = (0, b, c, d)$.

For any given element $q \in \mathbb{H}$, $q \neq 0$, its multiplicative inverse $q^{-1} \in \mathbb{H} \setminus \{0\}$ is given by

$$q^{-1} = \frac{q^*}{\|q\|^2},$$

where $q^* = (a, -b, -c, -d)$ denotes the conjugate of q, and $\|q\| = \sqrt{q^*q} = \sqrt{qq^*} = \sqrt{a^2 + b^2 + c^2 + d^2}$ denotes the norm in $\mathbb{H}$ defined as the Euclidean norm in $\mathbb{R}^4$; as a consequence, $qq^{-1} = q^{-1}q = 1, \forall q \neq 0$. By using the conjugate operator, the real and imaginary parts of $q \in \mathbb{H}$ can be written respectively as

$$\Re\{q\} = \frac{q + q^*}{2}, \ \Im\{q\} = \frac{q - q^*}{2}.$$

Remark 3. *The pair* $(\mathbb{H}, \cdot)$ *equipped with the identity element is a monoid under multiplication, while the inclusion of the multiplicative inverse makes* $(\mathbb{H} \setminus \{0\}, \cdot)$ *a group [Jacobson, 1943, Jacobson, 2009].*

Remark 4. *Since* $(\mathbb{H}, +)$ *is an Abelian group (Remark 1),* $(\mathbb{H}, \cdot)$ *is a group (Remark 3), and the quaternion product distributes over the sum (Remark 2), the triplet* $(\mathbb{H}, +, \cdot)$ *is a non-commutative division ring [Jacobson, 2009].*

Despite the lack of commutativity in $\mathbb{H}$, its division ring properties establish the basis for the design of estimation algorithms. Furthermore, $\mathbb{H}$ is one of the four normed division algebras over the real field, the other three being the real field $\mathbb{R}$, the complex field $\mathbb{C}$, and the non-associative unitary octonion ring $\mathcal{O}$ (see the Frobenius theorem [Frobenius, 1878]).

Quaternion-Valued Hilbert Spaces

To introduce the concept of quaternion Hilbert space, we first need to define quaternion vector spaces and their algebraic properties.

Since $(\mathbb{H}, +, \cdot)$ is a division ring **and not a field** (it lacks the commutativity property), strictly speaking it is not possible to construct a general vector space over $\mathbb{H}$; however, we can still construct a *left-module*. A module [Anderson and Fuller, 1992, Adkins and Weintraub, 1992] is a generalisation of vector space which allows for the scalar set to be a ring (rather than a field). We refer to a left-module $\mathcal{H}$ over $\mathbb{H}$ as vector

space [Friedman, 1982] in which the **non-commutative** scalar multiplication $\mathbb{H} \times \mathcal{H} \to \mathcal{H}$ is defined on the left hand side by $(q, \mathbf{x}) \mapsto q\mathbf{x}$.

We next set out to restate the concepts of inner product and left Hilbert space for quaternions, as these are required to define quaternion-valued RKHSs.

Definition 3 (Quaternion left Hilbert space)**.** *A nonempty set $\mathcal{H}$ is called a quaternion left Hilbert space if it is a quaternion left module (i.e. built over $\mathbb{H}$) and there exists a quaternion-valued function $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{H}$ with the following properties:*

1. *Conjugate symmetry: $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle^*$.*
2. *Linearity: $\langle p\mathbf{x} + q\mathbf{y}, \mathbf{z} \rangle = p \langle \mathbf{x}, \mathbf{z} \rangle + q \langle \mathbf{y}, \mathbf{z} \rangle$.*
3. *Conjugate linearity: $\langle \mathbf{x}, p\mathbf{y} + q\mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle p^* + \langle \mathbf{x}, \mathbf{z} \rangle q^*$.*
4. *$\langle \mathbf{x}, \mathbf{x} \rangle \geq 0$ and $\langle \mathbf{x}, \mathbf{x} \rangle = 0 \iff \mathbf{x} = 0$.*
5. *Completeness: If $\{\mathbf{x}_n\} \subset \mathcal{H}$ is a Cauchy sequence, then $\mathbf{x} = \lim_{n \to \infty} \mathbf{x}_n \in \mathcal{H}$.*

We refer to the function $\langle \cdot, \cdot \rangle$ as inner product and denote its induced norm by $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$.

Observation 1. *The space $\mathbb{H}^n$, with the inner product $\langle p, q \rangle = p^T q^*$ is a quaternion left Hilbert space.*

Observation 2. *The space of quaternion-valued square-integrable functions $L_2 = \{f : X \in \mathbb{H}^n \to \mathbb{H}, s.t. \int_X \|f(\mathbf{x})\|^2 \, d\mathbf{x} < \infty\}$ with the inner product $\langle f, g \rangle = \int_X f(\mathbf{x}) g^*(\mathbf{x}) d\mathbf{x}$ is a quaternion left Hilbert space.*[2]

Standard properties of real and complex Hilbert spaces such as the Cauchy-Schwarz inequality and the concept of orthogonality also extend to quaternion Hilbert spaces. In particular, we highlight two properties that will be helpful in the next section:

- The elements $\mathbf{x}, \mathbf{y} \in \mathcal{H}$ are orthogonal, denoted by $\mathbf{x} \perp \mathbf{y}$, if and only if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$.
- Two sets $A, B \in \mathcal{H}$ are orthogonal if and only if $\mathbf{x} \perp \mathbf{y}, \ \forall \mathbf{x} \in A, \mathbf{y} \in B$. We denote by $A^\perp$ the set of all elements that are orthogonal to $\mathbf{x} \in A$.

For the properties of complex Hilbert spaces which also apply to the introduced left quaternion Hilbert space see [Friedman, 1982].

[2] We considered the quaternion norm $\|q\| = \sqrt{q^* q}$ and the Lebesgue measure $\mathrm{d}\mathbf{x}$ in $\mathbb{H}^n$ defined in analogy to the Lebesgue measure in $\mathbb{R}^{4n}$ (for instance for $n = 1$, $\mathrm{d}\mathbf{x} = \mathrm{d}\mathbf{x}_r \mathrm{d}\mathbf{x}_i \mathrm{d}\mathbf{x}_j \mathrm{d}\mathbf{x}_k$).

4.2.2 Quaternion Reproducing Kernel Hilbert Spaces

We now introduce quaternion reproducing kernel Hilbert spaces to provide both theoretical support and physical insight for the design and implementation of quaternion-valued kernel estimation algorithms.

Definition 4 (Quaternion reproducing kernel Hilbert space)**.** *Let X be an arbitrary set and $\mathcal{H}$ a left quaternion Hilbert space of functions from X to $\mathbb{H}$. We say that $\mathcal{H}$ is a quaternion reproducing kernel Hilbert space (QRKHS) if the (linear) evaluation map*

$$\begin{aligned} L_{\mathbf{x}} : \mathcal{H} &\longrightarrow \mathbb{H} \\ f &\longmapsto f(\mathbf{x}) \end{aligned} \tag{4.8}$$

is bounded $\forall \mathbf{x} \in X$.

Riesz Representation Theorem

We can now introduce the following theorem in order to guarantee the existence of a reproducing kernel for any given QRKHS.

Theorem 3 (Quaternion Riesz representation theorem)**.** *For every bounded linear functional L defined over a quaternion left Hilbert space $\mathcal{H}$, there exists a unique element $g \in \mathcal{H}$ such that $L(f) = \langle f, g \rangle, \forall f \in \mathcal{H}$.*

Proof. The proof follows from [Friedman, 1982, Theorem 6.2.4] and the properties of the inner product in quaternion left Hilbert spaces stated in Definition 3.

Denote by $A = \{f \in \mathcal{H} : L(f) = 0\}$ the null space of L. By continuity[3] of L, A is a closed linear subspace of $\mathcal{H}$. If $A = \mathcal{H}$, then $L = 0$ and $L(f) = \langle f, 0 \rangle$. If $A \neq \mathcal{H}$, then there exists at least one element $g_0 \in \mathcal{H}$, such that $g_0 \neq 0$ and $g_0 \in A^{\perp}$ [Friedman, 1982, Corollary 6.2.3]. By definition of g_0, $L(g_0) \neq 0$, and for any $f \in \mathcal{H}$ the element $f - L(f)\big(L(g_0)\big)^{-1} g_0 \in A$. As a consequence,

$$\left\langle f - L(f)\big(L(g_0)\big)^{-1} g_0, g_0 \right\rangle = 0.$$

Applying the properties of the inner product space we have

$$L(f)\big(L(g_0)\big)^{-1} \langle g_0, g_0 \rangle = \langle f, g_0 \rangle,$$

[3] A bounded linear operator between normed spaces is always continuous, see [Friedman, 1982, Theorem 4.4.2].

then, replacing $\langle g_0, g_0 \rangle = \|g_0\|^2$ and right-multiplying both sides by $\frac{L(g_0)}{\|g_0\|^2}$ yields

$$L(f) = \langle f, g_0 \rangle \frac{L(g_0)}{\|g_0\|^2}. \tag{4.9}$$

Now, by denoting $g = L^*(g_0)g_0/\|g_0\|^2$ we arrive at the desired $L(f) = \langle f, g \rangle$.

To prove uniqueness, assume $g_1, g_2 \in \mathcal{H}$ such that $L(f) = \langle f, g_1 \rangle = \langle f, g_2 \rangle$. Therefore, $\langle f, g_1 - g_2 \rangle = 0$ for all $f \in \mathcal{H}$; in particular, by taking $f = g_1 - g_2$ we have $\|g_1 - g_2\|^2 = 0 \Rightarrow g_1 = g_2$. □

Remark 5. *Observe that the right-multiplication by $\frac{L(g_0)}{\|g_0\|^2}$, which yields Eq. (4.9), holds the key to differentiate the proof for Thm. 3 from that of the complex and real cases. Due to the non-commutative property of the quaternion ring, the element $g = L^*(g_0)g_0/\|g_0\|^2$ is different from $\bar{g} = g_0 L^*(g_0)/\|g_0\|^2$, which is used in the proof for the real/complex cases in [Friedman, 1982, Theorem 6.2.4].*

Corollary 1 (Reproducing property)**.** *For any $f \in \mathcal{H}$, there exists a unique element $K_{\mathbf{x}} \in \mathcal{H}$ such that the evaluation map $L_{\mathbf{x}} = f(\mathbf{x})$ in (4.8) can be expressed as $L_{\mathbf{x}} = \langle f, K_{\mathbf{x}} \rangle$.*

Proof. As $L_{\mathbf{x}}$ is itself a bounded linear operator, based on the quaternion Riesz representation theorem there exists an element $g \in \mathcal{H}$ such that $L_{\mathbf{x}}(f) = \langle f, g \rangle$. The element $g = g(\mathbf{x})$ is unique for a given functional $L_{\mathbf{x}}$, or equivalently, for a given $\mathbf{x} \in X$. Therefore, we can define $K_{\mathbf{x}} \triangleq g$ and write $L_{\mathbf{x}} = \langle f, K_{\mathbf{x}} \rangle$. □

Since $K_{\mathbf{x}}(\cdot) \in \mathcal{H}$, it can be evaluated for any $\mathbf{y} \in X$. This allows us to define

$$\begin{aligned} K : X \times X &\longrightarrow \mathbb{H} \\ (\mathbf{x}, \mathbf{y}) &\longmapsto K(\mathbf{x}, \mathbf{y}) = K_{\mathbf{x}}(\mathbf{y}), \end{aligned}$$

whereby the function K is referred to as the *reproducing kernel* of the QRKHS $\mathcal{H}$. Its existence and uniqueness properties are a direct consequence of the quaternion Riesz representation theorem (Theorem 3). Similarly to the standard real- and complex-valued cases, the reproducing property of K can be expressed as

$$\forall f \in \mathcal{H} \text{ and } \mathbf{x} \in X,\ f(\mathbf{x}) = \langle f, K_{\mathbf{x}} \rangle .$$

The following relationships are readily obtained by applying the reproducing property on the functions $K_{\mathbf{x}} = K(\mathbf{x}, \cdot) \in \mathcal{H}$ and $K_{\mathbf{y}} = K(\mathbf{y}, \cdot) \in \mathcal{H}$:

- $K(\mathbf{x}, \mathbf{y}) = \langle K_{\mathbf{x}}, K_{\mathbf{y}} \rangle = \langle K_{\mathbf{y}}, K_{\mathbf{x}} \rangle^* = K^*(\mathbf{x}, \mathbf{y})$.
- $K(\mathbf{x}, \mathbf{x}) = \langle K_{\mathbf{x}}, K_{\mathbf{x}} \rangle = \|K_{\mathbf{x}}\|^2 \geq 0$.

- $K_{\mathbf{x}} = 0 \iff f(\mathbf{x}) = \langle f, K_{\mathbf{x}} \rangle = 0, \ \forall f \in \mathcal{H}$.

We have therefore shown, through the quaternion Riesz representation theorem, that for an arbitrary QRKHS there exists a unique reproducing kernel. This makes it possible to compute inner products in a quaternion-valued feature space using the kernel trick.

Observe that although Theorem 3 gives theoretical support for quaternion kernel estimation, it is far from being useful in practice on its own, since the design of a QRKHS suited for a specific task can be rather difficult. To this end, we next complement the Riesz representation theorem with the Moore-Aronszajn theorem, in order to show that any quaternion kernel (within a certain class of kernels) generates a unique QRKHS.

Moore-Aronszajn Theorem

In the real-valued case, the existence of a unique QRKHS generated by a positive definite kernel is ensured via either (i) the Mercer theorem [Mercer, 1909], where the feature Hilbert space is spanned by the eigenfunctions of the kernel K, or (ii) the Moore-Aronszajn theorem, in which the feature Hilbert space is spanned by the functions $K_{\mathbf{x}} = K(\mathbf{x}, \cdot)$.

We now state two equivalent definitions of positive definiteness in order to introduce a key result in quaternion reproducing kernel Hilbert spaces: the quaternion Moore-Aronszajn theorem.

Definition 5 (Positive definiteness - integral form)**.** *A Hermitian kernel $K(\mathbf{x}, \mathbf{y}) = K^*(\mathbf{y}, \mathbf{x})$ is positive definite on the set X iff for any integrable function $\theta : X \to \mathbb{H}$, $\theta \neq 0$, it obeys*

$$\int_X \int_X \theta^*(\mathbf{x}) K(\mathbf{x}, \mathbf{y}) \theta(\mathbf{y}) d\mathbf{x} d\mathbf{y} > 0.$$

Definition 6 (Positive definiteness - matrix form)**.** *A Hermitian kernel $K(\mathbf{x}, \mathbf{y}) = K^*(\mathbf{y}, \mathbf{x})$ is positive definite on the set X iff the kernel matrix $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ is positive definite for any choice of the set $S_{\mathbf{x}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset X$, $m \in \mathbb{N}$.*

Theorem 4 (Quaternion Moore-Aronszajn theorem)**.** *For any positive definite quaternion-valued kernel K defined over a set X, there exists a unique (up to an isomorphism) left quaternion Hilbert space of functions $\mathcal{H}$ for which K is a reproducing kernel.*

Proof. The proof first generalises the idea behind the real-valued Moore-Aronszajn theorem [**?**], to show that the span of $K_{\mathbf{x}}$ is a QRKHS, and then presents the uniqueness proof.

(i) The span of $K_{\mathbf{x}}$ is a QRKHS. Define the set

$$\mathcal{H}_0 = \left\{ f \in \mathcal{F} : f = \sum_{i=0}^{n} \alpha_i K(\mathbf{x}_i, \cdot), \mathbf{x}_i \in X, \alpha_i \in \mathbb{H}, n \in \mathbb{N} \right\}$$

and the inner product between $f = \sum_{i=0}^{n} \alpha_i K(\mathbf{x}_i, \cdot)$ and $g = \sum_{i=0}^{m} \beta_i K(\mathbf{y}_i, \cdot)$ as

$$\langle f, g \rangle = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i K(\mathbf{x}_i, \mathbf{y}_j) \beta_j^*. \tag{4.10}$$

Note that the inner product $\langle \cdot, \cdot \rangle$ satisfies the properties in Definition 3 and the set $\mathcal{H}_0$ is a left inner product space. Its closure, denoted by $\mathcal{H} = \overline{\mathcal{H}_0}$, equips $\mathcal{H}_0$ with the limits of all its Cauchy sequences $\{f_n\} \subset \mathcal{H}_0$. As the elements added to form the closure are also bounded (Cauchy sequences are convergent), the elements of $\mathcal{H}$ can be written in the form $f = \sum_{i=0}^{\infty} \alpha_i K(\mathbf{x}_i, \cdot)$.

Observe that the evaluation functional (4.8) over the so-defined set $\mathcal{H}$ is bounded. Indeed, using the Cauchy-Schwartz inequality and the quaternion Riesz theorem (Thm. 3) we have

$$|f(\mathbf{x})|_{\mathbb{H}} = |\langle f, K_{\mathbf{x}} \rangle|_{\mathbb{H}} \leq \|f\|_{\mathcal{H}} \|K_{\mathbf{x}}\|_{\mathcal{H}} = \|f\|_{\mathcal{H}} \sqrt{K(\mathbf{x}, \mathbf{x})} < \infty.$$

(ii) Uniqueness. Consider two spaces $\mathcal{H}$ and $\mathcal{G}$ for which K is a reproducing kernel, and recall that the equation

$$\langle K_{\mathbf{x}}, K_{\mathbf{y}} \rangle_{\mathcal{H}} = K(\mathbf{x}, \mathbf{y}) = \langle K_{\mathbf{x}}, K_{\mathbf{y}} \rangle_{\mathcal{G}}$$

holds over the span of $\{K_{\mathbf{x}}, \mathbf{x} \in X\}$. As the closure of the span is unique and the inner product is linear, we have $\mathcal{H} = \mathcal{G}$.

We have therefore shown that given an arbitrary positive definite quaternion kernel K, there is a (unique) complete quaternion inner product space (i.e. a left Hilbert space), for which the evaluation functional is bounded, conditions for a QRKHS. □

Remark 6. *Due to the non-commutativity of $\mathbb{H}$, the inner product constructed in the proof of Thm. 4, Eq. (4.10), differs from the real/complex case in that it requires a particular form in order to fulfil the requirements of Definition 3.*

The so-constructed inner product supports the reproducing property of the QRKHS, that is,

$$f(\mathbf{x}) = \sum_{i=0}^{\infty} \alpha_i K(\mathbf{x}_i, \mathbf{x}) \overset{(a)}{=} \left\langle \sum_{i=0}^{\infty} \alpha_i K_{\mathbf{x}_i}, K_{\mathbf{x}} \right\rangle = \langle f(\cdot), K_{\mathbf{x}} \rangle ,$$

where the symbol $\overset{(a)}{=}$ refers to the definition of the inner product in (4.10).

Theorems 3 and 4 provide the existence and uniqueness conditions underpinning quaternion-valued kernel algorithms: the Riesz representation theorem allows us to simplify feature space operations into kernel evaluations and the Moore-Aronzsajn theorem ensures that for any (positive definite) kernel there is a unique QRKHS.

Remark 7. *Since the QRKHS is built upon a left-module, and not a field as the standard RKHS, the derivation of Theorems 3 and 4 confirms that commutativity is not a requirement for constructing feature spaces over division rings and also paves the way for the study of relationships between QRKHS built over left- and right-modules. We would also like to emphasise that the aim of the proofs provided is not claim a radical difference between Theorems 3 and 4, and their real versions, but to show that although the corresponding proofs follow the same criteria, the quaternion case requires more attention due to the lack of commutativity.*

4.2.3 Design of Quaternion-Valued Mercer Kernels

Theorem 4 gives the justification for the design and implementation of nonlinear kernel algorithms operating in QRKHS to simplify into the choice of a positive-semidefinite kernel. We next introduce and analyse the properties of some specific kernels of quaternion variable and justify their use within quaternion SVR algorithms.

Linear Quaternion Kernel

The linear kernel is the simplest reproducing kernel. For quaternion valued signals, the quaternion linear kernel K_Q and its real-valued counterpart K_R are respectively given by

$$K_Q(\mathbf{x},\mathbf{y}) = 1+\langle \mathbf{x},\mathbf{y}\rangle = 1+\mathbf{x}^H\mathbf{y} \tag{4.11}$$

$$K_R(\mathbf{x},\mathbf{y}) = 1+\langle \mathbf{x},\mathbf{y}\rangle_{\Re} = 1+\Re\{\mathbf{x}^H\mathbf{y}\}, \tag{4.12}$$

where $\langle \mathbf{x},\mathbf{y}\rangle_{\Re}$ is the inner product of the real-valued isomorphisms of $\mathbf{x}$ and $\mathbf{y}$, and $\Re\{q\}$ denotes the real part of the quaternion q. To show that the quaternion linear kernel is positive semidefinite, combine (4.11) and Definition 5, and use Fubini's Theorem to give

$$\begin{aligned}\int_{\mathbf{X}^2}\theta^*(\mathbf{x})(1+\mathbf{x}^H\mathbf{y})\theta(\mathbf{y})\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} &= \int_{\mathbf{X}^2}\theta^*(\mathbf{x})\theta(\mathbf{y})\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y}+\int_{\mathbf{X}^2}\left(\mathbf{x}\theta^*(\mathbf{x})\right)^H\mathbf{y}\theta(\mathbf{y})\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} \\ &= \left\|\int_{\mathbf{X}}\theta(\mathbf{x})\mathrm{d}\mathbf{x}\right\|^2+\left\|\int_{\mathbf{X}}\mathbf{x}\theta(\mathbf{x})\mathrm{d}\mathbf{x}\right\|^2.\end{aligned} \tag{4.13}$$

Remark 8. *The quaternion linear kernel in (4.11) admits the modelling of statistical interdependence in its imaginary parts, and has the ability to learn the relationship between the quadrivariate input variables, while preserving the mathematical simplicity of univariate kernel regression algorithms.*

Polynomial Kernel: The Quaternion Cubic Example

The polynomial kernel is standard in kernel-based estimation due to its robustness and ease of implementation. For real- and complex-valued samples $\mathbf{x}_r, \mathbf{y}_r$, the polynomial kernel is given by

$$K_P(\mathbf{x}_r, \mathbf{y}_r) = \big(1 + \mathbf{x}_r^T \mathbf{y}_r\big)^p$$

where $p \in \mathbb{N}$ is referred to as the order of the kernel. On the other hand, the real-valued polynomial kernel of quaternion samples $\mathbf{x}, \mathbf{y}$ $K_{RP} : X^2 \to \mathbb{R}$, that is, the polynomial kernel of the real-valued representations of $\mathbf{x}$ and $\mathbf{y}$, can be expressed as

$$K_{RP}(\mathbf{x}, \mathbf{y}) = \big(1 + \langle \mathbf{x}, \mathbf{y} \rangle_{\Re}\big)^p = \big(1 + \Re\{\mathbf{x}^H \mathbf{y}\}\big)^p \tag{4.14}$$

where $\langle \mathbf{x}, \mathbf{y} \rangle_{\Re}$ is the inner product in $\mathbb{R}^n$ and $\Re\{q\}$ denotes the real part of the quaternion q.

The extension to quaternion-valued polynomial kernels is not straightforward, as for the quaternion vectors $\mathbf{x}$ and $\mathbf{y}$ the factorisation

$$\big(1 + \mathbf{x}^H \mathbf{y}\big)^p = \phi^H(\mathbf{x})\phi(\mathbf{y})$$

may not be possible due to the noncommutativity of the quaternion ring, and therefore the positive definiteness of such kernel cannot be guaranteed in this manner.

For $p = 3$, we next propose a quaternion polynomial kernel which admits factorisation as an inner product, thus ensuring the required positive definiteness.

Consider the quaternion cubic kernel $K_{QP} : X^2 \to \mathbb{H}$ given by

$$K_{QP}(\mathbf{x}, \mathbf{y}) = \big(1 + \mathbf{x}^H \mathbf{x}\big)\big(1 + \mathbf{x}^H \mathbf{y}\big)\big(1 + \mathbf{y}^H \mathbf{y}\big). \tag{4.15}$$

To show that K_{QP} is positive semidefinite, we shall first consider its factorisation of the form $K_{QP}(\mathbf{x}, \mathbf{y}) = \phi^H(\mathbf{x})\phi(\mathbf{y})$. Indeed,

$$\begin{aligned} K_{QP}(\mathbf{x}, \mathbf{y}) &= \big(1 + \mathbf{x}^H \mathbf{x}\big)\big(1 + \mathbf{y}^H \mathbf{y}\big) \\ &\quad + \big(1 + \mathbf{x}^H \mathbf{x}\big)\mathbf{x}^H \mathbf{y}\big(1 + \mathbf{y}^H \mathbf{y}\big) \\ &= \big(1 + \mathbf{x}^H \mathbf{x}\big)^H \big(1 + \mathbf{y}^H \mathbf{y}\big) \\ &\quad + \big(\mathbf{x}\big(1 + \mathbf{x}^H \mathbf{x}\big)\big)^H \big(\mathbf{y}\big(1 + \mathbf{y}^H \mathbf{y}\big)\big) \\ &= \phi_1^H(\mathbf{x})\phi_1(\mathbf{y}) + \phi_2^H(\mathbf{x})\phi_2(\mathbf{y}), \end{aligned}$$

where $\phi_1(\mathbf{x}) = 1 + \mathbf{x}^H \mathbf{x}$ and $\phi_2(\mathbf{x}) = \mathbf{x}\big(1 + \mathbf{x}^H \mathbf{x}\big)$. Therefore, by setting $\phi(\mathbf{x}) = [\phi_1^T(\mathbf{x})\ \phi_2^T(\mathbf{x})]^T$ we arrive at

$$K_{QP}(\mathbf{x}, \mathbf{y}) = \phi^H(\mathbf{x})\phi(\mathbf{y}). \tag{4.16}$$

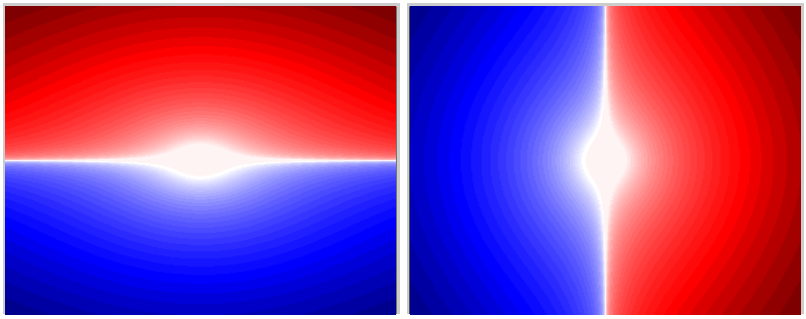

Figure 4.3: Real (left) and i-imaginary (right) parts of K_{QP}. The colourmap is dark blue for $-13 \cdot 10^3$, white for the interval $[-10, 10]$, and red for $13 \cdot 10^3$ with a logarithmic RGB interpolation.

Finally, by combining (4.16) and Definition 5 we have

$$\begin{aligned}
\int_{X^2} \theta^*(\mathbf{x}) K_{QP}(\mathbf{x}, \mathbf{y}) \theta(\mathbf{y}) \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} = \\
&= \int_{X^2} \theta^*(\mathbf{x}) \phi^H(\mathbf{x}) \phi(\mathbf{y}) \theta(\mathbf{y}) \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} \\
&\overset{(a)}{=} \int_X \theta^*(\mathbf{x}) \phi^H(\mathbf{x}) \mathrm{d}\mathbf{x} \int_X \phi(\mathbf{y}) \theta(\mathbf{y}) \mathrm{d}\mathbf{y} \\
&= \left(\int_X \phi(\mathbf{x}) \theta(\mathbf{x}) \mathrm{d}\mathbf{x} \right)^H \int_X \phi(\mathbf{y}) \theta(\mathbf{y}) \mathrm{d}\mathbf{y} \\
&= \left\| \int_X \phi(\mathbf{x}) \theta(\mathbf{x}) \mathrm{d}\mathbf{x} \right\|^2 \geq 0,
\end{aligned}$$

where $\theta(\cdot)$ is assumed to be Lebesgue integrable and bounded on the compact set $X \subsetneq \mathbb{H}^n$, while the identity $\overset{(a)}{=}$ is a consequence of Fubini's theorem [Friedman, 1982].

Remark 9. *Note from Eqs. (4.14) and (4.15) that, owing to its imaginary part, the quaternion cubic kernel K_{QP} provides enhanced data representation over the real-valued cubic kernel K_{RP}. Therefore, K_{QP} has the ability to learn the relationship between input variables, while preserving the mathematical simplicity of polynomial kernels.*

Fig. 4.3 visualises K_{QP} for the scalar case $x = 1, y = y_r + iy_i + jy_j + ky_k \in \mathbb{H}$, which gives $K_{QP}(1, y) = 2(1 + y)(1 + \|y\|^2)$. As $K_{QP}(1, y)$ is symmetric, we only plot the region $(y_r, y_i, y_j, y_k) \in [-15, 15] \times [-15, 15] \times \{0\} \times \{0\}$.

Real-Valued Gaussian Kernel

The Gaussian kernel in eq. (2.19) can be extended to operate on quaternion samples by accommodating the quaternion norm in its argument. Recall that $\|q\| = \sqrt{q^H q}$, so that

the real-valued Gaussian kernel K_{RG} can be defined as[4]

$$K_{RG}(\mathbf{x},\mathbf{y}) = \exp\left(-A_R\left(\mathbf{x}-\mathbf{y}\right)^H\left(\mathbf{x}-\mathbf{y}\right)\right), \tag{4.17}$$

where $A_R > 0$ is the kernel parameter.

We next use Definition 6 to show that K_{RG} is positive definite in the quaternion domain. First, observe that for an arbitrary, non-zero, vector $\mathbf{x} \in \mathbb{H}^n$ the quadratic form $\mathbf{x}^H\mathbf{K}\mathbf{x}$ is real. Indeed, due to the symmetry of the real matrix $\mathbf{K}$ we have

$$2\Im\{\mathbf{x}^H\mathbf{K}\mathbf{x}\} = \mathbf{x}^H\mathbf{K}\mathbf{x} - (\mathbf{x}^H\mathbf{K}\mathbf{x})^H = \mathbf{x}^H\mathbf{K}\mathbf{x} - (\mathbf{x}^H\mathbf{K}\mathbf{x}) = 0.$$

Now, by expanding the vector $\mathbf{x} = \mathbf{x}_r + i\mathbf{x}_i + j\mathbf{x}_j + k\mathbf{x}_k$ within $\Re\{\mathbf{x}^H\mathbf{K}\mathbf{x}\}$ using its real and imaginary parts, we can write

$$\Re\{\mathbf{x}^H\mathbf{K}\mathbf{x}\} = \mathbf{x}_r^T\mathbf{K}\mathbf{x}_r + \mathbf{x}_i^T\mathbf{K}\mathbf{x}_i + \mathbf{x}_j^T\mathbf{K}\mathbf{x}_j + \mathbf{x}_k^T\mathbf{K}\mathbf{x}_k.$$

Since $\mathbf{K}$ is positive definite in the real domain, the arbitrary components $\mathbf{x}_R, \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k$ are real-valued, and the quadratic form $\mathbf{x}^H\mathbf{K}\mathbf{x}$ is positive, we have $\mathbf{x}^H\mathbf{K}\mathbf{x} = \Re\{\mathbf{x}^H\mathbf{K}\mathbf{x}\} > 0$, proving the positive definiteness of the real Gaussian kernel K_{RG} in $\mathbb{H}$.

Quaternion-Valued Gaussian Kernel

Similarly to the complex-valued Gaussian kernel in (4.1), quaternion version of the Gaussian kernel can be expressed as

$$K_{QG}(\mathbf{x},\mathbf{y}) = \exp\left(-A_Q\left(\mathbf{x}-\mathbf{y}^*\right)^T\left(\mathbf{x}-\mathbf{y}^*\right)\right), \tag{4.18}$$

where $A_Q > 0$ is the kernel parameter, and the quaternion-valued argument allows for the kernel to be a full quaternion.

Similarly to the complex Gaussian kernel, $K_{QG}(\mathbf{x},\mathbf{y})$ can be decomposed by denoting $\mathbf{e}_R = \Re\{\mathbf{x}-\mathbf{y}^*\}$, $\mathbf{e}_I = \Im\{\mathbf{x}-\mathbf{y}^*\}$ and $\mathbf{x}-\mathbf{y}^* = \mathbf{e}_R + \mathbf{e}_I$ according to

$$\begin{aligned} K_{QG}(\mathbf{x},\mathbf{y}) &= \exp\left(-A_Q\left(\mathbf{e}_R+\mathbf{e}_I\right)^T\left(\mathbf{e}_R+\mathbf{e}_I\right)\right) \\ &= \exp\left(-A_Q\left(\mathbf{e}_R^T\mathbf{e}_R + \mathbf{e}_I^T\mathbf{e}_R + \mathbf{e}_R^T\mathbf{e}_I + \mathbf{e}_I^T\mathbf{e}_I\right)\right) \\ &= \exp\left(-A_Q\left(\|\mathbf{e}_R\|^2 - \|\mathbf{e}_I\|^2 + 2\mathbf{e}_R^T\mathbf{e}_I\right)\right) \\ &= e^{\delta}\left(\cos\|\Delta\| + \frac{\Delta}{\|\Delta\|}\sin\|\Delta\|\right), \end{aligned}$$

where $\delta = -A_Q\left(\|\mathbf{e}_R\|^2 - \|\mathbf{e}_I\|^2\right)$ and $\Delta = -2A_Q\mathbf{e}_R^T\mathbf{e}_I$.

[4]We use the notation $A_r = \sigma^{-2}$ for simplicity of presentation in this chapter.

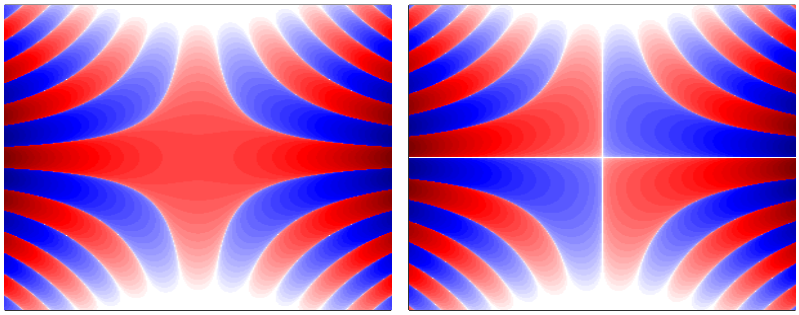

Figure 4.4: Real (left) and i-imaginary (right) parts of K_{QG}. The colourmap is dark blue for $-7 \cdot 10^{-4}$, white for 0, and red for $7 \cdot 10^{4}$, with a logarithmic RGB interpolation.

Observe that K_{QG} is not globally bounded, since its norm grows exponentially with $\|\mathbf{e}_I\|^2 = \|\Im\{\mathbf{x}\} + \Im\{\mathbf{y}\}\|^2$ (as $A_Q > 0$)—recall that this property was also present in the complex Gaussian kernel in eq. (4.1). This exponential property highlights both advantages and disadvantages regarding the implementation of kernel estimation algorithms: K_{QG} has the ability to model data with large dynamics and to boost the speed of learning due to its exponential growth; however, an incorrect choice of parameters will lead to unbounded estimates. From the point of view of a physically-meaningful representation, the real Gaussian kernel K_{RG} is better suited for interpolation applications as it can be regarded as a measure of similarity of samples (like the triangular kernel in similarity-based modelling [Tobar et al., 2011]), whereas the quaternion Gaussian kernel K_{QG} is useful for extrapolating nonlinear features.

Fig. 4.4 shows K_{QG} for the scalar case $\mathbf{y} = 0, \mathbf{x} = x_r + ix_i + jx_j + kx_k \in \mathbb{H}$, which gives $\delta = -A_Q(x_r^2 - x_i^2 - x_j^2 - x_k^2), \Delta = -2A_Q x_r(ix_i + jx_j + kx_k)$. As $K_{QG}(x, 0)$ is symmetric, we only plot the region $(x_r, x_i, x_j, x_k) \in [-15, 15] \times [-15, 15] \times \{0\} \times \{0\}$ where $A_Q = 0.05$.

4.3 Examples

4.3.1 Systems with Correlated and Uncorrelated Noise: Quaternion Linear Kernel

The quaternion-valued linear kernel proposed in (4.11) was validated against its real-valued counterpart in (4.12) in a least-squares kernel regression-setting for the prediction of as autoregressive process.

We considered the AR(1) process $x_{t+1} = Ax_t + B\eta_t$, where $x_t \in \mathbb{H}$, and η_t is a quaternion random variable whose components are uncorrelated and uniformly distributed in $[0, 1]$. Correlated and uncorrelated realisations of the process x_t were obtained by respectively setting $\Im\{A\} = \Im\{B\} = 0$ and by letting A, B to be full quaternions. Note

from Fig. 4.5 that the difference in MSEs of kernel algorithms for the uncorrelated case remained fairly constant for different support vectors, whereas for the correlated case this difference increased with the number of support vectors, hence highlighting the ability of the linear quaternion kernel to model coupled processes.

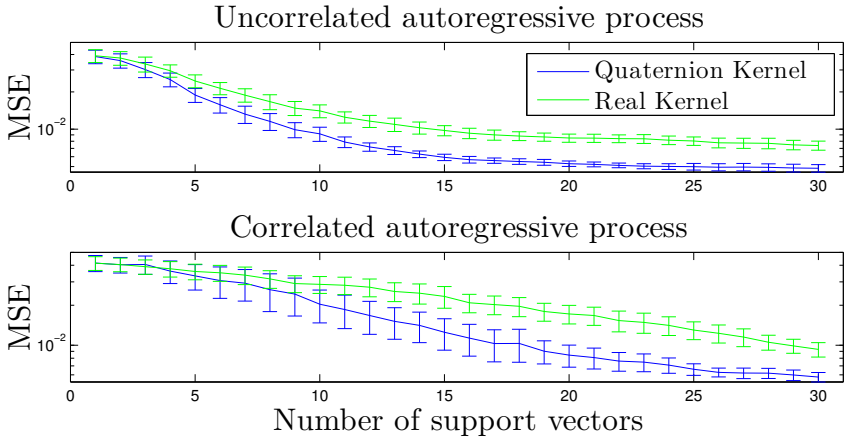

Figure 4.5: MSE $\pm$ 0.5 standard deviations for kernel algorithms as a function of the number of support vectors in the estimation of both uncorrelated (top, $A = 0.6808, B = 0.1157$) and correlated (bottom, $A = 0.6808 + i0.07321 + j0.6222 - k0.2157, B = 0.1157 + i0.1208 + j0.8425 - k0.5121$) AR(1) processes.

4.3.2 Nonlinear Channel Equalisation: Quaternion Gaussian Kernel

We next validated the real and quaternion Gaussian kernels for the problem of nonlinear channel equalisation in a ridge regression setting. A detailed account is given for the model of the channel, the choice of the kernel parameters, and the validation of the algorithm.

Channel Model

The transmission channel was modelled as a linear (moving average) filter with a memoryless nonlinearity stage corrupted by noise:

$$\begin{aligned} \mathbf{y}_n &= a_1\mathbf{x}_n + a_2\mathbf{x}_{n-1} \\ \mathbf{s}_n &= \mathbf{y}_n + a_3\mathbf{y}_n^2 + \epsilon_n, \end{aligned}$$

where $\{\mathbf{x}_n\}_{n\in\mathbb{N}}$ is the transmitted message (input to the channel), $\{\mathbf{y}_n\}_{n\in\mathbb{N}}$ is an unobserved latent process, $\{\epsilon_n\}_{n\in\mathbb{N}}$ is a noise process, and $\{\mathbf{s}_n\}_{n\in\mathbb{N}}$ is the received signal (output of the channel). This model has been previously considered for the validation of kernel learning algorithms including KRR [Lin et al., 2005], kernel LMS [Liu et al., 2008], and

its complex-valued extensions [Bouboulis and Theodoridis, 2010, Bouboulis et al., 2012]. The aim of channel equalisation is to identify the original message $\{\mathbf{x}_n\}_{n\in\mathbb{N}}$ from the noisy measurements $\{\mathbf{s}_n\}_{n\in\mathbb{N}}$.

We focused on the quadrivariate case, that is, $\mathbf{x}_n, \mathbf{y}_n, \epsilon_n, \mathbf{s}_n \in \mathbb{R}^4$, and assumed that the components of the input vector (message) $\mathbf{x}_n$ are jointly Gaussian, i.e. $\mathbf{x}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$, and uncorrelated with the (also Gaussian) noise $\epsilon_n \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma_2})$. The quadriavariate real signals $\mathbf{x}_n, \mathbf{s}_n \in \mathbb{R}^4$ were then expressed as univariate quaternion sequences $x_n, s_n \in \mathbb{H}$.

The model parameters were randomly chosen and had the values

$$\mathbf{\Sigma} = \begin{bmatrix} 1.2624 & -0.3541 & -0.1457 & -0.5030 \\ -0.3541 & 0.8487 & -0.1730 & 0.0402 \\ -0.1457 & -0.1730 & 0.4553 & -0.3892 \\ -0.5030 & 0.0402 & -0.3892 & 1.4336 \end{bmatrix},$$

$$\begin{aligned} a_1 &= 0.7466 + i0.3733 - j0.28 + k0.1867 \\ a_2 &= 0.4564 + i0.1521 - j0.6085 + k0.4564 \\ a_3 &= 0.5341 + i0.3204 + j0.1068 - k0.6409. \end{aligned}$$

With this choice of parameters, both the original message and the received signals were noncircular quaternion sequences [Vía et al., 2010, Cheong Took and Mandic, 2011].

Kernel Parameter Design

Within the KRR setting, once the optimal weights $\mathbf{a}$ are computed via eq. (3.41), the estimate is linear in the kernel evaluations. Accordingly, empirical criteria for kernel design were used to set the kernel parameters so that the kernel evaluations (entries of the kernel evaluation matrix $\mathbf{K}$) remained bounded, while at the same time captured enough data variance. We set the parameters of the Gaussian kernels to be $A_R = 6 \cdot 10^{-3}$ (real) and $A_Q = 10^{-4}$ (quaternion) by analysing the second moment of the kernel evaluations over a 200-sample realisation of the process $\mathbf{s}_t$, thus ensuring boundedness and sufficient variability. Fig. 4.6 analyses the features used for setting kernel parameters and shows the histogram of the kernel evaluations corresponding to the choice of parameters.

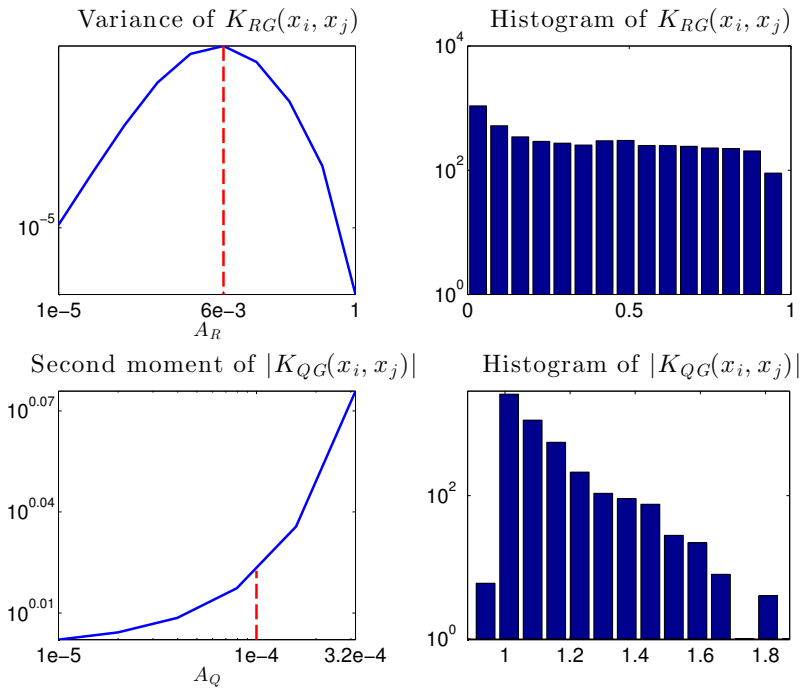

Figure 4.6: Choice of kernel parameters (red) A_R and A_Q based on the second moment of the kernel evaluations and histograms corresponding to the chosen parameters.

Validation

The ability of the different kernels to both (i) learn the relationship between the available input-output samples and (ii) generalise the estimates to new datasets of similar dynamics, was next assessed. Both kernels were also compared to the strictly- and widely-linear quaternion ridge regression. See Appendices A.3 for an introduction to quaternion widely-linear estimation.

Fig. 4.7 shows the training MSE averaged over 30 realisations as a function of the number of support vectors. The training MSE was computed from the estimate of a 200-sample sequence which **contained the support (training) vectors**. Observe that for more than 50 support vectors, the widely-linear ridge regression algorithm outperformed its strictly linear counterpart. Also note that the training performance of the quaternion Gaussian kernel was similar to that of the widely-linear ridge regression algorithm [Vía et al., 2010]. The real Gaussian KRR offered the best training performance, which improved monotonically with the number of support vectors.

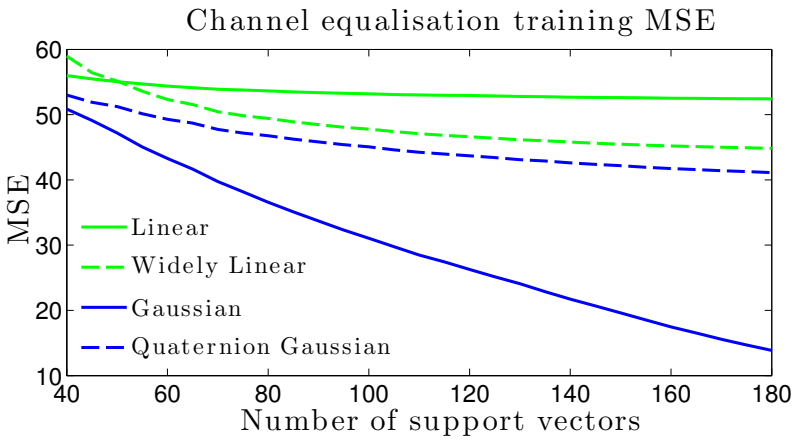

Figure 4.7: Training MSE of ridge regression algorithms for channel equalisation.

The validation MSE, also averaged over 30 realisations, is shown in Fig. 4.8 as a function of the number of support vectors. To compute the validation MSE, the support samples (together with the training samples) and the estimated signal **corresponded to different realisations** of 100 samples each, this way, the validation MSE assesses the ability of the regression algorithms to generalise the input-output dependency. Observe that, on average, the quaternion Gaussian kernel provided the best estimates, outperforming not only the linear ridge regression algorithms, but also to the standard, real-valued, Gaussian kernel.

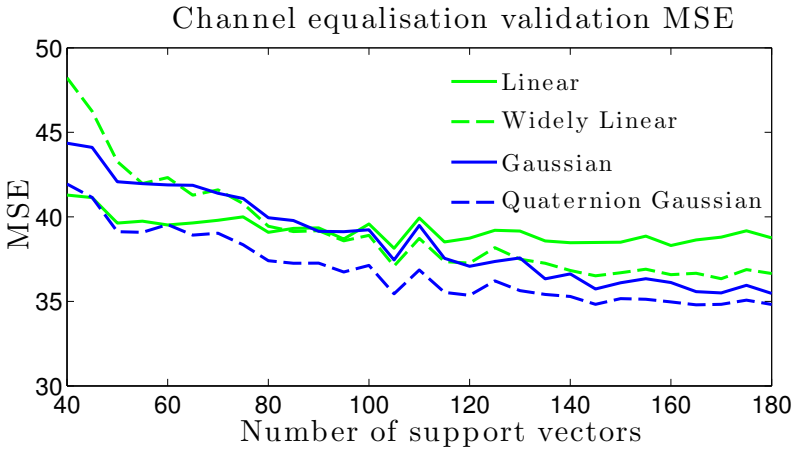

Figure 4.8: Validation MSE of ridge regression algorithms for channel equalisation.

The unbounded nature of the quaternion-valued Gaussian kernel allowed for the extrapolation of the nonlinear behaviour learned in the training stage. This property is not found in the real Gaussian kernel, which serves as a similarity measure and is therefore better suited for data interpolation.

Due to the enhanced modelling ability arising from the terms in their imaginary parts, the quaternion-valued kernels were less prone to overfitting than the real-valued kernels, as shown in both experiments. Furthermore, the superior performance of the quaternion SVR approach highlights the advantage of using high-dimensional feature spaces.

4.4 Discussion

The proposed hypercomplex-valued kernels provide a novel framework for learning in feature spaces. Even though RKHS are—in general—infinite-dimensional, considering complex- and quaternion-valued RKHS, and therefore kernels, is equivalent to performing estimation using two (complex) of four (quaternion) real-valued RKHS, whereby the estimation can only improve due to the additional degrees of freedom. The hypercomplex-kernel paradigm inherits the scalar-algebra simplicity of the standard RKHS approach and, despite considering higher dimensional features, does not require matrix operations; this means that standard kernel algorithms can be extended to operate on hypercomplex RKHS by only defining complex and quaternion kernels. The enhanced modelling ability of this proposed approach comes with the need of larger training sets: as in the general case the kernel mixing weights are also hypercomplex, there are two (complex) or four (quaternion) times more parameters to learn. This, however, should not be understood as a drawback, since an estimator that provides enhanced modelling would *benefit from*, rather than *require*, larger training sets.

Finally, we present the following lemma whereby the distinguishing properties of quaternion positive definite kernels are stated[5].

Lemma 3. *Let $K = K_r + iK_i + jK_j + kK_k$ be a quaternion kernel, then*

- **(a)** *K is Hermitian iff K_r is symmetric positive definite and $K_i = -K_i^T$, $K_j = -K_j^T$ and $K_k = -K_k^T$.*
- **(b)** *If K is Hermitian, K is positive definite iff the **real-valued** matrix representation[6] of its Gram matrix is positive definite.*

Lemma 3 highlights two distinguishing features of the QRKHS setting. First, the kernel imaginary parts are not positive definite, meaning that quaternion-valued kernels are not equivalent to an ensemble of four real-valued kernels. Second, the matrix representation of quaternions allows us to evaluate positive definiteness of quaternion kernels using standard tools for real matrices.

[5] See Appendix A.4 for the proof of Lemma 3.
[6] See [Ward, 1997, Page 91] for the real matrix representation of quaternions.

Chapter 5

Vector-Valued Kernels

The infinite! No other question has ever moved so profoundly the spirit of man; no other idea has so fruitfully stimulated his intellect; yet no other concept stands in greater need of clarification than that of the infinite.

-David Hilbert.
(Quoted in J. R. Newman's "The World of Mathematics," 1956.)

The hypercomplex-kernel approach proposed in Chapter 4 has proven advantageous when dealing with general bivariate and quadrivariate signals compared to the standard real-kernel regression paradigm. This confirms that the performance of general kernel regression algorithms can indeed be improved by augmenting the dimensionality of the feature space. The hypercomplex setting is elegant and simple in algebraic terms, however, it has two clear disadvantages: Firstly, it does not allow for arbitrary-dimension kernels but only 2D and 4D ones; secondly, it does not make use of the several known real kernels. We address these issues by introducing a novel vector-valued generalisation of RKHS, this space is then used as the basis the class of multiple-kernel (multikernel) regression algorithms, whereby constitutive *subkernels* operate in a collaborative fashion to compute estimates. The proposed vector-valued RKHS and multikernel algorithms have been introduced in [Tobar and Mandic, 2012, Tobar et al., 2014b]

5.1 A Hilbert Space of Vector-Valued Functions with a Reproducing Property

Consider the input set $X \in \mathbb{R}^n$ and an indexed collection of L RKHS over X denoted by $\mathcal{H} = \{\mathcal{H}_l\}_{l=1:L}$; recall that the elements $h_l \in \mathcal{H}_l$ are functions $h_l : X \longrightarrow \mathbb{R}$. Define the

space $\mathbf{H}_L$ as the set of vector-valued functions

$$\mathbf{H}_L = \left\{ \mathbf{h} = \begin{bmatrix} h_1 \\ \vdots \\ h_L \end{bmatrix}, h_l \in \mathcal{H}_l, l = 1, \ldots, L \right\} \tag{5.1}$$

that is, for $\mathbf{h} \in \mathbf{H}_L$, its l^{th} element $h_l \in \mathcal{H}_l$. For convenience of notation, we will denote $\mathbf{h} \in \mathbf{H}$ in terms of its components in compact notation by $\mathbf{h} = [h_l]_{l=1:L}$.

Note that the construction of $\mathbf{H}_L$ allows for the representation of its elements as vector-valued functions of the form:

$$\begin{aligned} \mathbf{h} : X &\longrightarrow \mathbb{R}^L \\ \mathbf{x} &\longmapsto \mathbf{h}(\mathbf{x}). \end{aligned} \tag{5.2}$$

We now introduce three properties (lemmas) of the space $\mathbf{H}_L$, that are a backbone of the proposed multikernel approach.

Lemma 4 (induced Hilbert space)**.** *For any L, X and $\mathcal{H}_L$ as defined above, $\mathbf{H}_L$ is a Hilbert space.*

Proof. As by construction $\mathbf{H}_L$ is a complete vector space, for it to become a Hilbert space we only have to equip it with an inner product inducing a norm.

Based on the definition of $\mathbf{H}_L$, $\mathcal{H}_l$ is also a Hilbert space and it is therefore equipped with an inner product. Without loss of generality, we denote the inner product between feature elements h_l, $g_l \in \mathcal{H}_l$ by $\langle h_l, g_l \rangle_{\mathcal{H}_l}$, and the corresponding norm by $\|h_l\|_{\mathcal{H}_l} = \sqrt{\langle h_l, h_l \rangle_{\mathcal{H}_l}}$. This allows us to define the inner product for $\mathbf{h} = [h_1]_{l=1:L}, \mathbf{g} = [g_l]_{l=1:L} \in \mathbf{H}_L$ by

$$\langle \mathbf{h}, \mathbf{g} \rangle = \sum_{l=1}^{L} \langle h_l, g_l \rangle_{\mathcal{H}_l} \tag{5.3}$$

where the inner product properties of $\langle \cdot, \cdot \rangle_{\mathcal{H}_l}$ are inherited by $\langle \cdot, \cdot \rangle$ due to its multilinear construction.

Based on the properties of the norms of the RKHS in $\mathcal{H}_L$, it then follows that the inner product in (5.3) induces a norm given by:

$$\|\mathbf{h}\| = \sqrt{\langle \mathbf{h}, \mathbf{h} \rangle} = \sqrt{\sum_{l=1}^{L} \langle h_l, h_l \rangle_{\mathcal{H}_l}} = \sqrt{\sum_{l=1}^{L} \|h_l\|_{\mathcal{H}_l}^2} \tag{5.4}$$

where the norm properties are inherited due to the multilinear construction of $\mathbf{H}_L$. □

Lemma 5 (boundedness)**.** *The evaluation functional $F_{\mathbf{x}}(\mathbf{h}) = \mathbf{h}(\mathbf{x}) \in \mathbb{R}^L$ is bounded for any $\mathbf{x} \in X$ and $\mathbf{h} \in \mathbf{H}_L$.*

Proof. The evaluation functional $F_{\mathbf{x}}(h_l) = h_l(\mathbf{x}) \in \mathbb{R}$ is bounded $\forall h_l \in \mathcal{H}_l$, that is,

$$\exists M_l \in \mathbb{R}^+,\ s.t.\ \|F_{\mathbf{x}}(h_l)\|_2 \leq M_l \|h_l\|_{\mathcal{H}_l},$$

where $\|\cdot\|_2$ is the L_2-norm. Combined with the norm definition in (5.4), this leads to

$$\begin{aligned}\|F_{\mathbf{x}}(\mathbf{h})\|_2^2 &= \sum_{l=1}^{L} \|F_{\mathbf{x}}(h_l)\|_2^2 \leq \sum_{l=1}^{L} M_l^2 \|h_l\|_{\mathcal{H}_l}^2 \\ &\leq \max_{l=1,\ldots,L}\{M_l^2\} \sum_{l=1}^{L} \|h_l\|_{\mathcal{H}_l}^2 = M^2 \|\mathbf{h}\|^2, \end{aligned} \tag{5.5}$$

where $M^2 = \max_{l=1:L}\{M_l^2\}$. Consequently, $\|F_{\mathbf{x}}(\mathbf{h})\|_2 \leq M \|\mathbf{h}\|$, that is, $F_{\mathbf{x}}(\mathbf{h})$ is bounded. □

Lemma 6 (reproducing property)**.** *The evaluation functional $F_{\mathbf{x}}(\mathbf{h}) = \mathbf{h}(\mathbf{x})$, $\forall \mathbf{x} \in X$, $\mathbf{h} \in \mathbf{H}_L$ can be written as a product between arrays $\mathbf{h} = [h_l]_{l=1,\ldots,L}$ and $K_L(\mathbf{x}, \cdot) = [k_l(\mathbf{x}, \cdot)]_{l=1,\ldots,L}$, where k_l is a reproducing kernel of $\mathcal{H}_l$.*

Proof. The reproducing property of the RKHS $\mathcal{H}_l$, $F_{\mathbf{x}}(h_l) = \langle h_l, k(\mathbf{x}, \cdot)\rangle_{\mathcal{H}_l}$, leads to

$$\begin{aligned} F_{\mathbf{x}}(H) &= [F_{\mathbf{x}}(h_l)]_{l=1:L} \\ &= [\langle h_l, k_l(\mathbf{x}, \cdot)\rangle_{\mathcal{H}_l}]_{l=1:L}, \end{aligned} \tag{5.6}$$

therefore, the evaluation functional can be expressed as a function of K_L and $\mathbf{h}$ given by

$$F_{\mathbf{x}}(\mathbf{h}) = [\mathbf{h}, K_L(\mathbf{x}, \cdot)], \tag{5.7}$$

where the operator $[\cdot, \cdot]$ denotes the element-wise inner product

$$\left[\begin{bmatrix} a_1 \\ \vdots \\ a_N \end{bmatrix}, \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix}\right] = \begin{bmatrix} \langle a_1, b_1\rangle \\ \vdots \\ \langle a_N, b_N\rangle \end{bmatrix}. \tag{5.8}$$

□

As a consequence of the reproducing property presented in Lemma 6, the *kernel trick for vector-valued RKHS* can now be implemented by choosing $\mathbf{h} = K_L(\mathbf{y}, \cdot), \mathbf{y} \in X$ in (5.7), leading to

$$K_L(\mathbf{y}, \mathbf{x}) = [K_L(\mathbf{y}, \cdot), K_L(\mathbf{x}, \cdot)]. \tag{5.9}$$

Remark 10. *The approach in (5.9) is generic, as for the dimension $L = 1$, $\mathbf{H}_L$ is by construction a (single kernel) RKHS and Lemmas 4-6 correspond to the standard reproducing kernel Hilbert*

space properties. This is observed from eq. (5.7), which collapses into the standard reproducing property of RKHS when (5.8) degenerates into a scalar product.

Observe that the standard reproducing property is defined for scalar-valued function spaces only [Aronszajn, 1950], and therefore it cannot be applied to general vector-valued function spaces, such as $\mathbf{H}_L$. However, if $K_L = [k_l]_{l=1:L}$ is regarded as a vector-valued extension of the concept of reproducing kernel, eq. (5.7) can be seen as a generalised reproducing property in Hilbert spaces of vector-valued functions, which we refer to as a vector-valued RKHS (VRKHS).

Remark 11. *The space of vector-valued functions $\mathbf{H}_L$ in (5.1) has a reproducing property presented in (5.7): For any $\mathbf{x} \in X, \mathbf{h} \in \mathbb{H}$, the evaluation functional $F_{\mathbf{x}}(\mathbf{h})$ can be expanded as a product between $\mathbf{h}$ and $K_L(\mathbf{x}, \cdot)$, where the uniqueness of K_L is guaranteed by construction from the uniqueness of the kernels k_l. This makes it possible to use the space $\mathbf{H}_L$ as a feature space in the construction of kernel regression algorithms, providing a theoretical backbone for their design.*

5.2 Multikernel Regression

We refer to the class of regression algorithms operating on the so-introduced VRKHS as *multikernel regression algorithms*, and introduce two approaches within such class based on ridge regression and least mean square.

5.2.1 Multikernel Ridge Regression

For convenience, we present the single-output version of the algorithm; this will serve as a basis for introducing a general multivariate case as an ensemble of multiple single-output algorithms. Consider the spaces $X \subseteq \mathbb{R}^n$, $D \subseteq \mathbb{R}$, a collection of training pairs $T = \{(\mathbf{x}_t, d_t)\}_{t=1:M}$, a dictionary $\mathcal{D} = \{\mathbf{s}^i\}_{i=1:N}$, and the vector-valued RKHS $\mathbf{H}_L$, as defined in (5.1), from the collection of RKHS $\{\mathcal{H}_i\}_{i=1,...,L}$. The aim of multikernel ridge regression (MKRR) is to learn the relationship of the training pairs by linearly combining multiple feature space elements in an optimal (regularised) least squares sense.

To that end, consider the **time-invariant** mapping given by:

$$\begin{aligned} \Psi : X &\longrightarrow \mathbf{H}_L \\ \mathbf{x} &\longmapsto \Psi(\mathbf{x}) = \begin{bmatrix} \Psi_1(\mathbf{x}) \\ \Psi_2(\mathbf{x}) \\ \vdots \\ \Psi_L(\mathbf{x}) \end{bmatrix}, \end{aligned} \tag{5.10}$$

where $\Psi_l(\mathbf{x}) \in \mathcal{H}_l$ is a real-valued (scalar) function, $l = 1, ..., L$. Based on the features

$\Psi(\mathbf{x}_t), \mathbf{x}_t \in X$, our aim is to approximate the output d_t by

$$y_t = \langle \mathbf{W}, \Psi(\mathbf{x}_t) \rangle , \tag{5.11}$$

where $\mathbf{W} \in \mathbf{H}_L$ is an array of coefficients. As an extension to the monokernel ridge regression algorithm, we propose to find a projection of the optimal weights $\mathbf{W}$ in the empirical [Kung, 2014] space $\mathbf{H}_L^e$ built upon the dictionary $\mathcal{D}$:

$$\mathbf{H}_L^e = \left\{ \mathbf{h} = \begin{bmatrix} \sum_{i=1}^N \omega_{1,i} \Psi_1(\mathbf{s}^i) \\ \vdots \\ \sum_{i=1}^N \omega_{L,i} \Psi_L(\mathbf{s}^i) \end{bmatrix}, \omega_{l,i} \in \mathbb{R}, \mathbf{s}^i \in \mathcal{D} \right\}. \tag{5.12}$$

This projection has the form $\mathbf{W} = [\sum_{i=1}^N \omega_{l,i} \Psi_l(\mathbf{s}^i)]_{l=1:L}$ and therefore the estimate (5.11) can be written as $y_t = \sum_{i=1}^N \sum_{l=1}^L \omega_{l,i} \left\langle \Psi_l(\mathbf{s}^i), \Psi_l(\mathbf{x}_t) \right\rangle$ which, by choosing the mappings Ψ_l to be expansion functions of the spaces $\mathcal{H}_l$, can be expressed via the kernel trick as

$$y_t = \sum_{i=1}^N \sum_{l=1}^L \omega_{l,i} K_l(\mathbf{s}^i, \mathbf{x}_t). \tag{5.13}$$

The optimal ridge regression coefficients $\omega_{l,i}$ are found via the minimisation of the cost function evaluated over the training

$$J = \frac{1}{2} \sum_{t=1}^M e_t^2 + \frac{\rho}{2} \sum_{i=1}^N \sum_{l=1}^L \omega_{l,i}^2, \tag{5.14}$$

where ρ is a regularisation coefficient and e_t denotes the estimation error for the training pair $(\mathbf{x}_t, d_t)$ given by

$$e_t = d_t - y_t. \tag{5.15}$$

Setting $\frac{\partial J}{\partial \omega_{l,i}} = 0$, the optimal weights become

$$\mathbf{\Omega} = (\mathbf{K}\mathbf{K}^T + \rho \mathbb{I})^{-1} \mathbf{K} Y, \tag{5.16}$$

where $\mathbb{I}$ is the identity matrix, and

$$\mathbf{\Omega} = \begin{bmatrix} [\omega_{1,i}]_{i=1..N} \\ \vdots \\ [\omega_{L,i}]_{i=1..N} \end{bmatrix} \in \mathbb{R}^{LN \times 1}, \; [\omega_{l,i}]_{i=1..T} = \begin{bmatrix} \omega_{l,1} \\ \vdots \\ \omega_{l,N} \end{bmatrix} \in \mathbb{R}^{N \times 1}, \tag{5.17}$$

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_1 \\ \vdots \\ \mathbf{K}_L \end{bmatrix} \in \mathbb{R}^{LN \times M}, \qquad Y = \begin{bmatrix} d_1 \\ \vdots \\ d_M \end{bmatrix} \in \mathbb{R}^{M \times 1},$$

while the sub-kernel Gram matrices (also referred to as kernel evaluation matrices) are $(N \times M)$-dimensional and given by $\{\mathbf{K}_l\}_{ij} = K_l(\mathbf{s}^i, \mathbf{x}_j)$.

The MKRR assigns a different kernel combination to each support vector and thus provides an estimation structure that cannot be achieved by standard monokernel ridge regression [Drucker et al., 1996].

Remark 12. *The multikernel ridge regression algorithm offers a physically meaningful framework to approximate non-homogeneous nonlinear mappings by combining subkernels in an optimal regularised least squares sense.*

5.2.2 Multikernel Least Mean Square

We now derive an adaptive multikernel regression algorithms based on the least mean square update. Consider the **time-varying** mapping given by:

$$\begin{aligned} \Psi^t : X &\longrightarrow \mathbf{H}_L \\ \mathbf{x} &\longmapsto \Psi^t(\mathbf{x}) = \begin{bmatrix} c_1^t \Psi_1(\mathbf{x}) \\ c_2^t \Psi_2(\mathbf{x}) \\ \vdots \\ c_L^t \Psi_L(\mathbf{x}) \end{bmatrix}, \end{aligned} \tag{5.18}$$

where t is the time index and $\{c_l^t\}_{t=1,2,\ldots}$ is a sequence of time-varying parameters, to approximate the output d_t by

$$y_t = \left\langle \mathbf{W}, \Psi^t(\mathbf{x}_t) \right\rangle. \tag{5.19}$$

Note that although the inclusion of both parameters $\{c_l^t\}_{l=1,\ldots,L}$ and $\mathbf{W}$ may at first appear redundant, they have different physical meaning: parameters $\{c_l^t\}_{l=1,\ldots,L}$ cater for the instantaneous contribution of each kernel within the multikernel algorithm at the time instant t, and therefore their update is crucial for the adaptive behaviour of the algorithm; the parameter matrix $\mathbf{W}$ (via its L elements) extracts information from the feature samples in order to replicate a nonlinear nature of the signal. Accordingly, the updates of these two weight structures are performed separately.

In a similar manner to the KLMS algorithm in Section 3.3.3, the update of $\mathbf{W}$ is performed using stochastic gradient, as

$$\begin{aligned} \mathbf{W}_t &= \mathbf{W}_{t-1} + \mu e_t \Psi^t(\mathbf{x}_t) \\ &= \mu \sum_{j=1}^{t} e_j \Psi^j(\mathbf{x}_j), \end{aligned} \tag{5.20}$$

where μ is the learning rate, thus leading to the following expression for the output esti-

mate

$$\begin{aligned} y_t &= \left\langle \mu \sum_{j=1}^{t-1} e_j \Psi^j(\mathbf{x}_j), \Psi^t(\mathbf{x}_t) \right\rangle \\ &= \mu \sum_{j=1}^{t-1} e_j \left\langle \Psi^j(\mathbf{x}_j), \Psi^t(\mathbf{x}_t) \right\rangle . \end{aligned} \tag{5.21}$$

The standard KLMS utilises the kernel trick to express the above product via a kernel evaluation. To restate the problem so that the kernel trick can be applied within MKLMS, we use the definition of Ψ^t in (5.18), to rewrite the inner product in (5.21) using the vector-valued RKHS inner product (5.3) as

$$\begin{aligned} \left\langle \Psi^j(\mathbf{x}_j), \Psi^t(\mathbf{x}_t) \right\rangle &= \sum_{l=1}^{L} \left\langle c_l^j \Psi_l(\mathbf{x}_j), c_l^t \Psi_l(\mathbf{x}_t) \right\rangle_{\mathcal{H}_l} \\ &= \sum_{l=1}^{L} c_l^j c_l^t k_l(\mathbf{x}_j, \mathbf{x}_t). \end{aligned} \tag{5.22}$$

By combining eq. (5.22) and (5.21), the output estimate y_t can be rewritten as

$$\hat{d}_i = \mu \sum_{j=1}^{i-1} e_j \sum_{l=1}^{L} c_l^i c_l^j k_l(\mathbf{x}_i, \mathbf{x}_j), \tag{5.23}$$

while by setting $\omega_{i,j,l} = e_j c_l^i c_l^j$, we have

$$y_t = \mu \sum_{j=1}^{t-1} \sum_{l=1}^{L} \omega_{t,j,l} k_l(\mathbf{x}_t, \mathbf{x}_j). \tag{5.24}$$

By virtue of introducing a time-varying optimisation of kernel combinations into MKLMS, the adaptive design of Ψ^t can be performed by updating $\omega_{t,j,l}$, and therefore parameters $\{c_l^t\}_{l=1:L}$ need not be explicitly updated. Notice that as a result of incorporating an LMS update for $\mathbf{W}$ into (5.19), the relation (5.24) models the estimate of d_t as a linear combination of multiple kernels. Therefore, (5.24) can be regarded as a multiple kernel generalisation of (3.50).

We now present the multivariate version of the proposed algorithm which represents an ensemble of multiple univariate MKLMS estimators, in which $\mathbf{d}_t \in \mathbb{R}^m$, for which the coefficient vector is given by $\boldsymbol{\omega}_{t,j,l} \in \mathbb{R}^m$ and the estimated output is in the form

$$\mathbf{y}_t = \mu \sum_{j=1}^{t-1} \sum_{l=1}^{L} \boldsymbol{\omega}_{t,j,l} k_l(\mathbf{x}_t, \mathbf{x}_j), \tag{5.25}$$

or in a more compact notation

$$\mathbf{y}_t = \mu \sum_{j=1}^{t-1} \mathbf{\Omega}_{t,j} K_L(\mathbf{x}_t, \mathbf{x}_j), \tag{5.26}$$

where $\mathbf{\Omega}_{t,j} = [\boldsymbol{\omega}_{t,j,1}, \ldots, \boldsymbol{\omega}_{t,j,L}] \in \mathbb{R}^{m \times L}$ is the coefficient matrix and $K_L(\mathbf{x}_t, \mathbf{x}_j) = [k_l(\mathbf{x}_t, \mathbf{x}_j)]_{l=1:L}$ is the Gram matrix. We refer to the expression in (5.25) and (5.26) as the multivariate-output MKLMS.

Online Implementation of MKLMS: Weight Update and Sparsification

A real-time update stage for the vector $\boldsymbol{\omega}_{t,j,l}$ is needed for the expression (5.25) to accurately represent the observed signal $\mathbf{d}_t$. Since this update is performed in an LMS fashion, the MKLMS architecture effectively comprises two cascaded LMS update stages. Furthermore, unlike the existing KLMS algorithm where the regressors are evaluations of a single (fixed) kernel, the regressors in the MKLMS vary in magnitude for different kernels, therefore, the optimal learning rate depends on the magnitude of these kernel evaluations. To that end, we implement a normalised version of MKLMS so that the value of the learning rate is robust to the kernel magnitude.

For illustration, consider the estimated output $\mathbf{y}_t$ in (5.25) and the estimation error defined by $\mathbf{e}_t = \mathbf{d}_t - \mathbf{y}_t \in \mathbb{R}^m$. By implementing a normalised LMS-based update for $\boldsymbol{\omega}_{t,j,l}$ we have

$$\boldsymbol{\omega}_{t,j,l} = \boldsymbol{\omega}_{t-1,j,l} + \hat{\mu} \mathbf{e}_t \frac{k_l(\mathbf{x}_t, \mathbf{x}_j)}{\epsilon + k_l^2(\mathbf{x}_t, \mathbf{x}_j)}, \tag{5.27}$$

where the learning rate μ is absorbed into the learning rate $\hat{\mu}$ and ϵ is a small positive constant. This also reflects the physical meaning of eqs. (5.25) and (5.27), namely the *estimation* and *update* stages.

Furthermore, for the MKLMS algorithm to be truly adaptive, the dictionary samples must accurately represent the current operating region in the state space. In this sense, we consider the proposed *presence-based* sparsification in Section 3.3.1 with the addition/rejection stage from the novelty criterion.

A pseudo-code implementation of the MKLMS is outlined in Algorithm 1, stating the novelty criterion and its corresponding sample addition and weight update stages, the computation of presence, and the sample elimination stage.

Algorithm 1 Multikernel Least Mean Square (MKLMS)

Initialisation
Dictionary: $\mathcal{D} = \{\mathbf{x}_0\}$
Kernels set: $K = \{k_1, k_2, ..., k_L\}$
Initial weights: $\boldsymbol{\omega}_{k,\mathbf{x}_1} = \hat{\mu}\mathbf{d}_1$ {for each kernel}
For simplicity, we denote by $\boldsymbol{\omega}_{k,\mathbf{x}}$ the weight corresponding to kernel k and support vector $\mathbf{x}$.

for Training pair $(\mathbf{x}_t, \mathbf{d}_t)$ **do**
  Sample deviation: $e_{\mathcal{D}} \leftarrow \min_{\mathbf{x}_j \in \mathcal{D}} \|\mathbf{x}_t - \mathbf{x}_j\|$
  Prediction: $\mathbf{y}_t \leftarrow \mu \sum_{\mathbf{x}_j \in \mathcal{D}} \sum_{k \in K} \boldsymbol{\omega}_{k,\mathbf{x}_j} k(\mathbf{x}_t, \mathbf{x}_j)$
  Error: $\mathbf{e}_t \leftarrow \mathbf{d}_t - \mathbf{y}_t$
  # *Novelty Criterion*
  if $\|\mathbf{e}_t\| \geq \delta_e \wedge e_{\mathcal{D}} \geq \delta_d$ **then**
    Add new sample: $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathbf{x}_t\}$
    for all $k \in K$ **do**
      Initialise new weight: $\boldsymbol{\omega}_{k,\mathbf{x}_t} \leftarrow \hat{\mu}\mathbf{d}_t$
    end for
  else
    for all $k \in K, \mathbf{x}_j \in \mathcal{D}$ **do**
      Update: $\boldsymbol{\omega}_{k,\mathbf{x}_j} \leftarrow \boldsymbol{\omega}_{k,\mathbf{x}_j} + \hat{\mu}\mathbf{e}_t \frac{k(\mathbf{x}_t,\mathbf{x}_j)}{\epsilon + k(\mathbf{x}_t,\mathbf{x}_j)^2}$
    end for
  end if
  # *Presence and Elimination*
  for all $\mathbf{x}_j \in \mathcal{D}$ **do**
    Instantaneous presence: $p_t(\mathbf{x}_j) \leftarrow K_G(\mathbf{x}_j, \mathbf{x}_t)$
    Presence: $P_t(\mathbf{x}_j) \leftarrow (1-\rho)P_{t-1}(\mathbf{x}_j) + \rho p_t(\mathbf{x}_j)$
  end for
  if Perform elimination **then**
    Eliminate sample: $\mathcal{D} \leftarrow \{\mathbf{x}_j \in \mathcal{D} : P_t(\mathbf{x}) \geq \delta_p\}$
  end if
end for

5.3 Implementation of Multikernel Algorithms

The proposed MKRR and MKLMS stem from the standard KRR and KLMS algorithms respectively, therefore, they have similar convergence properties but are more computationally demanding. We now review the convergence properties and complexity of the proposed multikernel algorithms.

5.3.1 Convergence Properties of Multikernel Methods

The existence and uniqueness of solutions for the MKRR [Tobar and Mandic, 2012] are based on the invertibility of the matrix $\mathbf{K}$ (see eqs. (5.16) and (5.17)), which is ensured via the selection of both the support and training sets and the regularisation factor in

a way that is analogous to that of the KRR algorithm—see Section 3.3.2. On the other hand, the convergence of the MKLMS algorithm is governed by the LMS update stage, since for stationary signals the sparsification criterion results in a fixed support dictionary and therefore the convergence of the algorithm depends on the finite-size weights vector. In this way, the convergence of the MKLMS (and also the standard KLMS) algorithm collapses into that of the standard LMS, where the regressors are the kernel evaluations. Therefore, convergence properties for MKLMS stem from the standard (normalised) LMS convergence [Widrow and Hoff, 1960, Haykin, 2001], by considering the transformed samples (or kernel evaluations) as the regressors. Furthermore, note that for stationary data, and therefore for a fixed support dictionary, the MKLMS converges to the non-regularised ($\rho = 0$) MKRR, this is because the relationship between MKRR and MKLMS can be thought of as the relationship between the standard LMS and the ridge regression algorithm with the kernel evaluation samples as regressors.

5.3.2 Computational Complexity of Proposed Algorithms

Computation of the weights in ridge regression algorithms involves both matrix multiplications and inversions. As the complexity of the multiplication of an $n \times m$ matrix by an $m \times p$ matrix is of order $\mathcal{O}(nmp)$, and the inversion of a $n \times n$ matrix is of order $\mathcal{O}(n^3)$, based on eq. (5.16) we summarise the complexity of weights computation for KRR algorithms in Table 5.1 for N support vectors, M training pairs and an L-kernel MKRR.

Table 5.1: Computational complexity of weight computation for KRR algorithms

Algorithm	Complexity
KRR	$\mathcal{O}(M^2N) + \mathcal{O}(MN) + \mathcal{O}(M^3)$
MKRR	$\mathcal{O}(L^2M^2N) + \mathcal{O}(LMN) + \mathcal{O}(L^3M^3)$

The computation complexity for optimal weights of an L-kernel MKRR is therefore L^3 times larger than that of the monokernel algorithm, however, this routine is only performed once and does not affect the algorithm implementation.

Regarding the computational complexity of the algorithm implementation, the number of kernel evaluations executed by an L-kernel MKRR algorithm is L times that of the monokernel version, that is, N kernel evaluations for the MKRR and LN for the KRR. Furthermore, for the adaptive MKLMS, as the dictionary size is time-varying and is given by the sparsification criteria, the ability of the algorithm to estimate an unknown mapping with fewer support vectors will affect its complexity. The chosen presence-based sparsification criterion, which eliminates samples not contributing to the estimation, reduces computational complexity and running time.

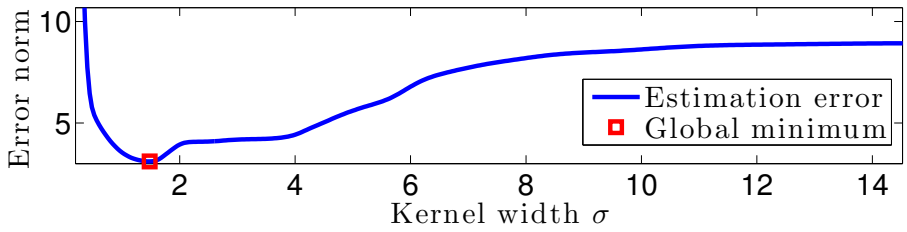

Figure 5.2: Training error norm for different values of the kernel width. The global minimum has the value of 3.11 and is reached for $\sigma = 1.48$.

5.4 Examples

5.4.1 Nonlinear Function Approximation: Multikernel Ridge Regression

Multikernel learning benefits from large training sets. We now illustrate the learning capability of the multikernel concept on the estimation of a piecewise-differentiable continuous function. The function and the support vectors are shown in fig. 5.1, all samples were considered for training.

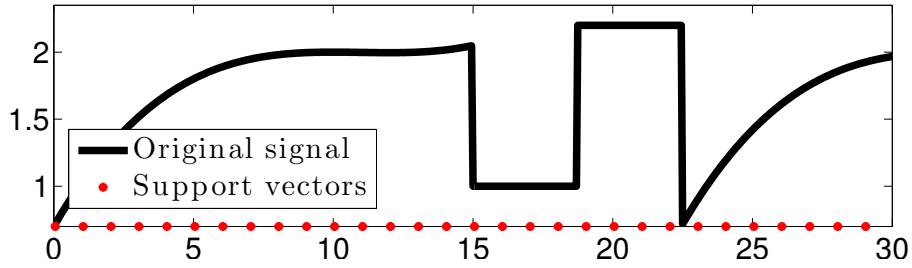

Figure 5.1: A nonlinear function and support vectors.

We first considered the Gaussian kernel, for which the optimal (least squares) width σ was found by means of exhaustive search, as this optimisation cannot be performed in closed form due to the nonlinear dependency of the kernel evaluation $K_G(\mathbf{x}, \mathbf{y}) = \exp(-\sigma^{-2} \|\mathbf{x} - \mathbf{y}\|^2)$ on σ. Fig. 5.2 shows the (single-kernel) estimation error for the considered support vectors as a function of the kernel width; the global minimum is highlighted. This reveals that the best single-kernel estimate has an associated error of 3.11 (for $\sigma = 1.48$).

We then implemented a monokernel algorithm using $\sigma = 1.48$ and a two-kernel multikernel algorithm using two Gaussian kernels of widths $\sigma_1 = 1.48$ and $\sigma_2 = 0.24$. The parameter σ_2 was found by analysing the residual of the monokernel estimate. The estimates for both KRR algorithms are shown in fig. 5.2 and the norm of the estimation error in Table 5.2.

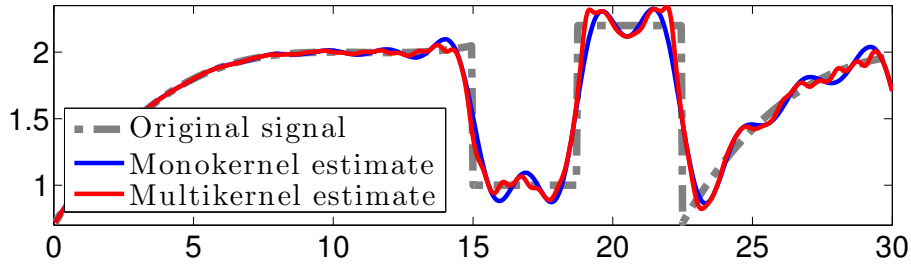

Figure 5.3: Nonlinear function estimation using mono- and multi-kernel ridge regression algorithms.

Algorithm	Monokernel	Multikernel
$\|\text{Error}\|$	3.11	**2.86**

Table 5.2: Estimation error for mono- and multi-kernel ridge regression algorithms.

Observe from fig. 5.3 that multikernel estimation provided both reduced overshoot around $x = 15$ and suppressed oscillations in $x \in [25, 30]$. Additionally, in terms of overall performance, the multikernel estimate was 8.7% more accurate than its monokernel counterpart. This experiment also reveals the ability of the multikernel algorithm to provide localised estimation, which benefits from large training sets in order to assign tailored kernel combinations for each region of the input space.

5.4.2 Prediction of Nonlinear Signals: Multikernel Least Mean Square

The aim of this example is to show contribution of individual kernels towards the overall estimation. To this end, we considered the prediction of a well understood nonlinear and chaotic trivariate signal, a discretised version of the Lorenz attractor:

$$l_{t+1} = \begin{bmatrix} l^x_{t+1} \\ l^y_{t+1} \\ l^z_{t+1} \end{bmatrix} = \begin{bmatrix} l^x_t \\ l^y_t \\ l^z_t \end{bmatrix} + s \begin{bmatrix} \alpha(l^y_t - l^x_t) \\ l^x_t(\beta - l^z_t) - l^y_t \\ l^x_t l^y_t - \gamma l^z_t \end{bmatrix} \tag{5.28}$$

where s is the discretisation step, α, β, γ are real-valued parameters, and l^x_t, l^y_t, l^z_t are the components of l_t.

The MKLMS and two versions of KLMS with different kernels were assessed on a one-step ahead prediction of the trivariate signal in (5.28) considering five regressors for each signal, that is, the task was to predict l_t using $l_{t-5:t-1}$. We implemented the KLMS algorithms with triangular and Gaussian kernels, and calculated the kernel parameters via exhaustive search minimisation of the predicted mean square error (MSE) for a pre-

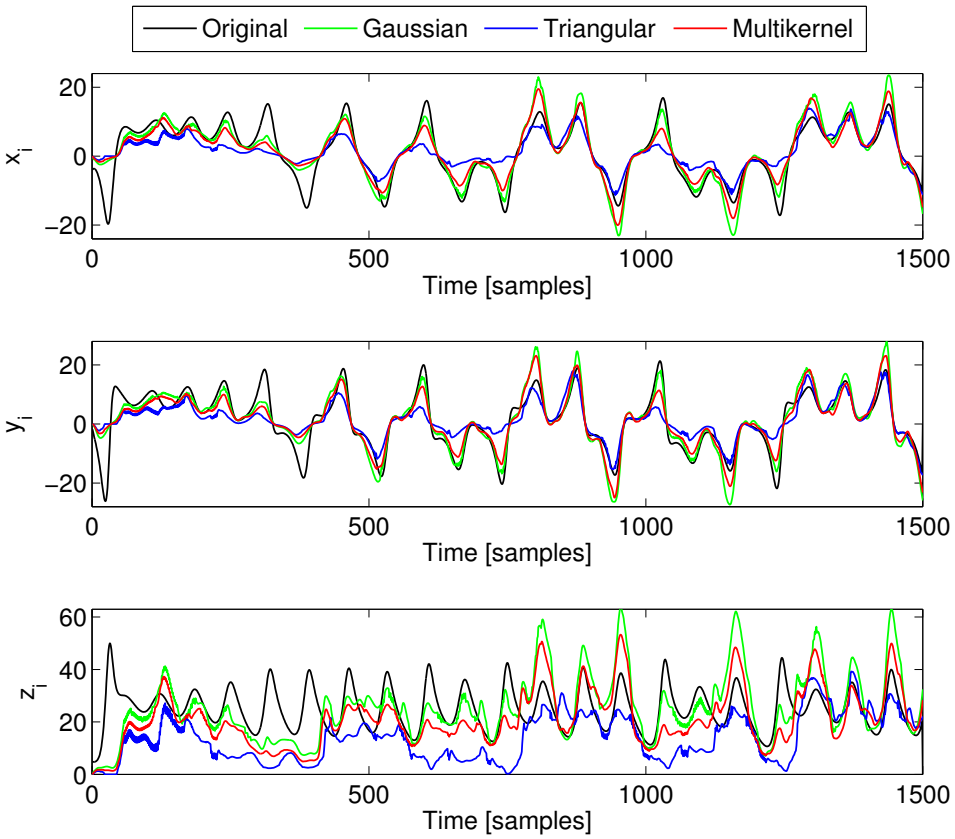

Figure 5.4: Trivariate original signal and KLMS estimates.

defined training set. The Gaussian kernel width was set to $\sigma^2 = 80$ and the triangular kernel threshold to $\Delta = 0.18$ (see eqs. (2.17) and (2.19)). The multikernel LMS comprised a triangular and a Gaussian kernel with the same parameters. All three kernel algorithms considered were implemented using the coherence sparsification criteria with parameters $\delta_e = 0.15, \delta_d = 1$ (see eqs. (3.31)-(3.32)), and learning rates $\mu = 0.3, \hat{\mu} = 0.5$; while the system parameters were $s = 0.01,\ \alpha = 10,\ \beta = 28$ and $\gamma = 8/3$. Figure 5.4 shows the original chaotic signal and the kernel estimates for a 1500-sample realisation.

We employed the norm of the columns of $\mathbf{\Omega}_{i,j}$ as a measure of contribution of each subkernel. Figure 5.5 shows that the kernel-wise contributions in both multikernel and single kernel cases were similar, illustrating that, as desired, different kernels within MKLMS account for different nonlinear features of the data and can be associated physical meaning. Figure 5.6 shows the averaged prediction MSE and size of the dictionary (in number of samples) for each of the algorithms considered over 30 independent 3000-sample trials. Observe that the multikernel LMS provided more accurate estimates in terms of the averaged prediction MSE, while using fewer samples than the existing KLMS algorithms. Notice that in the transient period the estimation error of the multikernel algorithm is greater than that of the monokernel ones, since the MKLMS adjusts

Table 5.3: Averaged prediction gains and running times for the Gaussian KLMS, triangular KLMS and MKLMS.

Algorithm	Gaussian	Triangular	Multikernel
R_p	1.01	0.73	**1.37**
Time [s]	31.18	30.93	50.03

more parameters than the monokernel algorithm and therefore needs more training data.

The steady state prediction performance of the kernel algorithms was assessed using the prediction gain

$$R_p = 10 \log_{10} \left(\frac{\sum_{t=T_0}^{T} \|l_t\|_2^2}{\sum_{t=T_0}^{T} \left\| l_t - \hat{l}_t \right\|_2^2} \right), \tag{5.29}$$

evaluated at the steady state (i.e. $T_0 = 1500$ and $T = 3000$) for each of the realisations. The average prediction gains over 30 realisations, as well as the running time, are given in Table 5.3. The multikernel approach provided the most accurate prediction at a lower computational complexity than the sum of its subkernels, due to the smaller support dictionary (Fig. 5.6, bottom).

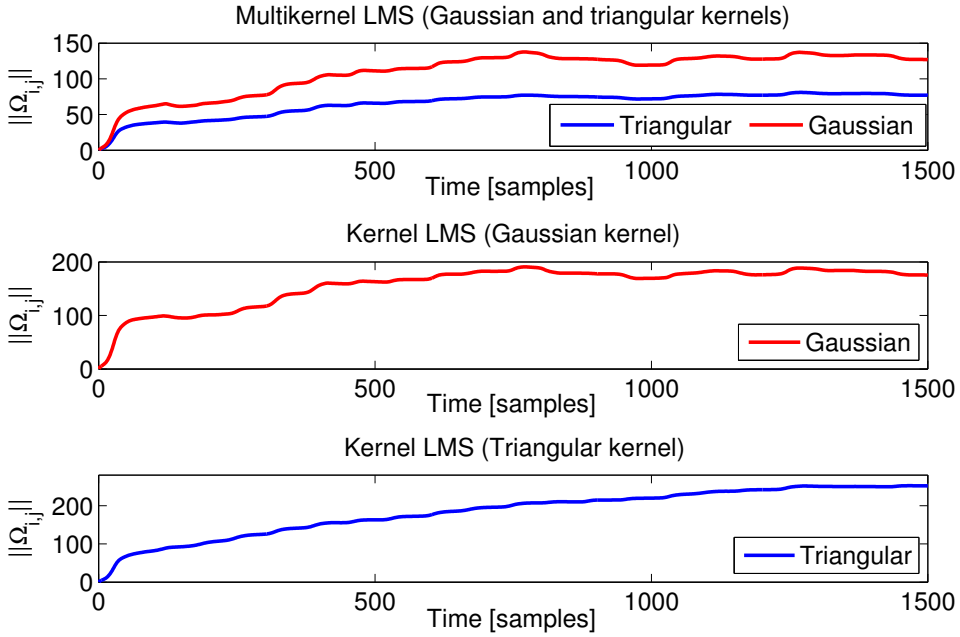

Figure 5.5: Time varying magnitude of the weight matrix for all the three implemented kernel algorithms.

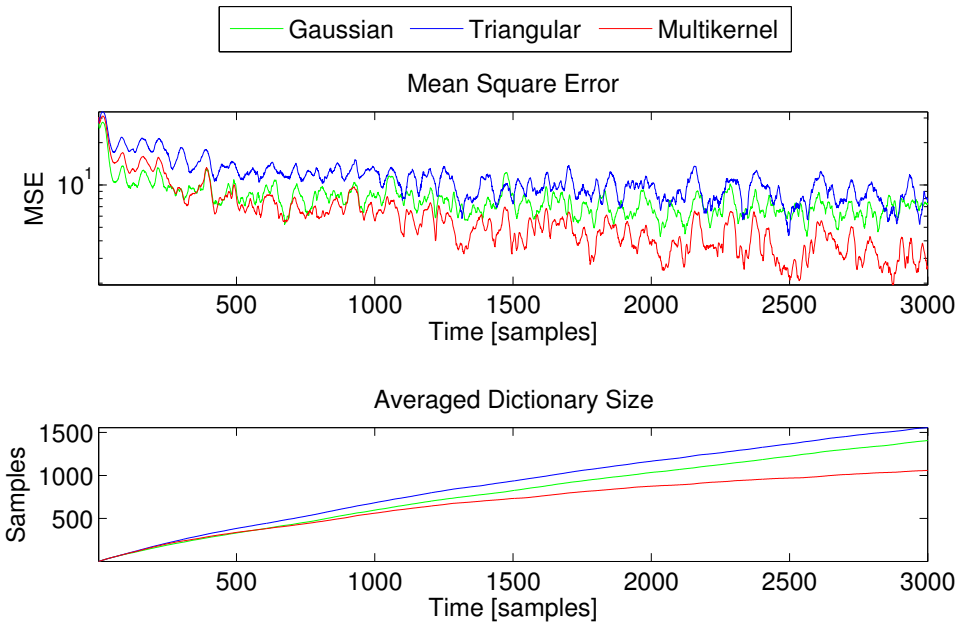

Figure 5.6: Averaged MSE and dictionary size (in number of samples) over 30 trials for the kernel algorithms.

5.5 Discussion

The multikernel estimator resulting from implementing regression algorithms on the proposed vector-valued RKHS is intuitive and has the ability to capture *different types* of nonlinear behaviour from the input data as documented in [Gönen and Alpaydın, 2011, Yukawa, 2012]. Furthermore, the approach is flexible, since the vector-kernel is not constrained to be two- or four-dimensional as in the hypercomplex case, but is only set as a design parameter based on the observed nonlinear features of the data and the available computational power.

The following lemma gives a sufficient condition for designing vector-kernels and helps to establish the difference between the VRKHS and QRKHS approaches.

Lemma 7. *$\overrightarrow{K} = [K_1, K_2, \ldots, K_L]^T$ is a valid vector-valued kernel if all its subkernels are positive definite scalar kernels.*

Proof. The proof follows from the construction of the vector-kernel. If every subkernel K_i is positive definite, then there exists a mapping ψ_i such that $K_i(\mathbf{x}, \mathbf{y}) = \psi_i^H(\mathbf{x})\psi_i(\mathbf{y})$. As a consequence, by denoting $\Psi = [\psi_1, \ldots, \psi_L]$ the array of the mappings ψ_i, we have $\overrightarrow{K}(\mathbf{x}, \mathbf{y}) = \text{diag}\left(\Psi^H(\mathbf{x})\Psi(\mathbf{y})\right)$, that is, a vector-valued kernel. □

We now address the following question: Can a quaternion kernel be written as 4D vector-kernel? Through the following theorem, we show that, despite sharing the same

number of degrees of freedom (four), they are correspond to different feature spaces.

Theorem 5. *Let K and $\overrightarrow{K}$ be arbitrary quaternion- and vector-kernels respectively given by*

$$K = K_r + iK_i + jK_j + kK_k \in \mathbb{H}$$
$$\overrightarrow{K} = [K_1\ K_2\ K_3\ K_4]^T \in \mathbb{R}^4,$$

where the real and imaginary parts of K and the subkernels of $\overrightarrow{K}$ are scalar, real-valued, functions.

The following statements about the $\mathbb{R}^4$ representation of K and the quaternion representation of $\overrightarrow{K}$ are true:

(a) $K' = [K_r\ K_i\ K_j\ K_k]^T \in \mathbb{R}^4$ *is not a vector-kernel,*

(b) $\overrightarrow{K}' = K_1 + iK_2 + jK_3 + kK_4 \in \mathbb{H}$ *is not a Hermitian positive definite quaternion-kernel.*

Proof. The proof follows from the properties of vector and quaternion kernels stated in Lemmas 3 and 7.

(a) Quaternion kernels are Hermitian and positive definite; consequently, according to Lemma 3(a) the imaginary parts of K given by K_i, K_j, K_k are not symmetric. Therefore, based on Lemma 7, the array $K' = [K_r\ K_i\ K_j\ K_k]$ is not a vector-kernel (since its subkernels are not symmetric and positive definite).

(b) As $\overrightarrow{K}$ is a vector-kernel, Lemma 7 states that K_1, K_2, K_3 are symmetric and positive definite. Therefore, $\overrightarrow{K}' = K_1 + iK_2 + jK_3 + kK_4$ is not Hermitian due to Lemma 3(b). □

Theorem 5 holds the key for identifying the differences between the QRKHS and VRKHS approaches: Although the $\mathbb{R}^4$ representation of the QRKHS generated by a quaternion kernel K of the form

$$\mathcal{H}_{\mathbb{R}} = \left\{ \begin{pmatrix} \Re f(\mathbf{x}) \\ \Im_i f(\mathbf{x}) \\ \Im_j f(\mathbf{x}) \\ \Im_k f(\mathbf{x}) \end{pmatrix} \in \mathbb{R}^4,\ \text{s.t. } f(\mathbf{x}) = \sum_{n=1}^{\infty} a_n K(\mathbf{x}_n, \mathbf{x}) \right\} \tag{5.30}$$

where $a_n \in \mathbb{H},\ \mathbf{x} \in X$, is indeed an RKHS (its evaluation functional is bounded), its real vector-valued reproducing kernel is not given by the $\mathbb{R}^4$ representation of the quaternion kernel $K' = [K_r\ K_i\ K_j\ K_k]$ of $K \in \mathbb{H}$, because K' **is not a vector kernel in** $\mathbb{R}^4$.

Remark 13. *The vector-valued representation of a QRKHS in eq. (5.30) is a VRKHS, however, finding its associated vector-valued kernel is not trivial.*

Part II

Kernel-Based State-Space Models

Chapter 6

Bayesian Filtering and Monte Carlo Methods

I think that it is a relatively good approximation to truth—which is much too complicated to allow anything but approximations—that mathematical ideas originate in empirics.

-John Von Neumann.
("The Mathematician," 1956.)

The purpose of filtering is to extract useful information from noisy measurements. Both linear and kernel adaptive filters considered in Chapter 3 proceed by modelling the estimate as a parametric function of the available information and then finding such parameters in offline or online manner depending on the application at hand. This parametric approach is only possible owing to the supervised nature of the applications that adaptive filters consider, since observed samples from both the regressor and desired processes are required to find meaningful parameters. We will now study a different approach to filtering, where rather than assuming a model for the estimate we assume a (stochastic) dynamic model relating the observed and the target processes, and then *derive* an expression for the estimate from such model. This approach, referred to as stochastic filtering, allows us to account for the case when the target process is unobserved and therefore supervised learning is not possible—a key requirement in this context is a dynamic model explaining the evolution of the hidden process and how it relates to the observations.

Both batch and sequential Monte Carlo methods benefit from a comprehensive literature [Doucet et al., 2001, Arulampalam et al., 2002, Andrieu et al., 2003, Djurić et al., 2003]. This chapter presents a brief introduction to Monte Carlo methods in a concise and unified fashion to give the theoretical background for Chapter 7. The reader familiar with (recursive) Monte Carlo simulation and Bayesian filtering can skip this chapter.

6.1 Bayesian Filtering

Stochastic Filtering [Bain and Crisan, 2009] refers to the estimation of a latent stochastic process $X_{1:t}$ based on the observed sequence[1] $Y_{1:t} = y_{1:t}$, where the processes $X_{1:t}$ and $Y_{1:t}$ are related according to some mathematical model $\mathcal{M}$. We consider the class of state-space models (SSM), where X_t is the Markovian state and Y_t the (noisy) observation.

The solution to the filtering problem is given by the posterior density of the latent process conditional to the observations, that is, $p(X_{1:t}|y_{1:t})$. This conditional distribution is the solution of the Kushner-Stratonovich equation [Stratonovich, 1960, Kushner, 1964], a nonlinear measure-valued differential equation that admits a closed-form solution only for a restricted class of systems, such as linear and Gaussian ones (Kalman filter [Kalman, 1960]) or those satisfying the Beneš condition (Beneš filter [Beneš, 1981]). For the general case, this distribution is mathematically intractable and numerical algorithms to find approximate solutions are required.

6.1.1 Filtering Equations

We consider the continuous-state discrete-time SSMs of the form

$$\begin{aligned} X_{t+1} &= f_t(X_t) + W_t, \\ Y_t &= h_t(X_t) + V_t \end{aligned} \tag{6.1}$$

where $f_t : \mathbb{R}^n \to \mathbb{R}^n$, $h_t : \mathbb{R}^n \to \mathbb{R}^m$ and V_t, W_t are noise processes, these allow for modelling of nonlinear and non-Gaussian systems. The unobserved signal $\{X_t; t \in \mathbb{N}\} \subset \mathcal{X}$ is modelled as a Markov processes with a transition density defined by f_t and the statistics of W_t and the observations $\{Y_t; t \in \mathbb{N}\} \subset \mathcal{Y}$ are assumed to be independent given the process $\{X_t; t \in \mathbb{N}\}$. See [Doucet et al., 2001].

We are interested in the marginal filtering density, or more specifically, in estimating it in a recursive fashion. In order to do so, let us first observe that the prediction density $p(x_{t+1}|y_{1:t})$ can be expressed by marginalising x_t out of $p(x_{t:t+1}|y_{1:t})$ and using

[1] We use uppercase letters Y_t to denote a random variable, and lowercase letters y_t for its (observed) realisation at time t; however, we may use these notations interchangeably when it does not lead to confusion, e.g. when writing conditional densities. Additionally, notice that in Part II of the thesis we have adopted the standard Bayesian filtering notation, that is, X_t for the hidden state and Y_t for the observation signal.

the definition of conditional density, that is

$$\begin{aligned} p(x_{t+1}|y_{1:t}) &= \int p(x_{t:t+1}|y_{1:t})\mathrm{d}x_t \\ &= \int p(x_{t+1}|x_t, y_{1:t})p(x_t|y_{1:t})\mathrm{d}x_t \\ &= \int p(x_{t+1}|x_t)p(x_t|y_{1:t})\mathrm{d}x_t \end{aligned} \tag{6.2}$$

where the last step is due to the Markovian property of $\{X_t; t \in \mathbb{N}\}$. Furthermore, let us now state the marginal density in terms of the prediction density using the Bayes theorem

$$\begin{aligned} p(x_{t+1}|y_{1:t+1}) &= \frac{p(y_{t+1}|x_{t+1}, y_{1:t})p(x_{t+1}|y_{1:t})}{p(y_{t+1}|y_{1:t})} \\ &= \frac{p(y_{t+1}|x_{t+1})p(x_{t+1}|y_{1:t})}{p(y_{t+1}|y_{1:t})} \end{aligned} \tag{6.3}$$

where the last step is due to the independence of the observations given $\{X_t; t \in \mathbb{N}\}$. Eqs. (6.2) and (6.3) are referred to as *prediction* and *update* stages respectively and they are the backbone of Bayesian filtering. Although the prediction and update stages seem to be fairly straightforward, they can be very difficult to implement for general SSM when the filtering solution is intractable.

6.1.2 Learning the Dynamical Model

The choice of the model $\mathcal{M}$ is crucial for within Bayesian inference and also challenging. A typical case is when a mathematical model for the observation and hidden processes can be only partially derived, that is, when the model known up to some unknown parameters θ.

These parameters can then be found by maximising their likelihood. Let us denote by $l(\theta)$ the log-likelihood function conditional to the observations $y_{1:T}$

$$\begin{aligned} l(\theta) &= \log p(y_{1:T}|\theta) \\ &= \log \prod_{t=1}^{T} p(y_t|y_{1:t-1}, \theta) \\ &= \sum_{t=1}^{T} \log p(y_t|y_{1:t-1}, \theta) \end{aligned} \tag{6.4}$$

Autoregressive Model for $Y_{1:T}$

When an autoregressive model is assumed for the observed process, as in the case of adaptive filtering, the likelihood function is known in closed form and depends on the

observations $y_{1:T}$. Consider the linear Gaussian model defined by $y_{t+1} = ay_t + \sigma\eta_t, \eta_t \sim \mathcal{N}(0,1)$, where the parameters to find are $\theta = [a, \sigma]$, $p(y_t|y_{1:t-1}, \theta) = \mathcal{N}(y_t; ay_{t-1}, \sigma)$; the log-likelihood w.r.t the observations $y_{1:T}$ then becomes

$$\begin{aligned} l(\theta) &= \sum_{t=1}^{T} \log\left(\frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-|y_t - ay_t|^2}{2\sigma^2}\right)\right) \\ &= -\sum_{t=1}^{T} \log \sigma\sqrt{2\pi} + \sum_{t=1}^{T}\left(\frac{-|y_t - ay_t|^2}{2\sigma^2}\right) \end{aligned} \tag{6.5}$$

and can be then maximised with respect to a and θ, thus yielding the maximum likelihood parameter estimates.

State-Space Model for $X_{1:T}$ and $Y_{1:T}$

When the observed process is modelled as the output of a SSM, the density $p(y_t|y_{1:t-1}, \theta)$ is not known generally in closed form, and the log-likelihood function has the form

$$l(\theta) = \sum_{t=1}^{T} \log \int_{\mathcal{X}^2} p(y_t|x_t, \theta)p(x_t|x_{t-1}, \theta)p(x_{t-1}|y_{1:t-1}, \theta)\mathrm{d}x_{t-1:t}. \tag{6.6}$$

The assumption of the hidden process $\{X_t\}_{t\in\mathbb{N}}$ results in a likelihood function that comprises intractable integrals which can only be maximised using numerical methods such as Expectation-Maximisation [Dempster et al., 1977]. In this sense, observe that the densities in eq. (6.6) are either known or they can be approximated (at least for a known *candidate* θ) and, consequently, Monte Carlo-based methods to sample from these integrals can be considered.

6.2 Monte Carlo Sampling

Monte Carlo methods [Robert and Casella, 1999] are a class of numerical algorithms that approximate a probability density $\pi(x)$ by the empirical measure

$$\widehat{\pi}_N(x) = \frac{1}{N}\sum_{i=1}^{N} \delta_{x^{(i)}}(x) \tag{6.7}$$

where $\delta_{x^{(i)}}(\cdot)$ denotes the Dirac delta located at $x^{(i)}$ and the samples $\{x^{(i)}\}_{i=1:N}$ are drawn from $\pi(x)$. The density $\pi(x)$ is known as target distributions and the samples $\{x^{(i)}\}_{i=1:N}$ as particles.

The Monte Carlo estimates allows us to approximate expectations taken with respect to the target density of the form $I(\psi) := \int \psi(x)\pi(x)\mathrm{d}x$, where ψ is a test function,

by replacing the target density by the MC approximation, thus yielding the estimate

$$\widehat{I}(\psi) := \int \psi(x)\widehat{\pi}_N(x)\mathrm{d}x = \frac{1}{N}\sum_{i=1}^{N}\psi(x^{(i)}) \tag{6.8}$$

which is unbiased and the variance of its approximation error decreases at a rate $O(1/N)$ [Doucet and Johansen, 2009].

The main problem of this basic Monte Carlo approach is that sometimes, especially when the target distribution is high-dimensional or complex, there is not a straightforward way to generate the samples $\{x^{(i)}\}_{i=1:N}$. We now review two ways to address this problem.

6.2.1 Importance Sampling

When samples $\{x^{(i)}\}_{i=1:N}$ distributed according to the target density are not available, an alternative to compute the estimate (6.8) is to express the true integral as

$$I(\psi) = \int \psi(x)\pi(x)\mathrm{d}x = \int \psi(x)\frac{\pi(x)}{q(x)}q(x)\mathrm{d}x \tag{6.9}$$

where $q(x)$ is a density from which it is easy to draw samples (e.g. a multivariate Gaussian or an uniform density) and such that $\pi(x) > 0 \Rightarrow q(x) > 0$. We can therefore compute the Monte Carlo estimate of the expression in (6.9) by using the particle approximation $\widehat{q}_N(x) = \frac{1}{c}\sum_{i=1}^{N}\delta_{x^{(i)}}(x)$, where $c > 0$, $x^{(i)} \sim q(x)$, and identifying the test function $\psi(x)\frac{\pi(x)}{q(x)}$, that is,

$$\widehat{I}^{IS}(\psi) := \int \psi(x)\frac{\pi(x)}{q(x)}\widehat{q}_N(x)\mathrm{d}x = \frac{1}{c}\sum_{i=1}^{N} w_i\psi(x^{(i)}) \tag{6.10}$$

where the weights are given by $w_i = \frac{\pi(x^{(i)})}{q(x^{(i)})}$ and the normalising constant can be computed by setting $\widehat{I}^{IS}(1) = I(1) = \int \pi(x)\mathrm{d}x = 1$, which gives $c = \sum_{j=1}^{N} w_j$.

The requirement of the proposal density $q(\cdot)$ to have a support that includes that of the target density can be understood from the fact that $q(\cdot)$ should generate samples in all regions of the sample space where the target density can generate samples, thus providing a reliable approximation. In this way, the importance weights can be seen as a compensation for the over-generation (or under-generation) of samples from the proposal with respect to the target densities

The IS approach [Geweke, 1989] addresses the problem of calculating approximations of $I(\psi)$ when sampling form π is not possible or too difficult, however, observe that the IS approach does not generate samples according to $\pi(\cdot)$, but only samples from $q(\cdot)$

which are the weighted to compute expectations with respect to $\pi(\cdot)$.

6.2.2 Markov Chain Monte Carlo (MCMC)

MCMC methods [Andrieu et al., 2003] allow us to draw samples from distributions by constructing a Markov chain which has the target distribution as steady-state (stationary) distribution; this way, the desired samples are obtained by simulating the chain until it converges. As a consequence, and unlike IS, MCMC operates by relaxing the requirement that se samples should be independent and generates a sequence of correlated samples [Murray, 2007] (i.e. states of the Markov chain). The design of the Markov chain involves the choice of a (Markovian) transition density denoted by $T(x' \leftarrow x)$, which represents the probability of moving from x to x'; thus, at each step the move form the current sample x is chosen by drawing $x' \sim T(x' \leftarrow x)$.

If the stationary distribution of the chain is denoted by π, then the transition density T needs to fulfil the requirement[2]

$$\pi\left(x^{(i+1)}\right) = \int \pi\left(x^{(i)}\right) T\left(x^{(i+1)} \leftarrow x^{(i)}\right) \mathrm{d}x^{(i)} \tag{6.11}$$

which means that if the chain is at the stationary distribution, the operator T will leave it in the same distribution.

Furthermore, for the chain to converge to the stationary distribution π, two properties are required: *Irreducibility*, the probability to reach any state in a finite number of steps is greater than zero, and *aperiodicity*, the chain does not get trapped in cycles of the space [Murray, 2007]. A sufficient, but not necessary, condition for the chain to have $\pi(x)$ as the desired stationary distribution is the detailed balance

$$\pi\left(x^{(i)}\right) T\left(x^{(i+1)} \leftarrow x^{(i)}\right) = \pi\left(x^{(i+1)}\right) T\left(x^{(i)} \leftarrow x^{(i+1)}\right), \tag{6.12}$$

where chains satisfying this condition are referred to as *reversible chains*. The detailed balance can be interpreted as if the chain is "in equilibrium" at $x^{(i)}$, then the probability of leaving this equilibrium (towards $x^{(i+1)}$) is the same probability of coming back (i.e. from $x^{(i+1)}$ to $x^{(i)}$). Observe that by integrating both sides of eq. (6.12) with respect to $x^{(i)}$, gives the stationarity density condition in eq. (6.11)

The most popular MCMC variant is the celebrated Metropolis-Hasting algorithm [Metropolis et al., 1953, Hastings, 1970], which generates samples distributed according to π by a two-step procedure: at time instant i, a candidate move x^c is proposed given

[2] In general, the MCMC literature assumes discrete state-spaces whereby the transition density T is a matrix, π a distribution and the condition in eq. (6.11) is expressed as a sum. For the continuous case, the transition operator is referred to as Markov kernel (rather than matrix); however, we will avoid this notation and refer by kernels to reproducing kernels only.

Algorithm 2 Metropolis-Hastings MCMC

1: Initialise $x^{(0)}$.
2: **for** $i = 0 : N - 1$ **do**
3: Sample $x^c \sim q(x^c|x^{(i)})$
4: Sample $u \sim \mathcal{U}_{[0,1]}$
5: **if** $u < A_{MH} = \min\left\{1, \frac{\pi(x^c)q\left(x^{(i)}|x^c\right)}{\pi\left(x^{(i)}\right)q\left(x^c|x^{(i)}\right)}\right\}$ **then**
6: Set $x^{(i+1)} = x^c$
7: **else**
8: Set $x^{(i+1)} = x^i$
9: **end if**
10: **end for**

the current state $x^{(i)}$ and then accepted with probability $A_{MH} = \min\left\{1, \frac{\pi(x^c)q\left(x^{(i)}|x^c\right)}{\pi\left(x^{(i)}\right)q\left(x^c|x^{(i)}\right)}\right\}$, when the proposal density is symmetric, i.e. $q\left(x^{(i)}|x^c\right) = q\left(x^c|x^{(i)}\right)$ the algorithm is known as the Metropolis algorithm and has an acceptance ratio $A_M = \min\left\{1, \frac{\pi(x^c)}{\pi\left(x^{(i)}\right)}\right\}$. The Metropolis-Hasting algorithm is presented in Algorithm 2.

6.3 Sequential Monte Carlo Methods

The marginal filtering density can be computed with the IS and MCMC methods revised in the previous section; however, these are not well-suited for recursive sampling, where, based on $p(x_t|y_{1:t})$ and a novel observation y_{t+1}, one would like to update the current samples to now resemble $p(x_{t+1}|y_{1:t+1})$ rather than drawing an entirely new set of samples from this marginal.

We will then consider sequential Monte Carlo methods [Doucet et al., 2000], a simulation-based method for sampling from a sequence of time-varying probability densities. In the Bayesian filtering context, SMC are refereed to as particle filters and allow for estimating the posterior density without requiring linearity or Gaussianty of the SSM. Particle filters (PF) compute approximations of the filtering distribution in the form of a discrete set of *weights* and *particles* that are recursively updated based on the prediction and update stages of Bayesian filtering presented above.

The PF procedure can be easily understood my modifying the IS approach to operate recursively. At time t, PF approximates the posterior at by the empirical measure

$$\hat{p}_N\left(x_t|y_{1:t}\right) = \sum_{i=1}^{N_p} w_t^{(i)} \delta_{x_t^{(i)}}(x_t) \tag{6.13}$$

where $w_t^{(i)}$ and $x_t^{(i)}$ are the weights and particles respectively. Then, the estimate of the

Algorithm 3 A Generic Particle Filter

1: Initialise particles $x_0^{(i)}$ and weights $\alpha_0^{(i)}$.
2: **for** time $t = 1, 2, \ldots$ **do**
3: **for** particle $i = 1 : N$ **do**
4: Sample particle $x_t^{(i)} \sim q(x_t | x_{1:t-1}^{(i)}, y_t)$
5: Calculate weight $\alpha_t^{(i)}$ according to eq. (6.14)
6: **end for**
7: Normalise weights $\alpha_t^{(i)} \longleftarrow \alpha_t^{(i)} / \sum_{j=1}^{N} \alpha_t^{(i)}, i = 1 : N$
8: **if** $\widehat{N}_{eff} < N_T$ **then**
9: $\left\{\left(x_t^{(i)}, \alpha_t^{(i)}\right)\right\}_{i=1:N} \longleftarrow \text{RESAMPLE}\left\{\left(x_t^{(i)}, \alpha_t^{(i)}\right)\right\}_{i=1:N}$
10: **end if**
11: **end for**

posterior at time $t+1$ can be computed by propagating the particles according to some proposal density $x_{t+1}^{(i)} \sim q\left(x_{t+1} \middle| x_t^{(i)}, y_{1:t}\right)$ and then calculating the importance weights according to

$$w_t^{(i)} \quad \propto \quad w_{t-1}^{(i)} \frac{p\left(y_t \middle| x_t^{(i)}\right) p\left(x_t^{(i)} \middle| x_{t-1}^{(i)}\right)}{q\left(x_t^{(i)} \middle| x_{t-1}^{(i)}, y_{1:t}\right)}. \tag{6.14}$$

A particular case is obtained when the proposal density is chosen to be equal to the prior distribution, that is $p\left(x_t^{(i)} \middle| x_{t-1}^{(i)}\right) = q\left(x_t^{(i)} \middle| x_{t-1}^{(i)}, y_{1:t}\right)$; in such case, weight update is given by $w_t^{(i)} \propto w_{t-1}^{(i)} p\left(y_t \middle| x_t^{(i)}\right)$. This recursive method is referred to as sequential importance sampling (SIS).

The main drawback of the SIS if the so-called particle degeneracy, this refers to the fact that the variance of the importance weights can only increase [Doucet et al., 2000] and therefore all particles except one do not contribute to the density estimate due to having almost zero weight. A natural approach to bypass this problem is to increase the number of particles, however, this is impracticable and will lead to a waste of computational resources. An alternative to avoid particle degeneracy is to resample the particle population whenever significant degeneracy is observed. The particle degeneracy can be assessed through the effective sample size [Kong et al., 1994, Liu, 1996], or its estimate given by

$$\widehat{N}_{eff} = \left(\sum_{i=1}^{N_p} \left(w_t^{(i)} \right)^2 \right)^{-1}.$$

Therefore, when $\widehat{N}_{eff}$ falls below a certain threshold, the resampling is performed. A basic resampling strategy is to choose (with restitution) from the set of particles where particle $x_t^{(i)}$ is chosen with probability $w_t^{(i)}$ and then setting all weights to $w_t^{(i)} = 1/N$ [Gordon et al., 1993], whereas more efficient resampling strategies include stratified sam-

pling and residual sampling [Douc and Cappe, 2005]. A generic particle filtering algorithm is presented in Algorithm 3.

Particle filters directly implement the Bayesian recursions and do not assume any particular form for the system functions or noise statistics, this means that PF do not require any model approximation such as the extended Kalman filter or grid-based methods. This benefit comes with an important requirement: special attention needs to be paid to the design of the particle filter, that is, to the proposal density and the resampling procedure.

Chapter 7

Unsupervised State-Space Modelling

Analyse data just so far as to obtain simplicity and no further.

-Henri Poincaré.
(La Science et L'Hypothèse, 1902.)

The flexibility of particle filters (PF) allows for approximating the posterior density $p(X_t|y_{1:t})$ with no rigid constraints on the functions f_t, h_t (such as linearity) or the distributions of the noise processes V_t, W_t (such as Gaussianity). This makes it possible to design SSMs using nonlinear functions and non-Gaussian noise, for which a posterior cannot be necessarily found in closed-form. By relying on the nonlinear filtering capability of PF methods, the model design can therefore *freely* focus on empirical evidence and prior knowledge (if any) of the nature of the signals, rather than assuming a simpler model to fulfil the stringent requirements of the filter. Specifically, the aim is to find a model that is general enough to account for all possible observations of the process Y_t, while at the same time *not being too uninformative*, as this will result in meaningless estimates of the hidden process X_t. Practical model design involves choosing a state-transition function f_t and an observation function h_t, the latter is usually known and given by the data-collection framework, whereas the former reflects the dynamical properties of the hidden process and is unknown when theoretical understanding of the process is scarce. This so-called *design of the prior* is a fundamental component of filtering applications, yet it remains an open challenge.

The design of the prior can be cast into a function approximation problem, and therefore admits the use of (data-driven) machine learning algorithms—kernel methods are particularly suited for this estimation task. The kernel adaptive filters in Chapter 3 allow for flexible nonlinear estimation, however, they perform supervised function approximation to provide point estimates, this is inadequate when the process of interest is latent within a probabilistic setting. We then address the system identification problem by first parametrising the SSM state-transition function using kernels, and then finding

the kernel mixing parameters using Monte Carlo methods in both an offline and online fashion. The choice of the support vectors and the prior density of the mixing weights is also discussed based on the empirical knowledge of the observed process.

7.1 Kernel State Space Models

We consider general SSMs of the form (6.1) and address the task of system identification for the case when the state-transition function f is unknown and the *sensor* function h is known. The analysis of this class of systems is motivated by the fact that, in real-world applications, the function h is usually available and given by the sensor chosen for a specific experiment; the sensor function is often even linear, such as the case when the observation is one of the states. We also assume that a *plausible* region exists where the state can be found, this is supported by combining the observations with the knowledge of the sensor function (likelihood) and known physical constrains on the latent state. This assumption allows for a straightforward design of the dictionary.

Within this setting, our aim is to find a meaningful estimate of the state-transition function f in eq. (6.1). As this is an infinite-dimensional optimisation problem, since $f \in C^0(\mathbb{R})$, we propose to approximate f by an element $f_{\mathbf{a}}$ in the reproducing kernel Hilbert space $\mathcal{H}$ induced by the Gaussian kernel

$$\mathcal{H} = \left\{ f_{\mathbf{a}} : X \to \mathbb{R}, \text{s.t. } f_{\mathbf{a}}(x) = \sum_{i=1}^{N} a_i K(x, s^i), a_i \in \mathbb{R} \right\} \tag{7.1}$$

where $\mathcal{D} = \{s^i\}_{i=1:N}$ is the dictionary, $\mathbf{a} = [a_1, \ldots, a_N]^T \in \mathbb{R}^N$ are the mixing parameters, and $K(x, y) = \exp\left(-\sigma^{-2}\|x - y\|^2\right)$, $x, y \in X, \sigma > 0$ is the Gaussian kernel. Observe that $\mathcal{H} = \mathcal{H}_{\mathcal{D},\sigma}$ depends on the dictionary $\mathcal{D}$ and the kernel width σ.

The assumption that $\mathcal{H}$ contains functions that approximate f arbitrarily well stems from the fact that the space spanned by of possible features, $\mathcal{H}_{X,\sigma}$, is dense in the space of continuous functions $C^0(\mathbb{R})$, meaning that if $f \in C^0(\mathbb{R})$, then there exists a sequence of functions $\{f_i \in \mathcal{H}_{X,\sigma}\}_{i\in\mathbb{N}}$ that converges to f. Therefore, by controlling the number of support vectors, that is, by choosing a dictionary $\mathcal{D} \subsetneq X$, we obtain an estimate which is reasonably close to the function f but at the same time allows for computationally-feasible implementations (see also [Steinwart et al., 2006, Steinwart, 2001, Minh, 2010]).

By parametrising the transition function f as $f_{\mathbf{a}} = \sum_{i=1}^{N} a_i K(s^i, \cdot)$, the approxi-

mated SSM takes the form

$$X_{t+1} = \sum_{i=1}^{N} a_i K(s^i, X_t) + W_t \tag{7.2}$$

$$Y_t = h(X_t) + V_t. \tag{7.3}$$

We refer to this class of models as kernel state space models (KSSM). Within this formulation, the system identification problem boils down to finding a set of fixed support vectors that is representative of the region where the state currently lies, and their corresponding mixing parameters $\mathbf{a}$. We now propose a procedure to find the posterior density of the mixing parameters conditional to the observed process in the offline case, and proceed to discuss how to choose the support vectors based on standard sparsification criteria employed by kernel adaptive filters.

7.1.1 Offline Learning of the State-Transition Function

The estimate $f_{\mathbf{a}}$ can be regarded as a mapping from $\mathbb{R}^N$ to $\mathcal{H}$ according to $\mathbf{a} \mapsto f_{\mathbf{a}} = \sum_{i=1}^{N} a_i K(s^i, \cdot)$. As a consequence, by considering $\mathbf{a}$ as a random vector, $f_{\mathbf{a}}$ becomes a random function, the posterior density of which can be found using Bayesian inference.

The posterior density $p(f_{\mathbf{a}}|y_{1:t})$ is then uniquely determined by the posterior density of the weights $p(\mathbf{a}|y_{1:t})$; this not only allows to find the transition-function posterior but also the expected value $f_* = \mathbb{E}[f_{\mathbf{a}}|y_{1:t}]$ given by

$$f_* = \int_{\mathbb{R}^N} f_{\mathbf{a}} p(\mathbf{a}|y_{1:t}) \mathrm{d}\mathbf{a} = \sum_{i=1}^{N} \mathbb{E}[a_i|y_{1:t}] K(s^i, \cdot) \tag{7.4}$$

meaning that f_* can be found by only computing $\mathbb{E}[\mathbf{a}|y_{1:t}]$, since $f_* = f_{\mathbb{E}[\mathbf{a}|y_{1:t}]}$.

We now investigate how to sample from the weights posterior $p(\mathbf{a}|y_{1:t})$. By virtue of the Bayes theorem, this density can be expressed as

$$p(\mathbf{a}|y_{1:t}) = p(y_{1:t}|\mathbf{a}) \frac{p(\mathbf{a})}{p(y_{1:t})} \tag{7.5}$$

and evaluated up to the normalising constant $p(y_{1:t})$; thus, we propose to sample from the posterior of $\mathbf{a}$ using MCMC.

The evaluation of (7.5) requires to assume a prior $p(\mathbf{a})$, which can be e.g. Gaussian or uniform, and to compute the likelihood $p(y_{1:t}|\mathbf{a}) = \prod_{k=0}^{t-1} p(y_{k+1}|y_{1:k}, \mathbf{a})$, where

$$p(y_{k+1}|y_{1:k}, \mathbf{a}) = \iint_{X^2} p(y_{k+1}|x_{k+1}, \mathbf{a}) p(x_{k+1}|x_k, \mathbf{a}) \times p(x_k|y_{1:k}, \mathbf{a}) \mathrm{d}x_{k:k+1}. \tag{7.6}$$

Recall that the observation function h is independent of the parameters $\mathbf{a}$, thus, $p(y_{k+1}|x_{k+1}, \mathbf{a}) = p(y_{k+1}|x_{k+1})$. Therefore, upon rearranging eq. (7.6) into expectations we obtain

$$p(y_{k+1}|y_{1:k}, \mathbf{a}) = \underbrace{\int \left(\underbrace{\int p(y_{k+1}|x_{k+1}) p(x_{k+1}|x_k, \mathbf{a}) \mathrm{d}x_{k+1}}_{\text{Expectation w.r.t } p(x_{k+1}|x_k, \mathbf{a})} \right) p(x_k|y_{1:k}, \mathbf{a}) \mathrm{d}x_k}_{\text{Expectation w.r.t } p(x_k|y_{1:k}, \mathbf{a})} \tag{7.7}$$
$$= \mathbb{E}_2 \left[\mathbb{E}_1 \left[p(y_{k+1}|x_{k+1}) \right] \right]$$

where $\mathbb{E}_1$ and $\mathbb{E}_2$ denote respectively the expectations with respect to $p(x_{k+1}|x_k, \mathbf{a})$ and $p\left(x_k|y_{1:k}, \mathbf{a}\right)$. Finally, the expression in (7.7) can be evaluated by:

i) Approximate the filtering density $p(x_k|y_{1:k}, \mathbf{a})$ using particle filters by $\hat{p}(x_k|y_{1:k}, \mathbf{a}) = \sum_{i=1}^{N_p} w_k^{(i)} \delta_{x_k^{(i)}}(x_k)$,

ii) For each particle $x_k^{(i)}$, compute the Monte Carlo estimate $\widehat{\mathbb{E}}_{1,i} \approx \mathbb{E}_1 \left[p(y_{k+1}|x_{k+1}) \right]$ by drawing particles $x_{k+1}^{(j)} \sim p(x_{k+1}|x_k^{(i)})$,

iii) Compute the sample estimate of $\mathbb{E}_2$ by $\widehat{\mathbb{E}}_2 = \sum_{i=1}^{N_p} w_i \widehat{\mathbb{E}}_{1,i}$.

The pseudocode for the proposed method using Metropolis-Hastings MCMC is given in Algorithm 4.

As a result, the posterior of the mixing parameters can be approximated by the empirical density

$$\hat{p}\left(\mathbf{a}|y_{0:t}\right) = \frac{1}{N_p} \sum_{i=1}^{N_p} \delta_{\mathbf{a}^{(i)}}(\mathbf{a}) \tag{7.8}$$

where the samples $\{\mathbf{a}^{(i)}\}_{i=1:N_p}$ are obtained through MCMC sampling from the density (7.5) as described above, and N_p is the number of particles. Consequently, the posterior mean of the KSSM transition function is given by

$$f_* = \sum_{i=1}^{N} \left(\frac{1}{N_p} \sum_{j=1}^{N_p} a_i^{(j)} \right) K(s^i, \cdot). \tag{7.9}$$

7.1.2 Choice of Support Vectors and Kernel Width

The support vectors can be chosen based on the observed process $y_{t \in \mathbb{N}}$ and the sensor function $h(\cdot)$. For each sample y_t, the (known) observation equation of the KSSM in eq. (7.3) allows us to compute a maximum likelihood estimate of x_t (conditional to y_t) by $\widehat{x}_t =$

Algorithm 4 Draw S samples from the posterior density $p(\mathbf{a}|y_{1:t})$ using Metropolis-Hastings

INPUT: Observations $y_{1:t}$, support vectors $\{s^i\}_{i=1:N}$ and kernel width σ.
Set: MCMC move $q(\mathbf{a}|\mathbf{a}^{(i)})$, first sample: $\mathbf{a}^{(1)} \sim q(\mathbf{a}|\mathbf{0})$, sample number $i = 1$.
while $i < S$ **do**
  Propose a candidate move: $\mathbf{a}^{(c)} \sim q(\mathbf{a}|\mathbf{a}^{(i)})$
  for all $k = 1 : t$ **do**
    Approximate $p(x_k|y_{1:k}, \mathbf{a}^{(c)})$ using N_p weighted particles $(x_k^{(i)}, w_k^{(i)})$ using particle filters.
    for all $x_k^{(i)}, i = 1 : N_p$ **do**
      Sample N_p particles $x_{k+1}^{(j)} \sim p(x_{k+1}|x_k^{(i)}, \mathbf{a}^{(c)})$
      Compute $\widehat{\mathbb{E}}_{1,i} = \frac{1}{N_p} \sum_{j=1}^{N_p} p(y_{k+1}|x_{k+1}^{(j)})$
    end for
    Compute $\widehat{\mathbb{E}}_{2,k} = \sum_{i=1}^{N_p} w_k^{(i)} \widehat{\mathbb{E}}_{1,i}$
  end for
  Compute $\widehat{p}(\mathbf{a}^{(c)}|y_{1:t}) = p(\mathbf{a}^{(c)}) \prod_{k=1}^{t} \hat{\mathbb{E}}_2$
  Set $\mathbf{a}^{(i+1)} = \mathbf{a}^{(c)}$ and $i = i + 1$ with probability $A = \min\left\{1, \frac{p(\mathbf{a}^{(c)}|y_{1:t}) q(\mathbf{a}^{(i)}|\mathbf{a}^{(c)})}{p(\mathbf{a}^{(i)}|y_{1:t}) q(\mathbf{a}^{(c)}|\mathbf{a}^{(i)})}\right\}$
end while

argmax $p(X_t|y_t)$. For a sequence of observations $y_{1:T}$, this procedure provides a collection of estimates for the state given by $\widehat{x}_{1:T}$, which, although not reliable as a filtering estimate due to not incorporating the state dynamics (eq. (7.2)), it provides insight about where the state can be found. We then apply standard kernel learning sparsification techniques, such as approximate linear dependence or the coherence criterion, to the sequence $\widehat{x}_{1:T}$ so as to choose the support vectors.

Observe that the support vectors can also be found in a Bayesian fashion together with the mixing parameters, however, this would require using reversible jump MCMC methods, thus increasing the computational complexity of the overall algorithm. On the other hand, the proposed heuristic approach for the choice of support vectors exploits knowledge of the sensor function, is straightforward to implement, and has low computational complexity.

After the dictionary is chosen, the kernel width can be set based on the desired smoothness of the estimate. An empirical approach to find a suitable kernel width is to analyse the distribution of the norm of the differences across the sequence $\widehat{x}_{1:T}$, that is, $\{\|\widehat{x}_i - \widehat{x}_j\|, i, j = 1 : T\}$, and then choose a kernel width according the the spread of these quantities. Furthermore, observe that due to the universal property of the Gaussian kernel, the performance of the kernel regression is not restricted to a particular value of the kernel width [Steinwart et al., 2006].

7.2 Online Update of the KSSM Model

To model time-varying state-transition functions when the observations y_t arrive sequentially, we propose the kernel-based time-varying SSM in the form

$$X_{t+1} = \sum_{i=1}^{N} a_{i,t} K(s^i, X_t) + W_t, \tag{7.10}$$

$$Y_t = h(X_t) + V_t \tag{7.11}$$

where $\mathbf{a}_t = [a_{1,t}, \ldots, a_{N,t}]^T$ is the vector of mixing parameters at time instant t. We next propose two ways of finding the mixing parameters: one based on artificial evolution [Liu and West, 2001] and one based on a sequential extension of the batch method proposed in Section 7.1.1.

7.2.1 Model Design by Artificial Evolution of Parameters

We consider the unknown kernel weights $\mathbf{a}_t$ to be part of the state of the KSSM, so that its posterior distribution $p(\mathbf{a}_t|y_{1:t})$ can be estimated using particle filters. This yields the state space model

$$\begin{aligned} X_{t+1} &= \sum_{j=1}^{N} a_{t,j} K(s^j, X_t,) + V_t, \;\; V_t \sim \mathcal{N}\left(0, \Sigma_X^2\right) \\ a_{t+1,j} &= a_{t,j} + \epsilon_t^a, \;\; \epsilon_t^a \sim \mathcal{N}\left(0, \sigma_a^2\right) \\ Y_t &= X_t + W_t, \;\; W_t \sim \mathcal{N}\left(0, \Sigma_Y^2\right) \end{aligned} \tag{7.12}$$

where $[X_t; \mathbf{a}_t]$ is the state of the KSSM, and $\Sigma_Y^2, \Sigma_X^2, \sigma_\omega^2$ are the covariances of the corresponding processes.

This approach for system identification, referred to as *self-organising state space models* [Kitagawa and Sato, 2001], assumes artificial evolution dynamics and is well known to provide reasonable estimates of model parameters when using particle filters in real-world applications [Liu and West, 2001].

Remark 14. *The model (7.12) offers two distinguishing advantages: (i) by allowing the parameters $\boldsymbol{\omega}_t$ to gradually change, the resulting time-varying model is suitable for nonstationary environments; (ii) by virtue of the parameters being part of the (extended) system state, their posterior density can be found using particle filters.*

To estimate the joint posterior of X_t and $\mathbf{a}_t$ in (7.12) conditional to the observed path $Y_{0:t} = y_{0:t}$, we can then use sequential importance resampling (SIR) particle filters [Gordon et al., 1993]. In this way, $p(X_t, \mathbf{a}_t|y_{0:t})$ is approximated by a weighted average of particles $\left(\mathbf{x}_t^{(j)}, \mathbf{a}_t^{(j)}\right)$, for which the weights $w_t^{(j)}$ are recursively calculated based on

the likelihood of each particle [Doucet et al., 2001]. Additionally, a Gaussian proposal density π that is sufficiently wide is considered so as to fully explore the state space and to avoid particle degeneracy,

Remark 15. *Notice that within the proposed estimator, the usual practice of choosing the proposal density π equal to the model prior is highly inappropriate, since the KSSM in (7.12) might not be able to resemble the desired model before a sufficient number of samples have been processed.*

7.2.2 Recursive Sampling From $p(\mathbf{a}_t|y_{1:t+1})$ using MCMC and SMC

The mixing weights can also be estimated through their posterior density conditional to the observed process. The following remark states the approach to define this posterior.

Remark 16. *Observe that within the proposed KSSM, posterior inference on the mixing parameters should be addressed by targeting the density $p(\mathbf{a}_t|y_{1:t+1})$, since $\mathbf{a}_t$ will only be available through x_{t+1}, and consequently, through y_{t+1} and not y_t—see eqs.* (7.10)-(7.11).

To find a recursive expression for the posterior $p(\mathbf{a}_t|y_{1:t+1})$, we assume (i) a prior density for the transition of the weights $p(\mathbf{a}_t|\mathbf{a}_{t-1})$, and (ii) a particle approximation for $p(\mathbf{a}_{t-1}|y_{1:t})$—such as eq. (7.8). This leads to the following recursion

$$\begin{aligned} p(\mathbf{a}_t|y_{1:t+1}) &= \frac{p(y_{t+1}|y_{1:t},\mathbf{a}_t)}{p(y_{t+1}|y_{1:t})}p(\mathbf{a}_t|y_{1:t}) \\ &= \frac{p(y_{t+1}|y_{1:t},\mathbf{a}_t)}{p(y_{t+1}|y_{1:t})}\int p(\mathbf{a}_t|\mathbf{a}_{t-1},y_{1:t})p(\mathbf{a}_{t-1}|y_{1:t})\mathrm{d}\mathbf{a}_{t-1} \end{aligned} \tag{7.13}$$

where, based on Remark 16, we can write $p(\mathbf{a}_t|\mathbf{a}_{t-1},y_{1:t}) = p(\mathbf{a}_t|\mathbf{a}_{t-1})$ since, conditional to $\mathbf{a}_{t-1}$, the sequence $y_{1:t}$ does not provide *posterior* evidence for $\mathbf{a}_t$.

As a consequence, eq. (7.13) gives a recursive expression for the posterior in integral form, thus, by using the particle approximation $p(\mathbf{a}_{t-1}|y_{1:t}) \approx \sum_{j=1}^{N_p} w_{t-1}^{(j)}\delta_{\mathbf{a}_{t-1}^{(j)}}(\mathbf{a}_{t-1})$ we can write the estimate

$$\hat{p}(\mathbf{a}_t|y_{1:t+1}) = \frac{p(y_{t+1}|y_{1:t},\mathbf{a}_t)}{p(y_{t+1}|y_{1:t})}\sum_{j=1}^{N_p} w_{t-1}^{(j)}p(\mathbf{a}_t|\mathbf{a}_{t-1}^{(j)}). \tag{7.14}$$

With the weighted samples $\left(\mathbf{a}_{t-1}^{(j)}, w_{t-1}^{(j)}\right) \sim p(\mathbf{a}_{t-1}|y_{1:t})$ available from the previous step, the evaluation of (7.14) is straightforward up to a normalising constant, since $p(y_{t+1}|y_{1:t},\mathbf{a}_t)$ can be computed using eq. (7.6) and the prior $p(\mathbf{a}_t|\mathbf{a}_{t-1})$ is assumed to be known. Therefore, we can also sample from (7.14) using MCMC. The proposed recursive method is presented in pseudocode form in Algorithm 5.

7.2.3 Online Sparsification and Choice of the Prior $p(\mathbf{a}_t|\mathbf{a}_{t-1})$

To avoid the large computational cost associated to Bayesian inference of model order, we next consider deterministic kernel adaptive filtering sparsification. Accordingly, for every observation y_t, we compute the MLE of the state given by $\widehat{x}_t$ to assess whether the state is in a region covered by the current dictionary; we then include $\widehat{x}_t$ as a support vector based on either the ALD or coherence criterion. Notice that this represents an adaptive version of the sparsification procedure explained in Section 7.1.2.

With this data-driven choice of support vectors, we can now design the prior transition $p(\mathbf{a}_t|\mathbf{a}_{t-1})$ based on whether a new sample $s = \widehat{x}_t$ is added to the dictionary at time t, in the following ways.

Sample $s = \widehat{x}_t$ is not added

When the dictionary is not modified, we assume $p(\mathbf{a}_t|\mathbf{a}_{t-1}) = \mathcal{N}(\mathbf{a}_t; \mathbf{a}_{t-1}, \mathbf{\Sigma})$ where $\mathbf{\Sigma}$ is a square-exponential covariance matrix that ensures smooth transitions moves; we can then write

$$\hat{p}(\mathbf{a}_t|y_{1:t+1}) \propto p(y_{t+1}|y_{1:t}, \mathbf{a}_t) \sum_{j=1}^{N_p} w_{t-1}^{(j)} \mathcal{N}(\mathbf{a}_t; \mathbf{a}_{t-1}^{(j)}, \mathbf{\Sigma}) \tag{7.15}$$

where $p(y_{t+1}|y_{1:t}, \mathbf{a}_t)$ is given by eq. (7.6).

The MCMC candidate can be then chosen by randomly seeting a particle $\mathbf{a}_{t-1}^{(r)}, r \in [1, N_p]$, and adding a random perturbation proportional to the likelihood of X_t with respect to the observation y_t, that is

$$\mathbf{a}_t^{(c)} \sim \mathcal{N}(\mathbf{a}_{t-1}^{(r)}, L_t) \tag{7.16}$$

where

$$L_t = \text{diag}\left([p(X_t = s^1|y_t), \ldots, p(X_t = s^m|y_t)]\right). \tag{7.17}$$

By choosing the MCMC moves in this way, the candidate sample still retains past information, since the mean of the transition operator in eq. (7.16) is a sample from the previous posterior, but at the same time explores zones from where the latest state sample may have been generated by using the information from the covariance of the candidate move. In this sense, sampling from eq. (7.15) can be seen as a trade-off between maintaining previous knowledge and learning from the latest observed sample.

Sample $s = \widehat{x}_t$ is added

When the support vector s^{N+1} is added to the dictionary, the dimension of the mixing weights increases to $(N + 1)$. For the choice of the prior $p(\mathbf{a}_t|\mathbf{a}_{t-1})$, we consider the first

Algorithm 5 Draw S samples from the sequence of posteriors $p(\mathbf{a}_t|y_{1:t+1}), t = 1, 2, \ldots$

INPUT: Kernel width σ and first observation y_1.
Initialise: Dictionary $\{\widehat{x}_1\}$, kernel width σ, and particles $\{x_1^{(i)}\} \sim p(x_1|y_1)$ and $\{\mathbf{a}_1^{(i)}\} \sim p(\mathbf{a}_1)$.
for all $y_t, t = 1, 2, \ldots$ **do**
  Calculate $\widehat{x}_t = \text{argmax}\, p(X_t|y_t)$
  if $\widehat{x}_t$ deviates form dictionary **then**
    Set $N = N + 1$ and add $s^N = \widehat{x}_t$ into the dictionary
    Draw samples from eq. (7.15) using MCMC and the proposal in (7.16)
  else
    Draw samples from eq. (7.18) using MCMC and the proposal in (7.19)
  end if
end for

N components of the mixing weights to evolve according to the fixed dictionary case, and the new coefficient, $a_{N+1,t}$, to be independent from the previous coefficients, thus making $p(a_{N+1,t}|\mathbf{a}_{t-1})$ a constant. As a consequence, we have

$$\hat{p}(\mathbf{a}_t|y_{1:t+1}) \propto p(y_{t+1}|y_{1:t}, \mathbf{a}_t) \sum_{j=1}^{N_p} \mathcal{N}\left(\begin{bmatrix} a_{1,t} \\ \vdots \\ a_{N,t} \end{bmatrix}; \mathbf{a}_{t-1}^{(j)}, \Sigma \right). \tag{7.18}$$

We then consider the MCMC move

$$\begin{bmatrix} a_{1,t}^{(c)} \\ \vdots \\ a_{N,t}^{(c)} \end{bmatrix} \sim \mathcal{N}(\mathbf{a}_{t-1}^{(r)}, V_t) \text{ and } a_{N+1,t}^{(c)} \sim \mathcal{N}(\mu, \sigma) \tag{7.19}$$

where the first N parameters are sampled according to the fixed-dictionary case, and the new parameter is sampled independently from the rest of the parameters and is given by a normal density of mean μ and variance σ^2.

Following the same concept as in the fixed-dictionary case, this MCMC proposal move aims to retain previous knowledge by relying on past estimates for the first N entries of the weights, while at the same time exploring values for the new parameter.

7.2.4 Multivariate KSSM for Autoregressive Modelling

Recall that the stochastic process satisfying the τ-order difference equation $X_{t+1} = f_t(X_t, \ldots, X_{t-\tau+1}) + W_t$ can be expressed as a first-order multivariate process of the form

$$\bar{X}_{t+1} = F_t(\bar{X}_t) + G_t \tag{7.20}$$

where $\bar{X}_t = [X_t, \ldots, X_{t-\tau+1}]^T$,

$$F_t(\bar{X}_t) = \begin{bmatrix} f_t(X_t, \ldots, X_{t-\tau+1}) \\ X_t \\ \vdots \\ X_{t-\tau+2} \end{bmatrix}, \text{ and } G_t = \begin{bmatrix} W_t \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{7.21}$$

The process $\bar{X}_t$ can then be considered as the hidden state of an SSM, thus yielding a SSM formulation of higher-order autoregressive process with noisy observations. Furthermore, if the state-transition function f_t is modelled using kernels, we obtain a multivariate version of the proposed KSSM suited for autoregressive time series given by

$$\begin{bmatrix} X_{t+1} \\ X_t \\ \vdots \\ X_{t-\tau+2} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} a_{i,t} K(s^i, \bar{X}_t) \\ X_t \\ \vdots \\ X_{t-\tau+2} \end{bmatrix} + \begin{bmatrix} W_t \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{7.22}$$

$$Y_t = h(\bar{X}_t) + V_t \tag{7.23}$$

where Y_t is the observed process, h_t is the observation function and V_t observation noise.

Remark 17. *The proposed KSSM method can be used for the design of nonlinear higher-order autoregressive models. Observe that the order of the process (τ) and the number of support vectors (N) are two independent design parameters and they can be chosen, respectively, according to the dynamics of the signal and the desired accuracy. As a consequence, the number of parameters to find is given by the number of support vectors and not by the order of the process.*

7.3 Examples

7.3.1 Offline Estimation of a Nonlinear State-Transition Function

The following example provides an experimental insight into the proposed algorithm. Consider the system

$$\begin{aligned} X_t &= 10\text{sinc}\left(\frac{X_{t-1}}{7}\right) + W_t \\ Y_t &= X_t + V_t \end{aligned} \tag{7.24}$$

where the state and observation noise variances are $\sigma_x^2 = 4$ and $\sigma_y^2 = 4$, and $x_1 \sim \mathcal{N}(0, 10)$. This SSM state-transition function was chosen because its state is bounded (as the norm of $f(x) = 10\text{sinc}(x/7)$ vanishes for $x \to \infty$) and the system does not converge, as the

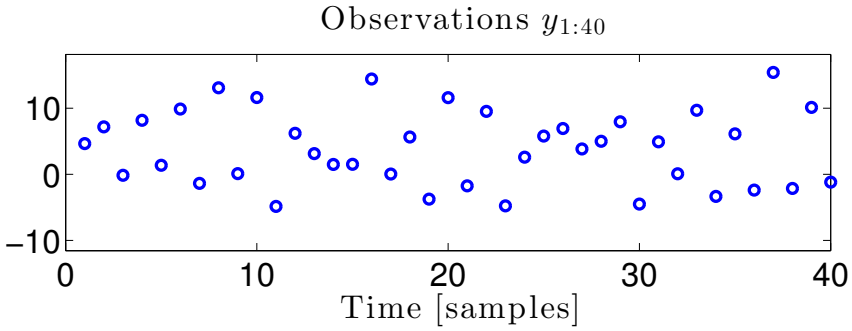

Figure 7.1: Observed process for the system in eq. (7.24).

sequence $X_{t+1} = f(X_t)$ has two accumulation points[1] about -1.7 and 8.7. We then parametrised the system (7.24) using a KSSM and estimated the mixing parameters of the kernel-based transition function $f_{\mathbf{a}} = \sum_{i=1}^{N} a_i K(\cdot, s^i)$ as explained in Section 7.1.1.

We considered 40 observations of the process Y_t denoted by $y_{1:40}$, shown in Fig. 7.1, and set the support vectors according to Section 7.1.2. Furthermore, the kernel with was $\sigma^2 = 10$, and we assumed a uniform prior $p(\mathbf{a})$, and a proposal density for the MCMC moves given by $p(\mathbf{a}^{(c)}|\mathbf{a}^{(i)}) = \mathcal{N}\left(\mathbf{a}^{(i)}, L\right)$, with the square-exponential[2] covariance matrix

$$L\left(a_i, a_j\right) = 0.2^2 \exp\left(-0.2\|a_i - a_j\|^2\right). \tag{7.25}$$

This candidate proposal allows the MCMC moves to be smooth, the kernel estimate was then computed according to Algorithm 1.

Fig. 7.2 shows the true SSM transition function, the hidden state samples corresponding to the considered time period, and the posterior mean of the KSSM transition function (f_*) with its one-standard-deviation confidence interval. Observe that the mean estimate f_* matches the true underlying transition function for the regions of the state space where the hidden state reside. The posterior variance of the kernel estimate is also consistent with the spread of the unobserved samples, as for the regions where the state samples were more disperse, the estimate of the posterior variance was larger.

Recall that MCMC sampling generates a sequence of samples that move to areas of high probability, and then explore such areas. Fig. 7.3 shows the value of the posterior $p(\mathbf{a}|y_{1:40})$ for both the samples drawn using MCMC and the supervised least squares solution using the hidden samples (i.e. kernel least squares). Observe that the Markov chain converges in about 100 samples to a zone of a probability similar to that of the supervised solution to then continue to draw samples of similar probability. Additionally,

[1] Recall that $\bar{x}$ is an accumulation point of the sequence $\{X_t\}_{t\in\mathbb{N}}$ if and only if any neighbourhood of $\bar{x}$ contains infinite elements of $\{X_t\}_{t\in\mathbb{N}}$.

[2] We refer to square exponential covariance functions and reserve the term Gaussian for reproducing kernels only.

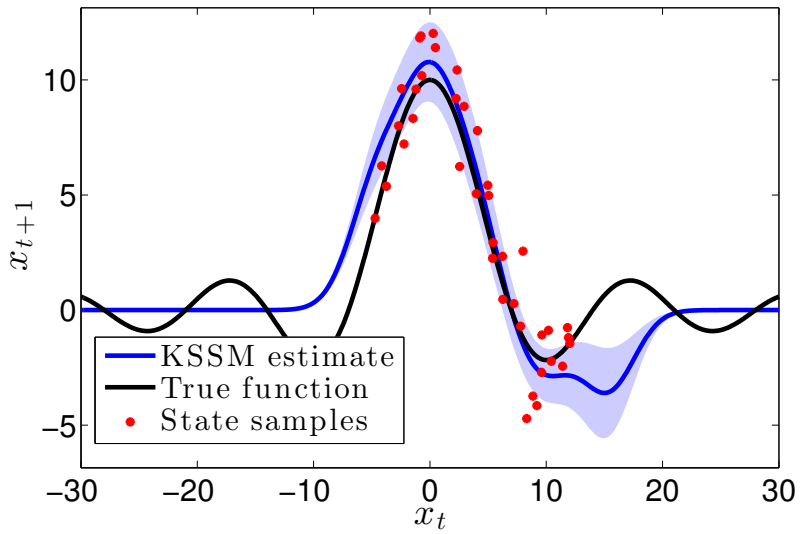

Figure 7.2: Original state-transition function, state samples and kernel-based approximation.

as an uniform prior $p(\mathbf{a})$ was assumed, the posterior is equal to the likelihood up to a normalising constant—see eq. (7.5). Fig. 7.3 illustrates that the supervised solution is different from the maximum likelihood estimate, as some of the samples drawn using the proposed method have a higher likelihood. This discrepancy arises from the fact that the supervised solution uses the hidden samples whereas the MCMC sampling does not.

Remark 18. *This example highlights the ability of the proposed method to learn the state-transition functions in a localised manner for regions containing state samples. Such an approach performs unsupervised learning to provide a sequence of estimates (i.e. samples of the posterior of $f_{\mathbf{a}}$) that are of a probability similar to that of the supervised solution (that uses the hidden state samples).*

For filtering applications, the estimate of the SSM needs to be updated so that it is representative of the region of operation. We now introduce an adaptive version of the proposed algorithm that incorporates new observations to learn the state-transition function in a recursive fashion.

7.3.2 Prediction of a Nonlinear Prediction: KSSM and Artificial Evolution

In the first set of simulations, the following benchmark nonlinear system [Doucet et al., 2001] was considered

$$\begin{aligned} X_{t+1} &= \frac{X_t}{2} + \frac{25X_t}{1+X_t^2} + 8\cos(0.8t) + V_t \\ Y_t &= X_t + W_t \end{aligned}$$

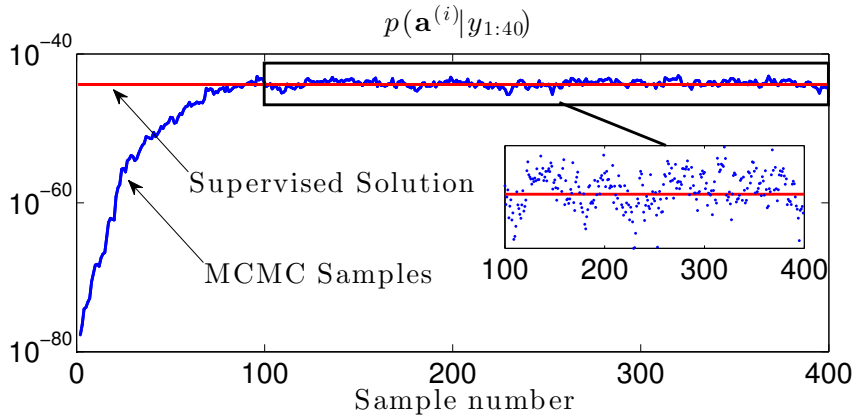

Figure 7.3: Value of the posterior $p(\mathbf{a}|y_{1:40})$ for the supervised solution and the samples generated by the proposed method.

where $V_t \sim \mathcal{N}(0,1)$ and $W_t \sim \mathcal{N}(0,9)$.

The goal was to validate the KSSM paradigm in the recursive prediction of X_{t+1} from the noisy observations $Y_{0:t} = y_{0:t}$ and compare it against the KLMS and KRLS. All three kernel algorithms used a kernel width $A = 0.8$ and ALD threshold $\delta = 0.1$. The algorithm parameters were $\mu = 0.6$ for the KNLMS and the variances $\Sigma_X^2 = 20, \sigma_\omega^2 = .01, \Sigma_Y^2 = 9$ for the KSSM with $N_p = 400$ particles.

Fig. 7.4 shows the hidden process X_t, the KSSM prediction, and the confidence interval corresponding to two standard deviations centred about the prediction. Observe that the estimated two-standard-deviation confidence interval progressively included the original signal as more information became available, highlighting the statistical prediction ability of the proposed KSSM as opposed to standard point predictions. The average prediction mean squared error (MSE) of the kernel algorithms considered was evaluated over 50 independent realisations and is given in Table 7.1; the KSSM outperformed the KNLMS and KRLS. Observe that the better performance of KNLMS over KRLS indicates the nonstationarity of the system.

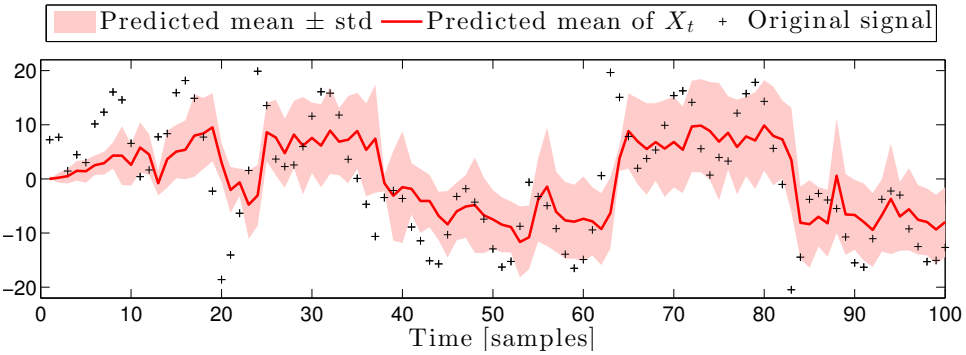

Figure 7.4: Prediction using the proposed KSSM.

Table 7.1: Prediction performance of kernel algorithms

	KSSM	KNLMS	KRLS
MSE	**8.54**	9.08	10.09

7.3.3 Identification of a Time-Varying State-Transition Function: KSSM and MCMC

We now test the proposed algorithm for online learning of a nonlinear state-transition function. Consider the time-varying SSM with an affine observation function given by

$$\begin{aligned} X_{t+1} &= f_t(X_t) + W_t \\ Y_t &= 5 + X_t/2 + V_t \end{aligned} \tag{7.26}$$

where the variances of the state and observation noises were set to $\sigma_x^2 = 1$ and $\sigma_y^2 = 0.5$, $x_1 \sim \mathcal{N}(0, 1)$, and

$$f_t(x) = \begin{cases} \frac{x}{2} + \frac{25x}{1+x^2} & t < 30 \\ \frac{60-t}{30}\left(\frac{x}{2} + \frac{25x}{1+x^2}\right) + \frac{t-30}{30} 10\text{sinc}\left(\frac{x}{7}\right) & 30 \leq t \leq 60 \\ 10\text{sinc}\left(\frac{x}{7}\right) & t > 60 \end{cases} \tag{7.27}$$

This time-varying function is motivated by real-world applications where systems switch between different operation conditions. In this example, the system is time-invariant and stable for $t \in [1, 30]$, time-varying for $t \in [31, 60]$, and time-invariant with two accumulation points for $t \in [61, 90]$ (as in Example 1). This case is typical in anomaly detection scenarios, where an early identification of the data relationship is crucial. The evolution of the state-transition function is shown in Fig. 7.5.

We approximated the function f_t in a parametric manner by $f_{\mathbf{a}} = \sum_{i=1}^{N} a_{i,t} K(\cdot, s^i)$, where the mixing weights were computed in a recursive fashion according to Section 7.2.2 and Algorithm 2. The procedures for choosing the support vectors and the MCMC candidates were those given in Section 7.2.3. Furthermore, for fast computation.

Fig. 7.6 shows the learnt state-transition function at two different time instants, $t = 30$ and $t = 90$. Observe that the function learnt until $t = 30$ is very localised; this is due to the system state being stable and therefore residing on a limited region of the state space. As t grows larger, the transition function changes according to eq. (7.27), thus becoming time varying for $t \in [30, 60]$. In the last third of the analysed period, $t \in [60, 90]$, the transition function is again time invariant and the state lies on a larger region of the state space. Fig. 7.6 also shows the estimate at $t = 90$, where the KSSM successfully learnt the dependency of the state samples through the observation sequence. Observe

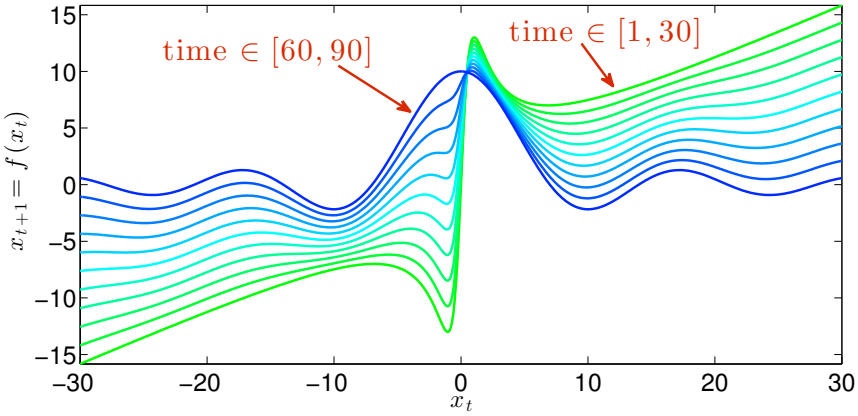

Figure 7.5: The state-transition function of the system in (7.26) is time-varying between $t = 30$ and $t = 60$, and constant for $t < 30$ (system stable) and for $t > 60$ (system unstable).

that the estimate at time $t = 90$ resembles the data samples corresponded to the region and successfully updated the value of the function learnt with previous measurements (i.e. those for $t < 30$); furthermore, this estimate also shows the ability to retain useful information by not relying exclusively on new samples only, this is exemplified by the smooth shape and wide support of the estimate.

Similarly to the offline learning case in the previous section, the recursive estimate provided by the proposed KSSM is localised, as the Bayesian learning paradigm only allows to learn the transition function for the regions *visited* by the hidden state—see eq. (7.7). In this sense, attempting to find a full-support estimate of the true transition function in fig. 7.5 is unrealistic, therefore, the proposed method aims to find an estimate of the transition function that is consistent with the observed sequence, even if it only learns the state dynamics for a limited region of the state space.

Recall that the proposed algorithm performs joint parameter identification and state estimation, as the samples from the filtering density are needed to sample from the posterior of the mixing parameters—see eq. (7.7). The filtered state signal resulting from the function estimation implementation is is shown in fig. 7.7 (mean and standard deviation) together with the true state.

7.4 Discussion

The proposed KSSM paradigm for unsupervised learning of state-space models is flexible and admits different criteria for finding the mixing parameters, designing PF and MCMC stages, and setting hyperparameters. We now give further insight into the presented approach and discuss possible modifications.

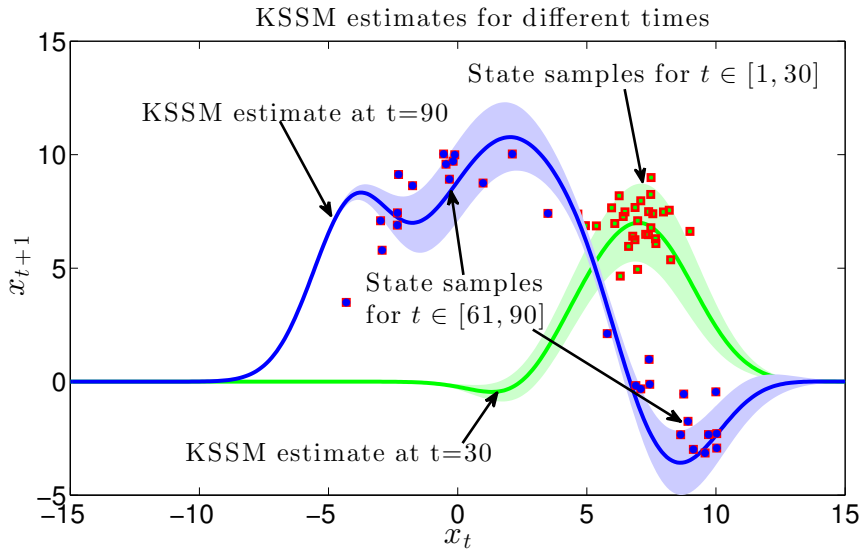

Figure 7.6: KSSM estimate (mean and standard deviation) of the time-varying state-transition function in (7.26) for $t = 30$ and $t = 90$. The true state samples are plotted with red borders for the regions $[1, 30]$ (green fill) and $[61, 90]$ (blue fill).

Criterion to Find the Mixing Parameters

As explained in Section 7.1.1, we performed parameter estimation by targeting the posterior density $p(\mathbf{a}|y_{1:t})$. This allows us to impose desired properties of the solution, such as e.g. regularisation, through the prior density $p(\mathbf{a})$, and then used the observations to compute the pdf of the mixing weights. An alternative approach is to find the maximum likelihood of the weights, this is achieved by maximising $p(y_{1:t}|\mathbf{a})$ and then replacing the random vector $\mathbf{a}$ in eq. (7.2) by the maximum likelihood estimate rather than *integrating out* the parameters as in eq. (7.4). This dilemma is common to many problems in Bayesian inference [MacKay, 1999] and either alternative has both advantages and disadvantages. The posterior density approach gives a complete pdf of the estimated transition function at a more expensive computational cost, whereas the maximum likelihood approach is of lower complexity but only gives a point estimate.

The artificial evolution approach also aims to estimate the posterior of the mixing parameters, by using a modified KSSM, where the parameters are time-varying. Although this method has a reduced computational complexity compared to the MCMC method, it must be taken into account that, due to the assumption of artificial evolution, additional (artificial) sources of uncertainty result in noisy estimates.

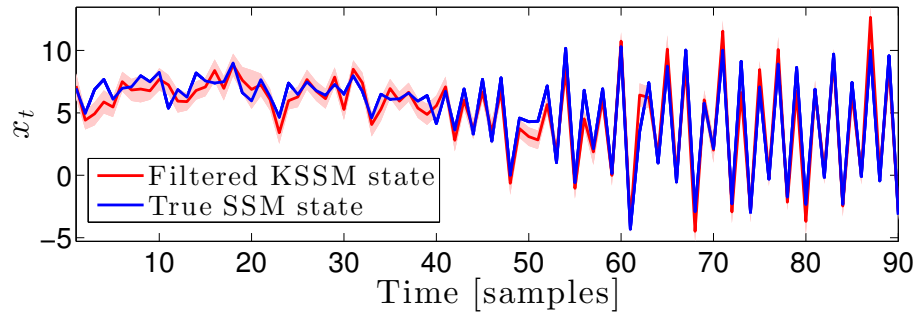

Figure 7.7: Filtered state signal using the KSSM and SIR particle filter. The original state is shown in blue and the posterior mean in red, with a one-standard-deviation confidence interval in light red.

Candidate Move for Markov Chain Monte Carlo

We considered a Metropolis-Hasting MCMC sampling from the posterior $p(\mathbf{a}|y_{1:t})$, where the candidate move plays an important role in both convergence of the chain and the area it explores. To boost the convergence of the chain, we can consider the sequence of maximum likelihood estimates of the state $\widehat{x}_{t\in\mathbb{N}}$ and perform supervised learning of this signal—we can then propose the MCMC moves in this region. This concept was applied in the presented online simulations, as it provided short convergence time and explored a localised region of the parameter space.

Choice of Hyperparameters: Noise Variances, Support Vector and Kernel Width.

The standard in Bayesian inference is to define prior densities for all the unknown parameters and then find their posterior with respect to the observed data, this includes model orders and variances. Although this leads to a full Bayesian model where all the quantities follow from the observations, this approach can be prohibitively expensive in practice, especially when the dimension of the parameters needs to be determined (as it is the case with the choice of support vectors). We then considered concepts from kernel adaptive filtering so as to maintain a reduced computational complexity; in particular, the support vectors were chosen by sparsifying the maximum likelihood estimates of the state $\widehat{x}_{t\in\mathbb{N}}$ and the kernel width and noise variances by trial-and-error. We have shown this approach is well suited for real-world signals and is also in line with the KSSM formulation since (i) the performance of kernel regression is robust to the choice of the kernel width [Steinwart et al., 2006] and (ii) the misadjustment of the choice of the noise variance can be corrected by the (posterior) variance of the mixing weights.

Estimation of the Filtering Density $p(x_t|y_{1:t})$

Within the proposed KSSM, the filtering density is computed not only to perform joint system identification and state estimation, but also because such a density is needed to sample from the posterior $p(\mathbf{a}|\mathbf{y}_{1:t})$ using MCMC—see eq. (7.7). Although the simulations in this paper considered the classic SIR approach [Gordon et al., 1993], any variant of particle filter can be used. In the online case, we can consider uninformative priors for the state when the current estimate is still in its transient [Tobar and Mandic, 2013], or risk-sensitive PF [Thrun et al., 2002], so as to explore critical regions of the state-space, as these provide advantageous in the identification of critical state behaviour [Orchard et al., 2013].

Part III

Real-World Simulation Examples and Concluding Remarks

Chapter 8

Experimental Validation

Nature is our kindest friend and best critic in experimental science if we only allow her intimations to fall unbiased on our minds. Nothing is so good as an experiment which, whilst it sets an error right, gives us (as a reward for our humility in being reproved) an absolute advancement in knowledge.

-Michael Faraday.
(In a letter to John Tyndall, 1851.)

We now illustrate the performance of the proposed estimation algorithms in real-world scenarios and compare their performance to existing kernel adaptive filters. We considered a wide range of scalar and multivariate signals exhibiting nonstationarity, nonlinearity, and cross-coupling, so as to test the ability of kernel-based method in both adaptive and Bayesian filtering. The performance assessment was performed on the basis of quantitative measures such as mean square error (MSE) and prediction gain.

8.1 Bivariate Wind Speed

We considered measurements corresponding to the wind speed in the north-south (V_N) and east-west (V_E) directions, that is

$$V = [V_N, V_E]^T. \tag{8.1}$$

The wind speed readings were taken[1] with a 2D ultrasonic anemometer with the sampling rate of 50 [Hz], over three different wind regimes: *low*, *medium* and *high* dynamics regions.

[1]The data is publicly available in `http://www.commsp.ee.ic.ac.uk/~mandic/research/wind.htm`

8.1.1 One-Step Prediction using Complex-Valued Kernels and KLMS

We first validated the complex-kernel paradigm proposed in Section 4.1.2 in a KLMS setting by comparing the Gaussian independent kernel in eq. (4.5) (iCKLMS) to the complex-valued Gaussian kernel in eq. (4.1) (CKLMS) and the real-valued Gaussian kernel in eq. (2.19) (RKLMS). The performances of these kernel algorithms were assessed in the prediction of the 2D wind speed signal. Accordingly, the bivariate wind measurements corresponding to the low-dynamics region were converted to complex-valued samples; the resulting signal was found to be noncircular and nonstationary [Mandic and Goh, 2009]. The KLMS algorithms were then implemented to perform a one-step-ahed prediction using six consecutive measurements (regressors) from a signal of 1000 samples. The coherence sparsification criterion was used [Richard et al., 2009], and a learning rate of $\mu = 0.05$.

The steady-state mean-square error (MSE) in the form $10\log_{10}$(MSE) is given in Table 8.1 for each kernel. In addition, fig. 8.1 shows the operation of all three KLMS algorithms for the first 350 samples when estimating the north-south component, also illustrating the convergence properties of the KLMS approach.

Table 8.1: $10\log_{10}$(MSE) for wind prediction (last 500 samples).

	RKLMS	iCKLMS	CKLMS
10log(MSE)	-21.1853	-22.5284	-23.5473

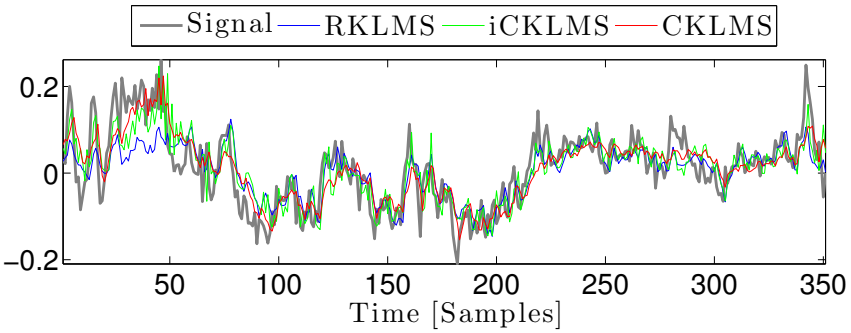

Figure 8.1: Original north-south wind speed component and KLMS estimates.

In terms of the average performance, observe that, due to their enhanced dimensionality, the complex kernels provided more accurate estimates than the real-valued one. Additionally, the complex Gaussian kernel provided more accurate estimates than the independent complex one, this is due to its exponential dependence on the sample deviation that boosts the learning performance.

8.1.2 Long-term Prediction using Multikernel LMS and Presence-Based Sparsification

The physical justification for a multikernel algorithm in this context is illuminated by considering the empirical distribution of the deviations of the input samples (i.e. regressors), evaluated through the set

$$R_{dyn} = \{\|a - b\|_2, a, b \in dyn \subset \mathbb{R}^{20\times 2}\}, \tag{8.2}$$

where the symbol dyn corresponds to a 1000-sample set in the *low*, *medium*, or *high* dynamics region. Figure 8.2 shows the histograms for the sets R_{dyn} and suggests that different kernels should be used for different wind dynamics regions. Furthermore, as the deviation sets corresponding to different dynamics regimes overlap, a kernel designed to fit a particular set may also be useful in the estimation of data from other regions.

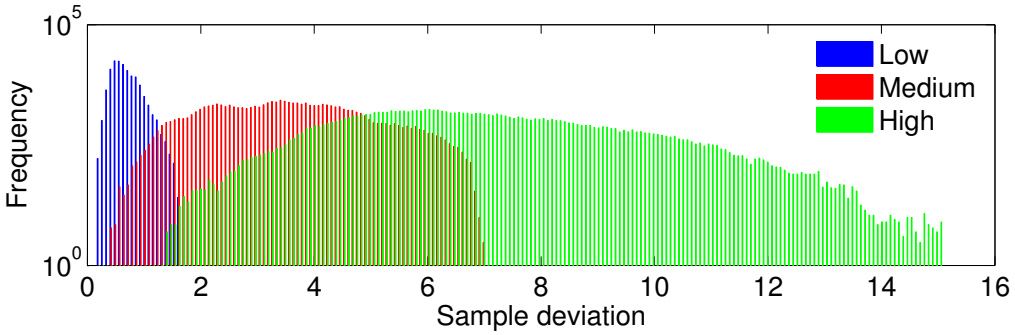

Figure 8.2: Input sample deviation for the low, medium, and high dynamics wind regime.

The multikernel LMS (MKLMS) approach proposed in Section 5.2.2 was then validated and compared to the standard KLMS for the 10-step-ahead prediction of the wind speed, using 20 past samples as regressors.

We fitted three Gaussian kernels (via exhaustive minimisation of the prediction MSE) to the low, medium, and high dynamics training sets; the parameters found for the wind regimes were respectively $\sigma_L^{-2} = 9.5, \sigma_M^{-2} = 0.75, \sigma_H^{-2} = 0.109$. We refer to such kernels (see eq. (2.19)) as the low-fitted (LF), medium-fitted (MF), and high-fitted (HF) kernel. The MKLMS algorithm was then constructed upon these three kernels and compared to three monokernel algorithms using each fitted kernels. To model the non-stationary nature of wind, all four KLMS algorithms were equipped with the presence-based adaptive sparsification criteria with coherence sample-addition stage presented in Section 3.3.1, with parameters $\delta_e = 0.15, \delta_d = 0.1, \delta_p = 0.01$ and $\mu = 0.2, \hat{\mu} = 0.1$.

Figure 8.3 shows performances for a 400-sample implementation of the kernel algorithms for the low dynamics wind. Observe that the MKLMS provided an accurate prediction of the signal even when compared to the specifically designed low-fitted KLMS,

confirming the benefits of a combination of kernels.

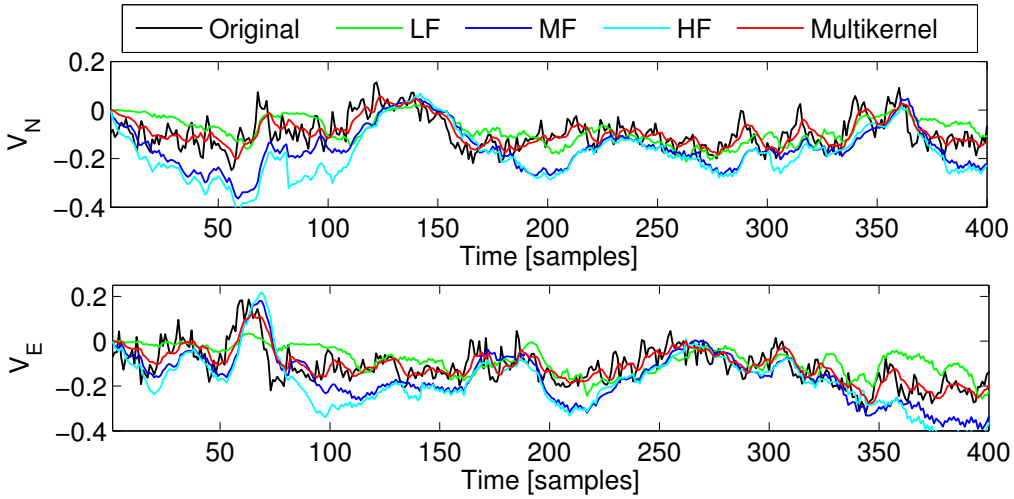

Figure 8.3: Original bivariate wind signal and its KLMS estimates for the low dynamics region.

To further highlight the physical meaning associated with the proposed multi-kernel approach, the kernel algorithms were used to predict mixed-regime wind data whose segments [1, 500], [501, 1000] and [1001, 1500] corresponded respectively to the low, medium and high dynamics regime. Figure 8.4 shows the magnitudes of the weights associated with each kernel within the MKLMS: the low-fitted kernel (blue) was active in the first third of the data duration, the medium-fitted and high-fitted kernel were active for the segment [501, 1000], and the high-fitted kernel in the last third of the data. This illuminates the ability of the MKLMS algorithm to assign more confidence (weights) to the kernel corresponding to the particular dynamics region. Notice also that the high-fitted kernel was also prominent for the second segment (samples [501, 1000]) due to the overlap of the sample deviation for the *medium* and *high* dynamics regions (see Figure 8.2).

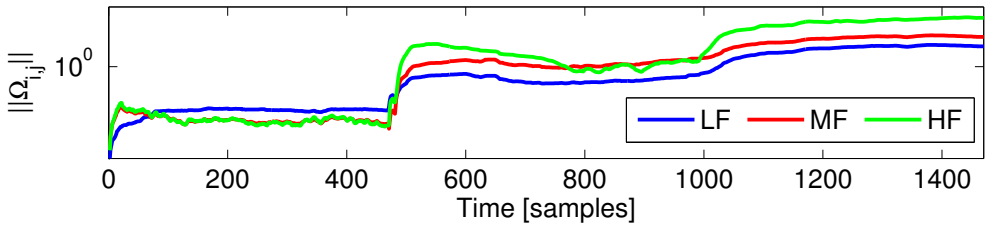

Figure 8.4: Weight magnitudes for the kernels within the MKLMS algorithm, evaluated for mixed regime wind, using the proposed adaptive sparsification criteria.

Figure 8.5 illustrates the efficiency of MKLMS for the prediction of mixed-regime

wind data in terms of the dictionary size for all the kernel algorithms considered. Observe that the proposed adaptive sparsification criterion allows for the dictionary to have a bounded number of samples throughout different dynamics regions, by rejecting samples that are not in use, thus avoiding kernel evaluations that do not improve the estimate. This desired feature of the adaptive sparsification criteria is present in all four implemented kernel algorithms.

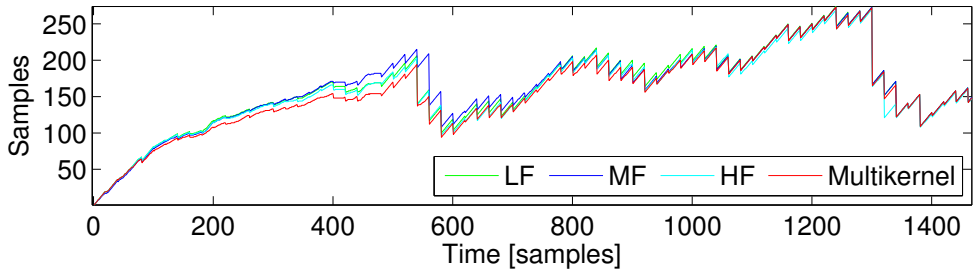

Figure 8.5: Dictionary size for all kernel algorithms for the prediction of mixed-regime wind using the proposed adaptive sparsification criteria.

The MSE performance of the MKLMS on the prediction of wind speed was considered over 10 non-overlapping segments of 470 samples for each regime (low, medium and high dynamics, 30 segments in total). Table 8.2 presents the averaged prediction gain given by

$$R_p = 10\log_{10}\left(\frac{\sum_{t=T_0}^{T}\|V_t\|_2^2}{\sum_{t=T_0}^{T}\left\|V_t-\hat{V}_t\right\|_2^2}\right), \tag{8.3}$$

where $\hat{V}_t$ is the wind estimate, $T_0 = 235$ and $T = 470$, calculated for all four kernel algorithms. Observe that, among the single kernel KLMS algorithms, each algorithm provided the most accurate estimates in the specific region used for its training, whereas the MKLMS performed better than any region-fitted KLMS, for all the three dynamics regions.

Table 8.2: Averaged prediction gains for the low-fitted KLMS (LF), medium-fitted KLMS (MF), high-fitted KLMS (HF), and MKLMS for all three dynamic regions

Algorithm	LF	MF	HF	Multikernel
Low dynamics	**5.53**	2.18	0.06	**9.55**
Medium dynamics	0.07	**3.24**	-1.86	**6.57**
High dynamics	0.01	0.37	**3.05**	**3.1**
Average	1.871	1.932	0.42	**6.41**

The averaged performance of all the algorithms considered was evaluated over all 30 segments (10 segments of each dynamics regime). The bottom row in Table 8.2

shows that the MKLMS had the highest averaged prediction gain, and Figure 8.6 shows that, as desired, the MKLMS exhibited the lowest MSE and smallest dictionary size. This is also reflected in the running times shown in Table 8.3, where the complexity of the MKLMS is bounded by the sum of the KLMS complexities. These results conform with the analysis, highlighting the power of the MKLMS in real world prediction applications in nonstationary environments, where there is no *a priori* information of the instantaneous wind dynamics.

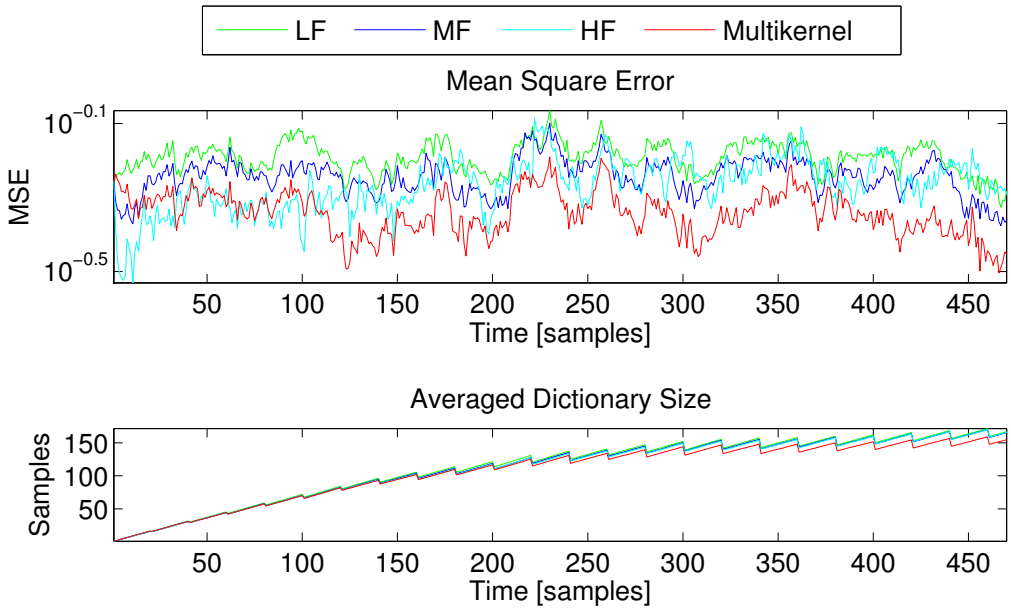

Figure 8.6: Averaged MSE and dictionary size over 30 trials of combined low, medium and high dynamics regions.

Table 8.3: Averaged running time (in seconds) for the low-fitted KLMS (LF), medium-fitted KLMS (MF), high-fitted KLMS (HF), and MKLMS for all three dynamic regions

Algorithm	LF	MF	HF	Multikernel
Low dynamics	0.734	0.737	0.734	**1.28**
Medium dynamics	1.11	1.079	1.09	**2.164**
High dynamics	0.98	0.95	0.941	**1.857**
Average	0.941	0.922	0.921	**1.767**

8.2 Bodysensor Signals

We implemented kernel ridge regression algorithms to validate both quaternion kernel and multikernel algorithms in the one-step-ahead prediction of the trajectory of limbs in Tai Chi sequences.

8.2.1 Data Acquisition and Preprocessing

Four accelerometers (placed at wrists and ankles) recorded the three Euler angles (Fig. 8.7), giving a total of 12 signals $\{\theta_s\}_{s=1,..,12}$ taking values in the range $[-\pi, \pi]$. The recorded signals were discontinuous in $\{-\pi, \pi\}$ and thus unsuitable for the application of continuous kernels, hence the angles data were conditioned through the mapping $\theta_s \mapsto (\sin\theta_s, \cos\theta_s)$. These new features also made it possible for the data to be resampled if needed.

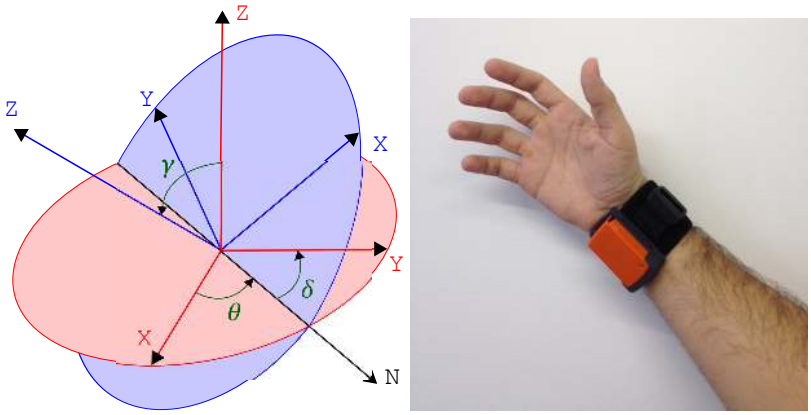

Figure 8.7: Inertial body sensor setting. **[Left]** Fixed coordinate system (red), sensor coordinate system (blue) and Euler angles (green). **[Right]** A 3D inertial body sensor at the right wrist.

Each of the scalar mappings $\theta_s \mapsto \sin\theta_s$, $\theta_s \mapsto \cos\theta_s$ is non-injective (and non-invertible) and therefore does not allow for the angle signal θ_s to be recovered. However, the considered 2D map $\theta_s \mapsto (\sin\theta_s, \cos\theta_s) \in \mathbb{R}^2$ is bijective and therefore invertible, hence, allowing us to recover the original angle signal θ_s. Additionally, the proposed map also allows us to preserve the dynamics of the signal. As illustrated in Fig. 8.8, sine and cosine preserve data variation successfully only when they behave in a linear-like fashion[2]; however, as such trigonometric functions are shifted versions of one another, by considering them together the signals dynamics are well preserved. The data corresponding to the so-mapped 12 angles were then represented by a 24-dimensional real signal.

[2]Recall that for $\theta \approx 0, \sin(\theta) \approx \theta$ and $\cos(1.5\pi + \theta) \approx \theta$.

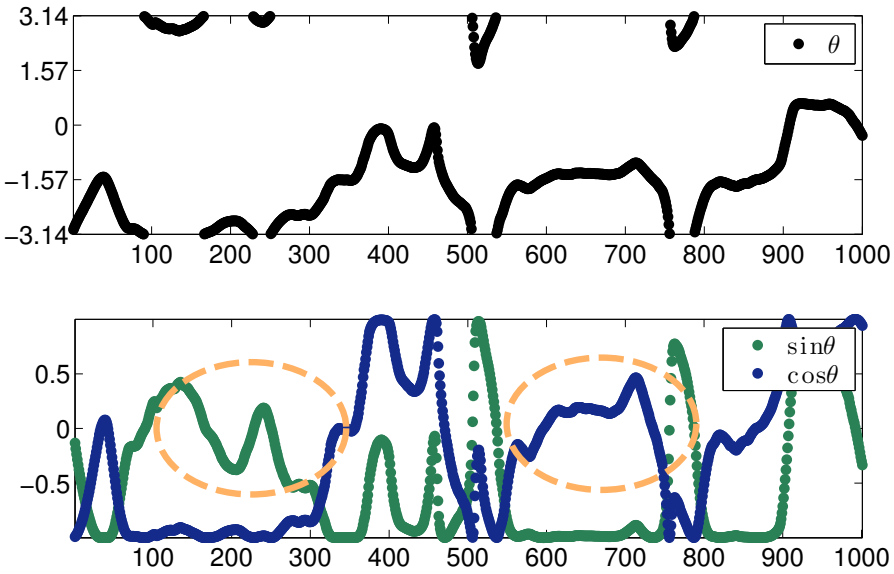

Figure 8.8: Raw angle measurements and considered features. **[Top]** Original discontinuous angle recording and **[Bottom]** the corresponding continuous sine and cosine mapping. Observe that in the right (left) circle only cosine (sine) preserves the dynamics of the angle signal accurately.

8.2.2 Signal Tracking using Quaternion Cubic Kernels and Ridge Regression

To perform estimation using quaternion kernels, the 24-dimensional real signal was converted into a six-dimensional quaternion signal. Therefore, by considering two delayed samples as regressors, the input and output pairs were respectively elements of $\mathbb{H}^{12}$ and $\mathbb{H}^{6}$.

Choice of Cubic Kernels

The quaternion KRR algorithm was compared to its real-, vector-valued, counterparts using cubic kernels. The real kernel used was the standard cubic kernel in eq. (4.14) for $p = 3$, that is[3], $K_{RP}(\mathbf{x}, \mathbf{y}) = \left(1 + \langle \mathbf{x}, \mathbf{y} \rangle_{\Re}\right)^3$, whereas the vector-kernel chosen was

$$K_M(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} \langle \mathbf{x}, \mathbf{y} \rangle_{\Re}^3 \\ \left(1 + \langle \mathbf{x}, \mathbf{y} \rangle_{\Re}\right)^3 \\ \left(10 + \langle \mathbf{x}, \mathbf{y} \rangle_{\Re}\right)^3 \\ \left(100 + \langle \mathbf{x}, \mathbf{y} \rangle_{\Re}\right)^3 \end{pmatrix}$$

[3]Recall that $\langle \mathbf{x}, \mathbf{y} \rangle_{\Re} = \Re\{\langle \mathbf{x}, \mathbf{y} \rangle\}$.

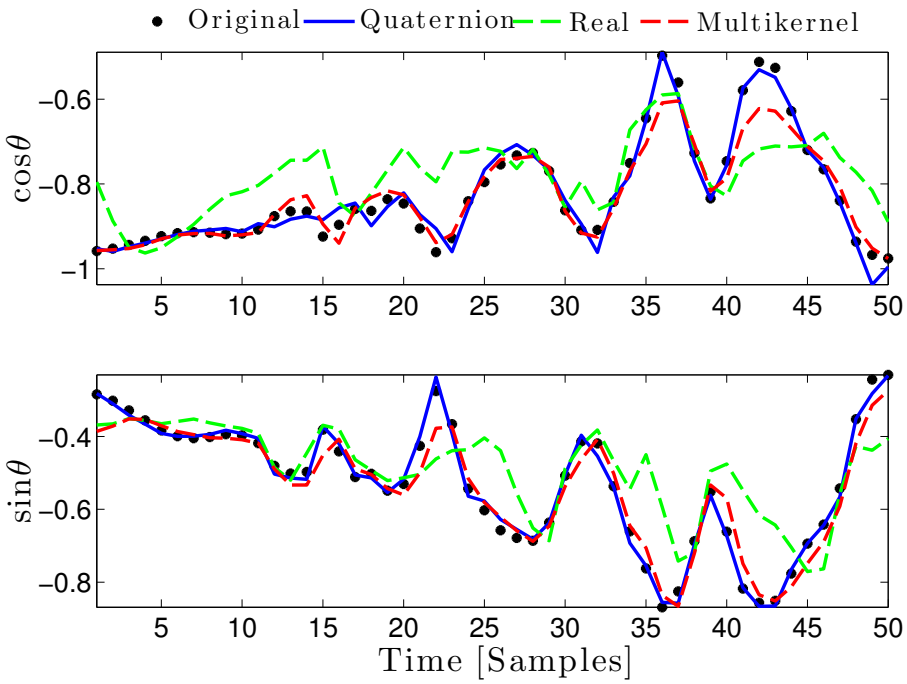

Figure 8.9: Body sensor signal tracking: Angle features $(\sin\theta, \cos\theta)$ and KRR estimates.

since its subkernels are a basis[4] of the space of cubic polynomials on $\langle \mathbf{x}, \mathbf{y} \rangle_{\Re}$ and performed better that other basis considered. Finally, we chose the quaternion kernel $K_{QP}(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^H\mathbf{x})(1 + \mathbf{x}^H\mathbf{y})(1 + \mathbf{y}^H\mathbf{y})$ introduced in eq. (4.15) to validate the quaternion kernel regression concept.

Results

We chose a regularisation parameter $\rho = 5$ as this suited all three algorithms and in particular allowed the multikernel not to suffer from overfitting. Fig. 8.9 shows the cosine and sine of one coordinate θ and their kernel estimates for 90 randomly chosen support vectors. Fig. 8.10 shows the averaged prediction mean square error (MSE) over 30 independent realisations, as a function of the number of support vectors for the same regularisation parameter. The support vectors and the validation set (50 samples) were randomly chosen, without repetition, for all realisations.

Observe that the scalar real kernel algorithm is outperformed by both the multikernel and quaternion ones due to their higher degrees of freedom. Moreover, note that the performance of the quaternion cubic kernel became progressively better than that of its real-valued counterpart as the number of support vectors (and therefore training samples) increased. The better performance of K_{QP} for a larger number of support vectors

[4] See Appendix A.2 for the proof that these subkernels are a basis for the space of cubic polynomials on $\langle \mathbf{x}, \mathbf{y} \rangle_{\Re}$.

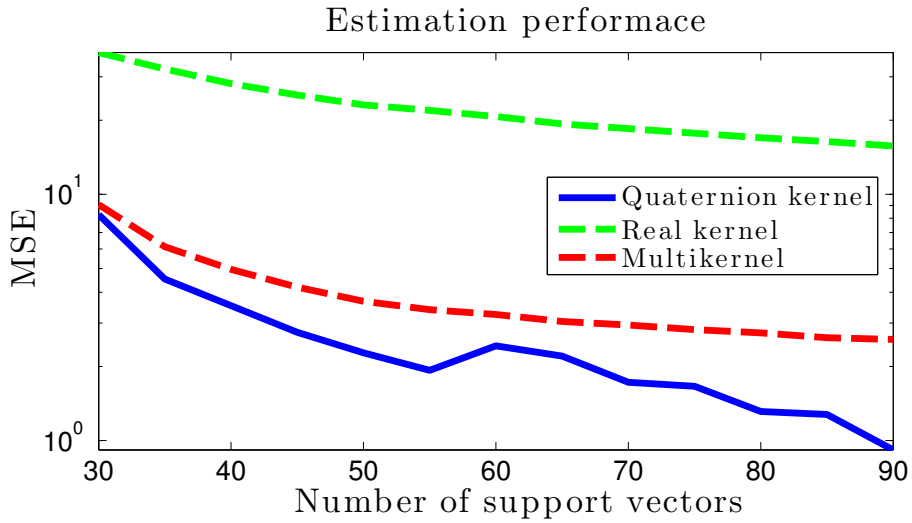

Figure 8.10: Performance of KRR algorithms for body sensor signal tracking as a function of the number of support vectors.

can be explained by the inability of K_{RP} to model cross-coupling between data components and the cross-coordinate terms, for which the quaternion cubic kernel is perfectly well suited (see Remark 9 in Section 4.2.3). Finally, Fig. 8.11 shows the computation time for all three algorithms. The complexities of the vector and quaternion kernels were found to be similar and greater than that of the real kernel.

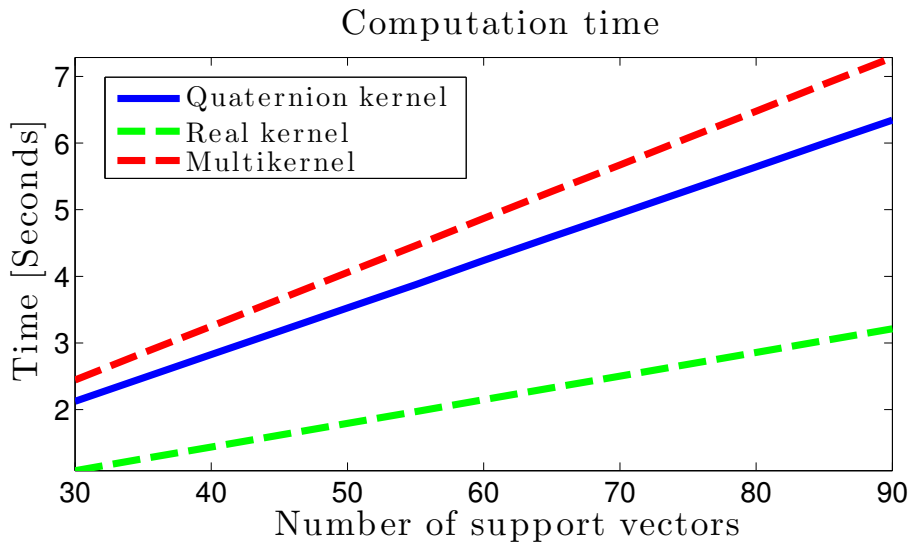

Figure 8.11: Computation time of KRR algorithms for body sensor signal tracking.

8.2.3 Signal Tracking using Multikernel Ridge Regression

For multikernel learning we considered one extra accelerometer (waist, see fig. 8.7), thus giving a total of 15 signals $\{\theta_s\}_{s=1,..,15}$ taking values in the range $[-\pi, \pi]$. After the preprocessing of Section 8.2.1 the signal was converted to be 30-dimensional signal. The training set used to learn the evolution of the recorded data was constructed using two regressors, that is, the training pairs were elements of $\mathbb{R}^{60} \times \mathbb{R}^{30}$.

Quadratic and cubic polynomial kernels were fitted to implement two monokernel and one two-kernel multikernel ridge regression algorithms. The regularisation parameter of KRR was set to $\rho = 10^{-3}$ (see eq. (5.16)) and the cubic kernel was normalised with $\eta = 30$. By randomly choosing both the support vector and training samples from the recorded data (the number of training samples was 15 times that of the support vectors), the optimal weights were computed according to (5.16). Fig. 8.12 shows the running time for the computation of the optimal least squares weights for all the three KRR algorithms in MATLAB. As discussed in Section 5.3.2, although the implementation of the MKRR is more complex than the standard KRR, the actual running time is below 10 [ms] which is desired for the learning of a 1200-sample sequence (80 support samples).

Fig. 8.13 shows the averaged training MSE and its standard deviation as a function of the number of support vectors for 100 independent trials (i.e. 100 independent choices of the support samples). The ability of the multikernel approach to replicate nonlinear non-homogeneous multivariate mappings is highlighted by the lower training error achieved by MKRR, compared to the KRR algorithms, for the same number of support vectors, and in particular for a lower number of support vectors.

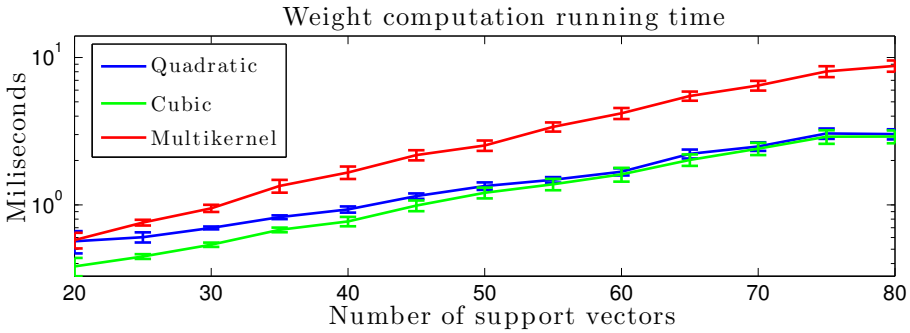

Figure 8.12: Average running time and standard deviation for kernel algorithms as a function of the number of support vectors.

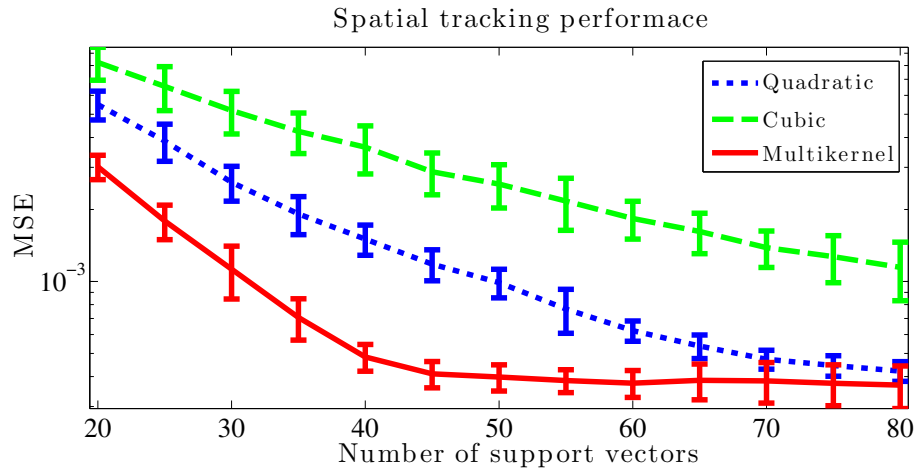

Figure 8.13: MSE and standard deviation for kernel algorithms as a function of number of support vectors.

8.3 Prediction of Bivariate Point-of-Gaze using KSSM and Artificial Evolution

We next validated the kernel SSM for the task of gaze prediction and compared it to the kernel normalised LMS (KNLMS) and kernel RLS (KRLS). A Tobii T60 Eye Tracker was used to acquire the horizontal and vertical point-of-gaze signals when reading a 20-word text arranged in two lines. This is a challenging task, as a *linear* reading of the task is followed by a *jump* to the beginning of the next line.

The data were centered, missing values were replaced by their previous value and a preliminary recording was used to set the empirical parameters: a) kernel width $A = 5 \cdot 10^{-4}$, b) KNLMS gain $\mu = 0.6$, c) ALD threshold $\delta = 0.1$, d) number of particles $N_p = 800$ and e) kernel SSM variances $\sigma_\omega^2 = 1$,

$$\Sigma_X^2 = 300 \begin{bmatrix} 10 & 1 \\ 1 & 1 \end{bmatrix}, \; \Sigma_Y^2 = 300 \begin{bmatrix} 5 & 2.45 \\ 2.45 & 2.8 \end{bmatrix}.$$

The performances of all three kernel algorithms considered were assessed in the presence of additive Gaussian noise of different power added to the point-of-gaze data; the averaged performances as a function of the signal-to-noise ratio (SNR) are shown in Fig. 8.14. The KSSM (in red) proved to be more robust to uncertainties in the observations than the KNLMS and KRLS. Fig. 8.15 illustrates the accuracy of KSSM compared to the KNLMS and KRLS, and its reduced level of noise in predictions.

The KSSM provides a full statistical description of the signal. Fig. 8.16 shows the prediction density and the original signal (SNR=26.02 [dB]). Observe that the prediction

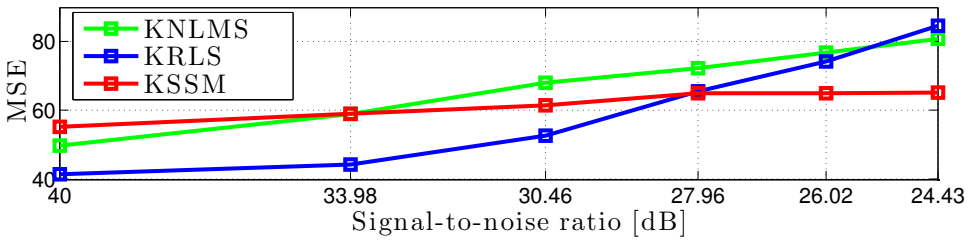

Figure 8.14: Performances of kernel algorithms for gaze prediction.

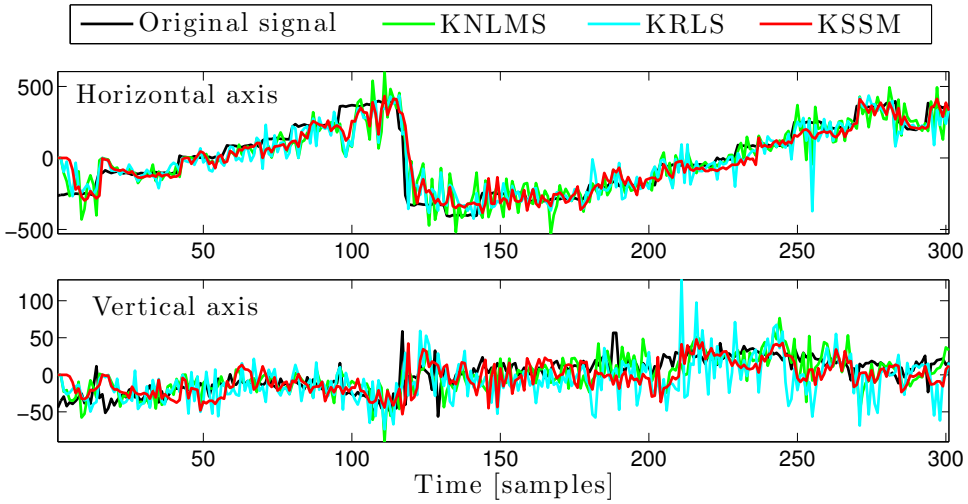

Figure 8.15: Kernel algorithms prediction and original gaze signal for KNLMS, KRLS and KSSM.

densities computed using KSSM successfully track the signal, as their probability mass is located around the original process. This makes possible for the computation of both reliable estimates and the associated confidence intervals.

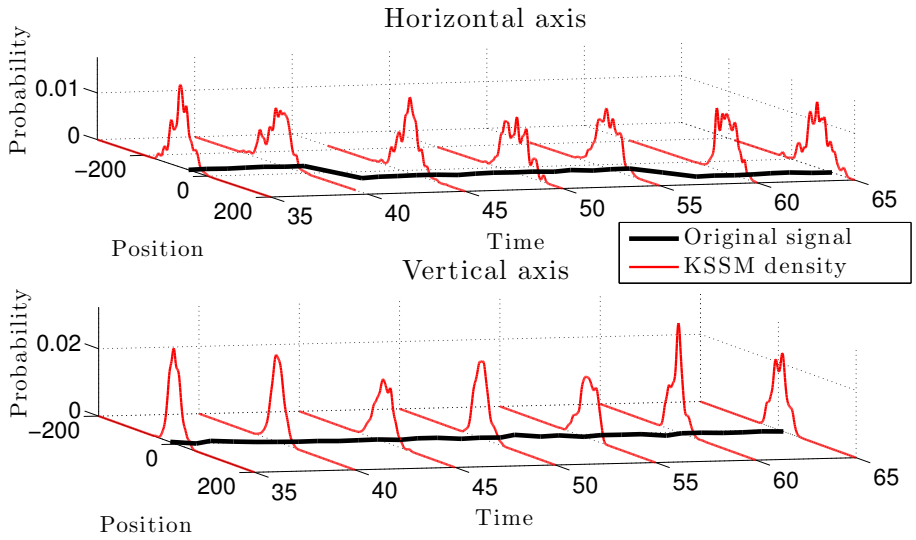

Figure 8.16: Original signal and kernel SSM densities for the prediction of gaze signals.

8.4 Prediction of Power-Grid Frequency using KSSM and MCMC

Frequency estimation is a key topic in signal processing and has become even more important in the last decade due to Smart Grids applications [Xia et al., 2012], where it allows for the planing and control of grid operation.

We considered the frequency signal of the UK national grid[5] for the day 17 July 2014, where measurements became available every five minutes. The frequency data, originally in the region [49.85 Hz, 50.15 Hz], were scaled according to

$$\text{Scaled frequency} = 50(\text{Raw frequency} - 50) \tag{8.4}$$

for simplicity of presentation; the scaled frequency was then in the region [-8,8]. The scaled frequency was modelled by the hidden process of a KSSM of order two, as shown in eq. (7.22), and the observation process was created by adding Gaussian noise of standard deviation $\sigma_y = 0.3$ to the state signal. The aim of this experiment was to recursively approximate the transition function of the frequency process using the method in Section 7.2 and to perform one-step ahead prediction.

[5]Data available from `http://www.gridwatch.templar.co.uk/`.

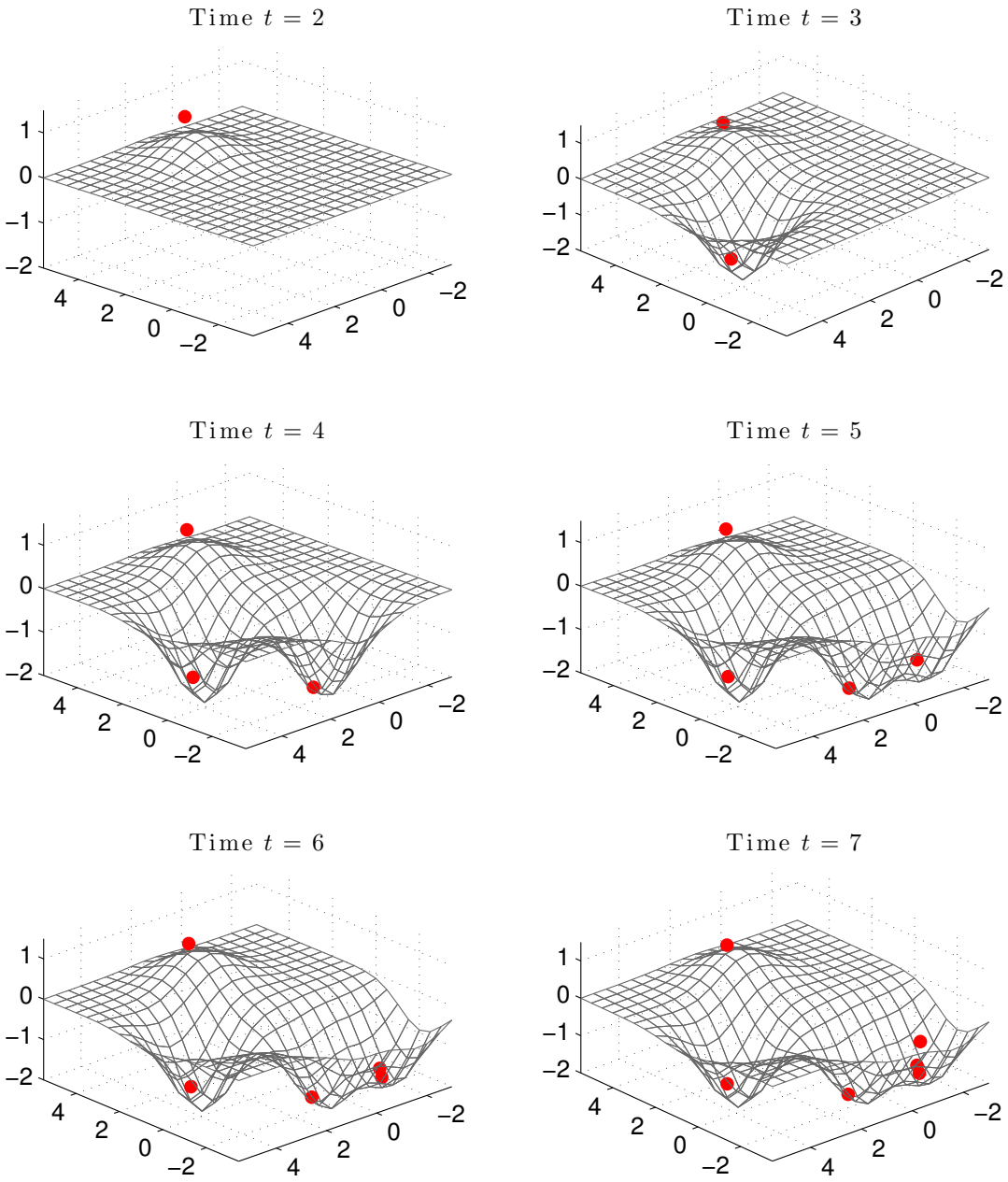

Figure 8.17: Online estimation of the state-transition function for $t \in [2, 7]$. The hidden state samples are shown in red.

Fig. 8.17 shows the estimates of the transition function for time instants $t \in [2, 7]$, together with the state samples until the corresponding time step.[6] Notice that support vectors were included for $t \in [2, 5]$, and that for these time steps the learnt mapping corresponded to a smooth mapping form $\mathbb{R}^2$ to $\mathbb{R}$ that represented the hidden state samples. Furthermore, observe that for $t \in [6, 7]$ (where support vectors were not added) the learnt

[6]The case $t = 1$ is omitted since at least two observations are needed to perform the inference on the mixing parameters—see eq.(7.24).

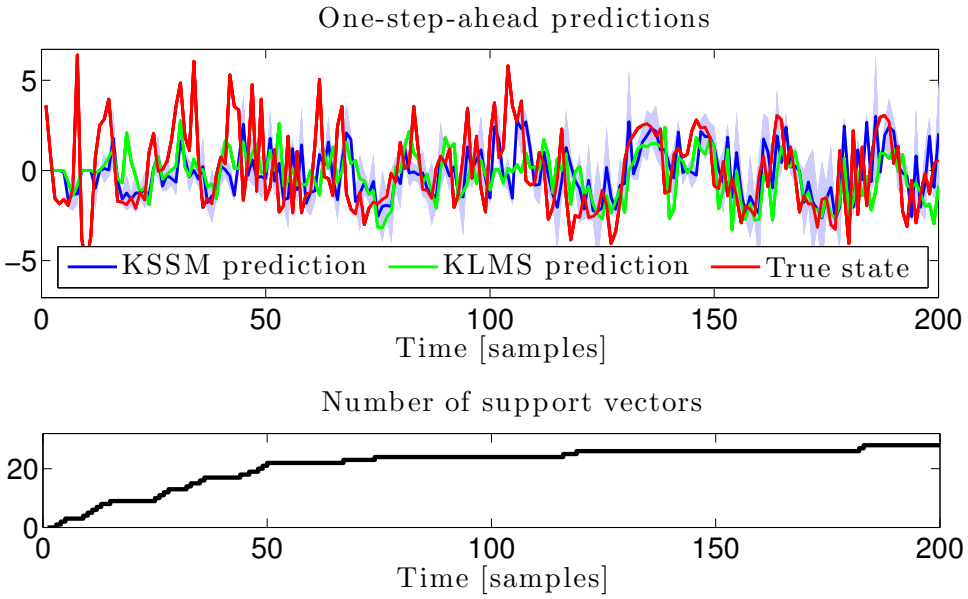

Figure 8.18: Kernel predictions of the UK National Grid frequency. The original signal is shown in red, the KLMS prediction in red, the KSSM prediction in blue, and the one-standard-deviation confidence interval in light blue.

mapping remained robust to new samples. As in the synthetic examples in the previous sections, the estimates were updated only locally, this is a consequence of the prior and MCMC move densities presented in Section 7.2.3.

The proposed KSSM was also implemented for Bayesian prediction of the frequency signal. We considered 200 samples, corresponding to approximately 16.7 hours, and validated the KSSM in the one-step-ahead prediction setting. The prediction algorithms considered were:

Persistent Estimation: The prediction at time t is simply the previous valued of the observed observed process y_{t-1}.

Kernel Normalised Least Mean Square (KLMS): A standard in kernel adaptive filtering, where the estimate is produced by performing nonlinear regression on the observation signal y_t using kernels and an LMS-based update rule.

Kernel State-Space Model (KSSM): The adaptive version of the proposed method, where the predictions are generated by propagating the particles of the state according to the estimated transition function. A sequential importance resampling (SIR) [Gordon et al., 1993] particle filter with stratified sampling was considered.

For a meaningful comparison, both KLMS and KSSM used the same dictionary set using coherence on the observed process y_t and the same kernel width $\sigma = 2$. Fig.

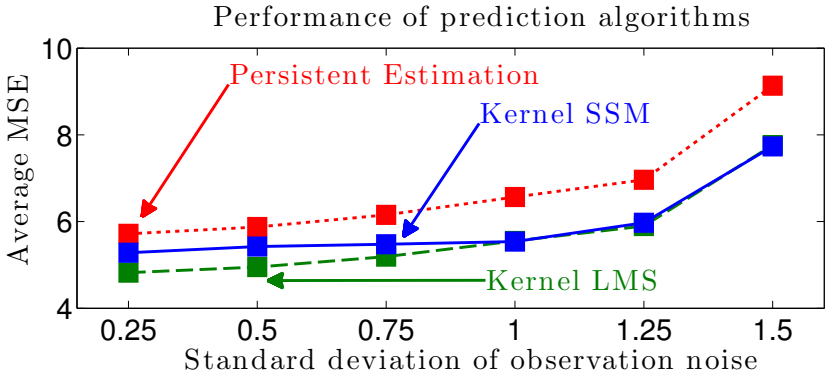

Figure 8.19: Prediction performance of the considered algorithms.

8.18 shows the one-step-ahead predictions using the considered kernel algorithms and the true frequency signal (top), and the number of support vectors (bottom). Observe that the kernel predictions were fairly inaccurate for $t < 50$, as not enough samples had been processed, moreover, notice that a large volume of support vectors were added during this period. We can therefore consider this period as the *transient* of the kernel adaptive estimators. As t increased, the kernel predictions became progressively better, in particular, the one-standard-deviation confidence interval of the KSSM prediction also became larger to include the true frequency signal.

The prediction performance was assessed for different levels of observation noise, over 25 realisations, in terms of the average mean square error (MSE)

$$\text{Average MSE} = \frac{1}{R}\sum_{r=1}^{R}\frac{1}{T}\sum_{t=1}^{T}(x_t - x^r_{t|t-1})^2 \tag{8.5}$$

where $R = 25$ is the number of realisations, $T = 200$ is the number of time steps, and $x^r_{t|t-1}$ the r^{th} realisation of the one-step-ahead prediction of the frequency signal. For each realisation, the observed signal was recreated by adding zero mean Gaussian noise of standard deviation in the range $\sigma_y \in [0.25, 1.5]$.

Fig. 8.19 shows the average MSE for all three considered algorithms and levels of observation noise. Observe that both kernel predictors outperformed the persistent estimate in all cases, this means that the kernel-based predictors are indeed capable of capturing the nonlinear dynamics of the signal. Furthermore, notice that for low levels of observation noise, the KLMS slightly outperformed the KSSM, whereas for higher levels of noise both kernel algorithms performed similarly. This validates the ability of the proposed KSSM algorithm to design meaningful dynamic models in an unsupervised learning setting, since the estimated model not only predicts real world signals as accurate as kernel adaptive filters (KLMS), but also it gives a full probabilistic description of the nonlinear dynamics.

Chapter 9

Conclusions

A thinker sees his own actions as experiments and questions—as attempts to find out something. Success and failure are for him answers above all.

-Friedrich Nietzsche.
("The Gay Science," 1882.)

Linear filtering assumes a known topology for the model, therefore, the task is to jointly find the model parameters and the values of the signal, performing either batch or online estimation. In nonlinear filtering, conversely, the topology of the model is unknown in the general case, and the first step is to consider a large class of models in order to find the *best* model within the chosen class to, ultimately, estimate the signal. Nonlinear filtering is therefore challenging but also enormously appealing, since, rather than being an *identification* problem (as in system identification), it is a *design* problem that requires us to combine both empirical evidence and theoretical knowledge of the underlying dynamical processes. We have considered data-driven techniques to design nonlinear filters, these are rooted in the very foundations of signal processing due to their ability to build models in cases when the derivation of a closed-form mathematical expression is unfeasible. In particular, this thesis has explored how kernel-based methods can be used to design enhanced filtering approaches both in the adaptive and Bayesian senses.

9.1 Hypercomplex Kernels

The existence and uniqueness of quaternion RKHS have been provided to give a foundational theoretical framework for the design of quaternion-valued reproducing kernels. As a direct consequence, it is now possible to implement estimation algorithms operating on quaternion-valued feature spaces in a straightforward manner, by relying on the computationally-simple SVR framework. More specifically, the so-proposed QRKHS

paradigm allows for performing non-linear adaptive filtering in QRKHS by only choosing a quaternion-valued positive definite kernel, analogously to the real- and complex-valued cases. With the standard RKHS and proposed QRKHS, we have also studied different ways of designing specific complex and quaternion kernels based on real kernels with well-known properties. The experimental validation has demonstrated that these hypercomplex kernels have the ability to capture the inherent data relationships in way that is a more accurate and physically-meaningful than using real kernels.

9.2 Multikernel Learning

To further investigate the benefits of higher-dimensionality feature spaces in kernel estimation, we have considered vector-valued RKHS constructed upon the Cartesian product of real (scalar) RKHS—in this way, the resulting VRKHS can be of an arbitrary dimension. Through a rigorous analysis of the reproducing properties of VRKHS, inherited from the scalar RKHSs, we have shown that it serves as a feature space for the multikernel learning paradigm and therefore gives theoretical support for the design of such algorithms. We have also illustrated how multikernel algorithms can be designed by performing SVR on the proposed VRKHS in both batch and adaptive scenarios, in particular, we introduced the multikernel ridge regression (MKRR) and the multikernel least mean square algorithm (MKLMS), these were further equipped with low-complexity adaptive sparsification criteria. The advantages of the multikernel estimation concept in capturing different *types* of nonlinear behaviours have also been illuminated in a experimental setting; this establishes multikernel estimators as a proven alternative for adaptive nonlinear filtering, where the underlying nonlinearity is not only unknown but also time-varying, and the estimators are required to identify the nonlinear behaviour in a recursive manner.

9.3 Learning and Prediction using Kernel State-Space Models (KSSM)

We have proposed a novel framework for the design of state-space models by parametrising the state-transition function using reproducing kernels, and then finding the mixing parameters in an unsupervised fashion with Monte Carlo methods. The KSSM formulation unifies the enhanced function approximation ability of reproducing kernels with the flexibility of sequential Monte Carlo methods. The resulting algorithm allows us to perform hybrid inference on the unknown quantities: the mixing weights and the hidden process are found in a Bayesian fashion, since their densities are required in several applications, whereas the model order (number of support vectors) and kernel width are

found with deterministic techniques from KAF, thus reducing the computational complexity.

Through illustrative examples, we have shown that the model estimates provided by KSSM are close (in likelihood terms) to the supervised solution, and also resemble the unobserved state samples. Additionally, the proposed KSSM formulation can be implemented in an online manner to perform joint system identification and state estimation; in this setting, the KSSM was also validated for the prediction of real-world signals, where it performed as accurately as standard KAF, while providing the desired probabilistic estimates.

9.4 Experimental Validation of the Proposed Methods

The proposed algorithms were first tested through illustrative case studies using synthetic data, thus allowing us to investigate their performance and accuracy in controlled and well-understood environments. Furthermore, the high-dimensional kernel and kernel SSM paradigms were validated in the estimation of real-world signals, where they were compared against standard KAF algorithms. This not only gives a quantitative evidence for the benefits that the proposed approaches represent in the field of nonlinear adaptive filtering, but also equips the contributions of this thesis with a proof-of-concept that enables researchers and practitioners alike to both benefit from and complement these findings in practical and theoretical manners

9.5 Future Research Directions

The field of kernel-based signal estimation lies between Signal Processing and Machine Learning, and the research opportunities it provides are manifold. Besides practical and theoretical contributions, this thesis opens new research possibilities summarised below.

A generic framework for designing quaternion kernels. The proposed complexification procedure to generate complex kernels from real kernels can be further extended to quaternions, thus equipping the QRKHS with a wide variety of kernels, based on real kernels, to construct meaningful quaternion kernel algorithms.

A study of the regularising properties of multikernel estimation. We considered multikernel estimators of ridge regression and least mean square types, these are well-known to provide regularised solutions and therefore do not suffer from overfitting. In this sense, the learning performance can be investigated as a function of the number of kernels, so as to give insight into the extent to which a multikernel estimator can learn, and when it suffers from overfitting.

Comparison with Gaussian process (GP). Within Gaussian processes [Rasmussen and Williams, 2005], kernels are considered as covariance functions and are commonly designed as a sum and multiplication of more elementary kernels [Duvenaud et al., 2013]. This suggests two aspects for future study: (i) the physical consequences of considering high-dimensional kernels in GP can be explored to determine how GP regression can benefit from the hypercomplex and multikernel approaches, and (ii) an improvement of the proposed KSSM through an analogy with GPs, since the KSSM can be understood as a parametric version of a GP for model identification.

Improved numerical methods for KSSM inference. We have proposed a kernel-based SSM and have validated its estimation ability using MCMC and PF; however, more numerically-efficient tools can be employed to estimate the kernel mixing parameters. In particular, the expectation-maximisation algorithm [Dempster et al., 1977] can be used to either find an approximate maximum-likelihood estimate, or to set a better initial condition for the MCMC that reduces its burn-in time.

Bibliography

[Adkins and Weintraub, 1992] Adkins, W. and Weintraub, S. (1992). *Algebra: An Approach Via Module Theory*. Graduate Texts in Mathematics Series. Springer-Verlag.

[Aizerman et al., 1964] Aizerman, A., Braverman, E. M., and Rozoner, L. I. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837.

[Anderson and Fuller, 1992] Anderson, F. and Fuller, K. (1992). *Rings and Categories of Modules*, volume 13 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2nd edition.

[Andrieu et al., 2003] Andrieu, C., de Freitas, N., Doucet, A., and Jordan, M. I. (2003). An introduction to MCMC for machine learning. *Machine Learning*, 50(1):5–43.

[Aronszajn, 1950] Aronszajn, N. (1950). Theory of reproducing kernels. *Trans. of the American Mathematical Society*, 68(3):337–404.

[Arulampalam et al., 2002] Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188.

[B. Schölkopf and Williamson, 2000] B. Schölkopf, R. H. and Williamson, R. (2000). A generalized representer theorem. Tech. rep. nc2-tr-2000-81, NeuroCOLT, Royal Holloway College, Univ. London, UK.

[Bain and Crisan, 2009] Bain, A. and Crisan, D. (2009). *Fundamentals of Stochastic Filtering*. Springer.

[Beneš, 1981] Beneš, V. E. (1981). Exact finite dimensional filters for certain diffusion with nonlinear drift. *Stochastics*, 5(1-2):65–92.

[Boser et al., 1992] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proc. of the 5th annual workshop on computational learning theory*, pages 144–152.

[Bouboulis and Theodoridis, 2010] Bouboulis, P. and Theodoridis, S. (2010). The complex Gaussian kernel LMS algorithm. In *Proc. of the 20th Intl. Conf on Artificial Neural Networks: Part II*, pages 11–20.

[Bouboulis and Theodoridis, 2011] Bouboulis, P. and Theodoridis, S. (2011). Extension of Wirtinger's calculus to reproducing kernel Hilbert spaces and the complex kernel LMS. *IEEE Transactions on Signal Processing*, 59(3):964–978.

[Bouboulis et al., 2012] Bouboulis, P., Theodoridis, S., and Mavroforakis, M. (2012). The augmented complex kernel LMS. *IEEE Trans. on Signal Processing*, 60(9):4962–4967.

[Candy, 2011] Candy, J. V. (2011). *Bayesian signal processing: Classical, modern and particle filtering methods*, volume 54. John Wiley & Sons.

[Cheong Took and Mandic, 2011] Cheong Took, C. and Mandic, D. P. (2011). Augmented second-order statistics of quaternion random signals. *Signal Processing*, 91(2):214–224.

[Cristianini and Shawe-Taylor, 2000] Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38.

[Djurić et al., 2003] Djurić, P., Kotecha, J. H., Zhang, J., Huang, Y., Ghirmai, T., Bugallo, M., and Miguez, J. (2003). Particle filtering. *IEEE Signal Processing Magazine*, 20(5):19–38.

[Douc and Cappe, 2005] Douc, R. and Cappe, O. (2005). Comparison of resampling schemes for particle filtering. In *Proc. of the 4th International Symposium on Image and Signal Processing and Analysis, 2005. ISPA 2005.*, pages 64–69.

[Doucet et al., 2001] Doucet, A., de Freitas, N., and Gordon, N. (2001). *Sequential Monte Carlo Methods in Practice*. Statistics for engineering and information science. Springer.

[Doucet et al., 2000] Doucet, A., Godsill, S., and Andrieu, C. (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and computing*, 10(3):197–208.

[Doucet and Johansen, 2009] Doucet, A. and Johansen, A. M. (2009). A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12:656–704.

[Douglas, 2009] Douglas, S. (2009). Widely-linear recursive least-squares algorithm for adaptive beamforming. In *Proc. of IEEE ICASSP*, pages 2041–2044.

[Drucker et al., 1996] Drucker, H., Burges, C., Kaufman, L., Smola, A., and Vapnik, V. (1996). Support vector regression machines. In *Proc. Advances in Neural Information Processing Systems 9*, pages 155–161.

[Duvenaud et al., 2013] Duvenaud, D., Lloyd, J. R., Grosse, R., Tenenbaum, J. B., and Ghahramani, Z. (2013). Structure discovery in nonparametric regression through compositional kernel search. In *Proc. of the 30th International Conference on Machine Learning*.

[Ell and Sangwine, 2007] Ell, T. A. and Sangwine, S. J. (2007). Quaternion involutions and anti-involutions. *Computers & Mathematics with Applications*, 53(1):137 – 143.

[Engel et al., 2002] Engel, Y., Mannor, S., and Meir, R. (2002). Sparse online greedy support vector regression. In *13th European Conference on Machine Learning*, pages 84–96.

[Engel et al., 2004] Engel, Y., Mannor, S., and Meir, R. (2004). The kernel recursive least-squares algorithm. *IEEE Transactions on Signal Processing*, 52(8):2275–2285.

[Epanechnikov, 1969] Epanechnikov, V. A. (1969). Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1):153–158.

[Fleuret and Sahbi, 2003] Fleuret, F. and Sahbi, H. (2003). Scale-invariance of support vector machines based on the triangular kernel. In *3rd International Workshop on Statistical and Computational Theories of Vision*.

[Friedman, 1982] Friedman, A. (1982). *Foundations of modern analysis*. Dover, 2nd edition.

[Frobenius, 1878] Frobenius, F. G. (1878). Über lineare substitutionen und bilineare formen. *Journal für die reine und angewandte Mathematik*, 84:1–63.

[Gabor et al., 1960] Gabor, D., Wilby, W. P. L., and Woodcock, R. (1960). A universal non-linear filter, predictor and simulator which optimizes itself by a learning process. *Proceedings of the IEE - Part B: Electronic and Communication Engineering*, 108(40):422–435.

[Galton, 1886] Galton, F. (1886). Regression towards mediocrity in hereditary stature. *The Journal of the Anthropological Institute of Great Britain and Ireland*, 15:pp. 246–263.

[Gelfand and Smith, 1990] Gelfand, A. E. and Smith, A. F. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409.

[Gershgorin, 1931] Gershgorin, S. A. (1931). über die abgrenzung der eigenwerte einer matrix. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, (6):749–754.

[Geweke, 1989] Geweke, J. (1989). Bayesian inference in econometrics models using Monte Carlo integration. *Econometrica*, 57:1317–1339.

[Gönen and Alpaydın, 2011] Gönen, M. and Alpaydın, E. (2011). Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268.

[Gordon et al., 1993] Gordon, N., Salmond, D., and Smith, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113.

[Hamilton, 1844] Hamilton, W. R. (1844). On quaternions, or on a new system of imaginaries in algebra. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, 25:10–13.

[Harter, 1974] Harter, W. L. (1974). The method of least squares and some alternatives: Part I. *International Statistical Review / Revue Internationale de Statistique*, 42(2):pp. 147–174.

[Hastings, 1970] Hastings, W. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

[Haykin, 1994] Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Macmillan.

[Haykin, 2001] Haykin, S. (2001). *Adaptive Filter Theory*. Prentice Hall, 4th edition.

[Jacobson, 1943] Jacobson, N. (1943). *The Theory of Rings*. American Mathematical Soc.

[Jacobson, 2009] Jacobson, N. (2009). *Basic algebra I*. Dover.

[Jahanchahi et al., 2010] Jahanchahi, C., Took, C., and Mandic, D. (2010). On HR calculus, quaternion valued stochastic gradient, and adaptive three dimensional wind forecasting. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–5.

[Javidi et al., 2008] Javidi, S., Pedzisz, M., Goh, S. L., and Mandic, D. P. (2008). The augmented complex least mean square algorithm with application to adaptive prediction problems. In *Proceedings of the Cognitive Information Processing Conference*, pages 54–57.

[Kailath, 1974] Kailath, T. (1974). A view of three decades of linear filtering theory. *IEEE Transactions on Information Theory*, 20(2):146–181.

[Kalman, 1960] Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45.

[Kimeldorf and Wahba, 1971] Kimeldorf, G. and Wahba, G. (1971). Some results on Tchebycheffian spline functions. *J. Math. Anal. Appl.*, 33:82–95.

[Kitagawa and Sato, 2001] Kitagawa, G. and Sato, S. (2001). *Sequential Monte Carlo Methods in Practice*, chapter Monte Carlo Smoothing and Self-Organising State-Space Model. Springer.

[Kolmogorov, 1931] Kolmogorov, A. (1931). Über die analytischen methoden in der wahrscheinlichkeitsrechnung. *Mathematische Annalen*, 104(1):415–458.

[Kolmogorov, 1941] Kolmogorov, A. N. (1941). Stationary sequences in Hilbert space. *Bull. Moscow State Univ.*, 2(6).

[Kong et al., 1994] Kong, A., Liu, J. S., and Wong, W. H. (1994). Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):pp. 278–288.

[Kuh and Mandic, 2009] Kuh, A. and Mandic, D. P. (2009). Applications of complex augmented kernels to wind profile prediction. In *Proc. of IEEE ICASSP*, pages 3581–3584.

[Kuhn, 1962] Kuhn, T. S. (21962). *The structure of scientific revolutions*. University of Chicago press.

[Kung, 2014] Kung, S.-Y. (2014). *Kernel Methods and Machine Learning*. Cambridge Press.

[Kushner, 1964] Kushner, H. (1964). On the differential equations satisfied by conditional probability densities of Markov processes, with applications. *SIAM Control Ser. A*, 21(1):106–119.

[Lin et al., 2005] Lin, C.-J., Hong, S.-J., and Lee, C.-Y. (2005). Using least squares support vector machines for adaptive communication channel equalization. *International Journal of Applied Science and Engineering*, 3(1):51–59.

[Liu, 1996] Liu, J. (1996). Metropolized independent sampling with comparison to rejection sampling and importance sampling. *Stat. Comput.*, 6:113–119.

[Liu and West, 2001] Liu, J. and West, M. (2001). *Sequential Monte Carlo Methods in Practice*, chapter Combined parameter and state estimation in simulation-based filtering. Springer.

[Liu et al., 2008] Liu, W., Pokharel, P. P., and Principe, J. C. (2008). The kernel least-mean-square algorithm. *IEEE Trans. on Signal Processing*, 56(2):543–554.

[Liu et al., 2010] Liu, W., Principe, J. C., and Haykin, S. (2010). *Kernel adaptive filtering: A comprehensive introduction*. Wiley.

[MacKay, 1992] MacKay, D. J. (1992). Bayesian interpolation. *Neural computation*, 4(3):415–447.

[MacKay, 1999] MacKay, D. J. C. (1999). Comparison of approximate methods for handling hyperparameters. *Neural Computation*, 11(5):1035–1068.

[Mahalanobis, 1936] Mahalanobis, P. C. (1936). On the generalised distance in statistics. In *Proceedings of the National Institute of Sciences of India*, volume 2, pages 49–55.

[Mandic, 2004] Mandic, D. P. (2004). A generalized normalized gradient descent algorithm. *IEEE Signal Processing Letters*, 11(2):115–118.

[Mandic and Goh, 2009] Mandic, D. P. and Goh, S. L. (2009). *Complex valued nonlinear adaptive filters: Noncircularity, widely linear and neural models*. John Wiley & Sons.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

[Mercer, 1909] Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Trans. of the Royal Society of London (A)*, 209:415–446.

[Metropolis et al., 1953] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

[Minh, 2010] Minh, H. Q. (2010). Some properties of Gaussian reproducing kernel Hilbert spaces and their implications for function approximation and learning theory. *Constructive Approximation*, 32(2):307–338.

[Murray, 2007] Murray, I. (2007). *Advances in Markov chain Monte Carlo methods*. PhD thesis, Gatsby computational neuroscience unit, University College London.

[Neal, 1995] Neal, R. M. (1995). *Bayesian learning for neural networks*. PhD thesis, University of Toronto.

[Orchard et al., 2013] Orchard, M., Hevia-Koch, P., Zhang, B., and Tang, L. (2013). Risk measures for particle-filtering-based state-of-charge prognosis in lithium-ion batteries. *IEEE Trans. on Industrial Electronics*, 60(11):5260–5269.

[Platt, 1991] Platt, J. (1991). A Resource-Allocating Network for Function Interpolation. *Neural Computation*, 3(2):213–225.

[Principe, 2001] Principe, J. C. (2001). Dynamic neural networks and optimal signal processing. In Hu, Y. H. and Hwa, J.-N., editors, *Handbook of Neural Network Signal Processing*. CRC Press.

[Rasmussen and Williams, 2005] Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning*. The MIT Press.

[Richard et al., 2009] Richard, C., Bermudez, J. C. M., and Honeine, P. (2009). Online prediction of time series data with kernels. *IEEE Trans. on Signal Processing*, 57(3):1058–1067.

[Robert et al., 2011] Robert, C., Casella, G., et al. (2011). A short history of Markov chain Monte Carlo: Subjective recollections from incomplete data. *Statistical Science*, 26(1):102–115.

[Robert and Casella, 1999] Robert, C. P. and Casella, G. (1999). *Monte Carlo statistical methods*. Springer.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

[Samuel, 1959] Samuel, A. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210229.

[Saunders et al., 1998] Saunders, C., Gammerman, A., and Vovk, V. (1998). Ridge regression learning algorithm in dual variables. *Proc. of the 15th International Conference on Machine Learning*, 75(2):515–521.

[Sayed, 2003] Sayed, A. H. (2003). *Fundamentals of Adaptive Filtering*. John Wiley, New Jersey.

[Schölkopf et al., 2001] Schölkopf, B., Herbrich, R., and Smola, A. (2001). A generalized representer theorem. In Helmbold, D. and Williamson, B., editors, *Computational Learning Theory*, volume 2111 of *Lecture Notes in Computer Science*, pages 416–426. Springer Berlin Heidelberg.

[Schölkopf and Smola, 2001] Schölkopf, B. and Smola, A. (2001). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press, Cambridge, MA, USA.

[Schölkopf et al., 1998] Schölkopf, B., Smola, A., and Müller, K. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319.

[Schürmann, 1996] Schürmann, J. (1996). *Pattern classification: A unified view of statistical and neural approaches*. Wiley.

[Sherman and Morrison, 1950] Sherman, J. and Morrison, W. J. (1950). Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127.

[Smee, 1850] Smee, A. (1850). *Instinct and reason: deduced from electro-biology*. Reeve and Benham.

[Steinwart, 2001] Steinwart, I. (2001). On the influence of the kernel on the consistency of support vector machines. *J. Mach. Learn. Res.*, 2:67–93.

[Steinwart and Christmann, 2008] Steinwart, I. and Christmann, A. (2008). *Support vector machines*. Springer.

[Steinwart et al., 2006] Steinwart, I., Hush, D., and Scovel, C. (2006). An explicit description of the reproducing kernel Hilbert spaces of Gaussian RBF kernels. *IEEE Trans. on Information Theory*, 52(10):4635–4643.

[Stratonovich, 1960] Stratonovich, R. (1960). Conditional Markov processes. *Theory of Probability and its Applications*, 5(156–178).

[Thrun et al., 2002] Thrun, S., J., L., and Verma, V. (2002). Risk sensitive particle filters. In *Advances in Neural Information Processing Systems 14*. MIT Press.

[Tobar et al., 2014a] Tobar, F., Djurić, P., and Mandic, D. (2014a). Unsupervised state-space modelling using reproducing kernels. (Submitted to IEEE Trans. on Signal Processing).

[Tobar et al., 2012] Tobar, F., Kuh, A., and Mandic, D. (2012). A novel augmented complex valued kernel LMS. In *Proc. of the 7th IEEE Sensor Array and Multichannel Signal Processing Workshop*, pages 473–476.

[Tobar et al., 2014b] Tobar, F., Kung, S.-Y., and Mandic, D. (2014b). Multikernel least mean square algorithm. *IEEE Trans. on Neural Networks and Learning Systems*, 25(2):265–277.

[Tobar and Mandic, 2012] Tobar, F. and Mandic, D. (2012). Multikernel least squares estimation. In *Proc. of the Sensor Signal Processing for Defence Conference*, pages 1–5.

[Tobar and Mandic, 2013] Tobar, F. and Mandic, D. (2013). The quaternion kernel least squares. In *Proc. of ICASSP*, pages 6128–6132.

[Tobar and Mandic, 2014a] Tobar, F. and Mandic, D. (2014a). A particle filtering based kernel HMM predictor. In *Proc. of ICASSP*, pages 7969–7973.

[Tobar and Mandic, 2014b] Tobar, F. and Mandic, D. (2014b). Quaternion reproducing kernel Hilbert spaces: Existence and uniqueness conditions. *IEEE Transactions on Information Theory*, 60(9):5736–5749.

[Tobar et al., 2011] Tobar, F., Yacher, L., Paredes, R., and Orchard, M. (2011). Anomaly detection in power generation plants using similarity-based modeling and multivariate analysis. In *Proc. of the American Control Conference (ACC)*, pages 1940–1945.

[Took and Mandic, 2010] Took, C. C. and Mandic, D. P. (2010). A quaternion widely linear adaptive filter. *IEEE Trans. on Signal Processing*, 58(8):4427–4431.

[Vapnik, 1982] Vapnik, V. (1982). *Estimation of Dependences Based on Empirical Data*. Springer-Verlag.

[Vapnik and Lerner, 1963] Vapnik, V. and Lerner, A. (1963). Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24(774–780).

[Vapnik, 1995] Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag.

[Vapnik, 1999] Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999.

[Vía et al., 2010] Vía, J., Ramírez, D., and Santamaría, I. (2010). Properness and widely linear processing of quaternion random vectors. *IEEE Trans. on Information Theory*, 56(7):3502–3515.

[Ward, 1997] Ward, J. P. (1997). *Quaternions and Cayley Numbers: Algebra and Applications*. Kluwer Academic Publishers.

[Widrow and Hoff, 1960] Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits. *IRE WESCON Convention Record*, 4:96–104.

[Wiener, 1949] Wiener, N. (1949). *Extrapolation, interpolation, and smoothing of stationary time series*, volume 2. MIT press Cambridge, MA.

[Woodbury, 1950] Woodbury, M. A. (1950). Inverting modified matrices. Memorandum Rept. 42, Statistical Research Group, Princeton University.

[Xia et al., 2012] Xia, Y., Douglas, S., and Mandic, D. (2012). Adaptive frequency estimation in smart grid applications: Exploiting noncircularity and widely linear adaptive estimators. *IEEE Signal Processing Magazine*, 29(5):44–54.

[Yukawa, 2012] Yukawa, M. (2012). Multikernel adaptive filtering. *IEEE Trans. on Signal Processing*, 60(9):4672–4682.

Appendix A

Additional Material

A.1 Algebraic Identities for the Trace Operator

Identity: Tr $\left\{\mathbf{M}^T(\mathbf{v}\mathbf{u}^T)\right\} = \mathbf{v}^T\mathbf{M}\mathbf{u}$

Denote

- $\mathbf{v} = [v_1, \ldots, v_{N_v}]^T \in \mathbb{R}^{N_v}$.
- $\mathbf{u} = [u_1, \ldots, u_{N_u}]^T \in \mathbb{R}^{N_u}$.
- $\mathbf{M} \in \mathbb{R}^{N_v \times N_u}$, with entries $\{\mathbf{M}\}_{ij} = M_{ij}$.

We proceed by applying the explicit summation expression for the trace:

$$\text{Tr}\left\{\mathbf{M}^T(\mathbf{v}\mathbf{u}^T)\right\} = \sum_{i=1}^{N_u}\{\mathbf{M}^T(\mathbf{v}\mathbf{u}^T)\}_{ii} = \sum_{i=1}^{N_u}\sum_{j=1}^{N_v}\{\mathbf{M}^T\}_{ij}\{(\mathbf{v}\mathbf{u}^T)\}_{ji} = \sum_{i=1}^{N_u}\sum_{j=1}^{N_v} M_{ji}v_j u_i = \mathbf{v}^T\mathbf{M}\mathbf{u} \tag{A.1}$$

Identity: $\frac{\partial \mathbf{Tr}\left\{\theta^T\theta\mathbf{R}\right\}}{\partial\theta_{i,j}} = 2\sum_{r=1}^{n}\theta_{i,r}\mathbf{R}_{r,j}$, $\mathbf{R}$ symmetric.

Denote

- $\theta \in \mathbb{R}^{m\times n}$, with entries θ_{ij}.
- $\mathbf{R} \in^{n\times n}$, with entries $\mathbf{R}_{ij}$.

We proceed by applying the explicit summation expression for the trace and its linearity property:

$$\frac{\partial \mathrm{Tr}\left\{\theta^T \theta \mathbf{R}\right\}}{\partial \theta_{i,j}} = \frac{\partial \sum_{q=1}^{m}\sum_{r=1}^{n}\sum_{s=1}^{n}\theta_{q,r}\theta_{q,s}\mathbf{R}_{r,s}}{\partial \theta_{i,j}} = \sum_{q=1}^{m}\sum_{r=1}^{n}\sum_{s=1}^{n}\frac{\partial \theta_{q,r}\theta_{q,s}}{\partial \theta_{i,j}}\mathbf{R}_{r,s} \tag{A.2}$$
$$= \sum_{q=1}^{m}\sum_{r=1}^{n}\sum_{s=1}^{n}\left(\frac{\partial \theta_{q,r}}{\partial \theta_{i,j}}\theta_{q,s} + \theta_{q,r}\frac{\partial \theta_{q,s}}{\partial \theta_{i,j}}\right)\mathbf{R}_{r,s}$$

As $\frac{\partial \theta_{q,s}}{\partial \theta_{i,j}} = 1$ when $(q,s) = (i,j)$ and $\frac{\partial \theta_{q,s}}{\partial \theta_{i,j}} = 0$ otherwise, we have

$$\frac{\partial \mathrm{Tr}\left\{\theta^T \theta \mathbf{R}\right\}}{\partial \theta_{i,j}} = \sum_{s=1}^{n}\theta_{i,s}\mathbf{R}_{j,s} + \sum_{r=1}^{n}\theta_{i,r}\mathbf{R}_{r,j} = 2\sum_{r=1}^{n}\theta_{i,r}\mathbf{R}_{r,j} \tag{A.3}$$

where the last step is given by symmetry of $\mathbf{R}$.

A.2 Basis of Cubic Polynomials in $\langle \mathbf{x}, \mathbf{y} \rangle_{\Re}$

We show that the polynomials $\langle \mathbf{x}, \mathbf{y} \rangle_{\Re}^3$, $\big(1+\langle \mathbf{x}, \mathbf{y} \rangle_{\Re}\big)^3$, $\big(10+\langle \mathbf{x}, \mathbf{y} \rangle_{\Re}\big)^3$, $\big(100+\langle \mathbf{x}, \mathbf{y} \rangle_{\Re}\big)^3$ are a basis of the space of cubic polynomials in $\langle \mathbf{x}, \mathbf{y} \rangle_{\Re}$. For simplicity we denote $\Delta = \langle \mathbf{x}, \mathbf{y} \rangle_{\Re}$.

For $[\alpha, \beta, \gamma, \delta]^T \in \mathbb{R}^4$, we need to find $[a, b, c, d]^T \in \mathbb{R}^4$ such that

$$a\Delta^3 + b\big(1+\Delta\big)^3 + c\big(10+\Delta\big)^3 + d\big(100+\Delta\big)^3 = \alpha\Delta^3 + \beta\Delta^2 + \gamma\Delta + \delta. \tag{A.4}$$

Upon expanding the left-hand side of Eq. (A.4) and factorising it with respect to the basis $[\Delta^3, \Delta^2, \Delta, 1]$, we obtain

$$\begin{aligned} a\Delta^3 + b\big(1+\Delta\big)^3 + c\big(10+\Delta\big)^3 + d\big(100+\Delta\big)^3 = \\ &= a\Delta^3 + b\big(1 + 3\Delta + 3\Delta^2 + \Delta^3\big) \\ &\quad + c\big(10^3 + 300\Delta + 30\Delta^2 + \Delta^3\big) \\ &\quad + d\big(100^3 + 3\times 100^2\Delta + 300\Delta^2 + \Delta^3\big) \\ &= \Delta^3(a+b+c+d) + \Delta^2(3b + 30c + 300d) \\ &\quad + \Delta(3b + 300c + 3\times 100^2 d) + 1(b + 10^3 c + 100^3 d). \end{aligned} \tag{A.5}$$

A comparison of the right-hand sides of eqs. (A.4) and (A.5) gives the linear equation

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 3 & 30 & 300 \\ 0 & 3 & 300 & 3\times 100^2 \\ 0 & 1 & 10^3 & 100^3 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix}$$

with an invertible matrix on the left-hand side.

As a consequence, the proposed basis is a linearly-independent basis of the set of cubic polynomials in Δ (the independence property can be verified by $[\alpha, \beta, \gamma, \delta] = \mathbf{0} \Rightarrow [a, b, c, d] = \mathbf{0}$).

Other linearly-independent bases of real cubic polynomials on Δ were also considered, including $\{1, \Delta, \Delta^2, \Delta^3\}$ and $\{(b_0+\Delta)^3, (b_1+\Delta)^3, (b_2+\Delta)^3, (b_3+\Delta)^3\}$ for **different** parameters $b_0, b_1, b_2, b_3 \in \mathbb{R}^+$. We have found that the chosen basis (subkernels) provided the best results, in the MSE sense, in the prediction setting considered.

A.3 Quaternion Widely-Linear Ridge Regression

Strictly-linear models assume that the minimum mean square estimator (MMSE) $E\{\mathbf{x}|\mathbf{s}\}$ of a vector $\mathbf{x}$ given an observation vector $\mathbf{s}$ is, regardless of the real or quaternion nature of the vectors, given by[1]

$$\hat{\mathbf{x}} = \mathbf{A}\mathbf{s}, \tag{A.6}$$

where $\mathbf{A}$ is a coefficient matrix. On the other hand, widely-linear quaternion models exploit the linear dependency between the vector $\mathbf{x}$ and each of the components of the regressor $\mathbf{s} = \mathbf{s}_r + i\mathbf{s}_i + j\mathbf{s}_j + k\mathbf{s}_k$, yielding an estimator which is linear in each of these components.

Alternatively, by considering the *involutions* of $\mathbf{s}$, given by [Ell and Sangwine, 2007]

$$\begin{aligned}
\mathbf{s}^i &= -i\mathbf{s}i = \mathbf{s}_r - i\mathbf{s}_i + j\mathbf{s}_j + k\mathbf{s}_k \\
\mathbf{s}^j &= -j\mathbf{s}j = \mathbf{s}_r + i\mathbf{s}_i - j\mathbf{s}_j + k\mathbf{s}_k \\
\mathbf{s}^k &= -k\mathbf{s}k = \mathbf{s}_r + i\mathbf{s}_i + j\mathbf{s}_j - k\mathbf{s}_k,
\end{aligned}$$

we can express the widely-linear estimator in the form [Took and Mandic, 2010]:

$$\hat{\mathbf{x}} = \mathbf{A}\mathbf{s} + \mathbf{B}\mathbf{s}^i + \mathbf{C}\mathbf{s}^j + \mathbf{D}\mathbf{s}^k = \mathbf{W}\mathbf{s}^a,$$

where

$$\mathbf{W} = [\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}], \quad \mathbf{s}^a = \begin{bmatrix} \mathbf{s} \\ \mathbf{s}^i \\ \mathbf{s}^j \\ \mathbf{s}^k \end{bmatrix}$$

are the so-called *augmented* quantities.

[1] We have maintained the notation $\mathbf{s}$ and $\mathbf{x}$ for consistency with the nonlinear channel equalisation simulation.

This way, the widely-linear estimator is theoretically equivalent to the quadrivariate real-valued estimator and is the best linear estimator in $\mathbb{H}$ [Jahanchahi et al., 2010]. For complex widely-linear algorithms, see [Douglas, 2009, Mandic and Goh, 2009].

In the ridge-regression setting, the weights $\mathbf{W}$ are computed in the regularised least-squares sense based on a set of available observation pairs $\{(\mathbf{s}_n, \mathbf{x}_n), n = 1, ..., N\}$, that is

$$\mathbf{W} = (\mathbf{S}^H\mathbf{S} + \rho\mathbf{I})^{-1}\mathbf{S}^H\mathbf{X},$$

where $\rho > 0$ is a regularization factor, and $\mathbf{S} = [\mathbf{s}_1, ..., \mathbf{s}_N]$ and $\mathbf{X} = [\mathbf{x}_1^T, ..., \mathbf{x}_N^T]^T$ are the matrices of available observations.

The widely-linear ridge regression can also be regarded as an approximation of the widely-linear Wiener filter [Jahanchahi et al., 2010], where the correlation matrix and the autocorrelation vector are approximated using the available data.

A.4 Proof of Lemma 7

Lemma. *Let $K = K_r + iK_i + jK_j + kK_k$ be a quaternion kernel, then*

- **(a)** *K is Hermitian iff K_r is symmetric positive definite and $K_i = -K_i^T$, $K_k = -K_k^T$ and $K_k = -K_k^T$.*
- **(b)** *If K is Hermitian, K is positive definite iff the **real-valued** matrix representation[2] of its Gram matrix is positive definite.*

Proof. **(a)** The Hermitian condition of the quaternion kernel $K = K^H$ can be expressed in terms of its real an imaginary parts as

$$K_r + iK_i + jK_j + kK_k = K_r^T - iK_i^T - jK_j^T - kK_k^T.$$

Therefore, since $\{1, i, j, k\}$ is a linearly-independent basis of $\mathbb{H}$, the above relationship is true if and only if the corresponding terms are equal coordinate-wise, that is:

$$K_r = K_r^T, K_i = -K_i^T, K_j = -K_j^T, K_k = -K_k^T.$$

(b) Let us now re-state the matrix definition (Definition 3) of positive definiteness $\mathbf{v}^H\mathbf{K}\mathbf{v} > 0, \forall \mathbf{v} \in \mathbb{H}^m \setminus \{0\}$ as

$$\Re\{\mathbf{v}^H\mathbf{K}\mathbf{v}\} > 0 \tag{A.7}$$

$$\Im\{\mathbf{v}^H\mathbf{K}\mathbf{v}\} = 0. \tag{A.8}$$

[2] See [Ward, 1997, Page 91] for the real matrix representation of quaternions.

where $\mathbf{K} \in \mathbb{H}^{m\times m}$ is the Gram matrix corresponding to an arbitrary collection of m points $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset X$. Observe that Eq. (A.7) is a necessary and sufficient condition for a Hermitian quaternion kernel $\mathbf{K}$ to be positive definite, since (A.8) always holds due to the Hermitian property of the kernel:

$$2\Im\{\mathbf{v}^H\mathbf{K}\mathbf{v}\} = \mathbf{v}^H\mathbf{K}\mathbf{v} - (\mathbf{v}^H\mathbf{K}\mathbf{v})^H = \mathbf{v}^H\mathbf{K}\mathbf{v} - (\mathbf{v}^H\mathbf{K}\mathbf{v}) = 0.$$

To analyse (A.7) in terms of real and imaginary parts of the quaternion kernel, we expand the vector $\mathbf{v} = \mathbf{v}_r + i\mathbf{v}_i + j\mathbf{v}_j + k\mathbf{v}_k$ and the kernel matrix $\mathbf{K} = \mathbf{K}_r + i\mathbf{K}_i + j\mathbf{K}_j + k\mathbf{K}_k$ using their real and imaginary parts and rewrite Eq. (A.7) as

$$\begin{aligned}\Re\{\mathbf{v}^H\mathbf{K}\mathbf{v}\} = {} & \mathbf{v}_r^T\mathbf{K}_r\mathbf{v}_r + \mathbf{v}_i^T\mathbf{K}_r\mathbf{v}_i + \mathbf{v}_j^T\mathbf{K}_r\mathbf{v}_j + \mathbf{v}_k^T\mathbf{K}_r\mathbf{v}_k \\ & + 2\mathbf{v}_i^T\mathbf{K}_i\mathbf{v}_r + 2\mathbf{v}_j^T\mathbf{K}_j\mathbf{v}_r + 2\mathbf{v}_k^T\mathbf{K}_k\mathbf{v}_r \\ & + 2\mathbf{v}_i^T\mathbf{K}_j\mathbf{v}_k + 2\mathbf{v}_j^T\mathbf{K}_k\mathbf{v}_i + 2\mathbf{v}_k^T\mathbf{K}_i\mathbf{v}_j > 0.\end{aligned} \tag{A.9}$$

Observe that Eq. (A.9) can be written as a quadratic positive-definite form $\vec{\mathbf{r}}_\mathbf{v}^T\mathbf{Q}\vec{\mathbf{r}}_\mathbf{v} \geq 0$, where

$$\vec{\mathbf{r}}_\mathbf{v} = \begin{pmatrix}\mathbf{v}_r \\ \mathbf{v}_i \\ \mathbf{v}_j \\ \mathbf{v}_k\end{pmatrix}, \mathbf{Q} = \begin{pmatrix}\mathbf{K}_r & -\mathbf{K}_i & -\mathbf{K}_j & -\mathbf{K}_k \\ \mathbf{K}_i & \mathbf{K}_r & -\mathbf{K}_k & \mathbf{K}_j \\ \mathbf{K}_j & \mathbf{K}_k & \mathbf{K}_r & -\mathbf{K}_i \\ \mathbf{K}_k & -\mathbf{K}_j & \mathbf{K}_i & \mathbf{K}_r\end{pmatrix}$$

are respectively an $\mathbb{R}^{4m}$ representation of $\mathbf{v}$ and the real-valued matrix representation of the Gram matrix of K.

We have therefore proved that the positive definiteness condition of a quaternion kernel can be verified through the positive definiteness of its real-valued matrix representation and vice versa.

□