Escuela Técnica Superior de Ingeniería
Departamento de Ingeniería de Sistemas y Automática

Doctoral Thesis

Probabilistic data-driven methods for forecasting, identification and control

Alfonso Daniel Carnerero Panduro

Supervised by:
Daniel Rodríguez Ramírez
Teodoro Álamo Cantarero

Seville, September 2022

Contents

Acknowledgements

Me voy a tomar la libertad de escribir esta sección en castellano debido a que algunas de las personas a las que va dirigida no tienen un conocimiento fluido de la lengua de Shakespeare y, por tanto, todo este esfuerzo podría resultar en vano. Además que, despues de escribir numerosas páginas en riguroso inglés, no está de más volver a las propias raíces por un tiempo.

Han pasado aproximadamente 9 años desde que entré a estudiar en la escuela de ingenieros y, hasta cierto punto, parece como si fuera ayer (lo cual es probablemente el tópico más usado de la historia). El cómo llegué aquí me parece, en cierto sentido, pura casualidad, pues durante mi más tierna infancia y adolescencia, mis intereres laborales y profesionales variaron con frecuencia. En cualquier caso, creo que debo gran parte de mi interés a la ingeniería y al control a Antonio Nuevo y a Juanma Escaño. Si no hubiera frecuentado aquel taller de automatización en el Colegio Altair cuando me encontraba en Primaria, probablemente mi trayectoria hubiera sido distinta.

Por otro lado, debo a mi hermano José María mi interés por la carrera investigadora, la cual él comenzó mucho antes que yo en otra facultad de esta misma universidad. Por algún motivo, pensé que se trataba de un trabajo fascinante con una calidad de vida bastante buena. En otro orden de cosas, mi padre resultó ser un gran apoyo durante mis primeros pasos en la escuela, debido a las experiencias similares que había sufrido durante su juventud. En cuanto a mis hermanas, Ana y Guadalupe, les debo el no vestir como un pordiosero (lo cual no es poco). Bueno, algo más habrá, pero eso es lo primero que se me ha venido a la mente. Para mi madre, sólo se me ocurre decir que es única en todos los sentidos.

También me gustaría agradecer especialmente a Sergio Lucía por su acogida en la Universidad Técnica de Dortmund durante mi estancia de doctorado. Fue una de esas experiencias positivas en las que uno descubre que el resto del mundo no es como Sevilla (quien tenga oídos que oiga).

Por supuesto, y como no podía ser de otra manera, también agradezco la labor que han tenido mis directores Dani y Teo en este trabajo y por la formación recibida a lo largo de todo este tiempo. Aunque nuestros caminos se separen temporalmente, espero que vuelvan a juntarse en un futuro cercano. También, me gustaría agradecer especialmente a Dani Limón por la oportunidad que me brindó en su momento. Gracias a eso, pude empezar a trabajar en este grupo de investigación. A su vez, me gustaría agradecer a los compañeros de doctorado que me ayudaron en mis comienzos, especialmente a Pablo y Pepe. De manera similar, tengo también un hueco particular para mis compañeros de cafés y/o desayunos Joaquín y José Antonio.

Por otra parte, existen a su vez gran cantidad de personas que han contribuido

infinitesimalmente a convertirme en lo que soy a día de hoy y a las que, por tanto, estoy muy agradecido. No quiero realizar una enumeración exhaustiva porque eso resulta sumamente peligroso (la probabilidad de olvidar a alguien es bastante alta y no es directamente proporcional a la importancia de la persona en cuestión). Por tanto, si estás leyendo esta aburrida sección, puedes concluir sin temor a equivocarte que eres una de esas personas.

Por último, y no por ello menos importante, me gustaría dedicar esta tesis a mis otros dos hermanos, Juanma y Alberto, los cuales no creo que tengan un interés especial por esta dedicatoria. En cualquier caso, ahí queda dicho.

Alfonso Daniel Carnerero Panduro,
Sevilla, septiembre de 2022.

Abstract

This dissertation presents contributions mainly in three different fields: system identification, probabilistic forecasting and stochastic control.

Thanks to the concept of dissimilarity and by defining an appropriate dissimilarity function, it is shown that a family of predictors can be obtained. First, a predictor to compute nominal forecastings of a time-series or a dynamical system is presented. The effectiveness of the predictor is shown by means of a numerical example, where daily predictions of a stock index are computed. The obtained results turn out to be better than those obtained with popular machine learning techniques like Neural Networks.

Similarly, the aforementioned dissimilarity function can be used to compute conditioned probability distributions. By means of the obtained distributions, interval predictions can be made by using the concept of quantiles. However, in order to do that, it is necessary to integrate the distribution for all the possible values of the output. As this numerical integration process is computationally expensive, an alternate method bypassing the computation of the probability distribution is also proposed. Not only is computationally cheaper but it also allows to compute prediction regions, which are the multivariate version of the interval predictions. Both methods present better results than other baseline approaches in a set of examples, including a stock forecasting example and the prediction of the Lorenz attractor.

Furthermore, new methods to obtain models of nonlinear systems by means of input-output data are proposed. Two different model approaches are presented: a local data approach and a kernel-based approach. A kalman filter can be added to improve the quality of the predictions. It is shown that the forecasting performance of the proposed models is better than other machine learning methods in several examples, such as the forecasting of the sunspot number and the Rössler attractor. Also, as these models are suitable for Model Predictive Control (MPC), new MPC formulations are proposed. Thanks to the distinctive features of the proposed models, the nonlinear MPC problem can be posed as a simple quadratic programming problem. Finally, by means of a simulation example and a real experiment, it is shown that the controller performs adequately.

On the other hand, in the field of stochastic control, several methods to bound the constraint violation rate of any controller under the presence of bounded or unbounded disturbances are presented. These can be used, for example, to tune some hyperparameters of the controller. Some simulation examples are proposed in order to show the functioning of the algorithms. One of these examples considers the management of a data center. Here, an energy-efficient MPC-inspired

policy is developed in order to reduce the electricity consumption while keeping the quality of service at acceptable levels.

Notation, conventions and definitions

Assuming that we have a column vector $v \in \mathbb{R}^n$, the transpose of vector v is denoted as $v^\top$. The subscript k in v_k denotes the value of vector v at time k. When the subscript is enclosed by parenthesis, i.e. $v_{(i)}$, it refers to the i-th element of vector v. Also, $\|v\|$ stands for the euclidean norm and $\|v\|_A$ stands for the euclidean norm weighted by a positive definite matrix A, that is, $\sqrt{v^\top A v}$.

A past sample of vector v is denoted as $\bar{v}$. When multiple samples of v are gathered, $\bar{v}_i$ refers to the i-th sample of v. On the other hand, $\tilde{v}$ stands for a prediction or estimation of v whereas $\hat{v}$ refers to a corrected version of $\tilde{v}$ after new information is available and some filtering is done (i.e. Kalman filtering). Similarly, $v_{i|k}$ corresponds to the prediction of v_{k+i}, e.g. $\tilde{v}_{k+i}$. However, $v_{i|k}$ also implies that the prediction is done at time instant k. Furthermore, $\breve{v}$ stands for a change of variables so that $\breve{v} = v - v_s$ where $v_s \in \mathbb{R}^n$ is a constant vector.

Suppose that $\mathcal{V}$ is a continuous set with infinite cardinality, it is possible to approximate this set with a discretized version of it denoted as $\breve{\mathcal{V}}$. Then, the members of this discretized set $\breve{\mathcal{V}}$ will be denoted as $\breve{v}_i$.

$\mathbf{1}$ and $\mathbf{0}$ stand for row vectors of appropriate dimensions whose elements are all equal to one or zero respectively. When a unique subscript appears, i.e. $\mathbf{0}_n$, it refers to a squared matrix of n dimensions. When two subscripts are present, i.e. $\mathbf{0}_{m\times n}$, it refers to a non-square matrix of dimensions $m \times n$. Also, the identity matrix of order n is denoted as $\mathrm{I_n}$.

Chapter 1

Introduction

1.1 Motivation and objectives

In this chapter, the main motivations and objectives of this dissertation are presented. First, some existing problems are introduced by means of different literature reviews. Once all the problems to be treated in this thesis are presented, the objectives of this dissertation are exposed, followed by the proposed solutions and obtained results.

1.1.1 The problem of system identification

System identification refers to the science of building mathematical models of the dynamical systems around us by means of input-output data [1]. These kinds of problems are of great importance within the control community since these models are used to design controllers that make the systems operate as we want them to. For this reason, this has been a very important research topic since the beginning of the control theory.

Depending on the previous knowledge of the system that we want to model, we consider different model categories. A first principles model could be initially considered. This is due to the fact that there are many situations where the dynamics of the system to be modelled are widely known (some mechanical or electrical systems, etc) [2]. Also, in the case of linear systems, a transfer function mapping the input of the system to the output can be obtained by applying the Laplace transform to the set of Ordinary Differential Equations (ODEs).

However, assuming that the set of equations governing the system are known is quite unrealistic in the general case. Even if the structure of the model is known, the parameters of the ODEs may be unknown in advance. Then, a series of experiments should be carried out in order to obtain the values of these parameters. This would correspond to the problem of parameter estimation or grey-box modeling [3, 4, 5]. These parameters can be estimated, for example, using the least

squares method. We notice that this methodology can be used even if the first-principles model is not linear as it suffices that the model is linear with respect to the parameters. Thus, for example, consider the model of a water tank

$$A\frac{\mathrm{d}h}{\mathrm{dt}} = -a\sqrt{h},$$

where h is the height of the tank, A is the cross-sectional area of the tank and a is a constant related to the flow out of the tank. Note that, here, the parameters appear linearly in the equations and, thus, the aforementioned method can be applied. The real experimental data will probably contain *noise*, transforming the estimation of these parameters into an uncertain problem. That is, the real parameters cannot be exactly obtained, but a set of parameters that provides small forecasting errors can be computed.

However, in most cases, the underlying dynamics of the system may be completely unknown. In these cases, a black-box model approach would be more suitable. Black-box models represent the external dynamic function of the system, i.e., the relation of the present output with past values of the input and past values of the output. These values can be grouped in a vector called the regressor,

$$r_k = \begin{bmatrix} y_{k-1}^\top & y_{k-2}^\top & \cdots & y_{k-n_y}^\top & u_{k-1}^\top & u_{k-2}^\top & \cdots & u_{k-n_u}^\top \end{bmatrix}^\top,$$

where y_k and u_k are the outputs and the inputs of the system at time instant k respectively, n_y is the number of past outputs and n_u is the number of past inputs. Now, for example, we could consider a Least-Squares Estimator [6] by solving the problem

$$\theta^* = \arg\min_\theta \sum_{i=1}^{N} \left(\bar{y}_i - \tilde{y}_i(\theta)\right)^\top \left(\bar{y}_i - \tilde{y}_i(\theta)\right),$$

where θ is a matrix of parameters, $\bar{y}_i$ is a certain sample obtained from a training set of cardinality N and $\tilde{y}_i(\theta) = \bar{r}_i^\top \theta$ is the prediction of $\bar{y}_i$ given θ. This estimator is known to provide good results in the linear case under simple identifiability conditions [1]. When the dimension of the system is much larger than the number of samples, Dynamic Mode Decomposition (DMD) techniques can be applied in order to tackle the problems that arise from working with high dimensional data [7, 8, 9].

These methods offer good results when identifying linear systems but, linear systems are only a small subclass of the systems that appear in control problems. Nonlinear system identification arises as a much harder problem [10, 11]. This is mainly due to the fact that there are countless different possible model structures and, sometimes, it is not possible to find a perfect answer to the identification problem. On the other hand, it is also a more interesting field because it allows us to model more complex systems.

First, for the sake of simplicity, we could assume that the system behaves as a linear-time-varying (LTV) system, that is, the system is linear but its parameters change over time. This kind of model, although simple, can lead to good results [12, 13]. Related to this, we could consider the system as an interpolation of different linear models, taking into account some nonlinear rules. This method leads to Takagi-Sugeno models based on fuzzy logic, which are known to be universal approximators [14, 15].

Another option would be to rely on Hammerstein-Wiener models [16, 17]. These models are compounded of a nonlinear block mapping the input, then a linear transfer function and, finally, another nonlinear block that maps the signal into the real output (see figure 1.1).

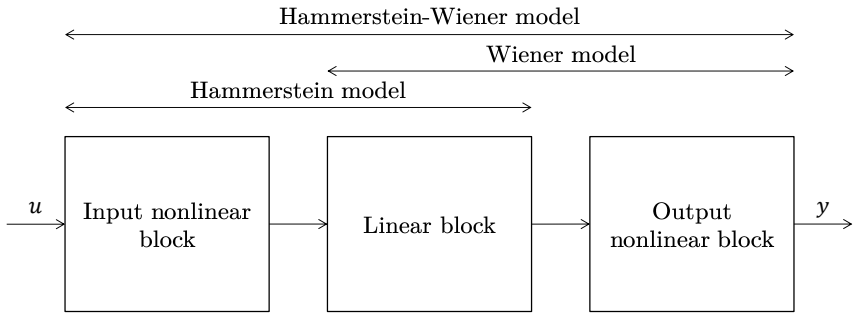


Figure 1.1: Hammerstein-Wiener models.

On the other hand, the nonlinearities of a system could be modeled as a Volterra series [18, 19]. These models are similar to a Taylor series expansion. However, a Taylor series expansion does not have any memory and thus dynamic systems cannot be considered. In this sense, Volterra series are the generalization of the Taylor series so that they can consider dynamic systems.

Furthermore, a kind of nonlinear autoregressive exogenous model (NARX) models can be used [20, 21]. Here, the outputs of the model are computed by means of a nonlinear function of the linear regressor, i.e. a wavelet network [22].

Also, Machine Learning (ML) and other data-driven techniques have been gaining increasing attention within the control field lately. For example, Gaussian Processes (GPs) [23, 24, 25] or Lipschitz Interpolation (LI) [26, 27] have been used in system identification and control. Other well-known techniques are neural networks [28, 29] and, more recently, deep neural networks [30, 31]. They are a popular tool in system identification because they are known for being able to reproduce any nonlinear function. However, it is not easy to find the right type of network, training algorithm or structure, leading to wrong results or overfitting in many cases [32]. As an alternative, one could rely on Reservoir Computing [33, 34] or Echo State Networks [35, 36] approaches which are easier to train.

Combinations of some of the aforementioned methods have been explored as well, i.e. neuro-fuzzy methods [37, 38] combining Takagi-Sugeno models and Neural Networks. Recently, another technique that is getting increasing attention is the Koopman operator [39, 40], where the nonlinear dynamics of the system are converted into linear dynamics in a infinite-dimensional state vector. Although the system becomes infinite-dimensional, it is possible to approximate them by a sufficiently large state-vector and apply methods from linear control theory.

All of these methods assume that the dynamics of the system are completely unknown. However, it may exist situations where some partial knowledge of the system can be used alongside some black-box model. This leads us to the hybrid modeling framework [41, 42]. Here, the outputs are computed as the sum of two terms, one corresponding to the known dynamics and another one corresponding to the unknown dynamics. Note that this is very different to the grey-box modeling approach mentioned before.

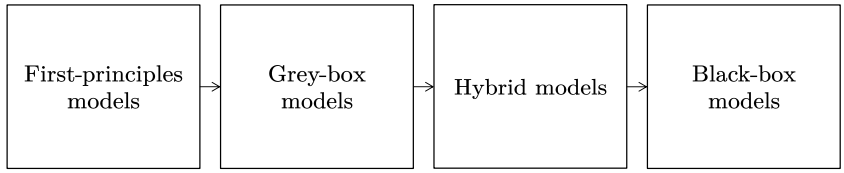


Figure 1.2: From first-principles models to black-box models.

1.1.2 Quantifying the uncertainty

However, sometimes we are interested not only in obtaining nominal predictions of a certain system, but also a quantification of their uncertainty. That is, when the considered system presents noisy measurements and/or additive disturbances, it would be useful to characterize a region where the real output of the system may be with a specified probability instead of only computing the expected value.

First, we focus on the univariate case, i.e. interval predictions. Given the regressor r_k, the objective is to compute an interval $I(r_k) = [y_k^-, y_k^+]$ such that we maximize the probability that y_k belongs to $I(r_k)$ while minimizing the interval width $(y_k^+ - y_k^-)$. These two conflicting objectives can be reconciled if one minimizes the interval width subject to the constraint that $I(r_k)$ contains y_k with a pre-specified probability.

Interval predictions play a relevant role in the control of uncertain systems. Zonotopes and DC Programming are used to obtain interval state estimators in [43] and [44] respectively. Interval observers for linear time-varying systems have been proposed in [45] and [46]. Fault detection methods based on zonotopic bounds

can be found in [47]. In [48], set theoretic approaches are also used in the context of fault detection. Set membership methods [49, 50] can also be used to obtain interval predictions. A mixed Bayesian/set-membership approach is proposed in [51].

There exists different methods in the literature that address the problem of obtaining interval predictions. For example, if the vector of uncertainty is bounded and the considered system satisfies some Lipschitz assumptions, one can resort to bounded error methods [52] that guarantee that y_k is always contained in $I(r_k)$. See, for example, [53] and [27]. Other bounded error strategies have been proposed in [54, 55, 56]. The statistical characterization of noise and disturbances can be used to enhance the performance of interval estimation methods. See [57, 58, 59] and references therein. Also, probabilistic validation methods can be used to assess the performance of the interval predictors [60, 61, 62, 63].

An important concept is that of quantiles [64, 65]. Denote $F_{y_k}(a|r_k)$ as the cumulative distribution function of the associated output y_k conditioned to the regressor r_k, that is,

$$F_{y_k}(a|r_k) = \text{Prob}\{y_k \leq a \mid r_k\},$$

where a is a scalar. Given r_k, we say that a_τ is the conditioned τ-quantile if

$$F_{y_k}(a_\tau|r_k) = \text{Prob}\{y_k \leq a_\tau \mid r_k\} = \tau.$$

The estimation of the conditioned quantiles is relevant in multiple applications (see [66] and [67]) and can be addressed using different methodologies. The most classical approach relies on the assumption that y_k and r_k are jointly normal. That is, the assumption that the (joint) probability density function of the (random) variables y and r is a multivariable normal probability density function. Under this assumption, the conditioned p.d.f. is a monovariable normal p.d.f. and the quantiles can be obtained in a simple and direct way [68]. Unfortunately, the methods based on normal distributions are very sensitive to the presence of outlier contamination. Moreover, in many long-tailed distributions, the normal assumption is not well suited to characterize confidence intervals and one has to resort to non-Gaussian distributions. In these cases, generalizations of the Chebyshev inequality can be used to obtain probabilistic bounds [69, 70].

The computation of the conditioned quantiles can be also addressed by means of parametric regression techniques [65], [66]. Assuming that there exists θ for which $y_k \approx \theta^\top r_k$, then parameter vector θ can be chosen as the one that minimizes a cost function of the error $\theta^\top r_k - y_k$. If one chooses a cost function that penalizes in an asymmetric way positive and negative errors then a quantile regressor is obtained. Given the training pairs $(\bar{y}_i, \bar{r}_i)$, $i = 1, \ldots, N$ and $\tau \in (0, 1)$, the quantile regressor is defined in terms of the following optimization problem

$$\min_\theta \sum_{i=1}^{N} (1-\tau)\max\{0, \theta^\top \bar{r}_i - \bar{y}_i\} + \tau \max\{0, \bar{y}_i - \theta^\top \bar{r}_i\}.$$

This linear optimization problem penalizes the (training) errors $e_i = \theta^\top \bar{r}_i - \bar{y}_i$, $i = 1, \ldots, N$ in an asymmetric way. The positive errors are weighted with coefficient $(1-\tau)$ and the negatives with coefficient τ. If $\tau \in (0,1)$ is close to zero, then the positive errors will be highly penalized (in comparison with the negative ones). This means that every optimal solution θ_τ to the linear optimization problem will tend to make most of the errors negative. This implies that $\theta_\tau^\top r_k$ could be used as a probabilistic lower bound for y_k. In a similar way, a probabilistic upper bound could be obtained taking $\tau \in (0,1)$ close to 1. Under rather mild assumptions, any minimizer θ_τ of the proposed optimization problem can be used to obtain an estimation of the τ quantile. That is, $\theta_\tau^\top r_k$ serves as an estimation of the τ quantile associated with y_k. See [65], [71] and [66] for further details.

One of the main limitations of quantile regression is that a large number of training samples N is required if one desires to obtain probabilistic guarantees of the method when τ is chosen close to the extremes of the interval $(0,1)$. This is due to the fact that estimating the probability of rare events requires a large number of samples. For example, the number of independent identically distributed samples required to obtain the $1-\epsilon$ quantile of a monovariable random variable grows with $\frac{1}{\epsilon}$ (see [72], [61] and [62]).

Comparatively, computing multivariate prediction regions becomes a harder task. The simplest way would be to obtain intervals considering each variable independently and then construct box-shaped regions (see section 2.2.3 in [73]). The main advantage of this method lies in its simplicity. However, the desired probability may not be attained or the size of the regions may be too large. Bootstrap methods are used in [74, 75] to construct regions that contain a path of a random variable with at least a certain probability. This means that the obtained regions are actually intervals for a $p-$step prediction instead of a multivariate system. In a similar manner, [76, 75] calculate prediction regions for Vector Auto Regression (VAR) models in the field of econometrics. Following similar methodologies, [77] provides intervals for an entire path of forecasts in Markov processes. The problem of refining previously estimated prediction regions in order to attain a coverage probability closer to the desired one is researched in [78] and [79]. However, all these aforementioned techniques do not deal with multivariate output systems. Instead, the computed regions corresponds to the prediction of the same variable for different time steps.

In the field of multivariate output systems, the literature is rather scarce. For example, in [80], prediction regions for a simple multivariate linear regression model are obtained. On the other hand, [81] proposes a method to calculate prediction regions of a system by computing the Jacobian of the Partial Least-Squares Regression (PLS) parameters. Thus, by means of this local linearization, an ellipsoid-shaped region can be obtained. Also, [82] manages to obtain ellipsoidal regions for dynamical systems by means of an Inverse Regression (IR) scheme. This IR scheme is more efficient and reliable than the classic regression approaches when

high dimensional data is available. In [83], a data-driven framework to generate and evaluate ellipsoidal prediction regions to characterize the uncertainty of a time-series is proposed. This methodology is applied to the electricity prices as a path forecasting problem. In the field of machine learning, Conformal Prediction techniques [84] are used to obtain prediction regions for any method producing a central prediction of the outputs. Similarly, it is proposed in [85] a framework to obtain a stochastic model based on a deterministic model of a dynamic system in the field of robotics. On the other hand, assuming that any finite set of samples follows a multivariate normal distribution, Gaussian processes (GPs) [23] can also be used. The main limitation of this approach is that it relies heavily on the knowledge of the first two moments of the underlying multivariate probability distribution.

1.1.3 Model predictive control

Once reached this point, it is important to recall that the purpose of finding a good model of a dynamic system is because we want to induce a certain behaviour to these systems, that is, we want to control them. In this dissertation, Model Predictive Control (MPC) [86, 87] has been chosen as the control algorithm to be used in the proposed control examples.

MPC corresponds to a set of computer control techniques sharing some common ideas. Its development exploded at the end of the 1970s thanks to the works of [88, 89], receiving attention from both academia and industry. In these works, they used predictions obtained by means of finite impulse response models or truncated step response models to obtain optimal control actions while minimizing the tracking error of the output of the system. However, most of these algorithms had an heuristic nature, they were not able to deal with disturbances and lacked stability guarantees.

This first generation of predictive controllers was followed by the Generalized Predictive Control (GPC) algorithm [90, 91] which comprised these previous controllers based on input-output models. On the other hand, the state-space interpretation of MPC [92, 93] appeared, and quickly became the standard formulation in MPC. A good review of some properties of the MPC controllers, including stability can be found in [94].

The main ideas around MPC are:

- MPC is an optimal controller in the sense that the input is calculated such that it minimizes a certain cost function, mainly penalizing the tracking error with respect to the desired reference and the control effort. This error can be computed thanks to a mathematical model of the system, which leads us to the following point.
- As the name suggests, an accurate mathematical model describing the behaviour of the system is needed. It is possible to use whether input-output

models or state-space models. Because of the numerical nature of the controller (i.e. an optimization problem is solved to compute the value of the input instead of obtaining an explicit control law), discrete-time models are usually used. Furthermore, as it is easier to guarantee the stability of the controller using state-space models, these are also widely used.

- Another interesting point of the MPC controllers is their ability to tackle constrained control problems easily. As the input is obtained by solving an optimization problem, it is very natural to add constraints. These constraints may reflect limited capability in the actuators, some physical limits of the system or even tackle security considerations.

- The last idea would be the receding horizon scheme. This means that even though the value of the input is computed for many future time instants, only the first one (that is, the one corresponding to the actual time instant) is applied, discarding the rest. Then, at the next sampling step, a new whole set of inputs is computed taking into account the new available information. Note that, if we applied the whole set of values computed at the first time instant, this would be open-loop control. Thus, the receding horizon scheme provides feedback to the controller.

From these properties of the MPC controllers, it is easy to see the many advantages that they present, i.e.

- The control problem is formulated in the time domain, in a flexible and intuitive manner in contrast to other control formulations that need to be designed in the frequency domain. Also, it is applicable to any system without regard to its open-loop stability, delays, etc.

- In the most general case, it allows to consider linear and nonlinear, univariate and multivariate systems alike using the same formulation of the controller.

- It presents delay compensation explicitly and measurable disturbances can also be easily compensated.

- It is possible to take into account the knowledge of the evolution of the reference in order to track the signal to the reference before it changes.

- It is possible to deal easily with constraints.

However, there are also some drawbacks

- A sufficiently precise model of the system to be controlled is needed. That is, the performance of the controller depends heavily on the quality of the model. An inappropriate model of the system can lead to undesired performance of the controller in closed-loop operation.

- It is necessary to solve an optimization problem at every time instant k. Even though nowadays the computational power of the computers have

been increasing enormously, many nonlinear MPC (NMPC) or robust MPC problems still remain intractable in practice due to the high computational cost of the algorithms.

As it was discussed previously, it is very common that the dynamics of the process to control are almost completely unknown, forcing us to rely on black-box models. Anyway, in most of these cases, there exists an appropriate model structure in the literature that is able to reflect the dynamics of the system and thus finding a good model is not an impossible task.

On the other hand, with respect to the computational cost, there are several ways to address these problems, like developing specific optimization algorithms to solve MPC problems [95, 96] or resorting to sub-optimal MPC schemes [97, 98]. Another way would be to rely on explicit MPC implementations [99, 100]. Here, an analytical solution of the MPC controller can be obtained if the optimization problem can be casted as a certain class of quadratic programming (QP) problems [101, 102]. The obtained control law is piece-wise linear, that is, different linear state feedback control policies are obtained for different polyhedral regions. However, the number of regions grows up largely with respect to the number of states, constraints, etc. becoming intractable for many MPC problems.

Besides the aforementioned classic MPC schemes, a huge number of MPC variants have been researched over time, i.e. approaches which considers disturbances and model mismatches. Actually, the classic deterministic MPC approach presents some inherent robustness [103, 104], which can be studied by means of the Input-to-State stability (ISS) [105] or robustly asymptotically stability [106] schemes. However, it is not possible to guarantee that the constraints are fulfilled in a robust manner. For that purpose, a description of the uncertainties is necessary. This led to min-max MPC, where the worst-case values of the disturbances were taken into account to compute the control actions [107, 108]. One step further was to optimize feedback control laws instead of control actions to improve the robustness of the controller [109]. The main drawback of this subclass of MPC controllers is that their online computational time is incredibly high, sometimes becoming intractable, especially when optimizing feedback control laws.

Since then, numerous robust MPC schemes have been proposed. The most important one is probably the tube-based MPC [110, 111, 112]. Here, a nominal control action is computed alongside a proportional policy. As the policy is assumed to be a proportional gain, the optimization problem is not infinite dimensional and thus it is generally tractable. Also, one could rely on constraint tightening schemes [27] to fulfill the constraints under the presence of any kind of disturbance.

However, robust MPC is, in general, very conservative because it addresses always the worst case, that is, it is always considering the extreme values of the disturbances. For that reason, stochastic MPC [113, 114] and scenario approaches [115, 116] which considers the probability distribution of the disturbances have

attracted the attention of many researchers. These techniques allow us to develop controllers that fulfill the constraints with a specified probability, i.e. relaxing the worst case optimization of robust MPC. For example, in the scenario approach, this can be done by taking a finite number of realizations of the disturbance during the online computation of the control action. In this approach, however, the number of scenarios to be generated increases with the dimension of the problem, leading to unaffordable computational times in many cases.

As most of these approaches complicate the online optimization problem to be solved online, one could wonder if it is possible to obtain a stochastic MPC controller while keeping the online optimization problem as simple as the deterministic MPC problem. This leads us to probabilistic validation approaches [117, 118] where, by means of offline simulations of the closed-loop system, it is possible to determine if the controller fulfills some probabilistic specifications. This means that the computational burden is outside the control loop and thus the online optimization problem can be solved easily. Also, it can be applied to any controller, even if it is not an MPC controller. However, this method only provides a "yes" or "no" answer without assessing numerically the performance of the controller, which makes it hard to compare with other controllers. In some cases, it could even lead to non feasible solutions if none of the controllers satisfies the probabilistic constraint.

Other popular variants of the standard MPC controller are the following:

- Economic MPC [119, 120] allows to optimize different cost functions in order to improve the profitability of a certain process. The stability results of standard MPC are not, in general, applicable for economic MPC. However, under some assumptions [121], it was proven that it is possible to find a Lyapunov function of the closed-loop system in order to ensure stability.

- Distributed MPC [122, 123] considers a decentralized control system, reducing the computational burden of the individual optimization problems and becoming more scalable and robust to failures, which may be helpful when controlling large-scale systems. However, the performance may be worse than the performance of the centralized controller.

- MPC for tracking [124, 125] tackles the problem of arbitrarily changing the reference signal of the controller. This approach preserves stability and guarantees that the system can be steered to any admissible equilibrium point, no matter what the initial equilibrium point is, without losing the recursive feasibility. This strategy also provides a larger domain of attraction for a given prediction horizon and, in the case of unreachable references, asymptotic convergence to the closest reachable reference is proven as well, only at the expense of adding a few more decision variables.

1.1.4 Objectives of this dissertation

Many different research fields have been presented throughout this introductory chapter: system identification, interval predictions, MPC, etc. This dissertation is oriented towards developing new methods that may improve the performance of the state-of-the-art methods in each one of the previously presented fields. Summarizing, the objectives of this dissertation are:

1. Developing new forecasting schemes for time-series and nonlinear systems. It is interesting to obtain not only the expected output but also a region where the output can be found with a certain probability.

2. Proposing new data-driven black-box models to be used in MPC strategies. As the proposed models may reflect better the behaviour of certain systems, they might increase the performance of certain controllers.

3. Obtaining probabilistic bounds on the constraint violation level for any kind of controller in a stochastic setting. These bounds could be used to tune the hyper-parameters of a controller in a soft-constraint approach.

Taking into account these objectives, the following results were obtained:

1. Corresponding to the first objective, some predictors based on dissimilarity functions are proposed. These include nominal predictions, interval predictions and prediction regions. All these methods were tested against some baseline techniques in many examples and showed a better performance.

2. For the second objective, it is shown that, by means of the aforementioned dissimilarity function, it is possible to identify a model from the available input-output data. Through different numerical examples, it can be seen that the model performs better than other machine learning techniques. The model has been used within a MPC for tracking framework to show the effectiveness of the approach in control problems.

3. Finally, two different sharp bounds on the constraint violation rate were obtained. These bounds were used in the context of data center energy optimization to quantify the quality of service provided to the users, proving its usefulness.

1.2 Outline

All the results of this dissertation converge to MPC at some point, that is, they are useful to improve the performance of MPC controllers or they allow to develop new MPC schemes that may attain better results than other state-of-the-art MPC controllers. For the sake of simplicity, the contributions are divided into three groups: novel algorithms for prediction and forecasting, new modeling methods

for controller design and, finally, randomized algorithms for stochastic control. Having this idea in mind, the text was divided into three different parts.

Part I presents the developed predictors based on dissimilarity functions. These proposed methodologies are related to the ones appearing in the context of Direct Weight Optimization [57] and the Kriging method [126, 127]. First, the concept of dissimilarity function is exposed, along with the proposed dissimilarity function for this dissertation. From that, a nominal predictor can be easily obtained. Given a regressor, a prediction is obtained as a combination of past outputs of the system using some weights that are obtained by means of an optimization problem. After that, a method to tackle the problem of uncertainty quantification is proposed. Again, by means of the aforementioned dissimilarity functions, it is shown that it is possible to obtain interval predictors for the univariate case and prediction regions for the multivariate case.

Part I is divided into two chapters:

Chapter 2 works as an introduction to chapter 3, introducing the basics of dissimilarity functions and proposing a first predictor to obtain the expected value of a time-series. A numerical example of the forecasting of the Dow Jones Industrial Average Index is shown to verify the good performance of the algorithm.

Chapter 3 presents the methodologies to quantify the uncertainty of the predictions. First, a method to obtain an empirical probability density function for a univariate function is discussed. From this empirical probability distribution, it is possible to compute quantiles of a desired probability, obtaining an interval predictor. Later, the methodology is extended to tackle the multivariate case. As obtaining the probability distribution of a multivariate random variable is generally intractable, an alternative method bypassing the probability density function allows us to obtain prediction regions of such variables. Some numerical examples are presented in order to show the improvements with respect to other baseline techniques.

On the other hand, **part II** presents novel methodologies to obtain models of nonlinear systems or time-series. Here, the methodologies proposed in part I are extended in order to obtain a model of the system by feed backing the 1-step ahead predictions and regularizing the optimization problem used to compute such predictions. Two different models are proposed in this chapter: a linear time varying (LTV) model obtained by weighting appropriately the local data within the data set and a kernel-based version of the state-space kriging where the non-linearity is modeled by means of a kernel function. The obtained models can be used easily to make predictions or design MPC controllers. Forecasting and Control examples are both provided to show the effectiveness of the proposed approaches.

Part II is divided into two chapters:

Chapter 4 presents the state-space kriging approach, named after the *kriging method* due to its similarities and also because the weights used to obtain the predictions become the state of the new model. This chapter is focused on autonomous systems and time-series. Thus, it presents methodologies to obtain a model of the outputs from past data of the system that can be used to make predictions. Numerical examples with comparisons are provided in order to show the effectiveness of the proposed approaches.

Chapter 5 introduces the state-space kriging for non-autonomous systems. It is shown that the input can be considered easily within the optimization problem with very few changes. As the approach considered in this chapter can tackle the problem of forecasting systems with manipulable inputs, the proposed approach is suitable to design, for example, MPC controllers whose prediction model is a state-space kriging model. The stability of the proposed controller is proven and a numerical example and a real experiment are provided to verify the performance of the controller.

Finally, **part III** presents novel bounds on the constraint violation rate that can be used to quantify the performance of different controllers in a stochastic setting in order to choose the most appropriate controller among them. These controllers are not necessarily MPC controllers, that is, any controller can be assessed by means of these techniques. Another advantage of these methods is that the computational burden of the process is completely offline and thus the online computation time of the controller does not change. The proposed methods are finally validated in the context of the management of a data center.

Part III is divided into two chapters:

Chapter 6 presents the concepts needed to obtain the bounds on the empirical constraint violation level based on the results by Chernoff in [128] and by Alamo et al. in [61]. These results provides two different expressions that can be used to measure the reliability of any controller.

Chapter 7 introduces the model of a data center (dealing with both the thermal modeling and the task model for the computers) that will be used as a benchmark for the previously developed bounds. In this chapter, not only the model of the data center and the implied constraints are presented, but a management approach based on MPC is also proposed. As the problem for this specific system is generally hard to solve (integer optimization, non-convex model, etc.) the solution of the problem is computed by means of a particle-based solver implemented in a Graphic Processing Unit (GPU).

1.3 Publications

The results shown throughout this thesis have been published in several journals and conference proceedings, although some of them are still under review.

The forecasting methods developed and the numerical examples presented in part I can be found in:

- [129] G. Alfonso, A. D. Carnerero, D. R. Ramirez, and T. Alamo, "Stock forecasting using local data," *IEEE Access*, vol. 9, pp. 9334–9344, 2020.
- [130] A. D. Carnerero, D. R. Ramirez, and T. Alamo, "Probabilistic interval predictor based on dissimilarity functions," *IEEE Transactions on Automatic Control*, 10.1109/TAC.2021.3136137, 2021.
- A. D. Carnerero, D. R. Ramirez, S. Lucia and T. Alamo, "Prediction regions based on dissimilarity functions," *Submitted to ISA Transactions*, 2022.

The proposed models and the MPC controllers developed in part II can be found in:

- [131] A. D. Carnerero, D. R. Ramirez, and T. Alamo, "State-space Kriging: A data-driven method to forecast nonlinear dynamical systems," *IEEE Control Systems Letters*, vol. 6, pp. 2258 – 2263, 2022.
- A. D. Carnerero, D. R. Ramirez and T. Alamo, "Kernel based State-Space Kriging: Application to predictive control," *2022 61th IEEE Conference on Decision and Control (CDC)*.
- A. D. Carnerero, D. R. Ramirez, D. Limon and T. Alamo, "Kernel-based State-Space Kriging for predictive control," *Submitted to IEEE/CAA Journal of Automatica Sinica*, 2022.

On the other hand, the results concerning the randomized approaches and development of model predictive controllers for energy-efficient management of data centers facilities can be found in:

- [132] A. D. Carnerero, D. R. Ramirez, D. Limon and T. Alamo, "Particle based optimization for predictive energy efficient data center management," in *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 2660–2665, IEEE, 2020.
- [133] A. D. Carnerero, D. R. Ramirez, T. Alamo, and D. Limon, "Probabilistically certified management of data centers using predictive control," *IEEE Transactions on Automation Science and Engineering*, 10.1109/TASE.2021.3093699, 2021.

Finally, another article related to this dissertation but not directly discussed here is:

- [134] G. Alfonso, A. D. Carnerero, D. R. Ramirez, and T. Alamo, "Receding horizon optimization of large trade orders," *IEEE Access*, vol. 9, pp. 63865-63875, 2021.

Part I

Probabilistic forecasting

Chapter 2

Forecasting using dissimilarity functions

In this chapter, the notion of dissimilarity functions and their usage in forecasting will be explored. As a result, a technique for forecasting any time series, with or without underlying dynamics, will be presented. This technique takes into account locality in data to better forecast processes with nonlinear behaviours.

Given a certain data set stored in a matrix $D = \begin{bmatrix} \bar{d}_1 & \bar{d}_2 & \dots & \bar{d}_N \end{bmatrix} \in \mathbb{R}^{n \times N}$, where each $\bar{d}_i \in \mathbb{R}^n$ is a past sample and N is the number of samples, we are interested in determining if a given vector $d \in \mathbb{R}^n$ can be considered to be similar to the other vectors of the data set D. In a more precise way, we are looking for a function

$$J_d(\cdot, \cdot) : \mathbb{R}^n \times \mathbb{R}^{n \times N} \to [0, \infty]$$

that measures the dissimilarity between a given point d and the data set D. Large values of $J_d(d, D)$ represent a high degree of dissimilarity, while small values correspond to a high degree of similarity (i.e., a small degree of dissimilarity). Clearly, from a dissimilarity function $J_d(d, D)$ one can obtain a similarity function $J_s(d, D)$. For example, given $\sigma > 0$, $J_s(x, D) = \mathrm{e}^{-\sigma J_d(d,D)}$ is small when d is not similar to the points in D and close to 1 when d is very similar to the elements of D. Another possibility would be $J_s(d, D) = (1 + \sigma J_d(d, D))^{-1}$, where $\sigma > 0$.

There exists a wide class of operators that can serve as dissimilarity functions for the particular case in which D is a singleton. For singleton D (i.e. $D = \bar{d}$), one popular choice is

$$J_d(d, \bar{d}) = \|d - \bar{d}\|,$$

where $\|\cdot\|$ is a given norm. One could also use the minimum distance with respect to each member in D. That is,

$$J_d(d, D) = \min_{i=1,\dots,N} \|d - \bar{d}_i\|. \tag{2.1}$$

Another possibility could be to consider as a dissimilarity function the mean value of the distances of d to each member of set D. See chapter 2 of [135] and chapter 2 of [136] for a review of similarity and dissimilarity functions applied in the field of image registration and in the context of cluster analysis, respectively.

Although these approaches could be valid for some applications, more sophisticated approaches are required in many situations. In this chapter, a convex optimization problem is proposed to obtain a measure of the dissimilarity between a point d and a data set D.

2.1 Proposed dissimilarity function

The dissimilarity function to be used throughout this dissertation is stated in the following definition:

Definition 2.1. Given $d \in \mathbb{R}^n$, a data set $D \in \mathbb{R}^{n\times N}$ and the scalar $\gamma \in [0, 1)$, the dissimilarity function $J_\gamma(d, D)$ is defined as

$$J_\gamma(d, D) = \min_{\lambda_{(1)},\dots,\lambda_{(N)}} (1-\gamma)\sum_{i=1}^{N} \lambda_{(i)}^2 + \gamma \sum_{i=1}^{N} |\lambda_{(i)}| \tag{2.2a}$$

$$\text{s.t.} \quad d = \sum_{i=1}^{N} \lambda_{(i)} d_i \tag{2.2b}$$

$$1 = \sum_{i=1}^{N} \lambda_{(i)}. \tag{2.2c}$$

Remark 2.2. *Note that non negative constant weights could be included into the cost function. That is, one could consider the cost function*

$$(1-\gamma)\sum_{i=1}^{N} \omega_i \lambda_{(i)}^2 + \gamma \sum_{i=1}^{N} |\lambda_{(i)}|,$$

where the scalars ω_i, $i = 1, \dots, N$ are used to weight different elements in D. These weights could be computed, for example, using a distance function between d and the singleton d_i (for example, $\omega_i = ||d - d_i||$).

This would be a way to incorporate local information into the analysis. Although the results are stated for the particular case in which $\omega_i = 1$, $i = 1, \dots, N$, the generalization to the general case is not difficult.

Remark 2.3. *There also exists a slight variation of the aforementioned dissimilarity function where the cost function becomes*

$$\sum_{i=1}^{N} \lambda_{(i)}^2 + \gamma \sum_{i=1}^{N} |\lambda_{(i)}|.$$

Note that, here, $\gamma \in [0, \infty)$. This works as a change in the scale of γ, but the dissimilarity function remains the same.

Remark 2.4. *Note that optimization problem (2.2) could be non-feasible. In order to rule out this possibility, it is assumed that the vectors that compose set D span all the space.*

Remark 2.5. *Optimization problem (2.2) is similar to the one appearing in the context of direct weight optimization and Kriging, where predictions of a certain variable are obtained by means of the solution of an optimization problem [57], [56], [137], [126], [138].*

It is important to remark that the proposed dissimilarity measure is invariant with respect to affine transformations. This is formally stated in the following property.

Property 2.6. Consider $d_{T,v}$ and $D_{T,v}$ obtained from d and D through the following affine transformation.

$$\begin{aligned} d_{T,v} &= Td + v \\ D_{T,v} &= \left[\begin{array}{cccc} T\bar{d}_1 + v & T\bar{d}_2 + v & \ldots & T\bar{d}_N + v \end{array} \right], \end{aligned}$$

where T is any non-singular matrix and v is any vector of adequate dimensions. Then

$$J_\gamma(d, D) = J_\gamma(d_{T,v}, D_{T,v}).$$

Proof. First, it is shown that any feasible solution $\lambda_{(i)}$, $i = 1, \ldots, N$ to the problem of computing $J_\gamma(d, D)$ is also a feasible solution for the computation of $J_\gamma(d_{T,v}, D_{T,v})$. Suppose that $d = \sum_{i=1}^{N} \lambda_{(i)} \bar{d}_i$ and $\sum_{i=1}^{N} \lambda_{(i)} = 1$. Then

$$\begin{aligned} d_{T,v} &= Td + v \\ &= T\left(\sum_{i=1}^{N} \lambda_{(i)} \bar{d}_i\right) + \left(\sum_{i=1}^{N} \lambda_{(i)}\right) v \\ &= \sum_{i=1}^{N} \lambda_{(i)} (T\bar{d}_i + v). \end{aligned}$$

It is clear that $T\bar{d}_i + v$, $i = 1, \ldots, N$, are the elements of $D_{T,v}$. Therefore $\lambda_{(i)}$, $i = 1, \ldots, N$, is also a feasible solution for the problem that defines $J_\gamma(d_{T,v}, D_{T,v})$. From this, it is possible to infer that $J_\gamma(d_{T,v}, D_{T,v}) \leq J_\gamma(d, D)$. On the other hand, since T is non-singular, a similar reasoning could be made to show that any feasible solution for $J_\gamma(d_{T,v}, D_{T,v})$ is a feasible solution for $J_\gamma(d, D)$. In this way, it is also proven that $J_\gamma(d, D) \leq J_\gamma(d_{T,v}, D_{T,v})$. Both inequalities prove the claimed equality. ■

This invariance property is important because it guarantees that the analysis based on the proposed dissimilarity function is not affected by the choice of the

coordinate system. However, many of the dissimilarity functions that can be found in the literature are not invariant. For example, any dissimilarity function based on the distance of d to the elements of D, such as that of equation (2.1), will be dependent on the particular choice of the coordinate system.

The proposed optimization problem (2.2) is a strict convex optimization problem subject to convex constraints. This means that it has a unique solution [139]. Note that the numerical resolution can be addressed using a dual formulation. In the dual formulation for this particular optimization problem, the number of dual decision variables is equal to the number of equality constraints $(n+1)$ which is in many situations much smaller than the number of primal variables (N). On the other hand, the gradient of the objective function in the dual formulation can be obtained in a direct way because once the dual variables are fixed, the optimal values for the primal variables are obtained by solving N separable optimization problems. The computations in this dissertation have been made using an accelerated gradient method in the dual variables (see [140], [141] and [142]).

As it is formally stated in the following property, the optimization problem has an explicit solution for the particular case $\gamma = 0$.

Property 2.7. $J_0(d, D)$ has the following explicit expression

$$J_0(d, D) = N^{-1} + (d - d_c)^\top (DD^\top - N d_c d_c^\top)^{-1}(d - d_c),$$

where $d_c = N^{-1} D \mathbf{1}^\top$ and $\mathbf{1}^\top$ is a column vector with all its N components equal to 1.

Proof. The optimization problem is solved using a dual formulation where $\mu \in \mathbb{R}^n$ denotes the multipliers associated with the equality constraint

$$d = \sum_{i=1}^{N} \lambda_{(i)} d_i = D\lambda,$$

and ν is the multiplier corresponding to the equality

$$1 = \sum_{i=1}^{N} \lambda_{(i)} = \mathbf{1}\lambda.$$

the Lagrange function is

$$\mathcal{L}(\lambda, \mu, \nu) = \lambda^\top \lambda + \mu^\top (D\lambda - d) + \nu(\mathbf{1}\lambda - 1).$$

Denote λ^*, μ^* and ν^* the optimal values for the primal and dual variables. From $\frac{\partial \mathcal{L}(\lambda^*, \mu^*, \nu^*)}{\partial \lambda} = 0$, it is clear that the optimal vector λ^* is given by

$$\lambda^* = -\frac{1}{2}(D^\top \mu^* + \mathbf{1}^\top \nu^*). \tag{2.3}$$

Since $\mathbf{1}\lambda^* = 1$, $D\mathbf{1}^\top = Nd_c$ and $\mathbf{1}\mathbf{1}^\top = N$, it is possible to premultiply both terms of the last equality by $\mathbf{1}$ to obtain

$$\begin{aligned} 1 &= -\frac{1}{2}(\mathbf{1}D^\top\mu^* + N\nu^*) \\ &= -\frac{N}{2}(d_c^\top\mu^* + \nu^*). \end{aligned}$$

Therefore,

$$\nu^* = -\frac{2}{N} - d_c^\top\mu^*.$$

Substituting the expression for ν^* in (2.3) yields

$$\begin{aligned} \lambda^* &= -\frac{1}{2}\left(D^\top\mu^* - \mathbf{1}^\top(\frac{2}{N} + d_c^\top\mu^*)\right) \\ &= \frac{\mathbf{1}^\top}{N} - \frac{1}{2}(D^\top - \mathbf{1}^\top d_c^\top)\mu^*. \end{aligned} \tag{2.4}$$

Premultiplying by D, the following equation is obtained

$$D\lambda^* = d_c - \frac{1}{2}(DD^\top - Nd_cd_c^\top)\mu^*. \tag{2.5}$$

From the equality constraint $D\lambda^* = d$ and (2.5), the value of μ^* can be obtained as

$$\mu^* = -2(DD^\top - Nd_cd_c^\top)^{-1}(d - d_c).$$

Substituting μ^* in equation (2.4), it is possible to compute λ^* as

$$\lambda^* = \frac{\mathbf{1}^\top}{N} + (D^\top - \mathbf{1}^\top d_c^\top)(DD^\top - Nd_cd_c^\top)^{-1}(d - d_c).$$

Finally, taking into account that

$$\mathbf{1}(D^\top - \mathbf{1}^\top d_c^\top) = (Nd_c^\top - Nd_c^\top) = 0,$$

the following expression is obtained

$$(D^\top - \mathbf{1}^\top d_c^\top)^\top(D^\top - \mathbf{1}^\top d_c^\top) = DD^\top - Nd_cd_c^\top.$$

From last equality and the expression for λ^*, the value of $J_0(d, D)$ can be computed as

$$J_0(d, D) = (\lambda^*)^\top\lambda^* = N^{-1} + (d - d_c)^\top(DD^\top - Nd_cd_c^\top)^{-1}(d - d_c). \quad \blacksquare$$

The previous result shows that the dissimilarity function is a quadratic function on the argument d for the particular case $\gamma = 0$. For the more general case in which $\gamma > 0$, it is possible to infer from the Karush-Kuhn-Tucker (KKT) optimality conditions [139] that the dissimilarity function $J_\gamma(d, D)$ is a piecewise convex quadratic function with respect to d.

2.1.1 Clarifying example

Figure 2.1 shows a cloud of points obtained from a certain probability distribution. On the left, the original cloud of points is shown and, on the right, this cloud of points is rotated an angle $\pi/4$.

Red markers correspond to the points in the data set D whereas blue markers correspond to two given values of d, i.e.

$$d_1 = \begin{bmatrix} 0.25 \\ 1 \end{bmatrix}, \quad d_2 = \begin{bmatrix} 0.75 \\ 1 \end{bmatrix}.$$

Note that these values correspond to the left figure, where they are not rotated. Assuming that $\gamma = 0$, the obtained values of the dissimilarity for these vectors d_1 and d_2 are

$$J_0(d_1, D) = 0.0056, \quad J_0(d_2, D) = 0.0191.$$

It is easy to see that the dissimilarity is smaller for d_1 because the concentration of data points is larger in that region, in contrast to d_2 where the concentration of red markers is small. This means that d_1 is a more likely event than d_2.

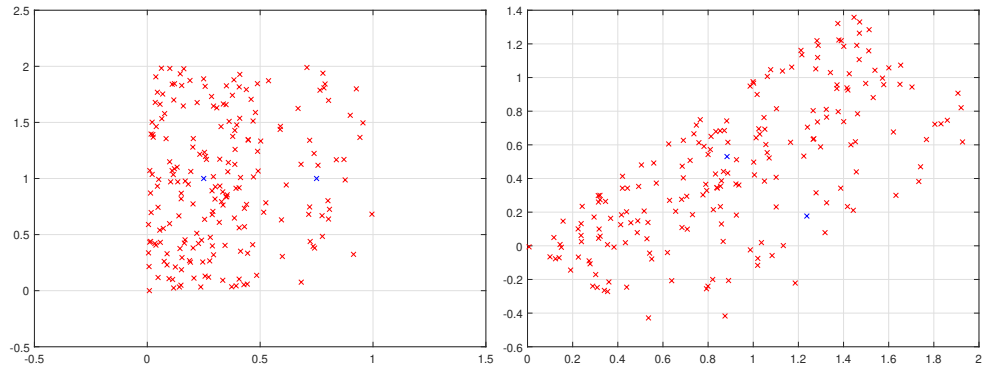


Figure 2.1: Left: original cloud of points. Right: rotated cloud of points.

Also, note that, as it was proven in property 2.6, the value of the dissimilarity function is invariant with respect to affine transformations.

2.2 Dissimilarity functions and regression

This section shows how dissimilarity functions can be used in the context of regression. Denoting the regressor as $z \in \mathbb{R}^{n_z}$ and the output as $y \in \mathbb{R}^{n_y}$, it is assumed that the following data sets Z and Y are available

$$Z = \begin{bmatrix} \bar{z}_1 & \bar{z}_2 & \dots & \bar{z}_N \end{bmatrix}, \; Y = \begin{bmatrix} \bar{y}_1 & \bar{y}_2 & \dots & \bar{y}_N \end{bmatrix},$$

where $\bar{y}_i \in \mathbb{R}^{n_y}$ and $\bar{z}_i \in \mathbb{R}^{n_z}$ $\forall i = 1, ..., N$ are past samples of the outputs and the regressor respectively.

Fixing a certain z and computing $J_\gamma\left(\begin{bmatrix} z \\ y \end{bmatrix}, \begin{bmatrix} Z \\ Y \end{bmatrix}\right)$ gives us the dissimilarity of a certain output y conditioned to the value of the regressor z. Thus, the dissimilarity function becomes

$$J_\gamma\left(\begin{bmatrix} z \\ y \end{bmatrix}, \begin{bmatrix} Z \\ Y \end{bmatrix}\right) = \min_{\lambda_{(1)},\dots,\lambda_{(N)}} (1-\gamma)\sum_{i=1}^{N}\lambda_{(i)}^2 + \gamma\sum_{i=1}^{N}|\lambda_{(i)}| \tag{2.6a}$$

$$\text{s.t. } z = \sum_{i=1}^{N}\lambda_{(i)}\bar{z}_i \tag{2.6b}$$

$$y = \sum_{i=1}^{N}\lambda_{(i)}\bar{y}_i \tag{2.6c}$$

$$1 = \sum_{i=1}^{N}\lambda_{(i)}. \tag{2.6d}$$

Given z, consider, from all the possible values of y, the one that minimizes the dissimilarity function defined by the previous optimization problem

$$\tilde{y}(z) = \arg\min_{y} J_\gamma\left(\begin{bmatrix} z \\ y \end{bmatrix}, \begin{bmatrix} Z \\ Y \end{bmatrix}\right). \tag{2.7}$$

Note that minimizing the dissimilarity corresponds to maximizing the similarity. Thus, $\tilde{y}(z)$ is the most likely value to happen given the regressor z and the data sets Z and Y. This $\tilde{y}(z)$ could be used as a forecasting of the next output.

Now, instead of $J_\gamma\left(\begin{bmatrix} z \\ y \end{bmatrix}, \begin{bmatrix} Z \\ Y \end{bmatrix}\right)$, consider $J_\gamma(z, Z)$, that is,

$$J_\gamma(z, Z) = \min_{\lambda_{(1)},\dots,\lambda_{(N)}} (1-\gamma)\sum_{i=1}^{N}\lambda_{(i)}^2 + \gamma\sum_{i=1}^{N}|\lambda_{(i)}| \tag{2.8a}$$

$$\text{s.t. } z = \sum_{i=1}^{N}\lambda_{(i)}\bar{z}_i \tag{2.8b}$$

$$1 = \sum_{i=1}^{N}\lambda_{(i)}. \tag{2.8c}$$

As the optimization problem in (2.8) is a less constrained version of that in (2.6), the optimal value of the cost of the more constrained problem is always greater or equal to the optimal value of the cost of the less constrained one, that is,

$$J_\gamma\left(z, Z\right) \leq J_\gamma\left(\begin{bmatrix} z \\ y \end{bmatrix}, \begin{bmatrix} Z \\ Y \end{bmatrix}\right).$$

Now, denote the vector of weights λ that minimizes the optimization problem in (2.8) as λ^*. From this optimum vector of weights λ^*, $\tilde{y}(z)$ can be computed as

$$\tilde{y}(z) = Y\lambda^* = \sum_{i=1}^{N} \lambda^*_{(i)} \bar{y}_i. \tag{2.9}$$

Then, it is easy to see that

$$J_\gamma(z, Z) = J_\gamma\left(\begin{bmatrix} z \\ \tilde{y}(z) \end{bmatrix}, \begin{bmatrix} Z \\ Y \end{bmatrix}\right)$$

and thus $\tilde{y}(z)$, computed using the optimal weights from (2.8), is the one that minimizes the dissimilarity function for a fixed z (see equation (2.7)). Note that this forecasting is made by means of a linear combination of past outputs whose weights are computed obtaining an affine envelope of the regressor z. As it is explained in what follows, strictly positive values of γ encourage the components of λ to be positive (which means that the central estimation is often obtained from an interpolation of points).

It is easy to see that when every $\lambda_{(i)} \geq 0$, $\forall i = 1, ..., N$, then $\sum_{i=1}^{N} |\lambda_{(i)}| = 1$. However, when some of the components of λ are negative, $\sum_{i=1}^{N} |\lambda_{(i)}| > 1$. In other words, the cost term $\sum_{i=1}^{N} |\lambda_{(i)}|$ becomes larger when extrapolating points (i.e. using negative values of $\lambda_{(i)}$). This means that convex combinations of $\lambda_{(i)}$ are encouraged and thus interpolation is preferred, which may improve the predictions when nonlinear systems are taken into account. The optimal value for $\gamma \geq 0$ could be obtained in different ways. A reasonable choice could be, for example, to choose the one that minimizes the error of the predictions with respect to the real outputs in a validation set.

As it is stated in the following property, the estimation obtained for the particular case $\gamma = 0$ matches the one given by the linear least squares method.

Property 2.8. Given z, the estimation

$$\tilde{y}(z) = \arg\min_{y} \; J_0\left(\begin{bmatrix} z \\ y \end{bmatrix}, \begin{bmatrix} Z \\ Y \end{bmatrix}\right),$$

matches the estimation obtained by linear least squares.

Proof. From equation (2.9), the optimal value for the estimation is $\tilde{y}(z) = \Sigma_{i=1}^{N} \lambda^*_{(i)} \bar{y}_i$, where for the particular case $\gamma = 0$, $\lambda^*_{(i)}$, $i = 1, \ldots, N$, are the optimal values of

the optimization problem

$$\begin{aligned} \min_{\lambda_{(1)},\ldots,\lambda_{(N)}} \quad & \sum_{i=1}^{N} \lambda_{(i)}^2 \\ \text{s.t.} \quad & z = \sum_{i=1}^{N} \lambda_{(i)} \bar{z}_i \\ & 1 = \sum_{i=1}^{N} \lambda_{(i)}. \end{aligned}$$

Defining

$$R = \begin{bmatrix} \bar{z}_1 & \bar{z}_2 & \ldots & \bar{z}_N \\ 1 & 1 & \ldots & 1 \end{bmatrix}, \quad r = \begin{bmatrix} z \\ 1 \end{bmatrix},$$

the equality constraints can be rewritten as

$$R\lambda = r. \tag{2.10}$$

From the KKT optimality conditions, the optimal solution is given by (see subsection 10.1.1 in [139])

$$\begin{bmatrix} \mathrm{I_N} & R^\top \\ R & \mathbf{0}_{(n_z+1)} \end{bmatrix} \begin{bmatrix} \lambda^* \\ \varphi^* \end{bmatrix} = \begin{bmatrix} 0 \\ r \end{bmatrix},$$

where φ^* corresponds to the optimal dual decision variables corresponding to the equality constraint (2.10). The previous equation can be rewritten as

$$\begin{aligned} \lambda^* &= -R^\top \varphi^* \\ R\lambda^* &= r. \end{aligned}$$

From here, it is inferred that $-RR^\top \varphi^* = r$, which implies $\varphi^* = -(RR^\top)^{-1} r$ and thus finally obtaining

$$\lambda^* = R^\top (RR^\top)^{-1} r.$$

Therefore,

$$\tilde{y}(z) = Y^\top \lambda^* = Y^\top R^\top (RR^\top)^{-1} r = r^\top (RR^\top)^{-1} RY.$$

This corresponds to the least squares estimation obtained when the considered regressors are the vectors $\begin{bmatrix} \bar{z}_i^\top & 1 \end{bmatrix}^\top$, $i = 1, \ldots, N$ (see [143], [64]). ■

Previous property shows that the proposed estimation method encompasses the least squares method for the particular case $\gamma = 0$. A family of optimal estimators is obtained if γ is considered a tuning parameter.

From now on, in order to make the manuscript more readable, the notation will be simplified by removing Z and Y from the dissimilarity function (since they are assumed to be fixed). That is, the notation becomes

$$J_\gamma \left(\begin{bmatrix} z \\ y \end{bmatrix}, \begin{bmatrix} Z \\ Y \end{bmatrix} \right) = J_\gamma (z, y).$$

Moreover, the minimal value of the dissimilarity function given z will be denoted as $J^*_\gamma(z)$, that is

$$J^*_\gamma(z) = J_\gamma(z, \tilde{y}(z)) = \min_y J_\gamma(z, y).$$

The following remark summarizes the process to use the dissimilarity functions in the context of regression.

Remark 2.9. *Given the regressor z, the dissimilarity function $J_\gamma(\cdot,\cdot)$ and the data sets Z and Y, solve optimization problem in* (2.8) *to obtain λ^*. Once λ^* has been obtained, the forecasting $\tilde{y}(z)$ is computed by means of equation* (2.9).

2.3 Application: forecasting stock prices using dissimilarity functions

In this section, a numerical example is presented to show the effectiveness of the proposed dissimilarity function as a predictor.

Consider the evolution of the price of a stock as a time series $p_k \in \mathcal{P}$, where k is a time unit and $\mathcal{P}$ is the range of values that the stock price can take. The state of this time series can be described as the value at time k of a series of technical indicators. These technical indicators can be past values of the price, stock price returns or more complex metrics in general. Thus, this group of technical indicators will form the regressor z_k. The objective is to be able to predict up to l-steps (i.e. l-days) ahead the price of a stock, that is, to obtain $\tilde{p}_{k+l}$ at time k. Thus, y_k corresponds to

$$y_k = \begin{bmatrix} p_{k+1} & p_{k+2} & \cdots & p_{k+l} \end{bmatrix}^\top.$$

As it was shown in the previous section, Z and Y are compounded of past samples of z and y, denoted as $\bar{z}_i$ and $\bar{y}_i$.

The approach considered in this example uses only a small subset of the database, denoted as $\Omega(z_k)$, to compute the predictions $\tilde{p}_{k+l}$. This subset $\Omega(z_k)$ is chosen by finding the N_Ω closest points to z_k within the data set by means of a certain distance measurement. Thus, $\Omega(z_k)$ is compounded of N_Ω pairs $(\bar{z}_i, \bar{y}_i)$ of the full data set.

Once the data to be included in $\Omega(z_k)$ are selected, the optimization problem (2.8) is solved with this reduced data set. Also, note that the forecasting of p_{k+l} is also made with this small portion of the complete data set. Algorithm 4 gives a formal description of the proposed approach.

Remark 2.10. *The distance can be any measure of how close are z_k to $\bar{z}_i$. Euclidean distance would be the most typical choice, but it is also possible to consider other aspects like the time span between samples. In this way, recent data could*

Algorithm 1: l-step ahead stock forecasting using local data

Data: Z, Y, z_k, N_Ω and γ.

Result: $\left[\begin{array}{cccc} \tilde{p}_{k+1} & \tilde{p}_{k+2} & \dots & \tilde{p}_{k+l} \end{array}\right]$.

1 Compute the distance for each $\bar{z}_i$ in the database to z_k;

2 Create a list of the entries in the data base sorted according to the computed distances. Denote $\bar{z}_j$ and $\bar{y}_j$ as the regressor $\bar{z}_i$ and the output $\bar{y}_i$ of the j-th entry in this ordered list respectively;

3 Build $\Omega(z_k)$ using the first N_Ω entries of the ordered list, that is,

$$\Omega(z_k) \triangleq \{(\bar{z}_j, \bar{y}_j)\} \quad \forall j = 1, \dots, N_\Omega.$$

4 Solve the following optimization problem:

$$\begin{aligned} \min_{\lambda_{(1)}, \dots, \lambda_{(N_\Omega)}} \quad & (1-\gamma) \sum_{j=1}^{N_\Omega} \lambda_{(j)}^2 + \gamma \sum_{j=1}^{N_\Omega} |\lambda_{(j)}| \\ \text{s.t.} \quad & \sum_{j=1}^{N_\Omega} \lambda_{(j)} = 1, \\ & \sum_{j=1}^{N_\Omega} \lambda_{(j)} \bar{z}_j = z_k. \end{aligned}$$

and compute the forecasted prices as:

$$\left[\begin{array}{cccc} \tilde{p}_{k+1} & \tilde{p}_{k+2} & \dots & \tilde{p}_{k+l} \end{array}\right]^\top = \tilde{y}_k = \sum_{j=1}^{N_\Omega} \lambda_{(j)} \bar{y}_j.$$

be prioritized when selecting the elements of $\Omega(z_k)$*. Other aspects like seasonality could also be taken into account.*

2.3.1 Results

The data set to be used in this example was obtained from the data provider Bloomberg and it is composed of the daily closing prices of the Dow Jones Index from 2005 to 2016. The data was divided into a training set (Z and Y) from 2005 to 2014 and a test set from 2015 to mid-2016. This period was chosen because the market does not follow a certain trend (bullish or bearish) that would make the forecasting trivial. In order to reduce the noise of the time series, the prices are smoothed using a 5-day Exponential Moving Average (EMA) computed as:

$$p_{\text{EMA},k}^m = \left(\frac{2}{m+1}\right) p_k + \left(1 - \frac{2}{m+1}\right) p_{\text{EMA},k-1}^m,$$

with $p^m_{\text{EMA},0} = p_0$ and $m = 5$ in this case. Note that the smoothing applied here is very light in comparison to the usual values of m used in short-term forecasting (12 and 26 day EMA [144]). This preserves fast price fluctuations at the cost of making the forecasting process harder.

The regressor z_k is composed of the last ten days smoothed prices, as well as the 5-day and 10-day relative difference percentage of unsmoothed prices (RDP) [145] i.e.

$$\text{RDP}^m_k = 100\left(\frac{p_k - p_{k-m}}{p_k}\right),$$

being m equal to 5 and 10 respectively. Thus,

$$z_k = \begin{bmatrix} p_k & p_{k-1} & \dots & p_{k-9} & \text{RDP}^5_k & \text{RDP}^{10}_k \end{bmatrix}^\top.$$

The size of $\Omega(z_k)$ is set to $N_\Omega = 250$ and $\gamma = 0$. The forecast and real prices can be seen in figures 2.2 and 2.3. The results show that the forecasting is accurate enough at first and it becomes worse as the prediction horizon increases, as it can be expected.

In order to compare the results obtained with the proposed approach, the results obtained with a persistence predictor and a Neural Network (NN) are shown. This persistence predictor works as a martingale, that is, $\tilde{p}_{k+l} = p_k$ whereas the NN corresponds to a multi-layer perceptron with 20 neurons in the hidden layer and trained following the Levenberg-Marquardt rule. The numerical results are shown in table 2.1. In particular, the mean squared errors (MSE) of the proposed approach and the aforementioned baselines are shown in table 2.1a. It can be seen that the proposed approach achieves the lowest errors. However, note that for long step ahead predictions, the persistence predictor achieves results as good as the other techniques, a consequence from the long term random walk nature of the financial market.

Table 2.1: Results of the Dow Jones forecasting example.

(a) Mean squared errors (MSE). (b) Standard deviation of the errors (σ).

k	**Proposed**	**MLP**	**Persistence**	**Proposed**	**MLP**	**Persistence**
1	3,294.8	3,577.0	5,296.4	57.4	59.9	72.8
2	12,433.2	14,190.8	16,867.4	111.5	119.3	130.0
3	26,659.0	30,829.6	31,975.0	163.1	175.6	178.9
4	45,475.9	50,794.9	49,072.4	212.9	224.9	221.6
5	66,859.6	77,017.5	66,943.4	257.8	276.0	258.8

On the other hand, table 2.1b shows the standard deviation of both the proposed approach and the baselines. Again, the proposed approach attains the lowest standard deviation in the errors, which means that the errors are not only smaller in mean but also they are more concentrated.

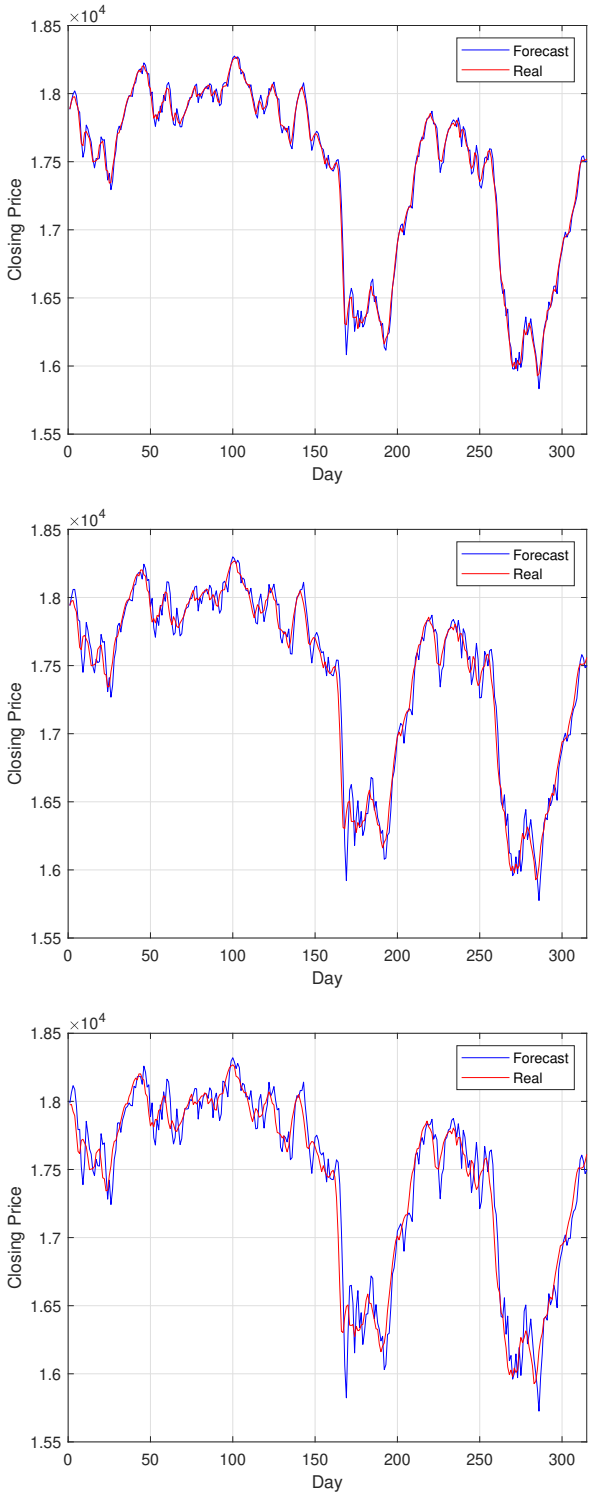


Figure 2.2: Forecasted and real prices (5-day EMA) for 1 to 3 days forecasting.

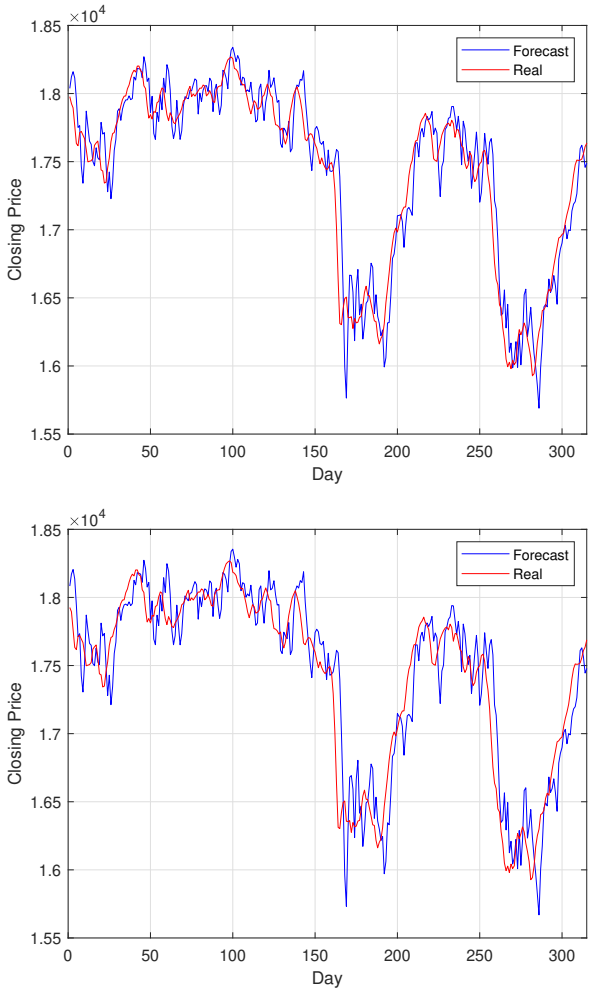


Figure 2.3: Forecasted and real prices (5-day EMA) for 4 and 5 days forecasting.

2.4 Conclusions

This chapter presented the notion of dissimilarity along with the dissimilarity function to be used in this dissertation. It was shown that the proposed dissimilarity function has many advantageous properties and that it can be used easily for regression or time-series forecasting. In order to obtain a prediction, it is only needed to compute a convex optimization problem that can be easily solved.

A forecasting example of the closing price of the Dow Jones index was proposed to compare the performance of the proposed predictor with a Neural Network and a persistence predictor. The results showed that the predictions obtained with the proposed approach are more accurate than those obtained by means of the baselines.

In the next chapter, the case of uncertain predictions is tackled. By means of the proposed dissimilarity function, methodologies to obtain interval predictions and prediction regions are presented.

Chapter 3

Probabilistic prediction regions

This chapter presents new methodologies to compute prediction regions of dynamical systems and time series. The univariate case corresponds to interval predictions. Here, the dissimilarity functions presented in chapter 2 are used to estimate the conditional probability density function of the outputs. This conditional probability density function is parameterized by means of two hyperparameters which are chosen so that the length of the intervals is minimized while guaranteeing that the empirical probability is fulfilled in a validation set. Also, the methodology is applied to some forecasting problems.

The case of computing multivariate prediction regions is a harder task. To avoid the integration of the multivariate conditional probability density function (which would be intractable), the region is defined implicitly through the dissimilarity function, resulting in a much lighter procedure in terms of computational burden. Again, the hyperparameters in which the methodology relies are computed to minimize the size of the regions while achieving the desired empirical probability in a validation set. Moreover, the approach does not make any assumption on the data or the shape of the region. Finally, a method to obtain ellipsoidal approximations of the regions and a numerical example are provided.

3.1 Univariate case

In the previous chapter, a function to measure the dissimilarity of a given vector with respect to a certain data set was presented. There, it was also shown that dissimilarity and similarity are complementary concepts. Actually, it turns out that similarity is very close to the concept of probability. Thus, it is possible to obtain a measure of the probability by using the proposed dissimilarity function and transforming it into a similarity function.

3.1.1 Empirical probability density function

Given z, the dissimilarity function $J_\gamma(\cdot,\cdot)$, the data sets Z and Y and the non negative scalars c and γ, the empirical **conditional** p.d.f. could be defined as

$$\mathrm{ecp}_{\gamma,c}\left(z, y, \left[\begin{array}{c} Z \\ Y \end{array}\right]\right) = \frac{e^{-cJ_\gamma(z,y)}}{\int\limits_{\check{\mathcal{Y}}} e^{-cJ_\gamma(z,\check{y})}d\check{y}},\ \forall y \in \mathcal{Y},$$

where $\mathcal{Y}$ is the set of possible values of y. As in chapter 2, the notation is simplified by removing Z and Y from the probability density function, that is

$$\mathrm{ecp}_{\gamma,c}\left(z, y\right) = \mathrm{ecp}_{\gamma,c}\left(z, y, \left[\begin{array}{c} Z \\ Y \end{array}\right]\right).$$

The empirical conditional p.d.f. serves to model the probability of y given the occurrence of z. Now, it is shown how to use this notion to compute, given z, an interval estimation of y.

First, in order to simplify the numerical integration required to compute the interval estimations, the set $\mathcal{Y}$ is approximated with a set $\check{\mathcal{Y}}$ of finite cardinality, that is,

$$\check{\mathcal{Y}} = \{\check{y}_1, \ldots, \check{y}_M\},$$

where $\check{y}_j < \check{y}_{j+1}$, $j = 1, \ldots, M-1$, and the extreme values of $\check{\mathcal{Y}}$ are chosen to guarantee that y belongs to $[\check{y}_1, \check{y}_M]$ with high probability. A reasonable procedure to construct set $\check{\mathcal{Y}}$ is to define $\check{y}_1, \ldots, \check{y}_M$ as follows

$$\begin{aligned}
\check{y}_1 &= \min_{i=1,\ldots,N} \bar{y}_i \\
\check{y}_M &= \max_{i=1,\ldots,N} \bar{y}_i \\
\check{y}_j &= \check{y}_1 + \left(\frac{\check{y}_M - \check{y}_1}{M-1}\right)(j-1),\ j = 2, \ldots, M-1,
\end{aligned}$$

where $\bar{y}_i$ are past samples of y. It is easy to see that larger values of M provide better approximations of $\mathcal{Y}$ at the expense of a larger computational burden. Thus, given z, the *discrete* empirical conditional distribution is defined at each point of $\check{\mathcal{Y}}$ as

$$\overline{\mathrm{ecp}}_{\gamma,c}\left(z, y\right) = \frac{e^{-cJ_\gamma(z,y)}}{\sum\limits_{j=1}^{M} e^{-cJ_\gamma(z,\check{y}_j)}}. \tag{3.1}$$

By construction,

$$\sum_{j=1}^{M} \overline{\mathrm{ecp}}_{\gamma,c}\left(z, \check{y}_j\right) = 1.$$

This discrete empirical conditional p.d.f. defines a conditioned probability distribution on $\check{\mathcal{Y}}$ that will be denoted as $\text{Prob}_{\check{\mathcal{Y}}|z}$. According to this discrete distribution, it is inferred that $\check{y}_\ell$ (i.e. the ℓ-th element of $\check{\mathcal{Y}}$) satisfies

$$\text{Prob}_{\check{\mathcal{Y}}|z}\{y \leq \check{y}_\ell\} = \sum_{j=1}^{\ell} \overline{\text{ecp}}_{\gamma,c}(z, \check{y}_j), \tag{3.2}$$

$$\text{Prob}_{\check{\mathcal{Y}}|z}\{y \geq \check{y}_\ell\} = \sum_{j=\ell}^{M} \overline{\text{ecp}}_{\gamma,c}(z, \check{y}_j). \tag{3.3}$$

Given z, $\gamma \geq 0$, $c \geq 0$ and $\tau \in (0,1)$, the empirical upper conditioned τ-quantile, denoted by y_τ^+, is defined as the *smallest* element of $\check{\mathcal{Y}}$ that satisfies

$$\text{Prob}_{\check{\mathcal{Y}}|z}\{y \leq y_\tau^+\} \geq 1 - \tau.$$

In a similar way, the empirical lower conditional τ-quantile, denoted by y_τ^-, is defined as the *largest* element of $\check{\mathcal{Y}}$ that satisfies

$$\text{Prob}_{\check{\mathcal{Y}}|z}\{y \geq y_\tau^-\} \geq 1 - \tau.$$

Then, the interval prediction for y is $[y_\tau^-, y_\tau^+]$. According to the discrete distribution $\text{Prob}_{\check{\mathcal{Y}}|z}$ and the definition of y_τ^- and y_τ^+

$$\text{Prob}_{\check{\mathcal{Y}}|z}\{y \in [y_\tau^-, y_\tau^+]\} \geq 1 - 2\tau.$$

What precedes illustrates how to compute the interval prediction $[y_\tau^-, y_\tau^+]$, for a given z, τ and pair (γ, c). See Algorithm 2 for a detailed description of the procedure.

Remark 3.1 (Conditioned median)**.** *Given the occurrence of z, a sensible estimation for y is the conditioned median, which can be approximated by the center of the interval $[y_{0.5}^-, y_{0.5}^+]$.*

The properties of the prediction intervals obtained using the procedure detailed above rely on the specific choice for γ and c, since they determine the underlining empirical distribution. Given $\tau \in (0,1)$, it is detailed in what follows how to obtain a pair $(\gamma_\tau^*, c_\tau^*)$ such that sharp interval estimations are obtained while meeting the probabilistic specifications (determined by τ). In order to circumvent the difficulty to generate i.i.d. samples, we consider a discrete probability described in a validation set

$$\mathcal{V} = \{(\bar{z}_1, \bar{y}_1), (\bar{z}_2, \bar{y}_2), \ldots, (\bar{z}_{N_\mathcal{V}}, \bar{y}_{N_\mathcal{V}})\},$$

where $N_\mathcal{V}$ is the number of samples in the validation set $\mathcal{V}$ and it is assumed to represent the true probability distribution.

Algorithm 2: Interval estimation $[y_\tau^-(z,\gamma,c), y_\tau^+(z,\gamma,c)]$.

Data: z, $\tau \in (0,1)$, $\gamma \geq 0$, $c \geq 0$, Z, Y, $\check{\mathcal{Y}}$.
Result: y_τ^-, y_τ^+.

1 Obtain the dissimilarity function (see Definition 2.1) for each element of $\check{\mathcal{Y}}$:

$$a_j = J_\gamma\left(z, \check{y}_j\right), \; j = 1, \ldots, M.$$

2 Compute the conditioned probabilities (see equation 3.1):

$$p_j = \overline{\text{ecp}}_{\gamma,c}\left(z, \check{y}_j\right) = \frac{e^{-ca_j}}{\sum_{\ell=1}^{M} e^{-ca_\ell}}, \; j = 1, \ldots, M.$$

3 Compute the indexes ℓ_τ^+ and ℓ_τ^- corresponding to the lower and upper conditioned τ-quantiles (see (3.2) and (3.3)):

$$\ell_\tau^+ = \text{smallest integer } \ell \text{ satisfying } \sum_{j=1}^{\ell} p_j \geq 1 - \tau,$$
$$\ell_\tau^- = \text{largest integer } \ell \text{ satisfying } \sum_{j=\ell}^{M} p_j \geq 1 - \tau.$$

4 Make $y_\tau^- = \check{y}_{\ell_\tau^-}$ and $y_\tau^+ = \check{y}_{\ell_\tau^+}$.

Consider now the role of parameter $c \geq 0$ in the discrete empirical conditioned distribution given in equation (3.1). On the one hand, the choice $c = 0$ provides a flat distribution in which each element of $\check{\mathcal{Y}}$ has a conditioned probability equal to $\frac{1}{M}$. On the other hand, large values of c provide narrow distributions centered around the point in $\check{\mathcal{Y}}$ that minimizes, given z, the dissimilarity function $J_\gamma(\cdot,\cdot)$, i.e. $\tilde{y}(z)$. Consequently, for a fixed value of γ, larger values of c reduce the size of the obtained interval at the expense of increasing the fraction of outputs that are not contained in the interval estimations. Therefore, given γ, the corresponding value for c should be chosen as the largest value of c that guarantees in the validation set that the obtained intervals contain the outputs with the desired probability.

From the discussion above, it is clear that the parameter c corresponding to a particular choice of $\gamma > 0$ (denoted c_γ) is determined by τ. As it is detailed in Algorithm 3, c_γ is chosen as the largest value of c (up to a given accuracy $\epsilon > 0$) that guarantees in the validation set that the obtained confidence intervals contain the outputs with the desired probability, that is, no smaller than $1 - 2\tau$.

Parameter $\gamma > 0$ can be obtained by maximizing the likelihood ratio which, for a

Algorithm 3: Optimal value of $c \geq 0$, for given $\gamma \geq 0$ and $\tau \in (0,1)$

Data: $\tau \geq 0$, $\gamma \geq 0$, $c_{\max} > 0$ and $\epsilon > 0$, Z, Y, $\check{\mathcal{Y}}$ and the validation set $\mathcal{V}$.
Result: c_γ.

$c_{\min} = 0$;
while $c_{\max} - c_{\min} \geq \epsilon$ **do**
  $c = \frac{1}{2}(c_{\max} + c_{\min})$;
  Compute, using Algorithm 2, the $N_{\mathcal{V}}$ intervals

$$I_i = [y_\tau^-(\bar{z}_i, \gamma, c), y_\tau^+(\bar{z}_i, \gamma, c)],\ i = 1, \ldots, N_{\mathcal{V}}.$$

  Make n_{viol}^+ equal to the number of violations of the upper constraints

$$\bar{y}_i \leq y_\tau^+(\bar{z}_i, \gamma, c),\ i = 1, \ldots, N_{\mathcal{V}},$$

  and n_{viol}^- equal to the number of violations of the lower constraints

$$\bar{y}_i \geq y_\tau^-(\bar{z}_i, \gamma, c),\ i = 1, \ldots, N_{\mathcal{V}}.$$

  if $\dfrac{\max\{n_{\text{viol}}^+, n_{\text{viol}}^-\}}{N_{\mathcal{V}}} < \tau$ **then**
    $c_{\min} = c$;
  else
    $c_{\max} = c$;
  end if
end while
$c_\gamma = c_{\min}$;

specific γ and corresponding c_γ, is defined as

$$L_\gamma = \sum_{i=1}^{N_\mathcal{V}} \log\left(\overline{\text{ecp}}_{\gamma,c}\left(\bar{z}_i, \bar{y}_i\right)\right).$$

Thus, using $\check{\mathcal{Y}} = \{\check{y}_1, \ldots, \check{y}_M\}$, the optimal value of γ is given by

$$\gamma_\tau^* \approx \arg\max_{\gamma\in\Gamma} \sum_{i=1}^{N_\mathcal{V}} \log\left(\frac{e^{-c_\gamma J_\gamma(\bar{z}_i,\bar{y}_i)}}{\sum_{j=1}^{M} e^{-c_\gamma J_\gamma(\bar{z}_i,\check{y}_j)}}\right), \tag{3.4}$$

where Γ is a set containing all the possible values considered for γ.

Remark 3.2. *Other criteria can be used to compute γ_τ^*. For example, γ_τ^* could be obtained by minimizing a cost function penalizing the average length of the intervals and/or the average prediction error with respect to the conditioned median introduced in Remark 3.1. However, explicitly minimizing the size of the intervals may translate into an increased violation rate when the validation set has not a sufficiently large number of samples, hence not being representative enough of the real distribution of y.*

3.1.2 Clarifying example: uniform distribution

A sample of 600 points in $\mathbb{R}$ is obtained from a uniform probability function with support $[0, 1]$. One half of the available points is used as a training set and the other half is used as a test set.

Figure 3.1 shows the empirical probability density functions estimated using different values of the parameters c and γ. In this case, $c = 1.5$ and $\gamma = 5$ is the pair that achieves the best fit for the distribution proposed in this example.

3.1.3 Numerical example: Lorenz attractor

The Lorenz attractor is a system of ODEs known for having chaotic solutions with certain values of the parameters of the system. The equations that define the system are the following

$$\begin{aligned} \frac{\mathrm{d}o}{\mathrm{d}t} &= \sigma(p - o) \\ \frac{\mathrm{d}p}{\mathrm{d}t} &= o(\rho - q) - p \\ \frac{\mathrm{d}q}{\mathrm{d}t} &= op - \beta q\,, \end{aligned} \tag{3.5}$$

where σ, ρ and β are real scalar parameters. In this example, these parameters take the values $\sigma = 10$, $\rho = 28$ and $\beta = 8/3$. Furthermore, in order to obtain the

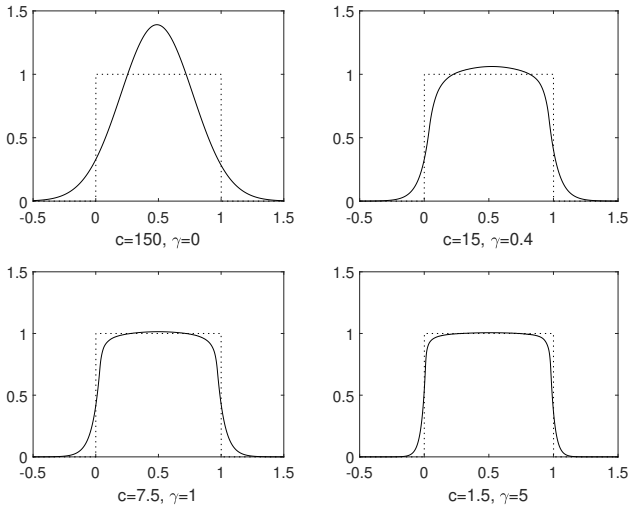


Figure 3.1: Estimated probability distribution functions.

necessary data, the ODEs have been integrated numerically with a fixed time step of $T_s = 0.1s$ and initial conditions $o_0 = 1$, $p_0 = 1$ and $q_0 = 1$. Here, it is considered the task of forecasting the one-step ahead value of o, i.e., $y_{k+1} = o_{k+1}$, using the two previous values of o, that is, the regressor vector will be $z_k = [y_k,\, y_{k-1}]^\top$. To start with, 2500 data points are considered, normalized in the $[0, 1]$ range. Different sizes for the data sets Z and Y are considered in this example (200, 350 and 500 points). The validation set $\mathcal{V}$ consists of 1000 data points and other 1000 data points are used as a test set, denoted by $\mathcal{T}$. Here, we consider the dissimilarity function of remark 2.3. The set Γ is taken from $[0, 3]$ using a 0.1 sampling step. On the other hand, $\check{\mathcal{Y}}$ is obtained from a grid of equally distant points in the interval $[-0.1893, 1.2298]$ sampled with a 1.4191×10^{-4} step.

Two different techniques will be considered as benchmarks. The first one is quantile regression [65], [66], a classical method for the estimation of conditioned quantiles. The second one is the set-membership method described in [49, 50]. This technique is a well-known method to generate interval bounds for a time series (usually produced from a dynamical system). For the sake of comparison, to guarantee that these bounds contain the output within a prescribed probability, the parameters ϵ, γ of [49] are chosen so that the resulting empirical probability of containing a sample within the validation set $\mathcal{V}$ is no smaller than $1 - 2\tau$.

The numerical results of the proposed approach and the two benchmark techniques are shown in table 3.1 for a $[o_{5\%}, o_{95\%}]$ interval, that is, $\tau = 0.05$, and in table 3.2 for $[o_{10\%}, o_{90\%}]$ ($\tau = 0.1$). The output of the test data should be contained in the first interval with a probability of 0.9 (0.8 for the second interval). The optimal value for γ has been chosen by maximizing the likelihood function L_γ (see equation (3.4) and figure 3.3).

Table 3.1: Results for the Lorenz Attractor, interval $[o_{5\%}, o_{95\%}]$.

	Proposed		**QR**		**SM**	
N	**E. P.**	**I. W.**	**E. P.**	**I. W.**	**E. P.**	**I. W.**
200	0.9140	2.0578	0.8290	3.0965	0.8960	2.9378
350	0.8990	1.9352	0.8260	3.0550	0.9100	2.4773
500	0.9070	2.0223	0.8410	3.2450	0.9120	2.5671

Table 3.2: Results for the Lorenz Attractor, interval $[o_{10\%}, o_{90\%}]$.

	Proposed		**QR**		**SM**	
N	**E. P.**	**I. W.**	**E. P.**	**I. W.**	**E. P.**	**I. W.**
200	0.8060	1.6053	0.7450	2.2776	0.8160	2.4248
350	0.8060	1.6164	0.7270	2.0607	0.7900	1.9797
500	0.8100	1.6195	0.7630	2.4371	0.8100	2.0021

The empirical probability in the case of the quantile regression clearly does not meet the probabilistic specifications. In the case of the proposed approach and the set-membership method, the observed fraction of outputs that fall into the predicted intervals is much closer to the desired one. Note that, for all techniques, the obtained empirical probability can be below the desired probability. This could be solved relying on a probabilistic scaling scheme [63] or probabilistic validation schemes [133], [117].

Regarding the interval width, the proposed approach clearly manages to obtain the smallest intervals for each data set. For the $[o_{5\%}, o_{95\%}]$ case, in comparison to the set-membership method, the mean interval width is 24.35% smaller. Also, it is 35.96% smaller with respect to the mean interval width obtained by means of quantile regression. On the other hand, for the $[o_{10\%}, o_{90\%}]$ case, the intervals obtained with the proposed technique are 23.75% smaller than those obtained by means of the set-membership method and 28.21% smaller than those obtained using quantile regression techniques. Taking into account the empirical probability values and the interval widths, it is possible to conclude that the proposed approach obtains the best results.

Finally, in figure 3.2, a fraction of the test set $\mathcal{T}$ is shown along with the computed intervals $[o_{5\%}, o_{95\%}]$ of the proposed approach. Note that the intervals are wider when there are trend changes in the output. Furthermore, figure 3.3 shows an example of the value of the maximum likelihood ratio L_γ as a function of γ (in this case for the data set of 200 points and interval $[o_{5\%}, o_{95\%}]$).

3.1.4 Numerical example: Dow Jones industrial average index

The example presented in the previous chapter will be used again to show the effectiveness of the proposed approach for interval forecasting. The parameters of

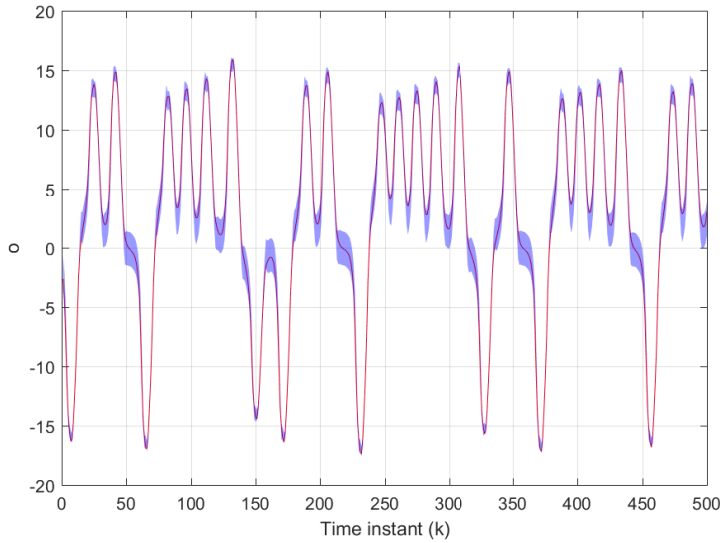


Figure 3.2: Test set and computed intervals for Lorenz Attractor (interval $[o_{5\%}, o_{95\%}]$).

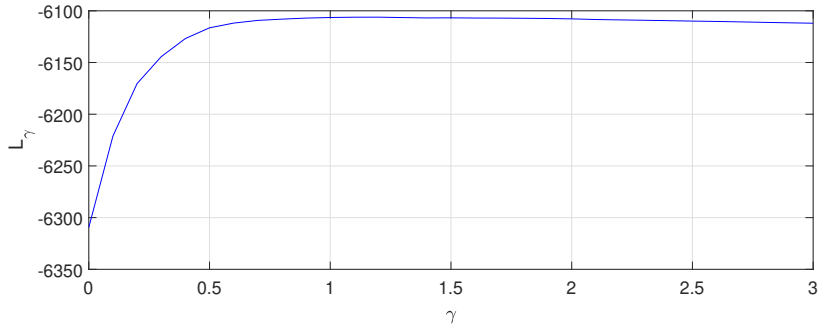


Figure 3.3: Maximum likelihood ratio as a function of γ.

the algorithms are $N_\Omega = 250$, $M = 1000$, $\check{y}_1 = 6684.3$, $\check{y}_M = 19445$, $c_{\max} = 15$. As in the previous example, the dissimilarity function used is the one corresponding to remark 2.3. The set Γ is taken from $[0, 5]$ using a 0.5 sampling step. The regressor is compounded as in chapter 2. The 10-th and 90-th percentiles (i.e. $\tau = 0.1$) corresponds to the desired probability for the price intervals. Note that, in order to compute the intervals with the proposed methodology, it is needed to consider each l-step prediction independently, obtaining l different predictors, each one for a certain l-step forecasting.

The results are shown in figures 3.4 and 3.5. There, the price intervals are presented as envelopes. Although these intervals are tight for $l = 1$, they become larger as l increases. This is straightforward due to the fact that the uncertainty increases as the prediction horizon does. Also, note that sometimes the real price is not included within the computed price interval. This is congruent to the fact that the probability of belonging to the interval is 80%.

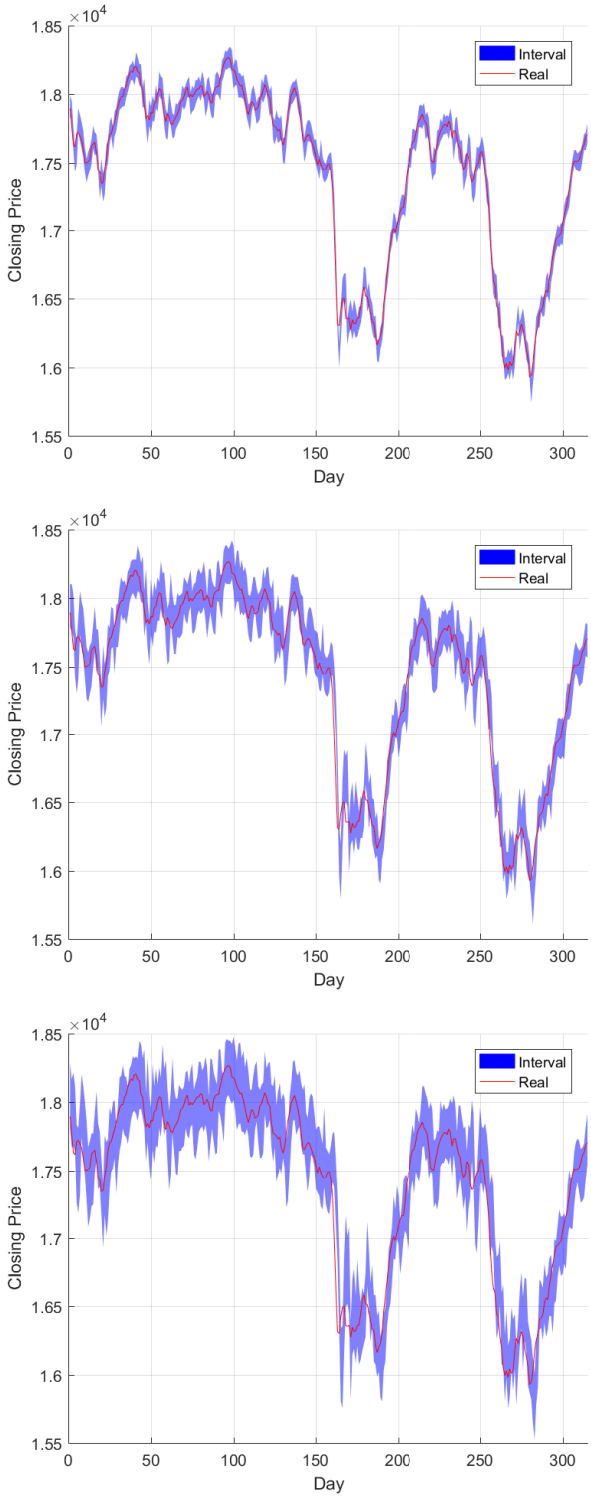


Figure 3.4: Price intervals for 1 to 3 days (5-day EMA).

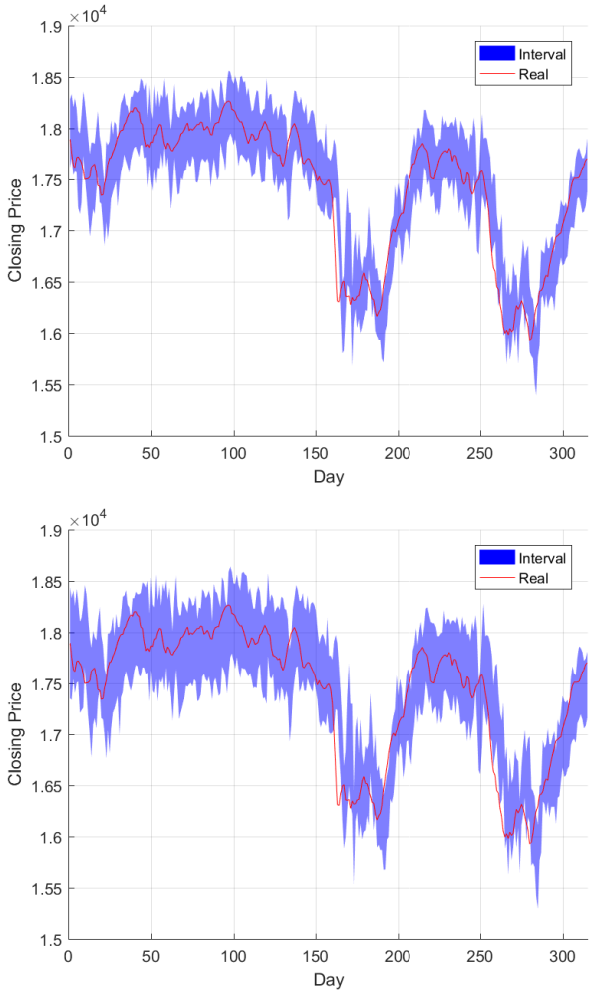


Figure 3.5: Price intervals for 4 and 5 days (5-day EMA).

As a baseline approach, a quantile regression approach is chosen to validate the obtained results. In table 3.3 it is shown the empirical probabilities and interval width of the proposed approach and the quantile regression scheme. Although the proposed approach contains the real price with a higher probability than the desired one, the quantile regression fails to obtain intervals meeting the specified probability for any l.

Table 3.3: Empirical probability and average interval width.

	Empirical probability		**Average interval width**	
k	**Proposed**	**Q. regression**	**Proposed**	**Q. regression**
1	0.8679	0.6006	162.4802	99.8091
2	0.8459	0.6101	315.7218	190.5443
3	0.8553	0.6321	477.8004	287.5774
4	0.8648	0.6761	638.8747	397.3780
5	0.8805	0.6950	813.5661	489.7816

3.2 Multivariate case

In this section, the problem of multidimensional systems or time-series is tackled, providing prediction regions for the multidimensional output. As the scheme proposed in the previous section for univariate outputs is based on a numerical integration in a unidimensional space, the generalization to the multidimensional setting is not trivial because of the well-known complexity of high-dimensional numerical integration. In what follows, it is shown how to obtain prediction regions in multidimensional spaces without resorting to numerical integration, providing a fundamental advantage with respect to the previous results.

First, implicit regions computed by means of the dissimilarity function are proposed. After that, an ellipsoidal approximation is proposed. Both approaches are tested in a numerical example.

3.2.1 Implicit regions

In order to characterize the desired prediction region, two issues have to be tackled: the choice of the region center and the computation of the region itself.

Choosing the center

In the previous chapter, $\tilde{y}(z)$ was denoted as the value of y that minimizes the dissimilarity function given z (see (2.7)). As a consequence of this, $\tilde{y}(z)$ can be considered as a good option to define the center of the region that will be obtained as it is the most likely value for the output given z, Z and Y.

This optimal value can be obtained as $\tilde{y}(z) = Y\lambda^*$ where λ^* corresponds to the

optimal solution of the strictly convex problem (2.8), that is

$$J_\gamma(z, Z) = \min_{\lambda_{(1)},\ldots,\lambda_{(N)}} (1-\gamma)\sum_{i=1}^{N}\lambda_{(i)}^2 + \gamma\sum_{i=1}^{N}|\lambda_{(i)}|$$
$$\text{s.t. } z = \sum_{i=1}^{N}\lambda_{(i)}\bar{z}_i$$
$$1 = \sum_{i=1}^{N}\lambda_{(i)}.$$

Computing the regions

In the previous section, it was shown that the dissimilarity function can be seen as a sort of surrogate of the probability distribution of y given z. Thus, this probability distribution peaks at $\tilde{y}(z)$ and decreases as the dissimilarity function increases. For that reason, the proposed prediction regions are defined as those points for which the dissimilarity function does not exceed more than a given factor α with respect to the value corresponding to the central prediction $\tilde{y}(z)$, that is, $J_\gamma^*(z)$. This is formally stated in the following definition.

Definition 3.3 (Prediction region)**.** For a given z, γ, data sets Z, Y and a tunable parameter $\alpha > 1$, the proposed prediction region is defined as the set

$$\Delta(z) = \left\{y : J_\gamma(z, y) \leq \alpha J_\gamma^*(z)\right\}, \tag{3.7}$$

that is, the points y that obtain a dissimilarity less or equal to $\alpha J_\gamma^*(z)$.

Since the dissimilarity function is convex in y, the obtained implicit regions are convex and can be used in different settings, e.g. in chance-constrained optimization problems. For example, in many cases, it is not necessary to compute a prediction region, but to verify if a certain point $\bar{y}$ belongs to it. This is an affordable task as it suffices to compute the dissimilarity $J_\gamma(z, \bar{y})$ and check that (3.7) holds.

Tuning the hyperparameters α and γ

The value of $\gamma \geq 0$ could be obtained in such a way that the prediction error corresponding to the central prediction in a validation set is minimized. For a given γ, smaller values of α make the regions smaller. Our objective is to obtain the smallest possible region that guarantees that a point $\bar{y}_j$ taken from a validation set $\mathcal{V}$ is contained in the computed region $\Delta(z_j)$ with a pre-specified probability $1-\tau$. Again, as in the univariate case, it is assumed that the validation set $\mathcal{V}$ represents the true probability distribution. Then, α is chosen so that it fulfills

$$\text{Prob}_{\mathcal{V}}(y \in \Delta(z)) \geq 1-\tau,$$

that is, guaranteeing that the fraction of points y in the validation set that fall out of the corresponding prediction region does not exceed τ. Thus, the number of points $\tilde{y}_j$ falling out should be no larger than $r_\tau = \lceil \tau N_\mathcal{V} \rceil$. Denote

$$\bar{\alpha}_{(j)} = \frac{J_\gamma(\bar{z}_j, \bar{y}_j)}{J^*_\gamma(\bar{z}_j)},\ j = 1, \ldots, N_\mathcal{V}.$$

Then, the value of α satisfying the probabilistic specification in the set $\mathcal{V}$ is the one corresponding to the r_τ largest value of $\bar{\alpha}_{(j)}$. If i.i.d. samples are available, one can resort to the concept of probabilistic scaling to choose the value of r_τ (see [61, 63, 146]).

3.2.2 Clarifying example: multivariate uniform distribution

To illustrate the role of γ, consider the easier case of a multivariate uniform distribution. As the example considered here is not a dynamic system, conditioned distributions are not taken into account. A number of samples $N = 500$ are extracted from a uniform distribution in $\mathbb{R}^2$ and gathered into a data set Y. This uniform distribution is generated considering that both dimensions are independent one with respect the other, with values ranging from 0 to 1. Then, the obtained distribution is rotated $\frac{\pi}{6}$ rad in order to misalign the previously computed distribution. The desired prediction region is

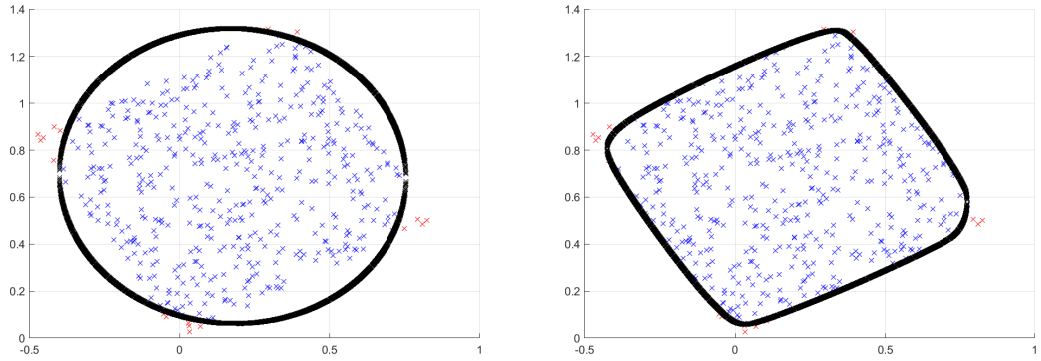
$$\Delta = \left\{ y : J_\gamma\left(y, Y\right) \leq \alpha J^*_\gamma \right\}.$$

Here, a finite family of two possible values of γ, denoted as $\Gamma = \{0, 0.8\}$ is considered. As it was stated in the previous sections, $\gamma = 0$ corresponds to the least-squares solution and thus an ellipsoid will be obtained. However, more appropriate shapes can be obtained with different values of γ. Also, note that the empirical expectation turns out to be the center of the region.

In this example, a value of $\tau = 0.05$ is considered, that is, the probability of the samples falling within the region should be 95%. The results are shown in figure 3.6. The blue crosses correspond to the points falling inside the region whereas the red crosses correspond to the points falling outside. It is easy to see that $\gamma = 0.8$ captures better the shape of the proposed probability distribution, leading to a tighter region.

3.2.3 Ellipsoidal prediction regions

Note that due to the term $\gamma \sum_{i=1}^{N} |\lambda_{(i)}|$ appearing in the definition of the dissimilarity function, the prediction regions cannot be easily computed. Instead, in order to check if a point y belongs to a prediction region, a convex optimization problem must be solved. In this section, by means of a quadratic upper bound on the dissimilarity function, it is possible to compute ellipsoidal regions that can be computed in an explicit way. The idea is to upper bound the absolute



(a) $\gamma = 0$ region. (b) $\gamma = 0.8$ region.

Figure 3.6: Prediction regions for different values of γ.

values $|\lambda_{(i)}|$ with scalar quadratic functions. This transforms the functional into a quadratic one and thus the optimization problem can be solved explicitly since the minimization of a convex quadratic function subject to linear constraints has an explicit solution. The following lemma makes possible to upper bound the absolute values.

Lemma 3.4 (Quadratic upper bound of $|\lambda_{(i)}|$)**.** Given a scalar $\lambda^c_{(i)}$ and $\nu > 0$,

$$|\lambda_{(i)}| \leq \frac{|\lambda^c_{(i)}| + \nu}{2} \left(\left(\frac{|\lambda_{(i)}|}{|\lambda^c_{(i)}| + \nu} \right)^2 + 1 \right), \quad \forall \lambda_{(i)} \in \mathbb{R}\,. \tag{3.8}$$

Proof. See Appendix B in [147].

In this case, $\lambda^c_{(i)}$ is chosen as the $\lambda^*_{(i)}$ used to obtain the central prediction $\tilde{y}(z)$ (see equation (2.9)). Note that the upper bound provided by lemma 3.4 is tight when $\lambda_{(i)} = \lambda^c_{(i)} = \lambda^*_{(i)}$ and $\nu \to 0$. Thus, ν is chosen as a small constant. Applying equation (3.8) to the optimization problem in (2.6), it is clear that $J_\gamma(z, y) \leq Q_\gamma(z, y)$, where

$$Q_\gamma(z, y) = \min_{\lambda_{(1)},\ldots,\lambda_{(N)}} \frac{1}{2}\lambda^\top H_\gamma \lambda + c_\gamma \tag{3.9a}$$

$$\text{s.t.} \quad \begin{bmatrix} Z \\ \mathbf{1} \end{bmatrix} \lambda = \begin{bmatrix} z \\ 1 \end{bmatrix} \tag{3.9b}$$

$$Y\lambda = y, \tag{3.9c}$$

H_γ is a diagonal matrix with entries

$$H_{\gamma|(i,i)} = 2(1-\gamma)\omega_{(i)} + \frac{\gamma}{|\lambda^c_{(i)}| + \nu}, \; i = 1, \ldots, N$$

and $c_\gamma = \gamma \sum_{i=1}^{N} \frac{|\lambda^c_{(i)}|+\nu}{2}$. The following lemma characterizes the new dissimilarity function $Q_\gamma(\cdot)$.

Lemma 3.5. Assume that $H_\gamma > 0$ and that the matrix $\left[\begin{array}{ccc} Z^\top & Y^\top & \mathbf{1}^\top \end{array}\right]^\top$ is full rank. Denote

$$\begin{aligned} Q_\gamma(z,y) = \min_{\lambda_{(1)},\ldots,\lambda_{(N)}} \; & \frac{1}{2}\lambda^\top H_\gamma \lambda + c_\gamma \\ \text{s.t.} \; & \left[\begin{array}{c} Z \\ \mathbf{1} \end{array}\right] \lambda = \left[\begin{array}{c} z \\ 1 \end{array}\right] \\ & Y\lambda = y, \end{aligned}$$

$$\begin{aligned} y^* &= \arg\min_y \; Q_\gamma(z,y), \\ Q^*_\gamma(z) &= Q_\gamma(z,y^*). \end{aligned}$$

Then,

$$\begin{aligned} y^* &= \Gamma_{1,2}^\top \Gamma_{1,1}^{-1} \left[\begin{array}{c} z \\ 1 \end{array}\right], \; Q^*_\gamma(z) = \frac{1}{2}\left[\begin{array}{c} z \\ 1 \end{array}\right]^\top \Gamma_{1,1}^{-1} \left[\begin{array}{c} z \\ 1 \end{array}\right] + c_\gamma \\ Q_\gamma(z,y) &= Q^*_\gamma(z) + \frac{1}{2}(y-y^*)^\top \Phi_{2,2}(y-y^*). \end{aligned}$$

where

$$\begin{aligned} \Gamma_{1,1} &= \left[\begin{array}{c} Z \\ \mathbf{1} \end{array}\right] H_\gamma^{-1} \left[\begin{array}{c} Z \\ \mathbf{1} \end{array}\right]^\top, \; \Gamma_{1,2} = \left[\begin{array}{c} Z \\ \mathbf{1} \end{array}\right] H_\gamma^{-1} Y^\top, \\ \Gamma_{2,2} &= Y H_\gamma^{-1} Y^\top, \; \Phi_{2,2} = \left(\Gamma_{2,2} - \Gamma_{1,2}^\top \Gamma_{1,1}^{-1} \Gamma_{1,2}\right)^{-1}. \end{aligned}$$

Proof. First, note that the optimization problem that defines $Q(z,y)$ is always feasible because of the full rank assumption on $\left[\begin{array}{ccc} Z^\top & Y^\top & \mathbf{1}^\top \end{array}\right]^\top$. Moreover, because of the definite positiveness of H_γ, the problem is feasible and the optimal value for λ is unique. For notational convenience, the following definitions are made

$$A_1 = \left[\begin{array}{c} Z \\ \mathbf{1} \end{array}\right], \; A_2 = Y, \; A = \left[\begin{array}{c} A_1 \\ A_2 \end{array}\right], \; b = \left[\begin{array}{c} z \\ 1 \end{array}\right].$$

Consider now a related optimization problem in which the equality constraint

$A_2\lambda = y$ is removed:

$$Q^*_\gamma(z) = \min_{\lambda_{(1)},\ldots,\lambda_{(N)}} \frac{1}{2}\lambda^\top H_\gamma \lambda + c_\gamma$$
$$\text{s.t } A_1\lambda = b.$$

Then, it is clear that $Q^*_\gamma(z) \leq Q_\gamma(z,y)$ for every y. The optimal vector λ^* corresponding to the optimization problem that defines $Q^*_\gamma(z)$ can be directly obtained from the the KKT conditions (see subsection 10.1.1 in [139]):

$$\lambda^* = H_\gamma^{-1}A_1^\top(A_1H_\gamma^{-1}A_1^\top)^{-1}b.$$

This leads to

$$Q^*_\gamma(z) = \frac{1}{2}(\lambda^*)^\top H_\gamma\lambda^* + c_\gamma = \frac{1}{2}b^\top(A_1H_\gamma^{-1}A_1^\top)^{-1}b + c_\gamma = \frac{1}{2}b^\top\Gamma_{1,1}^{-1}b + c_\gamma.$$

Denote now

$$y^* = A_2\lambda^* = A_2H_\gamma^{-1}A_1^\top(A_1H_\gamma^{-1}A_1^\top)^{-1} = \Gamma_{1,2}^\top\Gamma_{1,1}^{-1}b.$$

From $A_1\lambda^* = b$, it is easy to see that λ^* is also a feasible solution for the optimization problem corresponding to $Q_\gamma(z,y^*) = Q_\gamma(z,A_2\lambda^*)$. Thus,

$$Q_\gamma(z,y^*) = Q^*_\gamma(z).$$

This, together with inequality $Q^*_\gamma(z) \leq Q_\gamma(z,y)$, $\forall y$, provide the first two claims of the lemma.

Denote λ^*_y as the optimal value for λ in the optimization problem that provides $Q_\gamma(z,y)$. Using again the KKT conditions, the value of λ^*_y can be computed as

$$\lambda^*_y = H_\gamma^{-1}A^\top(AH_\gamma^{-1}A^\top)^{-1}\begin{bmatrix} b \\ y \end{bmatrix}.$$

Thus,

$$\begin{aligned} Q_\gamma(z,y) &= \frac{1}{2}\lambda_y^{*^\top}H_\gamma\lambda^*_y + c_\gamma = \frac{1}{2}\begin{bmatrix} b \\ y \end{bmatrix}^\top(AH_\gamma^{-1}A^\top)^{-1}\begin{bmatrix} b \\ y \end{bmatrix} + c_\gamma \\ &= \frac{1}{2}\begin{bmatrix} b \\ y \end{bmatrix}^\top\begin{bmatrix} \Gamma_{1,1} & \Gamma_{1,2} \\ \Gamma_{1,2}^\top & \Gamma_{2,2} \end{bmatrix}^{-1}\begin{bmatrix} b \\ y \end{bmatrix} + c_\gamma. \end{aligned}$$

Now, making

$$\begin{bmatrix} \Gamma_{1,1} & \Gamma_{1,2} \\ \Gamma_{2,1} & \Gamma_{2,2} \end{bmatrix}^{-1} = \begin{bmatrix} \Phi_{1,1} & \Phi_{1,2} \\ \Phi_{1,2}^\top & \Phi_{2,2} \end{bmatrix},$$

it is clear that

$$Q_\gamma(z,y) = \frac{1}{2}\begin{bmatrix} b \\ y \end{bmatrix}^\top\begin{bmatrix} \Phi_{1,1} & \Phi_{1,2} \\ \Phi_{1,2}^\top & \Phi_{2,2} \end{bmatrix}\begin{bmatrix} b \\ y \end{bmatrix} + c_\gamma\,.$$

It is well known that every quadratic function $q(y)$ with hessian Φ_q can be rewriten as $q(y) = q(y^*) + \frac{1}{2}(y-y^*)^\top \Phi_q (y-y^*)$. Thus,

$$Q_\gamma(z,y) = Q^*_\gamma(z) + \frac{1}{2}(y-y^*)^\top \Phi_{2,2}(y-y^*).$$

The matrix $\Phi_{2,2}$ can be obtained from the inverse of the partitioned matrix Γ using Schur complements (see e.g. Subsection 4.3.4 in [148]), that is

$$\Phi_{2,2} = \left(\Gamma_{2,2} - \Gamma_{1,2}^\top \Gamma_{1,1}^{-1} \Gamma_{1,2}\right)^{-1}.$$

This completes the proof. ■

By means of the new dissimilarity function $Q_\gamma(z,y)$, the following ellipsoidal regions are defined

$$\begin{aligned} \mathcal{E}_{\gamma,\alpha}(z) &= \left\{y : Q_\gamma(z,y) \leq \alpha Q^*_\gamma(z)\right\} \\ &= \left\{y : (y-y^*)^\top \Phi_{2,2}(y-y^*) \leq 2(\alpha-1)Q^*_\gamma(z)\right\}, \end{aligned}$$

where the values for $y^*, \Phi_{2,2}$ and $Q^*_\gamma(z)$ are given in Lemma 3.5.

Note that, given $\gamma \geq 0$, the optimal value for α could be obtained by means of a validation set $\{(\bar{z}_j, \bar{y}_j)\}_{j=1}^{N_\mathcal{V}}$ as in the case of implicit regions. In this case, the scalars $\bar{\alpha}_{(j)}$ are defined as

$$\bar{\alpha}_{(j)} = \frac{Q_\gamma(\bar{z}_j, \bar{y}_j)}{Q^*_\gamma(\bar{z}_j)},\ j = 1, \ldots, N_\mathcal{V}.$$

3.2.4 Numerical results

As an example, the application of the developed methodology is proposed to compute probabilistic prediction regions for the predictions of the outputs of a multivariable system. In this example, the forecasting technique is based on a Neural Network (NN) that it is trained to learn the dynamics of an autonomous uncertain nonlinear system. In this case, the NN is comprised of 2 hidden layers of 10 neurons each one. Furthermore, the inputs of the NN are the last 5 outputs of the system. The outputs of the last layer are used as the regressor z. The system to be forecasted using the NN is the non-linear model of a towing kite presented in [149]. This system has three states, which corresponds to the angles of the kite, θ, ϕ and φ, a control input u and two uncertain parameters. The set of ODEs governing the system are

$$\begin{aligned} \dot{\theta} &= \frac{v_a}{L_T}\left(\cos\varphi - \frac{\tan\theta}{E}\right), \\ \dot{\phi} &= -\frac{v_a}{L_T \sin\theta}\sin\varphi, \\ \dot{\varphi} &= \frac{v_a}{L_T}u + \dot{\phi}\cos\theta, \end{aligned}$$

where

$$v_a = v_0\, E \cos\theta,$$
$$E = E_0 - 0.028u.$$

From this system of equations, the parameter L_T (length of the tether) is considered to be known whereas v_0 (wind speed) and E_0 (base glide ratio) are considered to be uncertain parameters. Note that these uncertainties affect the states θ, ϕ and φ trough the effect of the wind (v_a) and the glide ratio (E). Only the states θ and ϕ are considered as measurable outputs. These outputs have an additive correlated gaussian noise so that

$$\mathcal{N} \sim (\mu, \Sigma):\ \mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix},\ \Sigma = \begin{bmatrix} 0.01 & -0.0085 \\ -0.0085 & 0.01 \end{bmatrix}.$$

As the system is considered to be autonomous, it is assumed that the control input u is computed following a certain stabilizing control policy whose exact nature is irrelevant for this paper. Thus, it is assumed that some stable past trajectories of the system for different values of the base glide ratio E_0 and the wind speed v_0 are available. These are used to build the data sets Z, Y. The sample time is chosen as $T_s = 0.15s$. Two different data sets are considered in this example, one with $N = 250$ samples and another one with $N = 500$ samples. Also, a validation set $\mathcal{V}$ compounded of $N_{\mathcal{V}} = 500$ samples to compute the optimal values of γ and α is considered. On the other hand, there is also a test set $\mathcal{T}$ comprised of $N_{\mathcal{T}} = 1000$ samples to compare the obtained results with other baseline strategies. The finite family of γ is compounded of the values $\Gamma = [0,\ 0.2,\ 0.4,\ 0.6,\ 0.8]$.

Three baseline strategies are considered for the sake of comparison. First, a quantile regression (QR) approach. This technique is well established for the single output case. In order to obtain regions, a probabilistic interval of probability τ is obtained for each output independently and then combined to form a box-shaped region. However, combining two intervals of probability τ does not turn out to be a prediction region of probability τ. Actually, a lower probability will be attained. To tackle this problem, a back-off parameter will be calculated so that the computed boxes fulfill the empirical probability in the validation set $\mathcal{V}$. The second one is based on GPs. Here, the matlab function "fitgrp" is used to obtain a model for each output independently. This function automatically finds an appropriate Kernel function for the input-output data and optimizes the values of the hyperparameters. Then, using "predict", it is possible to obtain intervals of a specified probability τ. Same as before, it is needed to compute a back-off parameter in the validation set $\mathcal{V}$ to achieve the desired empirical probability. Finally, an implementation of the prediction regions based on Inverse Regression (IR) of [82] is considered.

The results for $\tau = 0.1$ and $\tau = 0.2$ are shown in Table 3.4. It can be seen how the implicit regions of the proposed approach and the approximation ellipsoids

Table 3.4: Area of the regions and empirical probabilities (E.P.) obtained for the proposed methodology, the approximation with ellipsoidal regions, Gaussian processes (GPs), quantile regression (QR) and inverse regression (IR) .

		Proposed		**Approximation**		**GPs**		**QR**		**IR**	
	N	**Area**	**E.P.**	**Area**	**E.P.**	**Area**	**E.P.**	**Area**	**E.P.**	**Area**	**E.P.**
$\tau = 0.1$	250	0.1695	0.9200	0.1885	0.9170	0.2235	0.9250	0.5839	0.9120	0.4331	0.9031
	500	0.1551	0.9260	0.1656	0.9170	0.2092	0.9240	0.2645	0.8960	0.4101	0.9043
$\tau = 0.2$	250	0.1119	0.8070	0.1176	0.8070	0.1518	0.8450	0.2341	0.8550	0.2466	0.7446
	500	0.1053	0.8150	0.1142	0.8260	0.1403	0.8310	0.1590	0.8080	0.2335	0.7454

outperform the baselines considered for both values of τ and different data set sizes. That is, the proposed approach obtains regions of considerably smaller area while still fulfilling the specified probability given by τ (i.e. E.P. $\geq 1 - \tau$). This means that the proposed approach provides regions of smaller uncertainty in comparison to the baselines considered.

3.3 Conclusions

This chapter presented methods to obtain interval predictions and prediction regions, which can be considered as an extension of the previous chapter where only the expected value of the output of a time-series or a dynamic system was obtained. First, a method based on obtaining an empirical conditioned probability density function was proposed. However, this method turned out to be computationally expensive due to the fact that it required the numerical integration of this probability density function for every time instant. Two numerical examples were proposed: the forecasting of the closing price of the Dow Jones, which was presented in the previous chapter, and the prediction of the Lorenz attractor, which is a dynamic system known for its chaotic behaviour. Some comparisons are made with respect to some baseline techniques such as quantile regression and set-membership methods. In both examples, the proposed approach obtained better results.

In the second half of the chapter, it was proposed a method that not only allows us to obtain prediction regions of multivariate systems or time-series but also reduces considerable the computational burden. This is due to the fact that the second method no longer needs the integration of the empirical conditioned density function. Also, a method to obtain ellipsoidal approximations of such regions was proposed. Finally, a prediction example of a kite system was proposed. Other approaches including GPs, IR, etc. were considered as baselines. It was shown that the proposed approach and the approximated ellipsoidal regions achieved the smallest size while fulfilling the desired probabilistic specifications.

Part II

Kriging-based identification

Chapter 4

State-space kriging for autonomous systems

In part I, some methods to make predictions of time-series or dynamical systems based on dissimilarity functions were proposed. These included l-steps ahead forecastings, interval predictions, predictions regions, etc.

From now on, the objective will be to obtain a dynamic model of a system given some past data of its outputs and inputs. In the case of autonomous systems only past outputs will be considered, whereas for non-autonomous systems past inputs will be considered as well. The state-space of this new model will be compounded of the vector of weights λ appearing in the dissimilarity function presented in part I. Two different strategies are proposed, a linear time variant approach based on the weighting of the local data within the dissimilarity function and a kernel-based approach where the dissimilarity function is slightly modified to accommodate the kernel trick. Finally, some numerical examples are presented to show the effectiveness of the proposed approaches.

4.1 Dynamic kriging

Consider an autonomous discrete nonlinear system

$$x_{k+1} = h(x_k) \tag{4.1a}$$

$$y_k = g(x_k), \tag{4.1b}$$

where k is the time instant, $x_k \in \mathbb{R}^{n_x}$ is the state of the system, $y_k \in \mathbb{R}^{n_y}$ is the output of the system, whereas $h(\cdot)$ and $g(\cdot)$ are unknown nonlinear functions such that $h(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ and $g(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$.

The objective of this section is to obtain a model of the outputs of (4.1). It is assumed that the only available data are the measurable outputs. The regressor z_k is defined as a time delay embedding vector $z_k \in \mathbb{R}^{n_z}$ containing the

n_p past outputs of the system. That is, $z_k = [y_k^\top, y_{k-1}^\top, ..., y_{k-n_p+1}^\top]^\top \in \mathbb{R}^{n_z}$, where $n_z = n_p n_y$. Also, note that z_k^+ is denoted as the successor of z_k, i.e. $z_k^+ = [y_{k+1}^\top, y_k^\top, ..., y_{k-n_p+2}^\top]^\top \in \mathbb{R}^{n_z}$.

From now on, assume that some historical data of the plant is stored in a database in the form of matrices:

$$\begin{aligned} D &= \left[\begin{array}{cccc} \bar{z}_1 & \bar{z}_2 & \dots & \bar{z}_N \end{array} \right], \\ D^+ &= \left[\begin{array}{cccc} \bar{z}_1^+ & \bar{z}_2^+ & \dots & \bar{z}_N^+ \end{array} \right], \end{aligned}$$

where $N > n_z$ is the number of data points, $\bar{z}$ refers to a sample of z and matrix D^+ is the successor of D. The indexes of the columns of D and D^+ do not refer to the sample time, but to the position in the matrix. Therefore, $\bar{z}_{i+1}$ is not necessarily the successor sample of $\bar{z}_i$. At sample time k, an estimation of the successor of z_k, denoted as $\tilde{z}_{k+1}$, can be obtained by a linear combination of the columns of matrix D^+ using the vector of optimal weights $\lambda_k^* \in \mathbb{R}^N$ obtained from an optimization problem similar to those shown in section 2, i.e.

$$\tilde{z}_{k+1} = D^+ \lambda_k^*.$$

Considering definition 2.1 and assuming that $\gamma = 0$, the optimization problem from which λ_k^* is obtained can be posed as

$$\lambda_k^* = \arg\min_{\lambda_k} \quad \lambda_k^\top H_1 \lambda_k \tag{4.2a}$$

$$\text{s.t.} \quad \left[\begin{array}{c} D \\ \mathbf{1} \end{array} \right] \lambda_k = \left[\begin{array}{c} z_k \\ 1 \end{array} \right], \tag{4.2b}$$

where $H_1 \in \mathbb{R}^{N \times N}$ is a positive definite weighting matrix and $\mathbf{1}$ a row vector with all its components equal to 1. Forcing the components of λ_k to sum one is equivalent to including a bias term in the estimation process. The simplest choice is to make H_1 equal to the identity matrix $\mathrm{I_N}$ (see remark 2.2). This optimization problem can be rewritten as

$$\lambda_k^* = \arg\min_{\lambda_k} \quad \lambda_k^\top H_1 \lambda_k \tag{4.3a}$$

$$\text{s.t.} \quad C\lambda_k = b\,, \tag{4.3b}$$

with

$$C = \left[\begin{array}{c} D \\ \mathbf{1} \end{array} \right], \quad b = \left[\begin{array}{c} z_k \\ 1 \end{array} \right].$$

In order to guarantee that any point in $\mathbb{R}^{n_z+1}$ can be expressed as a linear combination of the columns of C, it is assumed that matrix C is full row rank. This equality constrained quadratic problem has an analytic solution that can be obtained computing the Lagrangian and its derivative:

$$\mathcal{L}(\lambda_k, \nu) = \lambda_k^\top H_1 \lambda_k + \nu^\top (C\lambda_k - b)$$

$$\frac{\mathrm{d}}{\mathrm{d}\lambda}\mathcal{L}(\lambda_k, \nu) = 2H_1\lambda_k + C^\top \nu\,,$$

where ν is the dual variable associated with the equality constraint. From the Karush-Kuhn-Tucker (KKT) conditions:

$$2H_1\lambda_k^* + C^\top \nu^* = 0\,, \tag{4.4}$$

which leads to

$$\lambda_k^* = \frac{-H_1^{-1}C^\top \nu^*}{2}.$$

Pre-multiplying this equality by C, and taking into account that $C\lambda_k^* = b$, the following is obtained

$$\nu^* = -2\left(CH_1^{-1}C^\top\right)^{-1} b,$$

which applied to equation (4.4) yields

$$\lambda_k^* = H_1^{-1}C^\top \left(CH_1^{-1}C^\top\right)^{-1} \begin{bmatrix} z_k \\ 1 \end{bmatrix}.$$

In order to predict z_{k+d}, with $d > 1$, one could use this approach in a recursive way. That is, the i-th ahead prediction $\tilde{z}_{k+i}$ could be used to compute

$$\lambda_{k+i}^* = H_1^{-1}C^\top \left(CH_1^{-1}C^\top\right)^{-1} \begin{bmatrix} \tilde{z}_{k+i} \\ 1 \end{bmatrix},$$

and thus obtaining $\tilde{z}_{k+i+1} = D^+\lambda_{k+i}^*$. In the next section, it is proposed a modification of this *naive* recursive method. The novel methodology relies on a time-varying state-space modelling of the optimal weighting vector parameter λ_k^*.

4.2 Linear state-space kriging

Suppose that the prediction $\tilde{z}_{k+1}$ of z_{k+1} is obtained from $\tilde{z}_{k+1} = D^+\lambda_k^*$, where the sum of the components of λ_k^* is assumed to be equal to one. In order to model how the dynamics of the optimal vector of weights λ_{k+1}^* depends on λ_k^*, a regularization term is added to optimization problem (4.2) that penalizes the difference between λ_{k+1}^* and λ_k^*. In this way, vector λ_{k+1}^* not only fulfills the required equality constraints, but also does not depart excessively from λ_k^*. This will reduce the sensitivity to noise of the identified dynamics. Thus, given $\tilde{z}_{k+1}$, λ_{k+1}^* is obtained from

$$\begin{aligned} \lambda_{k+1}^* = \arg\min_{\lambda_{k+1}} \quad & (\lambda_{k+1} - \lambda_k^*)^\top H_2(\lambda_{k+1} - \lambda_k^*) + \lambda_{k+1}^\top H_1 \lambda_{k+1} \\ \text{s.t.} \quad & \begin{bmatrix} D \\ \mathbf{1} \end{bmatrix} \lambda_{k+1} = \begin{bmatrix} \tilde{z}_{k+1} \\ 1 \end{bmatrix}, \end{aligned}$$

where $H_2 \in \mathbb{R}^{N\times N}$ is chosen as the identity matrix multiplied by a certain scalar that could be selected by cross-validation [1, §16.5]. Because of the assumptions on λ_k^*, the previous optimization problem can be rewritten as

$$\begin{aligned}\lambda_{k+1}^* = \arg\min_{\lambda_{k+1}} \quad & (\lambda_{k+1}-\lambda_k^*)^\top H_2(\lambda_{k+1}-\lambda_k^*) + \lambda_{k+1}^\top H_1 \lambda_{k+1} \\ \text{s.t.} \quad & \begin{bmatrix} D \\ \mathbf{1} \end{bmatrix} \lambda_{k+1} = \begin{bmatrix} D^+ \\ \mathbf{1} \end{bmatrix} \lambda_k^* .\end{aligned}$$

Thus, the problem becomes

$$\lambda_{k+1}^* = \arg\min_{\lambda_{k+1}} \quad (\lambda_{k+1}-\lambda_k^*)^\top H_2(\lambda_{k+1}-\lambda_k^*) + \lambda_{k+1}^\top H_1 \lambda_{k+1} \tag{4.5a}$$

$$\text{s.t.} \quad C\lambda_{k+1} = C^+\lambda_k^*, \tag{4.5b}$$

with

$$C = \begin{bmatrix} D \\ \mathbf{1} \end{bmatrix}, \; C^+ = \begin{bmatrix} D^+ \\ \mathbf{1} \end{bmatrix}.$$

Note that λ_{k+1}^* is determined only by λ_k^* and matrices C and C^+. Optimization problem (4.5) can be rewritten as

$$\begin{aligned}\lambda_{k+1}^* = \arg\min_{\lambda_{k+1}} \quad & \frac{1}{2}\lambda_{k+1}^\top H \lambda_{k+1} + f^\top \lambda_{k+1} \\ \text{s.t.} \quad & C\lambda_{k+1} = b ,\end{aligned}$$

with $H = 2(H_1 + H_2)$, $f = -2H_2\lambda_k^*$ and $b = C^+\lambda_k^*$. Also, note that the constant term ${\lambda_k^*}^\top H_2 \lambda_k^*$ is removed because it does not affect the solution λ_{k+1}^*. The Lagrangian of this problem is given by

$$\mathcal{L}(\lambda_{k+1}, \nu) = \frac{1}{2}\lambda_{k+1}^\top H \lambda_{k+1} + f^\top \lambda_{k+1} + \nu^\top (C\lambda_{k+1} - b) ,$$

where ν is the dual variable associated with the equality constraint. The derivative of the Lagrangian is

$$\frac{\mathrm{d}}{\mathrm{d}\lambda_{k+1}} \mathcal{L}(\lambda_{k+1}, \nu) = H\lambda_{k+1} + f + C^\top \nu .$$

In the optimum, the derivative fulfills the KKT conditions [139, §10.1.1]. That is,

$$H\lambda_{k+1}^* + f + C^\top \nu^* = 0 ,$$

and thus

$$\lambda_{k+1}^* = -H^{-1}f - H^{-1}C^\top \nu^* . \tag{4.7}$$

Pre-multiplying both sides of last equality by C yields

$$b = C\lambda_{k+1}^* = -CH^{-1}f - CH^{-1}C^\top \nu^* ,$$

and thus $\nu^* = (CH^{-1}C^\top)^{-1}(-CH^{-1}f - b)$. Substituting this into equation (4.7), the following expression for λ^*_{k+1} is obtained

$$\lambda^*_{k+1} = H^{-1}C^\top \left(CH^{-1}C^\top\right)^{-1} \left(CH^{-1}f + b\right) - H^{-1}f.$$

Taking into account that $f = -2H_2\lambda^*_k$ and $b = C^+\lambda^*_k$, the state-space equation of λ^*_k can be written as

$$\lambda^*_{k+1} = A\lambda^*_k\,,$$

with

$$A = 2H^{-1}H_2 + H^{-1}C^\top(CH^{-1}C^\top)^{-1}(C^+ - 2CH^{-1}H_2).$$

Note that this means that λ_k follows linear dynamics. Thus, a new model for the outputs of system (4.1) has been obtained using historical data of these outputs. This new autonomous system allows us to compute the next values of λ and z. Note that, as only the first term of z_k is needed (i.e. the term corresponding to y_k), the model can be posed as

$$\begin{aligned} \lambda^*_{k+1} &= A\lambda^*_k \\ y_k &= Y\lambda^*_k, \end{aligned}$$

where Y denotes a matrix compounded of only the first rows of D, that is, Y is a matrix containing the samples $\bar{y}_i$.

4.2.1 Initial condition

The initial vector of optimal weights λ^*_0 is computed by means of the optimization problem in (4.2) substituting k by 0, that is

$$\begin{aligned} \lambda^*_0 = \arg\min_{\lambda_0} \quad & \lambda_0^\top H_1 \lambda_0 \\ \text{s.t.} \quad & \begin{bmatrix} D \\ \mathbf{1} \end{bmatrix} \lambda_0 = \begin{bmatrix} z_0 \\ 1 \end{bmatrix}. \end{aligned}$$

4.2.2 Local-data approach

Note that the previous model is linear and time-invariant as the matrix A is constant. However, it is possible to weight the points in the data set with respect to z, which would encourage the use of local data and thus provide better results when identifying nonlinear systems.

This can be done by choosing H_1 appropriately. In remark 2.2, it was shown that it is possible to consider a vector ω within the dissimilarity function to weight different elements in the data set. Here, the squared Euclidean distance is chosen to measure the dissimilarity of a certain z to each point of the data set. That is,

$$\omega(z) = \begin{bmatrix} (z-\bar{z}_1)^\top (z-\bar{z}_1) \\ \vdots \\ (z-\bar{z}_N)^\top (z-\bar{z}_N) \end{bmatrix}, \tag{4.8}$$

and thus

$$H_1 = \mathrm{diag}(\omega(z)),$$

where $\mathrm{diag}(\cdot)$ denotes a diagonal matrix $\mathbb{R}^N \times \mathbb{R}^N$ whose non-zero entries are the components of the input vector. Note that, because of this change, this vector $\omega(z)$ needs to be computed at each time instant k, leading to a different matrix A for every k, and thus leading to LTV dynamics for λ, that is

$$\begin{aligned} \lambda^*_{k+1} &= A_k \lambda^*_k \\ y_k &= Y \lambda^*_k. \end{aligned}$$

4.3 Kernel-based state-space kriging

Kernels functions are widely used in the machine learning field. For example, Support Vector Machines (SVM) are supervised learning models that classify linearly separable data. That is, given a cloud of data points (each point belonging to a certain class), the problem of classification is defined as finding an hyperplane that divide the data into two sets. However, when considering nonlinear relations within the data, it is not possible to obtain an hyperplane that separates the data. In order to be able to classify this kind of data, it is needed to project this data into a high-dimensional space where it may become linearly separable.

Thanks to the so-called kernel trick (see chapter 3 in [150]), it is possible to operate in a high-dimensional feature space without computing the coordinates of such space. Instead, it is only needed to compute the inner product of the images of all pairs of the data samples, which bypasses the computation of the coordinates of the feature space that may be cumbersome or even impossible.

Thus, instead of using local data as it was presented in the previous section, one could resort to the use of kernels in order to model the nonlinear dynamics of the system. Also, the use of kernels will allow us to compute the matrices of the system only once, unlike the local data approach where the matrices needed to be recomputed at each sample time.

In order to include the kernels, the optimization problem is modified, becoming

$$\begin{aligned} \lambda^*_{k+1} = \arg\min_{\lambda_{k+1}} \; & (\lambda_{k+1} - \lambda^*_k)^\top H_2 (\lambda_{k+1} - \lambda^*_k) + \lambda^\top_{k+1} H_1 \lambda_{k+1} \\ & + \left\| \sum_{i=1}^{N} \varphi_{\bar{z}_i} \lambda_{k+1,(i)} - \sum_{i=1}^{N} \varphi_{\bar{z}^+_i} \lambda^*_{k,(i)} \right\|^2_{\Sigma_\varphi^{-1}} \\ \text{s.t.} \; & \mathbf{1}\lambda_{k+1} = 1, \end{aligned}$$

where $\varphi(\cdot) : \mathbb{R}^{n_z} \to \mathcal{H}$ refers to a nonlinear operator that maps $\mathbb{R}^{n_z}$ into a probably high dimensional space $\mathcal{H}$, $\varphi_{\bar{z}_i}$ and $\varphi_{\bar{z}_i^+}$ denote $\varphi(\bar{z}_i)$ and $\varphi(\bar{z}_i^+)$ respectively and Σ_φ is a positive definite matrix of appropriate dimensions. In what follows, it is shown that we do not need a precise knowledge of $\varphi(\cdot)$ to compute λ_{k+1} as it suffices to compute, for a given pair $a \in \mathbb{R}^{n_z}$ and $b \in \mathbb{R}^{n_z}$, the product

$$\langle \varphi_a, \varphi_b \rangle = \varphi_a \Sigma_\varphi^{-1} \varphi_b \,.$$

Note that the previous linear hard constraint on z_k has been changed to a penalty term on a high dimensional space by means of the kernel trick as it is no longer a linear constraint. Assuming that some data sets of the evaluation of $\varphi(\cdot)$ over the time delay embeddings of D and D^+ are available (which due to the kernel trick are not really necessary), it would be possible to denote them as

$$\varphi_{\bar{z}} = \begin{bmatrix} \varphi_{\bar{z}_1} & \varphi_{\bar{z}_2} & \cdots & \varphi_{\bar{z}_N} \end{bmatrix}, \varphi_{\bar{z}^+} = \begin{bmatrix} \varphi_{\bar{z}_1^+} & \varphi_{\bar{z}_2^+} & \cdots & \varphi_{\bar{z}_N^+} \end{bmatrix}.$$

Thus, the previous problem could be written in matrix form as

$$\begin{aligned} \lambda_{k+1}^* = \arg\min_{\lambda_{k+1}} \; & (\lambda_{k+1} - \lambda_k^*)^\top H_2 (\lambda_{k+1} - \lambda_k^*) + \lambda_{k+1}^\top H_1 \lambda_{k+1} \\ & + \|\varphi_{\bar{z}} \lambda_{k+1} - \varphi_{\bar{z}^+} \lambda_k^*\|^2_{\Sigma_\varphi^{-1}} \\ \text{s.t.} \; & \mathbf{1}\lambda_{k+1} = 1 \,. \end{aligned}$$

Now, operating with the term $\|\varphi_{\bar{z}} \lambda_{k+1} - \varphi_{\bar{z}^+} \lambda_k^*\|^2_{\Sigma_\varphi^{-1}}$, the following expression is obtained

$$\begin{aligned} \|\varphi_{\bar{z}} \lambda_{k+1} - \varphi_{\bar{z}^+} \lambda_k^*\|^2_{\Sigma_\varphi^{-1}} = & \lambda_{k+1}^\top \left(\varphi_{\bar{z}}^\top \Sigma_\varphi^{-1} \varphi_{\bar{z}} \right) \lambda_{k+1} - 2\lambda_{k+1}^\top \left(\varphi_{\bar{z}}^\top \Sigma_\varphi^{-1} \varphi_{\bar{z}^+} \right) \lambda_k^* \\ & + {\lambda_k^*}^\top \left(\varphi_{\bar{z}^+}^\top \Sigma_\varphi^{-1} \varphi_{\bar{z}^+} \right) \lambda_k^* \,. \end{aligned}$$

Again, the constant term ${\lambda_k^*}^\top \left(\varphi_{\bar{z}^+}^\top \Sigma_\varphi^{-1} \varphi_{\bar{z}^+} \right) \lambda_k^*$ is discarded because it does not affect the values of λ_{k+1}^*, leading to the following optimization problem

$$\lambda_{k+1}^* = \arg\min_{\lambda_{k+1}} \; \frac{1}{2} \lambda_{k+1}^\top H \lambda_{k+1} + f^\top \lambda_{k+1} \tag{4.12a}$$

$$\text{s.t.} \; \mathbf{1}\lambda_{k+1} = 1 \,. \tag{4.12b}$$

with $H = 2(H_1 + H_2) + 2\varphi_{\bar{z}}^\top \Sigma_\varphi^{-1} \varphi_{\bar{z}}, \quad f = -2(H_2 + \varphi_{\bar{z}}^\top \Sigma_\varphi^{-1} \varphi_{\bar{z}^+}) \lambda_k^*$.

Note that the kernel related terms can be computed because only cross products appear in the aforementioned equations. These terms would be computed as

$$\varphi_{\bar{z}}^\top \Sigma_\varphi^{-1} \varphi_{\bar{z}} = \begin{bmatrix} \langle \varphi_{\bar{z}_1}, \varphi_{\bar{z}_1} \rangle & \langle \varphi_{\bar{z}_1}, \varphi_{\bar{z}_2} \rangle & \cdots & \langle \varphi_{\bar{z}_1}, \varphi_{\bar{z}_N} \rangle \\ \langle \varphi_{\bar{z}_2}, \varphi_{\bar{z}_1} \rangle & \langle \varphi_{\bar{z}_2}, \varphi_{\bar{z}_2} \rangle & \cdots & \langle \varphi_{\bar{z}_2}, \varphi_{\bar{z}_N} \rangle \\ \vdots & \vdots & & \vdots \\ \langle \varphi_{\bar{z}_N}, \varphi_{\bar{z}_1} \rangle & \langle \varphi_{\bar{z}_N}, \varphi_{\bar{z}_2} \rangle & \cdots & \langle \varphi_{\bar{z}_N}, \varphi_{\bar{z}_N} \rangle \end{bmatrix},$$

$$\varphi_{\bar{z}}^{\top}\Sigma_{\varphi}^{-1}\varphi_{\bar{z}^+} = \begin{bmatrix} \langle\varphi_{\bar{z}_1},\varphi_{\bar{z}_1^+}\rangle & \langle\varphi_{\bar{z}_1},\varphi_{\bar{z}_2^+}\rangle & \cdots & \langle\varphi_{\bar{z}_1},\varphi_{\bar{z}_N^+}\rangle \\ \langle\varphi_{\bar{z}_2},\varphi_{\bar{z}_1^+}\rangle & \langle\varphi_{\bar{z}_2},\varphi_{\bar{z}_2^+}\rangle & \cdots & \langle\varphi_{\bar{z}_2},\varphi_{\bar{z}_N^+}\rangle \\ \vdots & \vdots & & \vdots \\ \langle\varphi_{\bar{z}_N},\varphi_{\bar{z}_1^+}\rangle & \langle\varphi_{\bar{z}_N},\varphi_{\bar{z}_2^+}\rangle & \cdots & \langle\varphi_{\bar{z}_N},\varphi_{\bar{z}_N^+}\rangle \end{bmatrix},$$

where $\langle\varphi_{\bar{z}_i},\varphi_{\bar{z}_j}\rangle$ is the result of applying a certain kernel function with samples $\bar{z}_i$ and $\bar{z}_j$ as inputs. By means of the KKT conditions, the solution of the optimization problem (4.12) is obtained;

$$\lambda_{k+1}^{*} = A\lambda_k^{*} + c,$$

with

$$A = 2H^{-1}(\mathrm{I_N} - \mathbf{1}^{\top}(\mathbf{1}\mathrm{H}^{-1}\mathbf{1}^{\top})^{-1}\mathbf{1}\mathrm{H}^{-1})(\mathrm{H_2} + \varphi_{\bar{\mathrm{z}}}^{\top}\Sigma_{\varphi}^{-1}\varphi_{\bar{\mathrm{z}}^+}),$$
$$c = H^{-1}\mathbf{1}^{\top}(\mathbf{1}H^{-1}\mathbf{1}^{\top})^{-1}.$$

As the output equation does not change, the complete model of the system is

$$\lambda_{k+1}^{*} = A\lambda_k^{*} + c$$
$$y_k = Y\lambda_k^{*}.$$

Remark 4.1. *Note that this system is only affine in the feature space. Assuming that the kernel functions are not linear, the model is nonlinear in the data space.*

Remark 4.2. *There are lots of possible kernel functions that can be used with the proposed methodology. In the following, some examples of kernel functions are shown.*

- Linear kernel:
$$\langle\varphi_a,\varphi_b\rangle = a^{\top}b.$$

- Polynomial kernel:
$$\langle\varphi_a,\varphi_b\rangle = (n + a^{\top}b)^p, \quad \text{where } n \in \mathbb{R} \text{ and } p \in \mathbb{Z}.$$

- Radial basis kernel:
$$\langle\varphi_a,\varphi_b\rangle = e^{\frac{-||a-b||}{2\sigma^2}}, \quad \text{where } \sigma > 0.$$

- Sigmoidal kernel:
$$\langle\varphi_a,\varphi_b\rangle = \tanh(n_1 a^{\top}b + n_2), \quad \text{where } n_1 \in \mathbb{R} \text{ and } n_2 \in \mathbb{R}.$$

4.3.1 Initial condition

As in the previous cases, an initial value λ_0^* is needed. Here, it is obtained from the following optimization problem

$$\begin{aligned}\lambda_0^* = \arg\min_{\lambda_0} \;& \lambda_0^\top H_1 \lambda_0 + \|\varphi_{z_0} - \varphi_{\bar{z}}\lambda_0\|^2_{\Sigma_\varphi^{-1}} \\ \text{s.t.} \;& \mathbf{1}^\top \lambda = 1.\end{aligned}$$

Operating with $\|\varphi_{z_0} - \varphi_{\bar{z}}\lambda_0\|^2_{\Sigma_\varphi^{-1}}$, we obtain

$$\begin{aligned}\lambda_0^* = \arg\min_{\lambda_0} \;& \frac{1}{2}\lambda_0^\top H \lambda_0 + f^\top \lambda_0 \\ \text{s.t.} \;& \mathbf{1}^\top \lambda_0 = 1,\end{aligned}$$

where

$$H = 2H_1 + 2\varphi_{\bar{z}}^\top \Sigma_\varphi^{-1} \varphi_{\bar{z}}, \quad f = -2\varphi_{\bar{z}}^\top \Sigma_\varphi^{-1} \varphi_{z_0},$$

$$\varphi_{\bar{z}}^\top \Sigma_\varphi^{-1} \varphi_{z_0} = \begin{bmatrix} \langle \varphi_{\bar{z}_1}, \varphi_{z_0} \rangle \\ \langle \varphi_{\bar{z}_2}, \varphi_{z_0} \rangle \\ \vdots \\ \langle \varphi_{\bar{z}_N}, \varphi_{z_0} \rangle \end{bmatrix}.$$

4.4 Kalman filter for SSK

To reduce the effect of noisy measurements in the data sets D and D^+ as well as disturbances and modelling mismatches, a kalman filter [151] is considered. Under the assumption that $z_{k+1} = D^+\lambda_k^*$ and $\mathbf{1}\lambda_k^* = 1$, a nominal LTV state-space model was derived in the previous sections:

$$\begin{aligned}\lambda_{k+1}^* &= A_k \lambda_k^* + c \\ y_k &= Y \lambda_k^*.\end{aligned}$$

Note that, here, the kernel-based approach is used without loss of generality. In order to address the existence of noise, disturbances and modelling mismatches, this model is augmented with disturbances, modelling errors (w_k) and measurement noise (v_k), that is,

$$\begin{aligned}\lambda_{k+1}^* &= A\lambda_k^* + c + w_k \\ y_k &= Y\lambda_k^* + v_k\,.\end{aligned}$$

As it is necessary to enforce the equality constraint $\mathbf{1}\lambda_k^* = 1$, the output vector is extended as

$$y_k' = \begin{bmatrix} Y \\ \mathbf{1} \end{bmatrix} \lambda_k^* + \begin{bmatrix} v_k \\ 0 \end{bmatrix}. \tag{4.13}$$

Defining $G = \begin{bmatrix} Y \\ \mathbf{1} \end{bmatrix}$ and $v'_k = \begin{bmatrix} v_k \\ 0 \end{bmatrix}$, the following extended system is obtained

$$\begin{aligned} \lambda^*_{k+1} &= A\lambda^*_k + c + w_k \\ y'_k &= G\lambda^*_k + v'_k \,. \end{aligned}$$

Due to noise, λ^*_k is also noisy and, thus, it may be helpful to use an optimal prediction of its value obtained with a kalman filter. The prediction of λ^*_k will be denoted as $\tilde{\lambda}^*_k$ whereas the corrected prediction will be denoted as $\hat{\lambda}^*_k$. It is assumed that w_k and v_k are uncorrelated white noise signals with bounded covariance matrices ς_ω and ς_v so that

$$\mathrm{E}(w_k w_k^\top) \leq \varsigma_w, \quad \mathrm{E}(v_k v_k^\top) \leq \varsigma_v,$$

where $\mathrm{E}(\cdot)$ denotes the mathematical expectation. From the bound on the covariance of v_k, it is easy to obtain a bound of v'_k:

$$\mathrm{E}(v'_k {v'_k}^\top) \leq \mathrm{E}\left(\begin{bmatrix} v_k v_k^\top & 0 \\ 0 & 0 \end{bmatrix} \right) = \begin{bmatrix} \varsigma_v & 0 \\ 0 & 0 \end{bmatrix} = \varsigma'_v \,.$$

The bound of the covariance of the estimation error $\lambda^*_k - \tilde{\lambda}^*_k$ is denoted as $\tilde{\sigma}_k$:

$$\mathrm{E}\left((\lambda^*_k - \tilde{\lambda}^*_k)(\lambda^*_k - \tilde{\lambda}^*_k)^\top \right) \leq \tilde{\sigma}_k.$$

Then, given an estimation $\tilde{\lambda}^*_k$, a corrected version $\hat{\lambda}^*_k$ is obtained from

$$\hat{\lambda}^*_k = \tilde{\lambda}^*_k + S_k \left(\begin{bmatrix} y_k \\ 1 \end{bmatrix} - G\tilde{\lambda}^*_k \right),$$

where S_k is the optimal gain, calculated as

$$S_k = \tilde{\sigma}_k G^\top \left(G\tilde{\sigma}_k G^\top + \varsigma'_v \right)^{-1}.$$

Note that the corrected vector $\hat{\lambda}^*_k$ fulfills the constraint $1 = \mathbf{1}\hat{\lambda}^*_k$ because the last component of the extended output (that is, $\mathbf{1}\lambda^*_k$) is artificial and thus there is no measurement noise (see (4.13)).

Applying the equations of the kalman filter, the matrix $\tilde{\sigma}_{k+1}$ is computed as

$$\tilde{\sigma}_{k+1} = A\hat{\sigma}_k A^\top + \varsigma_w \,,$$

where $\hat{\sigma}_k = \tilde{\sigma}_k - S_k G \tilde{\sigma}_k$. Finally, $\tilde{\lambda}^*_{k+1}$ is computed as

$$\tilde{\lambda}^*_{k+1} = A\hat{\lambda}^*_k + c \,. \tag{4.14}$$

The matrices $\tilde{\sigma}_0, \varsigma_w, \varsigma_v$ are set as diagonal matrices that are considered tuning parameters.

4.5 Numerical examples

In this section, two examples are provided to show the effectiveness of the proposed strategy. Four baselines based on Gaussian Processes (GPs) [23], Nonlinear ARX models (NARX) [20], Reservoir computing (RC) [33] and Dynamic Mode Decomposition (DMD) [7] are provided to compare the results obtained with our proposed approaches.

4.5.1 Sunspot number

Forecasting the sunspot number is considered quite difficult as the time series is nonstationary and because the nature of its dynamics is unknown. Monthly observations of the historical evolution of the number of sunspots since 1749 will be used in this example. The data sets are compounded of the first 2500 samples with a time-delay embedding $z_k = [y_k, \dots, y_{k-39}]^\top$. The next 150 samples will be used as a test set. The observations are assumed to be noise-free and thus the approach without kalman filter is used. The predictions will be made from time instant $k = 0$ exclusively. That is, the predictions will be k-step ahead, with $k = 1, \dots, 150$. Note that the forecasting horizon is quite long (it comprises more than a solar cycle), making the forecasting task even harder. Here, NARX and GPs are computed using Matlab functions ("nlarx" and "fitrgp") and the RC implementation considers a reservoir size of 300, a leakage rate of 0.9 and a spectral radius of 0.4. Both the Kernel-based State-Space Kriging (K-SSK) and Local-Data State-Space Kriging (LD-SSK) are tested in this example. K-SSK uses a *radial basis* kernel

$$\langle \varphi_{\bar{z}_i}, \varphi_{\bar{z}_j} \rangle = e^{\frac{-||\bar{z}_i - \bar{z}_j||}{2\sigma^2}},$$

with $\sigma = 0.4$. The value of the matrices H_1 and H_2 are $H_1 = 3.5 \cdot 10^{-12}\, \mathrm{I_N}$, $\mathrm{H_2} = 0.14\, \mathrm{I_N}$ and the data sets D and D^+ are normalized in the $[0, 1]$ range. On the other hand, the LD-SSK approach only has one tunable parameter whose value is $H_2 = 0.053\, \mathrm{I_N}$.

Figure 4.1 and table 4.1 show the forecasting results. Note that only the proposed approaches are shown in the figure for the sake of clarity. It can be seen that the proposed approaches perform better than the aforementioned baselines, obtaining smaller errors and standard deviations in general.

	LD-SSK	K-SSK	GP	NARX	DMD	RC
MSE	754.15	701.72	999.77	991.11	5795.6	6048.1
Std	24.671	25.843	35.147	26.490	72.897	62.552

Table 4.1: MSE for the Sunspot Number example.

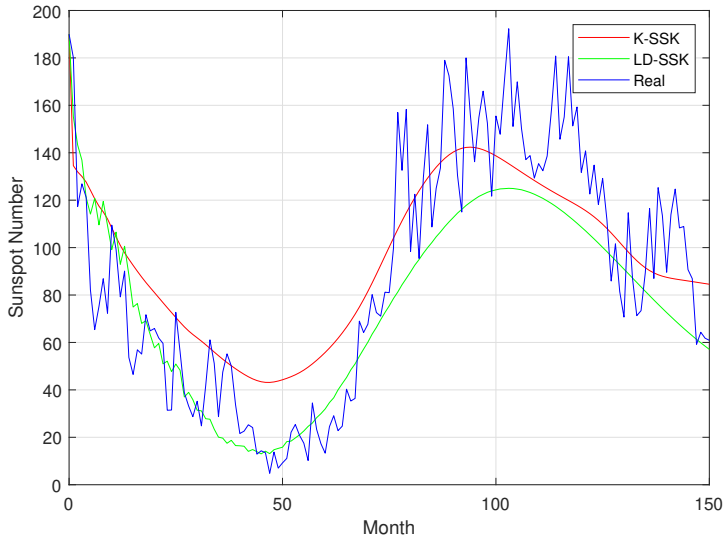


Figure 4.1: Forecasting the Sunspot Number 150 steps ahead.

4.5.2 Rössler attractor

Consider now the system described by the following set of differential equations

$$\begin{aligned} \dot{o} &= -p - l \\ \dot{p} &= o + ap \\ \dot{l} &= b + l(o - c)\,, \end{aligned}$$

also known as the Rössler attractor. The set of parameters considered in this example is $a = 0.2$, $b = 0.2$ and $c = 5.7$ which are known to correspond to a chaotic behaviour. In order to obtain samples of the continuous system, the system of ODEs is integrated numerically with a fixed sample time of 0.1 seconds during a total simulation time of 20 seconds starting from random initial points in the space, making a total of 1000 samples in the matrices D and D^+ (comprising 5 trajectories of 200 samples each one). The time-delay embedding considered here is $z_k = [o_k, p_k, l_k]^\top$.

The purpose of this example is to show the effectiveness of the proposed strategies to model nonlinearity with measurement noise. For that reason, it will be considered that the measurements obtained from the Rössler attractor are noisy. This noise will follow a normal distribution with zero mean and unit variance $\mathcal{N} \sim (0, 1)$ and is completely uncorrelated (that is, the noise of each state is also uncorrelated to that of the other states). As in the previous example, both proposed approaches are tested here. The LD-SSK scheme which incorporates the kalman filter is used here with parameters $H_2 = 9.79\,\mathrm{I_N}$, $\varsigma_w = 1.01 \cdot 10^{-5}\,\mathrm{I_N}$ and $\tilde{\sigma}_0 = 6.51 \cdot 10^{-6}\,\mathrm{I_N}$. Also, the variance of v_k is assumed to be known.

On the other hand, K-SSK uses a kernel such that

$$\langle \varphi_{\bar{z}_i}, \varphi_{\bar{z}_j} \rangle = \begin{cases} e^{\frac{-||\bar{z}_i - \bar{z}_j||}{2\sigma^2}} + \mathrm{E}(v_k^2) & \text{if } i = j\,, \\ e^{\frac{-||\bar{z}_i - \bar{z}_j||}{2\sigma^2}} & \text{else,} \end{cases}$$

where $\sigma = 0.119$. The value of the matrices H_1 and H_2 are $H_1 = 1.25{\cdot}10^{-5}\,\mathrm{I_N}$, $\mathrm{H}_2 = 2.24 \cdot 10^{-5}\,\mathrm{I_N}$ whereas the parameters of the kalman filter are $\varsigma_w = 1.23 \cdot 10^{-7}\,\mathrm{I_N}$ and $\tilde{\sigma}_0 = 4.09 \cdot 10^{-6}\,\mathrm{I_N}$

Note that the forecasting is done 1-step ahead, in contrast to the k-steps ahead predictions of the previous example, in order to be able to apply the methodology of the kalman filter. Thus, at each time instant k the value of the output is sampled and the forecasting at $k-1$ is corrected with this new measurement. After that, the prediction for $k+1$ is done with all the information available at k (which includes the corrected measurements). 100 trajectories of 15 seconds obtained from random initial conditions are considered. On the other hand, GPs are computed using a radial basis kernel with a regularization term equal to the true variance of the process. The RC implementation considers a reservoir size of 50, a leakage rate of 0.2 and a spectral radius of 0.3. Finally, a kalman filtering layer is coupled to the linear system obtained with the DMD in this section (K-DMD), where the variance of the noise is also assumed to be known.

The results are shown in figure 4.2, and table 4.2. Figure 4.2 shows the real evolution of the state along with the noisy measurements and the prediction obtained with the K-SSK approach for a representative trajectory. Table 4.2 summarizes the numerical results of the experiment. It can be seen that the K-SSK strategy achieves the best results both in MSE and standard deviations.

On the other hand, computational times of the different baselines for the Rössler atractor are provided in table 4.3. Although the proposed approaches seem to be the most costly methods, it remains in the order of 60 and 30 milliseconds, what can be considered fast enough for time series/dynamic systems forecasting. Also, it should be noted that most of the computation time is due to the kalman filter step. The computation times would drop to 12.477 ms for the LD-SSK and 0.2434 ms for the K-SSK without the kalman filter.

	LD-SSK	K-SSK	GP	NARX	K-DMD	RC
MSE	0.2504	0.2138	0.6423	0.6337	0.2734	0.3385
Std	0.5002	0.4622	0.8013	0.7960	0.5229	0.5816

Table 4.2: MSE for the noisy Rössler attractor.

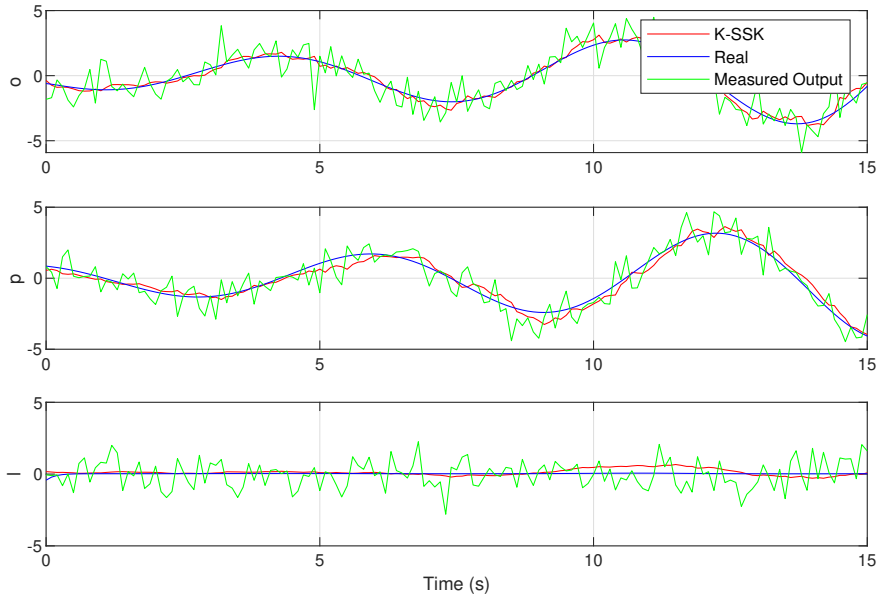


Figure 4.2: Forecasting the noisy Rössler attractor with kalman filtering.

LD-SSK	K-SSK	GP	NARX	K-DMD	RC
62.909	37.5912	2.591	14.872	0.036	0.868

Table 4.3: Average online computational time in milliseconds (Rössler attractor).

4.6 Conclusions

This chapter presented the state-space kriging method for autonomous systems. The proposed technique allowed us to obtain a model of the system by means of past data of the process. This could be done by manipulating appropriately the dissimilarity function presented in part I and by obtaining the explicit solution of the resulting optimization problems. Also, it was shown that a kalman filter can be used to improve the performance of the proposed models. Finally, two numerical examples were proposed in order to show the effectiveness of the SSK approaches. Both the LD-SSK and the K-SSK attained better performance than other existing machine learning models.

In the next chapter, the proposed methodology will be extended to tackle non-autonomous system, which will allow us to develop MPC controllers as well.

Chapter 5

State-space kriging for non-autonomous systems

The previous chapter introduced the concept of state-space kriging for autonomous systems and, by means of numerical results, it was shown that the performance is, if not better, comparable to many other forecasting methods. However, in order to be able to use the proposed approaches in a control scheme, it is needed to introduce the input term in the SSK formulation.

In this chapter, both methodologies presented before will be extended to tackle non-autonomous systems. It is also shown that the kernel-based SSK is more suitable for control due to the fact that the system matrices are computed only once, unlike the local data approach.

5.1 Non-autonomous linear SSK

Consider a non-autonomous discrete nonlinear system

$$x_{k+1} = h(x_k, u_k) \tag{5.1a}$$
$$y_k = g(x_k), \tag{5.1b}$$

where k is the time instant, $x_k \in \mathbb{R}^{n_x}$ is the state of the system, $u_k \in \mathbb{R}^{n_u}$ is the input of the system, $y_k \in \mathbb{R}^{n_y}$ is the output of the system and $h(\cdot)$ and $g(\cdot)$ are unknown nonlinear functions such that $h(\cdot) : \mathbb{R}^{n_x \times n_u} \to \mathbb{R}^{n_x}$ and $g(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$. Unlike in chapter 4, it is assumed that the vector of weights λ_k^* fulfill the following constraints:

$$z_k = D^+ \lambda_k^*,$$
$$1 = \mathbf{1}\lambda_k^*.$$

Furthermore, it is imposed that λ^*_{k+1} must be able to compute z_k and u_k as an affine combination of D and U. Thus, it must satisfy the constraints

$$\begin{bmatrix} D \\ \mathbf{1} \\ U \end{bmatrix} \lambda^*_{k+1} = \begin{bmatrix} z_k \\ 1 \\ u_k \end{bmatrix}.$$

where $U = \begin{bmatrix} \bar{u}_1 & \bar{u}_2 & \dots & \bar{u}_N \end{bmatrix} \in \mathcal{R}^{n_u \times N}$ is the data set of control actions, i.e. $\bar{u}_i$ are past samples of the input u. This set of constraints is similar to the constraints presented in the previous chapter but aimed to obtain the successor of λ^*_{k+1}, as a function of λ^*_k and u_k, that is, to consider non-autonomous systems. Thus, the proposed optimization problem leading to the state equation is

$$\lambda^*_{k+1} = \arg\min_{\lambda_{k+1}} \ (\lambda_{k+1} - \lambda^*_k)^\top H_2 (\lambda_{k+1} - \lambda^*_k) + \lambda^\top_{k+1} H_1 \lambda_{k+1} \tag{5.3a}$$

$$\text{s.t.} \quad C\lambda_{k+1} = C^+ \lambda^*_k + \begin{bmatrix} \mathbf{0}_{(n_z+1)\times n_u} \\ \mathrm{I}_{\mathrm{n_u}} \end{bmatrix} u_k, \tag{5.3b}$$

where $\mathbf{0}_{(n_z+1)\times n_u}$ is a matrix of zeros compounded of n_z+1 rows and n_u columns $C = \begin{bmatrix} D^\top & \mathbf{1}^\top & U^\top \end{bmatrix}^\top$ and $C^+ = \begin{bmatrix} D^{+\top} & \mathbf{1}^\top & \mathbf{0}^\top \end{bmatrix}^\top$. Note that u_k is the current value of the system input, not to be confused with $\bar{u}$, which are past values of u_k stored in the data base U.

This problem can be written in canonical form as

$$\begin{aligned} \lambda^*_{k+1} = \arg\min_{\lambda_{k+1}} \ & \frac{1}{2} \lambda^\top_{k+1} H \lambda_{k+1} + f^\top \lambda_{k+1} \\ \text{s.t.} \ & C\lambda_{k+1} = b\,, \end{aligned}$$

with $H = 2(H_1 + H_2)$, $f = -2H_2\lambda^*_k$ and $b = C^+\lambda^*_k + \begin{bmatrix} \mathbf{0}_{n_u\times(n_z+1)} & \mathrm{I}_{\mathrm{n_u}} \end{bmatrix}^\top u_k$. Note that the constant term is discarded because it does not affect the value of λ^*_{k+1}. By means of the KKT conditions, this problem has the solution

$$\lambda^*_{k+1} = -H^{-1}f + H^{-1}C^\top \left(CH^{-1}C^\top\right)^{-1} \left(CH^{-1}f + b\right).$$

Substituting f and b, the following state-space equation is obtained

$$\begin{aligned} \lambda^*_{k+1} &= A\lambda^*_k + Bu_k, \\ y_k &= Y^+ \lambda^*_k\,. \end{aligned}$$

where Y^+ denotes a matrix compounded of only the first n_y rows of D^+ and

$$\begin{aligned} A &= (2H^{-1}H_2 + H^{-1}C^\top (CH^{-1}C^\top)^{-1} \left(C^+ - 2CH^{-1}H_2\right)), \\ B &= H^{-1}C^\top (CH^{-1}C)^{-1} \begin{bmatrix} \mathbf{0}_{n_u\times(n_z+1)} & \mathrm{I}_{\mathrm{n_u}} \end{bmatrix}^\top. \end{aligned}$$

As the expression of the dynamics of λ has been obtained, the output y can be easily obtained for any time instant k assuming that the inputs are given.

5.1.1 Initial condition

It still remains to show how to obtain the initial value λ_0^*. For that purpose, consider the following optimization problem

$$\begin{aligned}\lambda_0^* = \arg\min_{\lambda_0} \ & \lambda_0^\top H_1 \lambda_0 \\ \text{s.t.} \ & \begin{bmatrix} D^+ \\ \mathbf{1} \end{bmatrix} \lambda_0 = \begin{bmatrix} z_0 \\ 1 \end{bmatrix}.\end{aligned}$$

where z_0 is the initial value of the time delay embedding.

5.2 Non-autonomous kernel-based SSK

Taking into account kernel functions, the optimization problem in (5.3) becomes the following

$$\begin{aligned}\lambda_{k+1}^* = \arg\min_{\lambda_{k+1}} \ & (\lambda_{k+1} - \lambda_k^*)^\top H_2 (\lambda_{k+1} - \lambda_k^*) + \lambda_{k+1}^\top H_1 \lambda_{k+1} \\ & + \|\varphi_{\bar{z}} \lambda_{k+1} - \varphi_{\bar{z}^+} \lambda_k^*\|^2_{\Sigma_\varphi^{-1}} \qquad (5.6a) \\ \text{s.t.} \ & \begin{bmatrix} U \\ \mathbf{1} \end{bmatrix} \lambda_{k+1} = \begin{bmatrix} u_k \\ 1 \end{bmatrix}. \qquad (5.6b)\end{aligned}$$

As in the previous chapter, the hard constraint becomes a penalty term in a high dimensional space due to the kernel trick [150]. Now, operating with the term $\|\varphi_{\bar{z}} \lambda_{k+1} - \varphi_{\bar{z}^+} \lambda_k^*\|^2_{\Sigma_\varphi^{-1}}$, the following expression is obtained

$$\begin{aligned}\|\varphi_{\bar{z}} \lambda_{k+1} - \varphi_{\bar{z}^+} \lambda_k^*\|^2_{\Sigma_\varphi^{-1}} = & \lambda_{k+1}^\top \varphi_{\bar{z}}^\top \Sigma_\varphi^{-1} \varphi_{\bar{z}} \lambda_{k+1} - 2\lambda_{k+1}^\top \varphi_{\bar{z}}^\top \Sigma_\varphi^{-1} \varphi_{\bar{z}^+} \lambda_k^* \\ & + \lambda_k^{*\top} \varphi_{\bar{z}^+}^\top \Sigma_\varphi^{-1} \varphi_{\bar{z}^+} \lambda_k^*,\end{aligned}$$

Discarding the constant term $\lambda_k^{*\top} \varphi_{\bar{z}^+}^\top \Sigma_\varphi^{-1} \varphi_{\bar{z}^+} \lambda_k^*$, we obtain the following optimization problem

$$\begin{aligned}\lambda_{k+1}^* = \arg\min_{\lambda_{k+1}} \ & \frac{1}{2} \lambda_{k+1}^\top H \lambda_{k+1} + f^\top \lambda_{k+1} \\ \text{s.t.} \ & T\lambda_{k+1} = \begin{bmatrix} \mathbf{0}^\top \\ 1 \end{bmatrix} + \begin{bmatrix} \mathrm{I}_{\mathrm{n_u}} \\ \mathbf{0} \end{bmatrix} u_k \,.\end{aligned}$$

with $H = 2(H_1 + H_2) + 2\varphi_{\bar{z}}^\top \Sigma_\varphi^{-1} \varphi_{\bar{z}}$, $\quad f = -2(H_2 + \varphi_{\bar{z}}^\top \Sigma_\varphi^{-1} \varphi_{\bar{z}^+}) \lambda_k^*$, $T = \begin{bmatrix} U \\ \mathbf{1} \end{bmatrix}$.

Note that the kernel related terms can be computed because only cross products appear in the aforementioned equations. These terms would be computed as

$$\varphi_{\bar{z}}^\top \Sigma_\varphi^{-1} \varphi_{\bar{z}} = \begin{bmatrix} \langle \varphi_{\bar{z}_1}, \varphi_{\bar{z}_1} \rangle & \langle \varphi_{\bar{z}_1}, \varphi_{\bar{z}_2} \rangle & \cdots & \langle \varphi_{\bar{z}_1}, \varphi_{\bar{z}_N} \rangle \\ \langle \varphi_{\bar{z}_2}, \varphi_{\bar{z}_1} \rangle & \langle \varphi_{\bar{z}_2}, \varphi_{\bar{z}_2} \rangle & \cdots & \langle \varphi_{\bar{z}_2}, \varphi_{\bar{z}_N} \rangle \\ \vdots & \vdots & & \vdots \\ \langle \varphi_{\bar{z}_N}, \varphi_{\bar{z}_1} \rangle & \langle \varphi_{\bar{z}_N}, \varphi_{\bar{z}_2} \rangle & \cdots & \langle \varphi_{\bar{z}_N}, \varphi_{\bar{z}_N} \rangle \end{bmatrix},$$

$$\varphi_{\bar{z}}^{\top}\Sigma_{\varphi}^{-1}\varphi_{\bar{z}^+} = \begin{bmatrix} \langle\varphi_{\bar{z}_1},\varphi_{\bar{z}_1^+}\rangle & \langle\varphi_{\bar{z}_1},\varphi_{\bar{z}_2^+}\rangle & \cdots & \langle\varphi_{\bar{z}_1},\varphi_{\bar{z}_N^+}\rangle \\ \langle\varphi_{\bar{z}_2},\varphi_{\bar{z}_1^+}\rangle & \langle\varphi_{\bar{z}_2},\varphi_{\bar{z}_2^+}\rangle & \cdots & \langle\varphi_{\bar{z}_2},\varphi_{\bar{z}_N^+}\rangle \\ \vdots & \vdots & & \vdots \\ \langle\varphi_{\bar{z}_N},\varphi_{\bar{z}_1^+}\rangle & \langle\varphi_{\bar{z}_N},\varphi_{\bar{z}_2^+}\rangle & \cdots & \langle\varphi_{\bar{z}_N},\varphi_{\bar{z}_N^+}\rangle \end{bmatrix}.$$

Applying the KKT conditions, the solution of the optimization problem (5.6) is obtained;

$$\lambda_{k+1}^* = A\lambda_k^* + Bu_k + c \tag{5.8a}$$
$$y_k = Y^+\lambda_k^*\,, \tag{5.8b}$$

with

$$A = 2H^{-1}(\mathrm{I_N} - \mathrm{T}^{\top}(\mathrm{TH}^{-1}\mathrm{T}^{\top})^{-1}\mathrm{TH}^{-1})(\mathrm{H}_2 + \varphi_{\bar{\mathrm{z}}}^{\top}\Sigma_{\varphi}^{-1}\varphi_{\bar{\mathrm{z}}^+}),$$
$$B = H^{-1}T^{\top}(TH^{-1}T^{\top})^{-1}\begin{bmatrix} \mathrm{I_{n_u}} \\ \mathbf{0} \end{bmatrix},$$
$$c = H^{-1}T^{\top}(TH^{-1}T^{\top})^{-1}\begin{bmatrix} \mathbf{0}^{\top} \\ 1 \end{bmatrix}.$$

Remark 5.1. *Note that adding the input to the optimization problem is compatible with both the Local-data approach and the kernel-based approach, leading to a family of different predictors. However, in the local-data approach, it is needed to compute the matrices A and B at each time instant k, becoming cumbersome when considering MPC control problems. For that reason, in the following, it is only considered the kernel-based SSK.*

Remark 5.2. *Note that the kalman filter for the non-autonomous SSK is equivalent to the one proposed in the previous chapter. In this case, equation* (4.14) *becomes*

$$\tilde{\lambda}_{k+1}^* = A\hat{\lambda}_k^* + Bu_k + c\,. \tag{5.9}$$

5.2.1 Initial condition

As in the non-autonomous linear SSK, an initial value λ_0^* is needed. This initial condition can be obtained from the following optimization problem

$$\lambda_0^* = \arg\min_{\lambda_0}\ \lambda_0^{\top}H_1\lambda_0 + \|\varphi_{z_0} - \varphi_{\bar{z}^+}\lambda_0\|_{\Sigma_{\varphi}^{-1}}^2$$
$$\text{s.t. } \mathbf{1}^{\top}\lambda = 1\,.$$

Operating with $\|\varphi_{z_0} - \varphi_{\bar{z}^+}\lambda_0\|_{\Sigma_{\varphi}^{-1}}^2$, we obtain

$$\lambda_0^* = \arg\min_{\lambda_0}\ \frac{1}{2}\lambda_0^{\top}H\lambda_0 + f^{\top}\lambda_0$$
$$\text{s.t. } \mathbf{1}^{\top}\lambda_0 = 1\,,$$

with

$$H = 2H_1 + 2\varphi_{\bar{z}^+}^\top \Sigma_\varphi^{-1} \varphi_{\bar{z}^+}, \quad f = -2\varphi_{\bar{z}^+}^\top \Sigma_\varphi^{-1} \varphi_{z_0},$$

$$\varphi_{\bar{z}^+}^\top \Sigma_\varphi^{-1} \varphi_{\bar{z}^+} = \begin{bmatrix} \langle \varphi_{\bar{z}_1^+}, \varphi_{\bar{z}_1^+} \rangle & \langle \varphi_{\bar{z}_1^+}, \varphi_{\bar{z}_2^+} \rangle & \dots & \langle \varphi_{\bar{z}_1^+}, \varphi_{\bar{z}_N^+} \rangle \\ \langle \varphi_{\bar{z}_2^+}, \varphi_{\bar{z}_1^+} \rangle & \langle \varphi_{\bar{z}_2^+}, \varphi_{\bar{z}_2^+} \rangle & \dots & \langle \varphi_{\bar{z}_2^+}, \varphi_{\bar{z}_N^+} \rangle \\ \vdots & \vdots & & \vdots \\ \langle \varphi_{\bar{z}_N^+}, \varphi_{\bar{z}_1^+} \rangle & \langle \varphi_{\bar{z}_N^+}, \varphi_{\bar{z}_2^+} \rangle & \dots & \langle \varphi_{\bar{z}_N^+}, \varphi_{\bar{z}_N^+} \rangle \end{bmatrix},$$

$$\varphi_{\bar{z}^+}^\top \Sigma_\varphi^{-1} \varphi_{z_0} = \begin{bmatrix} \langle \varphi_{\bar{z}_1^+}, \varphi_{z_0} \rangle \\ \langle \varphi_{\bar{z}_2^+}, \varphi_{z_0} \rangle \\ \vdots \\ \langle \varphi_{\bar{z}_N^+}, \varphi_{z_0} \rangle \end{bmatrix}.$$

5.3 Application to MPC

Assuming that we have an arbitrary nonlinear system in a discrete time setting like the system in equation (5.1), then, the objective of an MPC controller is to steer the state x to an equilibrium point (x_s, u_s) that it is called the *reference* of the controller. The control actions applied to the system to achieve this objective u_k are computed by means of an optimization problem.

The obtained optimal inputs depend on both the actual state of the plant and the optimization criteria. This step cost function is designed to penalize the deviation of the state and the inputs to the reference at each time instant k for a certain prediction horizon N_p. In order to ensure stability for the controller, a terminal cost function is usually added to the previously defined step cost function. Denoting $V(\cdot)$ as the total cost function, $\mathbf{u}$ as the sequence of inputs $\{u_i\}_{i=k}^{k+N_p-1}$, $\mathbf{x}$ as the sequence of predicted states $\{x_{i|k}\}_{i=0}^{N_p}$, $x_{i|k}$ as the prediction of x_{k+i} made at instant k and considering that the system may have constraints in the inputs and in the states so that $x \in \mathcal{X}$ and $u \in \mathcal{U}$, the optimal control inputs $\mathbf{u}^*$ are obtained by solving the following optimization problem

$$\mathbf{u}^* = \arg\min_{\mathbf{x},\mathbf{u}} \; V(\mathbf{x}, \mathbf{u}) \tag{5.11a}$$

$$\text{s.t. system model} \tag{5.11b}$$

$$x_{i|k} \in \mathcal{X} \quad \forall i = 0, \dots, N_p \tag{5.11c}$$

$$u_{k+i} \in \mathcal{U} \quad \forall i = 0, \dots, N_p - 1. \tag{5.11d}$$

From this sequence of optimal inputs $\mathbf{u}^*$, only the first component will be applied to the system, discarding the rest as it is usual in MPC due to the receding horizon strategy.

In this section, a tracking MPC controller [124, 125] that uses the K-SSK model is presented in this section. The main difference of the tracking MPC with respect to

traditional MPC is that the reference to be tracked is considered as an additional decision variable in the optimization problem, that is, it becomes an artificial reference. In order to enforce that the artificial reference converges to the target reference, an additional cost is added to the MPC cost function, penalising the deviation between them. Among the many advantages of this formulation, the fact that the recursive feasibility is guaranteed for any change of the desired reference and a significantly larger domain of attraction for short prediction horizons are probably the most important ones.

The cost function in the proposed controller is compounded of three ingredients:

- A step cost function $l_s(\cdot,\cdot)$ to penalize tracking error. Tracking error with respect to a certain reference is tackled by means of a change of variables $\breve{y} = \tilde{y} - y_s$, $\breve{u} = u - u_s$ where $\tilde{y}$ is a prediction of y, y_s is the artificial output reference and u_s is the artificial input reference. Here, a quadratic cost is considered. This cost penalizes the distance to the artificial input and output reference by means of some weighting matrices of appropriate dimensions Q and R (i.e. $Q \in \mathbb{R}^{n_y \times n_y}$ and $R \in \mathbb{R}^{n_u \times n_u}$), i.e.

$$l_s(\breve{y}, \breve{u}) = \breve{y}^\top Q \breve{y} + \breve{u}^\top R \breve{u}.$$

- An offset function penalizing the difference between the artificial output y_s and the target desired reference,

$$l_o(y_s, r) = (y_s - r)^\top O\,(y_s - r)\,.$$

  Under some mild conditions [124], it is proven that the artificial reference converges to the true reference as time goes by.

- A weighted terminal cost function. This function measures the closeness of the terminal state $\lambda^*_{N_p|k}$ to the artificial steady state λ^*_s. Weighting appropriately the terminal cost allows us to omit the terminal equality constraint [125], simplifying the design of the controller.

$$l_t(\lambda^*_{N_p|k}, \lambda^*_s) = \gamma \left(\lambda^*_{N_p|k} - \lambda^*_s\right)^\top P \left(\lambda^*_{N_p|k} - \lambda^*_s\right),$$

  where $\gamma \geq 1$.

Denoting $\mathbf{y} \in \mathbb{R}^{n_y N_p}$ and $\mathbf{u} \in \mathbb{R}^{n_u (N_p+1)}$ as

$$\mathbf{y} = [y^\top_{0|k}, \ldots, y^\top_{N_p-1|k}]^\top, \quad [u^\top_k, \ldots, u^\top_{k+N_p-1}, u^\top_s]^\top,$$

it is possible to define a total cost function $V_{N_p}(\mathbf{y}, \mathbf{u}, r, \lambda^*_k)$ as the sum of the aforementioned three functions for a finite prediction horizon N_p

$$V_{N_p}(\mathbf{y}, \mathbf{u}, r, \lambda^*_k) = \sum_{i=0}^{N_p-1} l_s(\breve{y}_{i|k}, \breve{u}_{k+i}) + l_o(y_s(u_s), r) + l_t(\lambda^*_{N_p|k}(\lambda^*_k, \mathbf{u}), \lambda^*_s(u_s)).$$

Thus, an MPC control problem with inequality constraints in the outputs and box constraints in the inputs is presented here

$$\mathbf{u}^* = \arg \min_{\mathbf{y},\mathbf{u},\lambda_{N_p|k}} \quad V_{N_p}(\mathbf{y}, \mathbf{u}, r, \lambda_k^*) \tag{5.12a}$$

$$\text{s.t.} \quad \lambda_{i+1|k}^* = A\lambda_{i|k}^* + Bu_{k+i} + c \quad \forall i = 0, \ldots, N_p - 1 \tag{5.12b}$$

$$y_{i|k} = Y^+ \lambda_{i|k}^* \quad \forall i = 0, \ldots, N_p \tag{5.12c}$$

$$\lambda_s^* = A\lambda_s^* + Bu_s + c \tag{5.12d}$$

$$\psi\, y_{i|k} \leq \delta \quad \forall i = 0, \ldots, N_p \tag{5.12e}$$

$$u_{\min} \leq u_{k+i} \leq u_{\max}, \quad \forall i = 0, \ldots, N_p - 1 \tag{5.12f}$$

which is a parametric quadratic optimization problem whose parameters are r and λ_k^*. Appendix A shows how to pose this problem in canonical form.

From this optimization problem, a sequence of optimal control actions $\mathbf{u}$ is obtained. However, due to the receding horizon scheme characteristic of any MPC controller, only the first component is applied to the system, computing a whole new sequence at next time instant.

5.3.1 Nominal stability analysis

For the nominal stability analysis, it is assumed that there are no mismatches between the dynamics of the real system and the obtained prediction model. Furthermore, the state vector λ_k^* is assumed to be known. This implies that the kalman filter is not necessary, and thus it is not taken into account. First, note that the proposed model is not strictly linear with respect to the weights λ_k. However, taking into account that

$$\begin{aligned} \lambda_{k+1}^* &= A\lambda_k^* + Bu_k + c \\ \lambda_s^* &= A\lambda_s^* + Bu_s + c, \end{aligned}$$

and subtracting the equations, it is clear that

$$(\lambda_{k+1}^* - \lambda_s^*) = A(\lambda_k^* - \lambda_s^*) + B(u_k - u_s).$$

Making

$$\breve{\lambda}_k^* = \lambda_k^* - \lambda_s^*, \quad \breve{u}_k = u_k - u_s,$$

it is easy to see that the dynamics of $\breve{\lambda}_k^*$ are linear, that is

$$\breve{\lambda}_{k+1}^* = A\breve{\lambda}_k^* + B\breve{u}_k.$$

Now, some additional assumptions are made in order to prove the nominal stability of the controller:

Assumption 5.3. The system described in (5.8) is observable and the pair (A, B) is stabilizable.

Assumption 5.4. Q, R and O are positive definite matrices.

Assumption 5.5. It exists a stabilizing matrix K so that the matrix $(A + BK)$ is Hurwitz.

Assumption 5.6. It exists a matrix P so that the Lyapunov equation

$$(A + BK)^\top P(A + BK) - P = -(Q + K^\top RK) \tag{5.13}$$

is fulfilled.

Assumption 5.7. It exists at least one feasible equilibrium point satisfying the constraints of the controller.

The previous assumptions makes possible to establish the following closed-loop stability lemma.

Lemma 5.8 (Closed-loop stability)**.** The system (5.8) controlled with the proposed MPC controller (5.12) converges asymptotically to the desired reference. In the case where the desired reference is unreachable, the system converges to the feasible reference which minimizes the offset cost while maintaining the asymptotic stability.

Proof. Assuming that the assumptions are satisfied, the proof follows from [124] and [125]. In [124], it is proven the asymptotic stability of the tracking controller for linear systems using a terminal cost term and a terminal constraint. Then, in [125], it is proven that the stability can be maintained even if there is no terminal constraint. In that case, increasing the weighting of the terminal cost makes the domain of attraction to grow. ■

5.3.2 Robust stability analysis

In this section, it is considered that modeling errors may appear, due to the fact that the model of the system is not perfect. For this purpose, the *Robustly asymptotically stability* (RAS) notion showcased in [106] is chosen. Denote e as measurement errors and d as additive disturbances. Also, denote $\mathbf{e}$ and $\mathbf{d}$ as sequences of N_p elements of e and d. Then, the definition of RAS is the following.

Definition 5.9. The origin of the closed loop nonlinear system $x_{k+1} = h(x_k, \kappa(x_k))$ is considered to be RAS in the interior of a set $\mathcal{F}$ with respect to both measurement errors and additive disturbances if it is possible to find a $\mathcal{KL}$ function β [152] and for each $\epsilon > 0$ and compact set $\mathcal{C} \subset \mathcal{F}$ there exists $\delta > 0$ such that

1. $\max(\mathbf{d}) \leq \delta$, $\max(\mathbf{e}) \leq \delta$.
2. Any admissible trajectory is bounded by $\beta(|x|, k) + \epsilon$.

Lemma 5.10 (Robust asymptotically stability)**.** The system (5.8) joined together with the kalman filter observer and controlled with the proposed MPC controller (5.12) is RAS with respect to measurement noise and additive disturbances.

Proof. Proposition 8 in [106] establishes some sufficient conditions to guarantee RAS. Actually, it is only needed to prove the continuity of the Lyapunov function. Here, in the proposed controller, taking into account that the prediction model is linear, then, the parametric quadratic optimization problem is convex and the aforementioned condition holds. Thus, the system is RAS with respect to measurement noise and additive disturbances. ■

5.4 Examples

This section presents two application examples of the proposed controller. First, the application to a simulated single-input single-output system, a continuously stirred tank reactor, is presented, followed by the application to a laboratory temperature control equipment. This is a multivariable system with two inputs and two outputs. Some considerations are taken into account for both examples in order to guarantee that the assumptions are fulfilled:

1. The models obtained for both systems are open-loop stables, i.e. all the eigenvalues in A are less than the unity. Thus, their modes are stable and hence the system is stabilizable (assumption 5.3).

2. Matrices Q, R and O are chosen as the identity matrix of appropriate dimensions times some positive constant, thus they are positive definite (assumption 5.4).

3. K and P are computed using the Matlab "dlqr" function. This returns a stabilizing gain K and the matrix P that satisfies the Lyapunov equation (5.13) (assumptions 5.5 and 5.6).

4. Taking into account the previous statements and that the following examples only consider box constraints in the inputs, it is easy to see that assumption 5.7 is also fulfilled.

5.4.1 Continuously-stirred tank reactor

First, an MPC controller for a Continuously-Stirred Tank Reactor (CSTR) [153] is designed. The dynamics of the system are given by the following set of differential equations

$$\begin{aligned}\frac{\mathrm{d}C_A(t)}{\mathrm{dt}} =& \frac{q_0}{V_T}(C_{Af} - C_A(t)) - k_0 e^{\frac{-E}{RT(t)}} C_A(t),\\ \frac{\mathrm{d}T(t)}{\mathrm{dt}} =& \frac{q_0}{V_T}(T_f - T(t)) + \frac{(-\Delta H_r)k_0}{\rho_d C_p} e^{\frac{-E}{RT(t)}} C_A(t) + \frac{UA}{V_T \rho_d C_p}(T_c(t) - T(t)),\end{aligned}$$

whose parameters are shown in table 5.1. The system has a unique input T_c (temperature of the cooling jacket) whereas the output is C_A, the concentration of the reactant. The regressor is compounded of the last two values of the output $z_k^\top = [y_k,\, y_{k-1}]$. The sampling time is $t_s = 0.5$ min. The data sets D, D^+ and U are obtained from a simulation experiment where random amplitude step input signals were applied to the system. A total of $N = 500$ samples are considered. Also, it is assumed that the measurements are noisy, having an additive Gaussian noise v_k whose variance is $2.5 \cdot 10^{-5}$. The controller considers input constraints as

$$u_{\min} \leq u_k \leq u_{\max},\ \forall k = 1, ..., N_p,$$

where $u_{\min} = 341.5K$, $u_{\max} = 365.5K$ and $N_p = 10$. The weighting parameters Q, R, O, H_1, H_2 and γ are

$$Q = 500, \quad R = 0.02, \quad O = 5000, \quad H_1 = 0.002\,\mathrm{I_N}, \quad \mathrm{H_2} = 0.007\,\mathrm{I_N}, \quad \gamma = 10.$$

On the other hand, the chosen *radial basis* kernel is

$$\langle \varphi_{\bar{z}_i}, \varphi_{\bar{z}_j} \rangle = e^{\frac{-||\bar{z}_i - \bar{z}_j||}{2\sigma^2}},$$

with $\sigma = 7.6741$. The output and input data is scaled as

$$y = \frac{C_A - C_A^{\min}}{C_A^{\max} - C_A^{\min}},\ u = T_c - 353.5\,\mathrm{K}\,.$$

For the kalman filtering, the following parameters are considered

$$\varsigma_w = 5 \cdot 10^{-6}\,\mathrm{I_N},\ \varsigma_\mathrm{v} = 2.5 \cdot 10^{-5},\ \tilde{\sigma}_0 = 0.1\,\mathrm{I_N}.$$

The results are shown in figure 5.1. The blue line correspond to the reference both in the input and the output whereas the red line corresponds to the real output and the inputs applied to the system. It is clear that the controller steers the system to the desired references. However, due to the noisy measurements and the discrepancies of the model with respect to the real system, it is possible to see some offset. This is something to be expected and several strategies could be used to solve this, e.g. using an appropriate disturbance model [154] or augmenting the state [155] (see [156] for a tutorial on the subject).

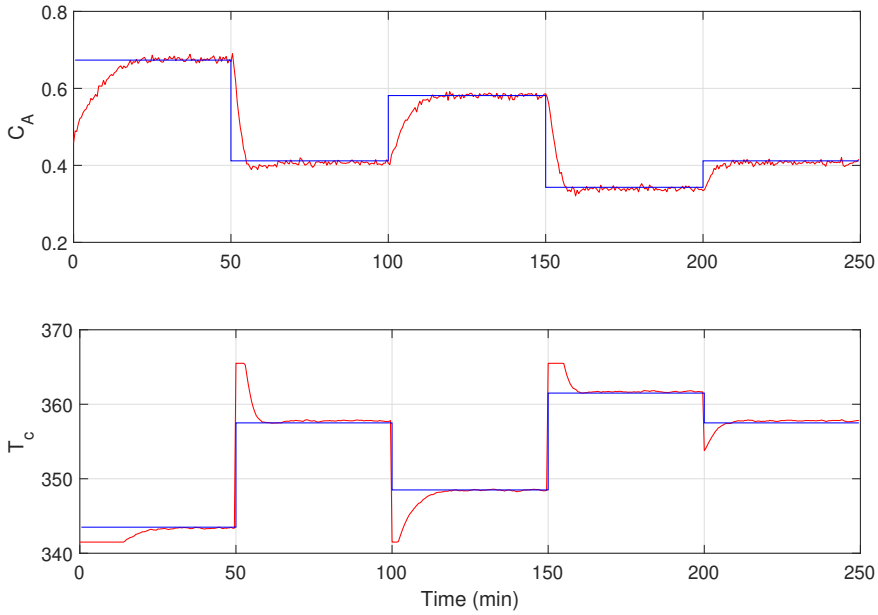


Figure 5.1: Top figure: output of the system. Bottom figure: input applied to the system.

5.4.2 Temperature control lab

The temperature control lab is an application of feedback control compounded of an Arduino and a shield with two heaters and two temperature sensors. The heaters correspond to the inputs whereas the temperatures are the outputs of the system. The heater power output can be adjusted so that a certain temperature reference is attained. The thermal energy within the system is transferred by conduction, convection, and radiation. Also, heat is transferred away from the device to the surroundings [157]. Thus, the dynamics of the system are nonlinear.

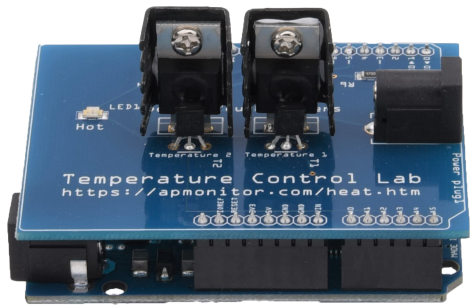


Figure 5.2: Temperature Control Lab.

The data sets D, D^+ and U are comprised of past trajectories of the real system. These are obtained from an experiment where random step signals were applied to the system as it can be seen in figure 5.3. There are a total of $N = 500$

Parameter	Meaning	Value	Units
q_0	Input flow of the reactive	10	$\mathrm{l\ min^{-1}}$
V_T	Liquid volume in the tank	150	l
k_0	Frecuency constant	6×10^{10}	$\mathrm{min^{-1}}$
E/R	Arrhenius constant	9750	K
$-\Delta H_r$	Enthalpy of the reaction	10000	$\mathrm{J\ mol^{-1}}$
UA	Heat transfer coefficient	70000	$\mathrm{J\ min^{-1}K^{-1}}$
ρ_d	Density	1100	$\mathrm{g\ l^{-1}}$
C_p	Specific heat	0.3	$\mathrm{J\ g^{-1}K^{-1}}$
C_{Af}	C_A in the input flow	1	$\mathrm{mol\ l^{-1}}$
T_f	Temperature (input flow)	370	K

Table 5.1: Parameters of the CSTR model

of data points. The regressor is compounded of the last values of the output $z_k^\top = [T_{1,k}, T_{2,k}]$ where T_1 and T_2 correspond to the aforementioned output temperatures. The sampling time is set to $t_s = 0.5$ min. Same as before, input constraints are considered as

$$u_{\min} \leq u_k \leq u_{\max},\ \forall k = 1, ..., N_p$$

where $u_{\min} = 0$, $u_{\max} = 100$ and $N_p = 10$. The weighting parameters Q, R, O, H_1, H_2 and γ are

$$Q = 10\,\mathrm{I_{n_y}},\ \ \mathrm{R} = 1\,\mathrm{I_{n_u}},\ \ \mathrm{O} = 1000\,\mathrm{I_{n_y}},\ \ \mathrm{H_1} = 0.005\,\mathrm{I_N},\ \ \mathrm{H_2} = 0.008\,\mathrm{I_N},\ \ \gamma = 1.$$

Again, a *radial basis kernel* is used with $\sigma = 0.413$ and the output and input data is scaled, making the data to range from 0 to 1. The parameters of the kalman Filter are the following

$$\varsigma_w = 10^{-6}\,\mathrm{I_N}, \quad \varsigma_\mathrm{v} = \mathrm{I_{n_y}}, \quad \tilde{\sigma}_0 = 0.05\,\mathrm{I_N}.$$

The results are shown in figure 5.4. The red line corresponds to the temperature of the first heater T_1 and the value of the first input u_1, the blue line to the temperature of the second heater T_2 and the second input u_2, the black-dashed line is the reference for T_1 and u_1 and the green-dashed line is the reference for T_2 and u_2. The MPC controller activates at $t = 5$ min. As in the previous example, the controller is able to steer the outputs of the system to the desired references. Also, there is some minor offset, which is more evident in the control actions plots, where it is evident that their values at equilibrium (y_s, u_s) do not match with those expected from the dataset (dotted plot). This is due to the unknown ambient temperature that may vary a lot from one day to another or even within the same day. The experiment of figure 5.4 and the dataset were obtained in different days, thus, the equilibrium points are not exactly congruent.

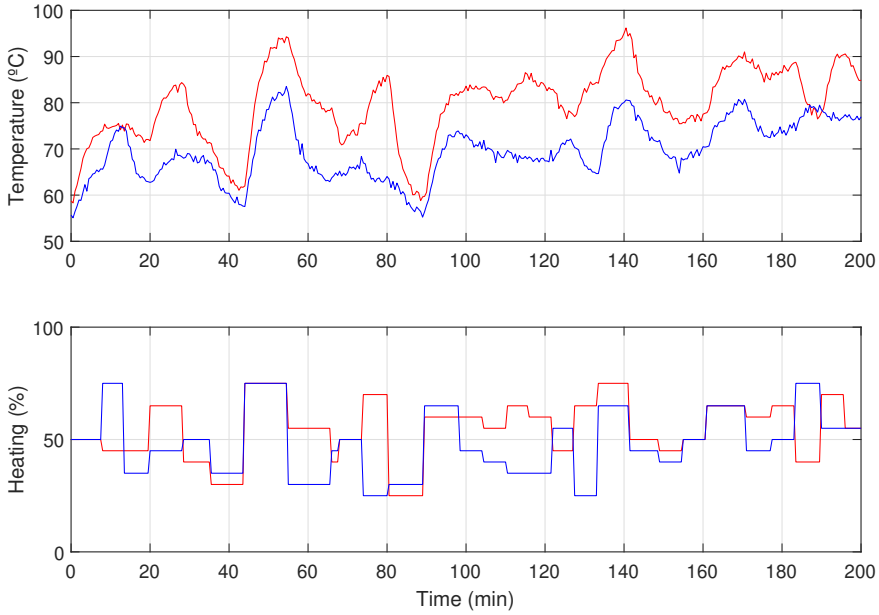


Figure 5.3: Experimental data obtained: Top figure: output of the system. Bottom figure: input applied to the system.

5.5 Conclusions

In this chapter, the state-space kriging method for non-autonomous systems was presented. It was shown that the input term could be taken into account by choosing appropriately the constraints within the dissimilarity function. As the new proposed models could handle external inputs, they became suitable for control. An MPC controller where the prediction model is a K-SSK model was proposed. Several properties like nominal stability and robust stability were proved. Finally, both a simulation example and a real experiment were conducted in order to show the effectiveness of the proposed controller.

Appendix A

As shown in the following, problem (5.12) can be posed as a canonical quadratic programming problem. For that purpose, using the previous definition of the system shown in (5.8), it is clear that, given a certain λ_k^*, $\lambda_{j|k}^*$ can be obtained by iterating λ_k^* repeatedly forward

$$\lambda_{j|k}^* = A^j \lambda_k^* + \sum_{i=0}^{j-1} A^{j-1-i} B u_{k+i} + \sum_{i=0}^{j-1} A^{j-1-i} c\,.$$

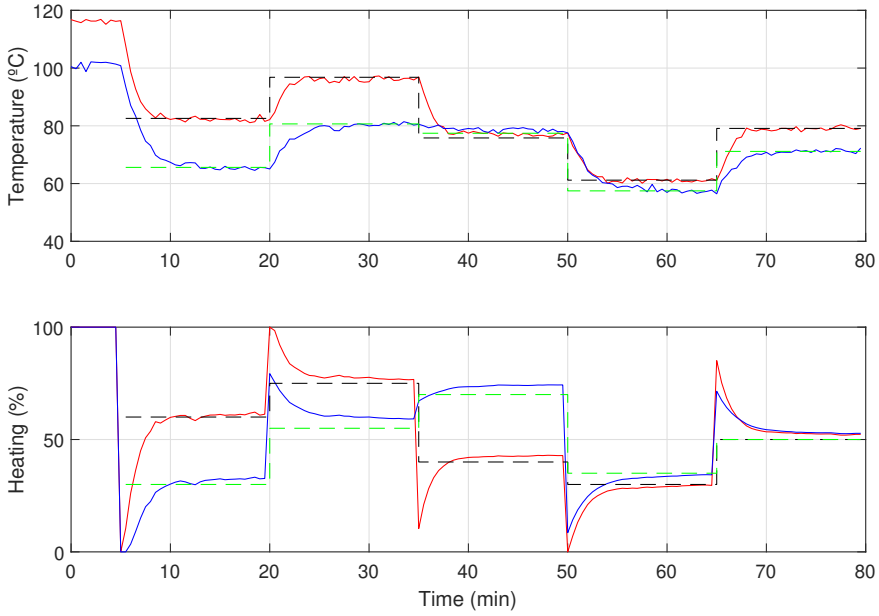


Figure 5.4: Top figure: outputs of the system. Bottom figure: inputs applied to the system.

As $y_k = Y^+\lambda_k^*$, then

$$\tilde{y}_{j|k} = Y^+A^j\lambda_k^* + \sum_{i=0}^{j-1} Y^+A^{j-1-i}Bu_{k+i} + \sum_{i=0}^{j-1} Y^+A^{j-1-i}c\,. \tag{5.15}$$

Equation (5.15) can be written in matrix form as

$$\mathbf{y} = \mathbf{n} + \mathbf{M}\mathbf{u}\,,$$

where

$$\mathbf{n} = \begin{bmatrix} Y^+\lambda_0^* \\ Y^+A\lambda_0^* + c \\ \vdots \\ Y^+A^{N_p-1}\lambda_0^* + \sum_{i=0}^{N_p-2} Y^+A^{N_p-2-i}c \end{bmatrix},$$

$$\mathbf{M} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ Y^+B & 0 & 0 & \cdots & 0 & 0 \\ Y^+AB & Y^+B & 0 & \cdots & 0 & 0 \\ Y^+A^2B & Y^+AB & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & & \vdots & \vdots \\ Y^+A^{N_p-2}B & Y^+A^{N_p-3}B & \cdots & Y^+B & 0 & 0 \end{bmatrix}.$$

Then, the constraints (5.12e) and (5.12f) can be posed as

$$\Psi \mathbf{M}\mathbf{u} \leq \Delta - \Psi \mathbf{n}$$

where

$$\Psi = \begin{bmatrix} \psi & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \psi \end{bmatrix}, \quad \Delta = \begin{bmatrix} \delta \\ \vdots \\ \delta \end{bmatrix},$$

and

$$\underline{\mathbf{u}} \leq \mathbf{u} \leq \overline{\mathbf{u}},$$

where

$$\underline{\mathbf{u}} = \begin{bmatrix} u_{\min} \\ \vdots \\ u_{\min} \end{bmatrix}, \quad \overline{\mathbf{u}} = \begin{bmatrix} u_{\max} \\ \vdots \\ u_{\max} \end{bmatrix}.$$

On the other hand, as it was previously stated, (λ_s^*, u_s) is an equilibrium pair and thus they satisfy

$$\lambda_s^* = A\lambda_s^* + Bu_s + c.$$

Also, they can be obtained as

$$\begin{aligned} \lambda_s^* &= \left(\mathrm{I_N} - \mathrm{A}\right)^{-1} \left(Bu_s + c\right), \\ y_s &= Y^+ \left(\mathrm{I_N} - \mathrm{A}\right)^{-1} \left(Bu_s + c\right), \end{aligned}$$

which can be written as

$$\begin{aligned} \lambda_s^* &= W\mathbf{u} + v, \\ y_s &= Y^+ W\mathbf{u} + Y^+ v, \end{aligned}$$

where

$$W = \begin{bmatrix} 0 & 0 & \dots & 0 & \left(\mathrm{I_N} - \mathrm{A}\right)^{-1} B \end{bmatrix},$$

$$v = \left(\mathrm{I_N} - \mathrm{A}\right)^{-1} c.$$

Also, it is convenient to define

$$\mathbf{y}_s = \begin{bmatrix} y_s \\ \vdots \\ y_s \end{bmatrix} = \mathbf{W}_Y \mathbf{u} + \mathbf{v}_Y,$$

where

$$\mathbf{W}_{Y^+} = \begin{bmatrix} Y^+ W \\ \vdots \\ Y^+ W \end{bmatrix}, \quad \mathbf{v}_Y = \begin{bmatrix} Y^+ v \\ \vdots \\ Y^+ v \end{bmatrix}.$$

Once reached this point, it is possible to write the output tracking error term in matrix form. Now, making

$$\mathbf{L} = \begin{bmatrix} \mathrm{I}_{\mathrm{n_u}} & 0 & 0 & -\mathrm{I}_{\mathrm{n_u}} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \mathrm{I}_{\mathrm{n_u}} & -\mathrm{I}_{\mathrm{n_u}} \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} Q & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & Q \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} R & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & R \end{bmatrix},$$

we obtain

$$\begin{aligned} &\sum_{i=0}^{N_p-1} l_s(\breve{y}_{i|k}, \breve{u}_{k+i}) = (\mathbf{y} - \mathbf{y}_s)^\top \mathbf{Q} (\mathbf{y} - \mathbf{y}_s) + \mathbf{u}^\top \mathbf{L}^\top \mathbf{R} \mathbf{L} \mathbf{u} \\ &= (\mathbf{M}\mathbf{u} + \mathbf{n} - \mathbf{W}_Y \mathbf{u} + \mathbf{v}_Y)^\top \mathbf{Q} (\mathbf{M}\mathbf{u} + \mathbf{n} - \mathbf{W}_Y \mathbf{u} + \mathbf{v}_Y) + \mathbf{u}^\top \mathbf{L}^\top \mathbf{R} \mathbf{L} \mathbf{u}. \end{aligned}$$

Also, $l_o(y_s, r)$ can be written as

$$l_o(y_s, r) = \left(Y^+ W \mathbf{u} + Y^+ v - r\right)^\top O \left(Y^+ W \mathbf{u} + Y^+ v - r\right).$$

It only remains to obtain a matrix expression for $l_t(\lambda^*_{N_p|k}, \lambda^*_s)$. As it was already shown how to obtain λ^*_s, it is only needed to show how to obtain $\lambda^*_{N_p|k}$. Using equation (5.15), it is clear that

$$\lambda^*_{N_p|k} = G\mathbf{u} + h,$$

where

$$G = \begin{bmatrix} A^{N_p-1}B & A^{N_p-2}B & \cdots & B & 0 \end{bmatrix},$$

$$h = A^{N_p-1}\lambda^*_k + \sum_{i=0}^{N_p-1} A^{N_p-1-i} c,$$

and thus

$$l_t(\lambda^*_{N_p|k}, \lambda^*_s) = \gamma \left(G\mathbf{u} + h - W\mathbf{u} - v\right)^\top P \left(G\mathbf{u} + h - W\mathbf{u} - v\right).$$

It is easy to see that the optimization problem in (5.12) becomes

$$\begin{aligned} \min_{\mathbf{u}} \ & \frac{1}{2}\mathbf{u}^\top H \mathbf{u} + f^\top \mathbf{u} + \mathrm{cnt} \\ \text{s.t.} \ & \Psi \mathbf{M} \mathbf{u} \leq \Delta - \Psi \mathbf{n} \\ & \underline{\mathbf{u}} \leq \mathbf{u} \leq \overline{\mathbf{u}}. \end{aligned}$$

where

$$\begin{aligned} H &= 2((\mathbf{M} - \mathbf{W}_Y)^\top \mathbf{Q} (\mathbf{M} - \mathbf{W}_Y) + \mathbf{L}^\top \mathbf{R} \mathbf{L} + W^\top {Y^+}^\top O Y^+ W \\ &\quad + \gamma((G - W)^\top P (G - W))), \\ f^\top &= 2(\mathbf{n} - \mathbf{v}_Y)^\top \mathbf{Q} (\mathbf{M} - \mathbf{W}_Y) + 2(Y^+ v - r)^\top O Y^+ W + 2\gamma (h - v)^\top P (G - W), \\ \mathrm{cnt} &= (\mathbf{n} - \mathbf{v}_Y)^\top \mathbf{Q} (\mathbf{n} - \mathbf{v}_Y) + (Y^+ v - r)^\top O (Y^+ v - r) + \gamma (h - v)^\top P (h - v). \end{aligned}$$

Thus, the control action is computed as

$$\begin{aligned} \mathbf{u}^* = \arg\min_{\mathbf{u}} \ & \frac{1}{2}\mathbf{u}^\top H\mathbf{u} + f^\top \mathbf{u} \\ \text{s.t.} \ & \Psi\mathbf{M}\mathbf{u} \leq \Delta - \Psi\mathbf{n} \\ & \underline{\mathbf{u}} \leq \mathbf{u} \leq \overline{\mathbf{u}}, \end{aligned}$$

where only the first component of the minimizer sequence $\mathbf{u}^*$ is applied following a receding horizon scheme as usual in any predictive controller.

Part III

Probabilistically-certified data center management

Chapter 6

Bounds on the constraint violation level

6.1 Introduction

Bounds on the constraint violation levels are useful in constrained control problems as they provide a measure on how likely are the constraints to be violated in practice. Being MPC the most popular control technique using constraints, it will be chosen for the example in this chapter.

Consider a nonlinear system which is also affected by noises and disturbances, that is

$$\begin{aligned} x_{k+1} &= h(x_k, u_k, w_k) \\ y_k &= g(x_k, v_k), \end{aligned}$$

where k is the time instant, $x_k \in \mathbb{R}^{n_x}$ is the state of the system, $u_k \in \mathbb{R}^{n_u}$ is the input of the system, $y_k \in \mathbb{R}^{n_y}$ is the output of the system, w_k are disturbances in the state, v_k is measurement noise and $h(\cdot)$ and $g(\cdot)$ are nonlinear functions such that $h(\cdot) : \mathbb{R}^{n_x \times n_u} \to \mathbb{R}^{n_x}$ and $g(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$.

In this case, the control action computed using a deterministic MPC law may not be optimal for every possible realization of w_k and v_k. Also, it is not possible to guarantee that the constraints will be satisfied. For that reason, it may be needed to resort to robust MPC schemes, which are known to be overly conservative in most of the cases as it was showcased in the introduction. Another possibility would be to rely on strategies based on randomized settings such as stochastic MPC or chance-constrained MPC, which may reduce this conservatism. However, this may become an intractable problem sometimes.

Instead, in this chapter, two methodologies to bound the constraint violation rate of a finite family of controllers by means of offline simulations of the closed-loop

system are presented. This finite family of controllers can be compounded of any controller, not necessarily MPC controllers. The obtained bounds will allow us to compare the proposed controllers in order to obtain the best one according to a pre-specified criterion that could weight performance and constraint violation. Also, it does not increase the computation burden of the control problem online.

6.2 Main results

Assume that a certain controller is parameterized by means of a decision vector $\theta \in \Theta$, where Θ is a set compounded of all the possible values that the vector θ can take. The disturbance w follows a certain probability distribution whose sample space is $\mathcal{W}$. No assumptions are made with respect to the size or shape of $\mathcal{W}$. From this, a function $q : \Theta \times \mathcal{W} \to \{0, 1\}$ is defined. This will be helpful to formulate the problems of this section in a general setting. Thus, the function $q(\cdot)$ is as follows

$$q(\theta, w) = \begin{cases} 0 & \text{if } \theta \text{ fulfills some design specifications for } w \\ 1 & \text{otherwise.} \end{cases}$$

This specification could be, for example, the satisfaction of a certain constraint subject to the realization w. Then, given θ, the probability of violation of the aforementioned constraint for any $w \in \mathcal{W}$ can be written as

$$\psi = \text{Prob}\left\{q(\theta, w) = 1 \,|\, \theta\right\}.$$

As it is almost impossible to compute or measure this quantity exactly, we rely on the empirical mean to approximate this value. Consider w_i as the i-th realization of the disturbance w, and assuming a number of N samples, then

$$\underline{\psi} = \frac{1}{N} \sum_{i=1}^{N} q(\theta, w_i),$$

where $\underline{\ }$ denotes the empirical mean. Thus, $\underline{\psi}$ would be the empirical mean of ψ for any given experiment. Note that as $\underline{\psi}$ is also a random variable, any set of experiments, that is, N different realizations of $q(\theta, w_i)$ lead to different values for $\underline{\psi}$.

The objective of this chapter is to obtain tight bounds for this empirical violation rate.

Definition 6.1. Failure. Assuming that

$$\text{E}(\psi) > \rho + \Delta\rho,$$

where $\text{E}(\cdot)$ is the expectation operator, $\rho \in [0, 1]$ and $\Delta\rho > 0$. A failure is considered to happen when an empirical violation rate of the controller $\underline{\psi}$ fulfills

$$\underline{\psi} \leq \rho.$$

That is, a *failure* happens when the empirical violation rate leads to misleading results, i.e., when given a empirical violation rate $\underline{\psi} \leq \rho$, the real violation rate ψ increases a quantity larger than $\Delta\rho$. The following theorems prove that, with a properly chosen $\Delta\rho$, failures occur with a probability lower than a small confident parameter δ. This definition of failure is slightly different than the one presented, for example, in [158]. Then, the following theorems summarize the results of this chapter.

6.2.1 A first bound on constraint violation rate

Theorem 6.2 (Bound on the empirical constraint violation level)**.** Assuming that

$$E(\psi) > \rho + \Delta\rho\,,$$

with $\rho \in [0,1]$ and $\Delta\rho$ satisfying

$$\Delta\rho \geq \frac{1}{N}\log\frac{1}{\delta} + 2\sqrt{\frac{\rho}{N}\log\frac{1}{\delta}}\,. \tag{6.1}$$

Then,

$$\text{Prob}\left\{\underline{\psi} \leq \rho\right\} < \delta\,,$$

which implies that the real constraint violation level is bounded by $\rho + \Delta\rho$ with probability greater than $1 - \delta$.

Proof. The objective is to upper bound $\text{Prob}\left\{\underline{\psi} \leq \rho\right\}$ under the assumption that $E(\psi) > \rho + \Delta\rho$. Since the number of empirical violations grows with $\text{E}(\psi)$, it is clear that this probability is lower bounded by the probability when $\text{E}(\psi) = \rho + \Delta\rho$. That is,

$$\text{Prob}\left\{\underline{\psi} \leq \rho \,|\, \text{E}(\psi) > \rho + \Delta\rho\right\} < \text{Prob}\left\{\underline{\psi} \leq \rho \,|\, \text{E}(\psi) = \rho + \Delta\rho\right\}\,.$$

Once reached this point, it is easy to notice that the probability of a certain constraint to be violated can be interpreted as a Bernoulli random variable. Therefore, the probability of having less than a certain number of violations for some number of trials can be expressed as the binomial tail.

On the other hand, consider a random variable X following a binomial random distribution $\mathcal{B} \sim (N, p)$ where N is the number of trials and p is the probability of success. Then, given a constant x, it is obtained from Chernoff's bound [128, 159] that, if $x \leq p$, then

$$\text{Prob}\left\{\frac{X}{N} \leq x \,|\, \text{E}\left(\frac{X}{N}\right) = p\right\} \leq e^{-N\varphi(\rho,\rho+\Delta\rho)},$$

where

$$\varphi(x,p) = x\log\frac{x}{p} + (1-x)\log\frac{1-x}{1-p}\,.$$

The convergence rate provided by this bound is known to be tight from Cramér's theorem of large deviations when $N \to \infty$ ([160], chapter 23 in [161]). It is easy to see that the above Chernoff's bound can be applied to our problem making $\frac{X}{N} = \underline{\psi}$ and $\Delta\rho > 0$, obtaining

$$\text{Prob}\left\{\underline{\psi} \leq \rho \,|\, \text{E}\left(\psi\right) = \rho + \Delta\rho\right\} \leq e^{-N\varphi(\rho,\rho+\Delta\rho)}\,.$$

Thus, this upper-bound can be designed to be lower than the confidence parameter δ by imposing

$$e^{-N\varphi(\rho,\rho+\Delta\rho)} \leq \delta\,.$$

Taking logarithms in both sides and rearranging the terms

$$\varphi(\rho, \rho + \Delta\rho) \geq \frac{1}{N}\log\frac{1}{\delta}\,. \tag{6.2}$$

As $\Delta\rho$ is embedded within the function $\varphi(\rho, \rho + \Delta\rho)$, the bound presented by Okamoto (Lemma 2 in [159]) is applied to obtain a closed expression for $\Delta\rho$,

$$\varphi(\rho, \rho + \Delta\rho) \geq \left(\sqrt{\rho + \Delta\rho} - \sqrt{\rho}\right)^2.$$

Thus, the following sufficient condition for equation (6.2) is obtained

$$\left(\sqrt{\rho + \Delta\rho} - \sqrt{\rho}\right)^2 \geq \frac{1}{N}\log\frac{1}{\delta}\,.$$

For $\Delta\rho \geq 0$, this is equivalent to

$$\sqrt{\rho + \Delta\rho} \geq \sqrt{\rho} + \sqrt{\frac{1}{N}\log\frac{1}{\delta}}\,,$$

and thus

$$\rho + \Delta\rho \geq \left(\sqrt{\rho} + \sqrt{\frac{1}{N}\log\frac{1}{\delta}}\right)^2.$$

This is easily rewritten as

$$\Delta\rho \geq \frac{1}{N}\log\frac{1}{\delta} + 2\sqrt{\frac{\rho}{N}\log\frac{1}{\delta}}\,.$$

This completes the proof. ■

6.2.2 A different bound

A different bound will be given in the following. This new bound is not guaranteed to be tighter than the previous one but it can yield better results in some cases.

Theorem 6.3. Suppose that

$$E\left(\psi\right) > \rho + \Delta\rho\,,$$

with $\rho \in [0,1]$ and

$$\Delta\rho \geq \frac{\log\frac{1}{\delta} + \lfloor \rho N \rfloor \log a}{N\left(1-\frac{1}{a}\right)} - \rho, \quad \forall a \geq 1\,. \tag{6.3}$$

Then,

$$\text{Prob}\left\{\underline{\psi} \leq \rho\right\} < \delta\,,$$

As in theorem (6.2), this implies that the real constraint violation level is bounded by $\rho + \Delta\rho$ with probability greater than $1-\delta$.

Proof. Assuming that $E\left(\psi\right) > \rho + \Delta\rho$, one could write this probability as

$$\text{Prob}\left\{\underline{\psi} \leq \rho \,|\, \text{E}\left(\psi\right) > \rho + \Delta\rho\right\}\,.$$

Again, as the number of empirical violations grows with $\text{E}\left(\psi\right)$, it is clear that this probability is lower bounded by the probability when $\text{E}\left(\psi\right) = \rho + \Delta\rho$, i.e.

$$\text{Prob}\left\{\underline{\psi} \leq \rho \,|\, \text{E}\left(\psi\right) > \rho + \Delta\rho\right\} < \text{Prob}\left\{\underline{\psi} \leq \rho \,|\, \text{E}\left(\psi\right) = \rho + \Delta\rho\right\}\,.$$

Similarly, as in theorem 6.1, it is easy to notice that this probability can be seen as a Bernoulli random variable and thus the probability of having less than a certain number of violations for some number of trials can be expressed as the binomial tail

$$\text{B}(N,\eta,m) = \sum_{y=0}^{m} \binom{N}{y} \eta^y (1-\eta)^{N-y}\,,$$

where $B(N,\eta,m)$ represents the mass probability function of the Binomial distribution. From the definition of $\underline{\psi}$, it is easy to see that $\lfloor \underline{\psi} N \rfloor = \underline{\psi} N$ whereas $\lfloor \rho N \rfloor \leq \rho N$. Then,

$$\text{Prob}\left\{\underline{\psi} \leq \rho \,|\, \text{E}\left(\psi\right) = \rho + \Delta\rho\right\} = \text{Prob}\left\{\lfloor \underline{\psi} N \rfloor \leq \lfloor \rho N \rfloor \,|\, \text{E}\left(\psi\right) = \rho + \Delta\rho\right\}.$$

Now, it is possible to find the equivalence between this probability and the binomial tail as

$$\text{Prob}\left\{\lfloor \underline{\psi} N \rfloor \leq \lfloor \rho N \rfloor \,|\, \text{E}\left(\psi\right) = \rho + \Delta\rho\right\} = \text{B}(N, \rho + \Delta\rho, \lfloor \rho N \rfloor)\,. \tag{6.4}$$

Lemma 1 in [61] shows that the binomial tail can be bounded by the following expression

$$\text{B}(N,\eta,m) \leq a^m \left(\frac{\eta}{a} + 1 - \eta\right)^N, \quad \forall a \geq 1\,.$$

Then, $\mathrm{B}(N, \eta, m)$ can be guaranteed to be lower or equal than a certain δ if:

$$a^m \left(\frac{\eta}{a} + 1 - \eta\right)^N \leq \delta \,.$$

It is also shown in [61] that the value of a is very close to the optimal if it is chosen as

$$a = 1 + \frac{\log(\frac{1}{\delta})}{\lfloor \rho N \rfloor} + \sqrt{2\frac{\log(\frac{1}{\delta})}{\lfloor \rho N \rfloor}} \,. \tag{6.5}$$

The previous inequality can be rewritten as

$$\left(1 - \eta\left(1 - \frac{1}{a}\right)\right)^N \leq \frac{\delta}{a^m} \,.$$

Taking logarithms in both sides

$$N\log\left(1 - \eta\left(1 - \frac{1}{a}\right)\right) \leq \log\delta - m\log a \,. \tag{6.6}$$

Now, the following logarithm inequality can be applied

$$\log(1 - x) \leq -x \,, \quad \forall x \in [0, 1) \,,$$

where $x = \left(1 - \frac{1}{a}\right)\eta$ in this case. Then, a sufficient condition to fulfill equation (6.6) is the following

$$-N\left(1 - \frac{1}{a}\right)\eta \leq \log\delta - m\log a \,,$$

and thus

$$\eta \geq \frac{\log\frac{1}{\delta} + m\log a}{N\left(1 - \frac{1}{a}\right)} \,.$$

Applying $\eta = \rho + \Delta\rho$, $m = \lfloor \rho N \rfloor$

$$\Delta\rho \geq \frac{\log\frac{1}{\delta} + \lfloor \rho N \rfloor \log a}{N\left(1 - \frac{1}{a}\right)} - \rho \,.$$

This completes the proof. ■

6.3 Clarifying Example

Consider a discrete time linear system

$$x_{k+1} = Ax_k + Bu_k + w_k$$

where

$$A = \begin{bmatrix} 0.9 & 0.4 \\ -0.4 & 0.85 \end{bmatrix}, \quad B = \begin{bmatrix} -0.1 \\ -0.3 \end{bmatrix},$$

and w_k is a uniformly distributed random variable so that $w \in [-0.05, 0.05]$.

Also, consider a naive MPC controller

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} \; V_N(x_{N_p|k}) + \sum_{i=0}^{N_p-1} l(x_{i|k}, u_{k+i}) \tag{6.7a}$$

$$\text{s.t.} \; x_{i+1|k} = Ax_{i|k} + Bu_{k+i} \tag{6.7b}$$

$$\begin{bmatrix} 1 & 0 \end{bmatrix} x_{i|k} \geq -0.05 + \gamma \quad \forall i = 1, \ldots, N_p \tag{6.7c}$$

$$-2 \leq u_{k+i} \leq 2 \quad \forall i = 0, \ldots, N_p \tag{6.7d}$$

where $\gamma \geq 0$ is a back-off parameter. The stage cost is chosen as

$$l(x, u) = x^\top Q x + u^\top R u,$$

with

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = 0.1,$$

and the terminal cost is chosen as

$$V_N(x_{N_p|k}) = x_{N_p|k}^\top P x_{N_p|k},$$

with

$$P = \begin{bmatrix} 3.6681 & 0.5318 \\ 0.5318 & 1.6995 \end{bmatrix},$$

which is obtained by solving the Lyapunov equation.

The objective is to characterize the probability of violating the constraint (6.7c) by using the proposed bounds. The possible lose of recursive feasibility due to a constraint violation is out of the scope of this example. However, note that the constraint is relaxed for $i = 0$ in order to prevent this.

A number of 100 experiments were carried out, each one with a duration of 50 time steps. The initial conditions are random points ranging from $x_{0,(1)} \in [0, 2]$ and $x_{0,(2)} \in [0, 2]$. The mean closed-loop total cost is computed at the end of the experiments, denoted as J. The results are shown in tables 6.1 and 6.2 and figure 6.1.

Table 6.1 and figure 6.1 presents the results only for the first 10 experiments (thus $N = 500$, and ρ will be computed over these 500 samples). The red line corresponds to $\gamma = 0$, the blue line corresponds to $\gamma = 0.025$, the green line corresponds to $\gamma = 0.05$ and the black line corresponds to the constraint (6.7c). It can be seen how as the back-off parameter γ increases, the resulting trajectories

are pushed to the right side, reducing the probability of violating the constraint but also increasing the cost. In the table, it is shown the obtained values of the bounds for theorems 6.2 and 6.3. The bounds obtained by means of the theorem 6.3 are generally sharper than the ones obtained by using theorem 6.2, being worse only for the value of $\gamma = 0.05$, that is, when ρ is very close to zero.

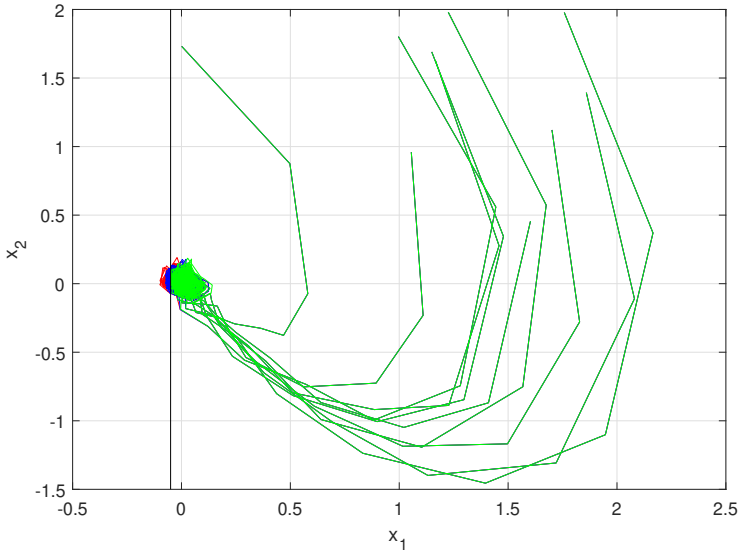


Figure 6.1: 10 trajectories for different back-offs parameters.

	ρ	$\rho + \Delta\rho$ (2.2)	$\rho + \Delta\rho$ (2.3)	J
$\gamma = 0$	0.1200	0.2628	0.2210	0.0055
$\gamma = 0.025$	0.0700	0.1856	0.1520	0.0066
$\gamma = 0.05$	0.0000	0.0276	0.0307	0.0109

Table 6.1: Bounds on the probabilistic constraint with $\delta = 10^{-6}$ and $N = 500$.

Finally, table 6.2 shows the results for the 100 experiments ($N = 5000$). As making more experiments increases the number of samples N, the bounds become much smaller as it is expected.

	ρ	$\rho + \Delta\rho$ (2.2)	$\rho + \Delta\rho$ (2.3)	J
$\gamma = 0$	0.0968	0.1323	0.1218	0.0058
$\gamma = 0.025$	0.0568	0.0846	0.0764	0.0073
$\gamma = 0.05$	0.0000	0.0028	0.0031	0.0109

Table 6.2: Bounds on the probabilistic constraint with $\delta = 10^{-6}$ and $N = 5000$.

6.4 Conclusions

In this chapter, two different sharp bounds on the constraint violation of a certain constraint to be used with any controller were proposed. It was shown that we do not need to make any assumption about the probability distribution as long as we can obtain independent identically distributed samples, which is a standard assumption in the field. Also, by means of a simple MPC example, it was shown how the proposed algorithms can be used in a real setting.

In the following chapter, the proposed bounds will be used to quantify the Quality of Service (QoS) of an energy-efficient management approach in a data center. This could help the user to choose the hyperparameters of the controllers so that they fulfill the QoS constraint with a certain probability while minimizing the power consumption as much as possible.

Chapter 7

Energy-efficient management of data centers

7.1 Introduction

Data centers are facilities composed by a large amount of servers and the associated support infrastructure. The variety of tasks that can be carried out by these infrastructures, such as batch and interactive computation, web portals, etc [162] have led to a sustained growing of the number of active data centers in the last decades [163]. Some studies predict that data centers within the U.S. will consume around 3000 TWh electricity by 2030 [164]. In addition to the economic burden derived from the electricity prices (which is particularly high nowadays), this extremely high power consumption also leads to serious concerns due to environmental issues [165]. Thus, improvements in the energy efficiency of data center management are critical for a sustainable society.

The energy consumption of a data center can be broadly associated to Information Technology (IT) equipment (i.e. servers) and infrastructure facilities, mainly the cooling equipment [166, 167]. The total amount of energy consumed will depend on both the design and the efficiency of the policies used to manage these facilities. An integrated data center management must consider not only the thermal constraints of the IoT equipments and the total energy consumption but also keep the performance near the required levels at all times. Thus, the way in which both the IT and cooling equipments are used to operate the data center will have a great impact on the overall energy consumption.

Many works in the literature are focused on developing cooling policies to minimize the energy consumption within data centers using only linear thermal models [168, 169, 170]. In spite of being simple first order linear models, interesting improvements were reported. However, better solutions could be attained with a unified data center management. In this field, works are rather scarce. For

example, the paper [162] proposes a control architecture where both thermal and tasks management are taken into account. The data center is posed as a linear first-order continuous system in its thermal component and in the computational part. Task arrival rates are considered deterministic, thus tasks arrive at fixed intervals. A Quality of Service (QoS) constraint is used to establish stability limits to the system. On the other hand, in [171], several control policies assuming that the maximum temperature of the servers are soft constraints are proposed. Other techniques available are based on workload distribution with some thermal aware criteria [172, 173]. Also, a solution to the problem under the assumption that the re-circulation of hot air is at constant temperature can be found in [174].

Model Predictive Control [86] has also been applied to this problem. In [175], a scheduling methodology based on MPC and electricity prices is proposed in order to reduce the economic impact. On the other hand, [176] unifies the management with the cooling scheme, assuming implicitly that servers have no limits to be overclocked, which is rather unrealistic. Due to the complexity of the problem, in [177] it is provided an approximation algorithm focused on providing fast evaluations of the complex constraints that have to be taken into account. However, the models appearing in the literature, under different assumptions, usually evade the fact that the system is actually a queue model, simplifying drastically the problem at the expense of a less realistic modeling.

This chapter proposes an MPC framework for the optimization of cold aisle data centers. The optimization objective is to guarantee a certain QoS to the users and keep a suitable temperature of the servers while consuming the least amount of energy possible. The data center is modelled as a queue system where the arriving tasks can be computed by multiple servers at the same time. Although it moves away from the usual M/M/c queue and makes the treatment more complicated, it allows more generality within the model, where it is possible to take into account some specific kind of data centers like render farms [178]. Unlike other works available in the literature, the full queue model is used to predict the evolution of the system through the horizon, thus achieving a more realistic modeling. In order to tackle random arrival rates and workloads, two different strategies are proposed. First, it is considered that the arrival rates and workloads are deterministic, that is, it is assumed that they take always the value of the expectation of their respective random variables. On the other hand, a scenario-based approach where a certain number of realizations of the disturbances are drawn is proposed (see chapter 1). The QoS is managed by imposing a hard constraint on the number of powered servers in the optimization problem, thus guaranteeing that tasks are executed within the pre-specified time limits. Although there exist very optimized mixed-integer solvers nowadays such as Gurobi [179], they are not suitable for the related optimization problem. For that reason, particle algorithms were chosen to deal with the optimization problem [180]. These algorithms are highly parallelizable and can obtain important speed ups with a parallel implementation [181].

Finally, the bounds obtained in chapter 6 will be used to quantify the constraint violation rate of the proposed approaches. This makes possible to obtain a different characterization of the constraint violation rate for any proposed controller, which could be helpful to tune some parameters of the controllers like the control horizon or the number of scenarios to be considered within the optimization problem.

7.2 Data center description

Data centers are compounded of thermally-isolated units in which server racks are allocated typically following a Cold Aisle (CA) structure as it can be seen in figure 7.1. Here, the air flow is separated into two different flows, being the first a cold one which reaches every server that it is blown from below the floor by means of suitable built-in fans. For this purpose, a Computer Room Air Conditioning (CRAC) unit is responsible of generating this cold flow. Then, the cold air travels through the servers and getting heated throughout this process and, finally, this hot air returns to the CRAC unit through the ceiling.

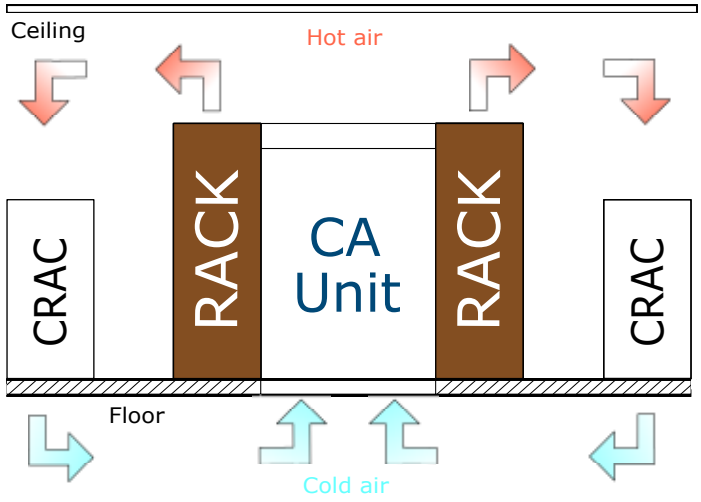


Figure 7.1: Scheme of a cold aisle data center structure.

The CRAC unit is essential within the data center system because the server temperatures should be kept below certain security levels in order to ensure the servers reliability. As almost all power consumed by servers is dissipated as heat, CRAC operation is very costly because of the high number of servers that typical data center has [166]. Thus, any measure aimed to achieve an efficient management of the data center will have a great impact not only in operating costs and environmental impact, but also in the QoS provided to the clients. In the next sections, a discretized model of the data center dynamics with an integration time step of t_s is presented. The discrete time unit will be denoted as k.

7.2.1 Tasks model

In this section, the queue model of the data center operation is presented [171]. It is assumed that the data center has M available servers and m, the number of booted servers, it is considered as an input of the system. In this way, the number of servers working in a certain time instant can be tuned according to the needs of the users (workload) and taking into account efficiency and QoS constraints.

For the whole data center, there is a queue where the tasks wait until they can be processed. That is, until there is an available server. As it is accepted in the literature [171, 176], the time between arrivals is assumed to follow an exponential distribution with mean k_a and a probability density function

$$f(k) = \frac{1}{k_a} e^{-\frac{k}{k_a}}. \tag{7.1}$$

Also, let L_k be the request rate at instant k. That is, the number of tasks arriving to the data center at instant k. Because the time between arrivals was previously assumed to follow an exponential distribution, L_k can be modeled as a Poisson random variable with mean $\frac{1}{k_a}$ and probability mass function

$$g(n) = \frac{1}{n!} \left(\frac{1}{k_a}\right)^n e^{-\frac{1}{k_a}}. \tag{7.2}$$

The number of tasks arriving in the interval $[k,\, k - k_u]$ is denoted as $L_{k_u,k}$ for an interval of length $k_u > 0$, $k_u \in \mathbb{Z}$. Also, the probability mass function is

$$g_{k_u}(n) = \frac{1}{n!} \left(\frac{k_u}{k_a}\right)^n e^{-\frac{k_u}{k_a}}, \tag{7.3}$$

with mean $\frac{k_u}{k_a}$.

As it holds in practice, it is considered that tasks have a different workload. Then, for a certain task, the computational time required to complete it in a single server is assumed to follow an exponential distribution like (7.1) with mean $\frac{1}{\mu}$. Thus, W_k corresponds to the average workload of the L_k tasks that arrived at instant k. Similarly, $W_{k_u,k}$ corresponds to the average workload of the $L_{k_u,k}$ tasks that arrived between k and $k - k_u$. Also, the parameters k_a and μ define the minimum number of servers that should be turned on within the data center so that the queue does not grow infinitely. Then, M must satisfy the condition

$$M \geq \frac{1}{k_a\, \mu}. \tag{7.4}$$

7.2.2 Server model

For a certain server i, the state $x_{k,(i)}$ at instant k is considered to be compounded of:

1. The number of tasks currently running in the data center (α_k).
2. The number of tasks in the queue (β_k).
3. The temperature of the cold air ($T_{c,k}$).
4. The temperature of the server ($T_{k,(i)}$).
5. The time instant in which the server i is turned on (s_i).

Obtaining the state of the whole data center is easy as it suffices to include the remaining $T_{k,(i)}$ and $s_{(i)}$. Therefore, $x_k^\top$ is obtained as

$$x_k^\top = \left[\alpha_k,\ \beta_k,\ T_{c,k},\ T_{k,(1)}\ \ldots\ T_{k,(M)},\ s_{(1)}\ \ldots\ s_{(M)}\right].$$

On the other hand, servers switch between four possible working conditions:

- Off. The server does not draw any power.
- Booting. This working condition appears due to the fixed delay k_{on} that it takes to turn on a server from off to working or idle. While in this transition time, the server draws power at the same rate as the idle condition.
- Working. The server is "on" and it is processing a certain task. For simplicity reasons, it is assumed that the power is drawn at a constant rate. In practice, this is equivalent to assume that the CPU frequency is constant.
- Idle. The server is "on" but it is not processing any tasks, drawing less power than in the working state. However, the server is consuming energy for doing nothing.

As usual in the literature and for simplicity reasons, the transition from "on" to "off" is assumed to be instantaneous. Also, the transition from working to idle and vice-versa is considered negligible. Note that because of the transition time k_{on} and the power drawn throughout the booting process, it may be advantageous to keep a certain number of servers in idle state in the case when they are expected to be needed in a near future. Furthermore, the transition from "off" to "on" is always considered to be immediately available (although it takes a k_{on} time to be completed), but the reverse, that is, from "on" to "off", is done in a deferred way, because the server must finish the remaining tasks.

Taken into account the previous working conditions, let $u_{k,(i)}$ be an input that indicates if the server i is switched on ($u_{k,(i)} = 1$) or off ($u_{k,(i)} = 0$) at a certain time instant k. Then, the power consumption of a server i is defined by the following conditions

$$p_{k,(i)}(x_{k,(i)}, u_{k,(i)}) = \begin{cases} 0 & \text{if off } (u_{k,(i)} = 0) \\ a_2 & \text{if booting } (u_{k,(i)} = 1,\ \alpha_k \geq 0,\ k - s_i < k_{\text{on}}) \\ a_2 & \text{if idle } (u_{k,(i)} = 1,\ \alpha_k = 0,\ k - s_i \geq k_{\text{on}}) \\ a_1 + a_2 & \text{if working } (u_{k,(i)} = 1,\ \alpha_k > 0,\ k - s_i \geq k_{\text{on}}) \end{cases}$$

where a_1 is the marginal consumption and a_2 the minimum consumption.

The management approach presented in this thesis does not deal with the task scheduling of the data center [182]. For that reason, it is assumed that the distribution of the tasks among the servers follows some known rules. It is also assumed that a task can be split among multiple servers (up to M). However, a single server can work only for the completion of a single task. Because of the previous assumption, the data center will not work as a $M/M/c$ queue in which each task is scheduled to be executed in a single server, complicating the mathematical modelling. However, it will result in a more general data center model.

These server assignment policies imply the existence of a pool of running tasks whose length is equal to the number of servers in "on" condition (i.e., "working" or "idle"). Once a task is ready to be processed, that is, it is at the front of the queue and the pool has at least one empty slot, it is assigned to at least one server and it never leaves the pool until its completion. Thus, the remaining workload of each task is strictly decreasing.

Denoting m_k as the number of servers turned on at instant k (i.e. working or idle), the processing of a task is done in the following way. Assume that only a certain task is within a pool of m_k servers ($\alpha_k = 1$). The workload of the task is the number of "work packages" that need to be processed to complete the task. At instant k, m_k servers are assigned to this task (because it is the only task in the pool). Thus, every server will compute a "work package" resulting in m_k "work packages" executed. If there are no "work packages" left for the instant $k+1$, the task is completed and ejected from the pool. If the pool is full at a certain instant k (i.e. $\alpha_k = m_k$), the assignation is trivial because every task can only be assigned one server. For the case where $1 < \alpha_k < m_k$, the tasks will be assigned to one server and the task with largest remaining time will be assigned to $m_k - \alpha_k$ servers. This makes sense because it is easy to see that it will achieve better QoS.

7.2.3 Thermal model

The thermal model of the servers is derived from the thermal balance equations

$$K_t \frac{\mathrm{d}T_{(i)}(t)}{\mathrm{dt}} = c_p\, q_a(t)(T_c(t) - T_{(i)}(t)) + p_{(i)}(t)\,, \tag{7.5}$$

where T_c and q_a are the temperature and flow of the cold air provided by the CRAC unit, K_t the server thermal capacity, $T_{(i)}$ and $p_{(i)}$ are the temperature and power consumption of server i respectively and c_p the air heat capacity. Considering a discretization scheme with an integration step t_s, equation (7.5) turns into

$$T_{k+1,(i)} = T_{k,(i)} + \frac{t_s}{K_t}\left(c_p\, q_{a,k}\left(T_{c,k} - T_{k,(i)}\right) + p_{k,(i)}\right)\,. \tag{7.6}$$

A hard constraint is used to guarantee the server reliability, keeping the temperatures under certain safety levels. That is

$$T_{(i)} \leq 80°\mathrm{C} \quad \forall i \in M\,.$$

From the variables affecting $T_{(i)}$ in (7.6), the flow $q_{a,k}$ is assumed to be constant and only $p_{(i)}$ and T_c can be considered manipulable, the first one through the state of the server and the second one is assumed to be regulated by a set point T_r. That is, T_c follows T_r with a first order closed loop dynamics with a time constant τ and unity gain

$$\tau \frac{\mathrm{d}T_c(t)}{\mathrm{dt}} = T_r(t) - T_c(t)\,. \tag{7.7}$$

Similarly, equation (7.7) turns into

$$T_{c,k+1} = T_{c,k} + \frac{t_s}{\tau}\left(T_{r,k} - T_{c,k}\right). \tag{7.8}$$

On the other hand, hard constraints in the input T_r are considered

$$15°\mathrm{C} \leq T_r \leq 25°\mathrm{C}\,.$$

Also, the coefficient of perfomance (CoP) of the CRAC unit will change depending on the cold air temperature (T_c). The CoP represents how expensive is the cooling process of the air flow until a certain temperature. Usually, more power consumption is required to reach lower temperatures. As the CoP increases, the cost will decrease. It can be calculated from the following equation

$$\mathrm{CoP}(T_{c,k}) = 0.0068\,T_{c,k}^2 + 0.0008\,T_{c,k} + 0.458\,, \tag{7.9}$$

which is widely adopted in the literature [173]. Thus, let $\sum_{i=1}^{M} p_{k,(i)}(x_{k,(i)}, u_{k,(i)})$ be the server power consumption at time instant k. The power consumption at the CRAC unit can be computed as

$$\frac{\sum_{i=1}^{M} p_{k,(i)}(x_{k,(i)}, u_{k,(i)})}{\mathrm{CoP}(T_{c,k})}\,.$$

Thus, the total power consumption corresponds to the power drawn by the servers added to the CRAC power consumption, leading to

$$\sum_{i=1}^{M}\left(1 + \frac{1}{\mathrm{CoP}(T_{c,k})}\right) p_{k,(i)}(x_{k,(i)}, u_{k,(i)})\,.$$

7.2.4 Quality of service

The QoS of a task is defined as the time required to finish it since its arrival until its completion. It includes both the waiting time within queue and the execution time once it is in the pool. Guaranteeing that this time will be lower than an agreed one is a necessary operating condition.

In the case where the tasks are assigned to just one server, like in an $M/M/c$ queue (or Erlang-C model [183]), the mean service time follows the expression

$$t_{c,k} = \frac{1}{\frac{m_k}{W_k} - L_k}. \tag{7.10}$$

This measurement could be used in practice with estimations of W_k and L_k, denoted as $\tilde{W}_k$ and $\tilde{L}_k$ respectively. However, in the proposed approach, as a certain task can be executed concurrently in many servers, the aforementioned measure is used to provide an upper bound and thus, in this case, it can only be lower or equal

$$t_{sv,k} \leq \frac{1}{\frac{m_k}{\tilde{W}_k} - \tilde{L}_k},$$

where $\tilde{W}_k$ is the estimated value of W_k at instant k for the next $\tilde{L}_k$ requests and $\tilde{L}_k$ is the estimation of L_k at instant k. It is considered that the QoS is satisfied at instant k if the mean service time is not larger than a specified value D. In other words, the QoS constraint is satisfied if

$$\frac{1}{\frac{m_k}{\tilde{W}_k} - \tilde{L}_k} \leq D,$$

which implies that

$$m_k \geq \tilde{W}_k \left(\frac{1}{D} + \tilde{L}_k \right).$$

The term on the right represents the minimum number of servers to fulfill the QoS constraint with a certain $\tilde{L}$, $\tilde{W}$ and D. This can be written as

$$m_{D,k} = \tilde{W}_k \left(\frac{1}{D} + \tilde{L}_k \right).$$

7.3 Management approach

In this chapter, an optimal management policy inspired on predictive control strategies is proposed. This controller decides the number of servers that should be on or off and the set point temperature of the CRAC unit. The optimization objective used to decide the optimal values of the inputs will be the minimization of the energy consumption and the control effort of the inputs (i.e. server switching and temperature reference changes), subject to thermal and QoS constraints.

First of all, it is assumed that the dynamics of the queue, task arrivals, etc. run much faster than the process of switching on a server (i.e. $k_{\text{on}} \gg 1$). Having such a large delay in the control actions, the control decisions have to be made in a superior time scale and separate them over time. This leads us to a scheme where the predictive controller is not executed at every instant k but it is executed at every instant $k_m t_s$ where k_m is the number of instants between the controller execution. Thus, the sample time of the controller is $k_m t_s$. In order to avoid switching off a server that is still booting (that is, never reached the "on" state and did nothing but consuming power), a sample time larger than the switching on time is chosen, i.e. $k_m \geq k_{\text{on}}$.

Based on the model, the expected evolution of the data center can be estimated and then, it is possible to associate a predicted cost to a sequence of candidate future control inputs. For clarity purposes, denote $\ell_j = k + jk_m$. This will work as the time scale of the MPC controller. In this chapter, the following predicted cost function to measure the expected performance of the data center will be used:

$$\begin{aligned} V(\ell_0, \mathbf{x}, \mathbf{u}, \mathbf{T_r}) = & \sum_{j=0}^{N_p} \sum_{i=0}^{M} \left(1 + \frac{1}{\text{CoP}(T_{c,\ell_j})}\right) p_{\ell_j,(i)}(x_{\ell_j,(i)}, u_{\ell_j,(i)}) \\ & + \kappa_u \sum_{j=0}^{N_c} |\Delta u_{\ell_j}| + \kappa_{T_r} \sum_{j=0}^{N_c} |\Delta T_{r,\ell_j}| \,, \end{aligned} \tag{7.11}$$

where N_p and N_c are the prediction and control horizons, $\mathbf{x}$ is the sequence of x_{ℓ_j} over the prediction horizon, $\mathbf{u}$ and $\mathbf{T_r}$ are the sequences of "on"-"off" control actions of all servers and temperature set points through the control horizon respectively, Δu_{ℓ_j} is the total number of commutations to either "on" or "off" at instant ℓ_j, $\Delta T_{r,\ell_j}$ is the increment in T_{r,ℓ_j}, κ_u is a term weighting Δu_{ℓ_j} and κ_{T_r} is a term weighting $\Delta T_{r,\ell_j}$. Also, the control actions further from the control horizon are considered to remain constant.

In order to derive the proposed controller, it is necessary to determine a prediction model such that for a given state at time ℓ_j, x_{ℓ_j} and for given control actions u_{ℓ_j} and T_{r,ℓ_j} (that will remain constant throughout the sampling time k_m), the state of the data center predicted at the next sampling time $\tilde{x}_{\ell_{j+1}}$ is calculated depending on the estimation of the number of tasks and their workload $\tilde{L}_{k_m,\ell_{j+1}}$ and $\tilde{W}_{k_m,\ell_{j+1}}$. This prediction model can be posed as:

$$\tilde{x}_{\ell_{j+1}} = h(x_{\ell_j},\, u_{\ell_j},\, T_{r,\ell_j}, \tilde{L}_{k_m,\ell_{j+1}},\, \tilde{W}_{k_m,\ell_{j+1}}) \,,$$

being $h(\cdot)$ the function that compute the following state given the previous one, the inputs and the realisations of L_{k_m} and W_{k_m}. Note that the function $h(\cdot)$ must compute all events happening thorough the interval k_m for every instant k in order to be able to return the state at the following sample time. In practice, this function is evaluated by means of an open-loop simulation of the complete queue model of the data center.

The optimal predicted number of servers and set point temperatures for the CRAC will be then computed as the solution of the optimization problem

$$\min_{m_{\ell_j}, T_{r,\ell_j}} \quad V(\ell_0, \mathbf{x}, \mathbf{u}, \mathbf{T_r}) \tag{7.12a}$$

$$\text{s.t.} \quad \tilde{x}_{\ell_{j+1}} = h(x_{\ell_j}, u_{\ell_j}, T_{r,\ell_j}, \tilde{L}_{k_m,\ell_{j+1}}, \tilde{W}_{k_m,\ell_{j+1}}) \tag{7.12b}$$

$$m_{\ell_j} \in [1,\, M] \quad \forall j \in [0, N_c] \tag{7.12c}$$

$$u_{\ell_j} = \text{switch}(m_{\ell_j}) \quad \forall j \in [0, N_c] \tag{7.12d}$$

$$15^{\circ}C \leq T_{r,\ell_j} \leq 25^{\circ}C \quad \forall j \in [0, N_c] \tag{7.12e}$$

$$T_{\ell_j,(i)} \leq 80^{\circ}\text{C} \quad \forall i \in M \quad \forall j \in [1, N_p] \tag{7.12f}$$

$$m_{\ell_j} \geq m_{D,\ell_j} \quad \forall j \in [1, N_p]\,, \tag{7.12g}$$

where switch$(\cdot)$ represents the policy to select which servers are to be switched on. Here, a simple scheme is proposed for such policy. That is, for a given number of servers, the first m_k will be switched on.

As it is customary in predictive controllers, the solution of (7.12) is applied in a receding horizon manner, meaning that only the first control actions and temperature set points are actually applied (i.e., $u_{\ell_0,(i)}$ and T_{r,ℓ_0}) while the remaining decision variables ($u_{\ell_1,(i)}, \ldots, u_{\ell_{N_c-1},(i)}$ and $T_{r,\ell_1}, \ldots, T_{r,\ell_{N_c-1}}$) computed at time k are discarded. The optimization of (7.12) is then repeated at each sampling time so that the decision variables to be applied are computed using the real state of the data center at that sampling time.

It should be noted that in this optimization problem some of the variables are integer (number of servers on) whereas other are real valued (the set-point temperatures). This, together with the complexity of the data center model motivates the use of specialized optimization algorithms to solve (7.12), such as a particle based optimization technique [180, 181] that will be exposed in the following section.

7.4 Particle based solvers for complex optimization problems

In these techniques of iterative nature, a set of possible candidate solutions (called particles) are evaluated at each iteration and used to generate a new candidate solution set that may be closer to the solution of the optimization problem. Here, the evaluation of each particle will be done by means of simulations that will be used to assess the performance of each candidate solution. The main ingredients of the proposed technique are:

- *Particles* are candidate solutions of the optimization problem. That is, a sequence of control actions over the control horizon. At higher problem

complexity (i.e., more servers and longer control horizons), more particles are needed in order to obtain a solution close to the optimal one [184].

- *Weights.* Particles have associated a weight that represents how good is a particle compared with the others. In this work, the weight is based on the performance cost of each particle computed by means of a computer simulation of the data center model. These simulations predict the evolution of the data center along the prediction horizon if the decision variables are those of the particles. Once the simulations for all particles are completed (this can be done in parallel), a scaling of the performance costs of the particles is made. Let $V_{\max}$ and $V_{\min}$ be the maximum and minimum costs attained on the set of particles under evaluation. If a given particle $z \triangleq (\mathbf{m}, \mathbf{T_r})$ has a performance cost $V_{(z)}$, then the weight of the particle z will be

$$\sigma_{(z)} = 1 - \frac{V_{(z)} - V_{\min}}{V_{\max} - V_{\min}}, \tag{7.13}$$

  meaning that particles with higher costs will have lower weights and vice versa. This is very important for the resampling phase.

- *Feasibility checking.* Those particles that do not satisfy the constraints in (7.12) will be assigned a zero weight so that they cannot be selected in the resampling stage.

- *Resampling* is the step where a new generation of particles is created based on the performance (weights) at the previous iteration. This is done by means of the Kitagawa resampling algorithm with stratified scheme [185]. The aim of this method is to form groups of particles with good performance in different areas of the feasible set of solutions. In this way, the risk of getting stuck at a local minimum is reduced because the algorithm considers all particles and not only the best one. Particles with higher weights are more likely to be resampled at next iteration (i.e., selected to be included in the next particle set).

- *Perturbation.* This process is inherently connected to the previous one. Once particles are resampled, it is necessary to "move" them along the local search space in order to discover new feasible solutions with potentially lower costs. In this work, the perturbation is made by adding a Gaussian white noise. Figure 7.2 shows the main ideas of the resampling and perturbation steps in a 2 degree of freedom minimization example. At first, there exists a set of random-generated particles. As the iterations continue, particles move towards better possible solutions according to the costs obtained previously. In our case, the optimization variables are integer, thus this process ends rounding to the nearest integer.

- *Reseeding.* The initial set of particles should be generated in a random manner. However, for the subsequent sample times, the reseeding can be

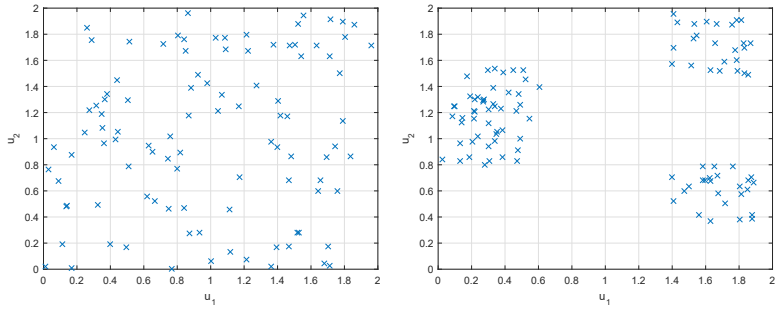


Figure 7.2: Optimization example with two real decision variables.

done by exploiting the receding horizon nature of the predictive controller. Then consider the solution of (7.12) for a given sampling time k

$$\mathbf{m}^*_{\ell_0} = \begin{bmatrix} m^*_{\ell_0\,|\,k} \\ m^*_{\ell_1\,|\,k} \\ \vdots \\ m^*_{\ell_{N_c-1}\,|\,k} \end{bmatrix}, \; \mathbf{T}^*_{r,\ell_0} = \begin{bmatrix} T^*_{r,\ell_0\,|\,k} \\ T^*_{r,\ell_1\,|\,k} \\ \vdots \\ T^*_{r,\ell_{N_c-1}\,|\,k} \end{bmatrix}.$$

Thus, a particle generated as the shifted version of the solution for ℓ_0

$$\mathbf{m}_{\ell_1} = \begin{bmatrix} m^*_{\ell_1\,|\,k} \\ \vdots \\ m^*_{\ell_{N_c-1}\,|\,k} \\ m^*_{\ell_{N_c-1}\,|\,k} \end{bmatrix}, \; \mathbf{T}_{r,\ell_1} = \begin{bmatrix} T^*_{r,\ell_1\,|\,k} \\ \vdots \\ T^*_{r,\ell_{N_c-1}\,|\,k} \\ T^*_{r,\ell_{N_c-1}\,|\,k} \end{bmatrix}.$$

would be a good candidate solution for (7.12) in ℓ_1 although it may not be optimal. Note that the last control action is repeated. Thus, the reseeding scheme is based on including in the initial particle set some shifted versions of the best particles (those with greater weight) of the final set obtained in the previous sampling time. These reseed particles will not be perturbed so that they can not be lost in the resampling process. Furthermore, in the resampling step, the reseed particles will lead to more particles close to them, and those may attain a higher weight. Note that the fact that a set of reseed particles is used instead of just one reduces the probability of getting stuck in a local minima.

The first particle algorithm is shown in Algorithm 4. In this case, it is assumed that tasks arrive uniformly every k_a time units (that is, the mean time of the exponential distribution (7.1)). Also, the workload of these incoming tasks is assumed to be the expected value of the corresponding exponential probability distribution, that is, $\frac{1}{\mu}$. From this, it is possible to obtain the values of $\tilde{L}_{k_m}$ and $\tilde{W}_{k_m}$ needed in line 3, which will be used to evaluate the constraints.

Algorithm 4: Particle based optimization (Pbo).

Data: χ (Number of particles), N_p , N_c (Prediction and Control Horizon respectively), $V(\cdot)$ (Cost Function), χ_t (Number of particles to reseed).

Result: Proposed control sequence $\mathbf{m}, \mathbf{T_r}$.

1 Reseed χ_t particles that showed best performance at previous time step;
2 Generate randomly new control actions for $\chi - \chi_t$ particles;
3 Estimate $\tilde{L}_{k_m}$ and $\tilde{W}_{k_m}$;
4 **repeat**
5 Predict the system evolution for each particle along the prediction horizon;
6 Verify the feasibility of each particle and set weight zero for the unfeasible ones;
7 Calculate cost and weight for each particle;
8 Resampling process considering all particles;
9 Perturbation of particles excluding the χ_t reseed particles;
10 **until** *a stop condition is fulfilled*;
11 Choose as result the particle with greatest weight;

The stop condition can be just a fixed number of iterations or a more elaborated scheme such as a small improving rate of the cost of the best particle at each iteration.

7.4.1 Scenario-based approach

It is clear that the previous proposed approach may have many constraint violations due to the fact that it is only taking into account the expected value of the considered distributions. That is, the true realizations of these random variables can take values very far away from the expectation, thus considering only the mean values of the probability distributions will not be a realistic setting. For this reason, another scheme for the estimation of L_{k_m} and W_{k_m} is proposed here at the expense of a heavier computation burden.

As the probability distribution of L_{k_m} and W_{k_m} are assumed to be known, it is possible to draw random sequences of those variables. These will be called scenarios. These scenarios represent hypothetical realizations of the variables in the future and will be considered in the computation of the optimal number of servers and temperature set points. Algorithm 5 shows how to consider scenarios.

The main change from Algorithm 4 is the addition of a "for" loop over all the scenarios. As the first step in each iteration of this "for" loop, each scenario is drawn according to the probability distributions of L_{k_m} and W_{k_m}. Once this has been done, the evolution of the system is predicted like in algorithm 4. The same

goes for the feasibility verification of the particles.

Algorithm 5: Scenario PbO (S-Pbo).

Data: χ (Number of particles), N_p , N_c (Prediction and Control Horizon respectively), $V(\cdot)$ (Cost Function), χ_t (Number of particles to reseed), S (Number of scenarios).

Result: Proposed control sequence $\mathbf{m}, \mathbf{T_r}$.

```
Reseed χ_t particles that showed best performance at previous time step;
Generate randomly new control actions for χ − χ_t particles;
Set the weights to an initial value 1/χ;
repeat
    for every scenario do
        Extract a sequence of arrivals and workloads from the distributions;
        Estimate L̃_{k_m} and W̃_{k_m};
        Predict the system evolution for each particle along the prediction
         horizon;
        Verify the feasibility of each particle and set weight zero for the
         unfeasible ones;
        Calculate cost and update weight for each particle;
    end for
    Scale particle weights;
    Resampling process considering all particles;
    Perturbation of particles excluding the χ_t reseed particles;
    Reset the values of the weights to 1/χ;
until a stop condition is fulfilled;
Choose as result the particle with greatest weight;
```

There are also several changes about the computation and meaning of costs and weights. In algorithm 4, the weight was assigned directly once the cost was calculated applying equation (7.13). This could be done because there was only one scenario (the expected values of the random variables). Now, it is necessary that weights show the performance of each particle for all scenarios. For that reason, the weight of each particle is given an initial value of $\frac{1}{\chi}$ and it is updated at each iteration of the "for" loop with the information obtained from the scenario considered at that iteration. The particle with the best cost will maintain its weight meanwhile the rest will decrease theirs according to how good their performance were. This is done by means of the following equation:

$$\sigma_{(z)} \leftarrow \sigma_{(z)} \left(1 - \frac{V_{(z)} - V_{\min}}{V_{\max} - V_{\min}}\right) . \tag{7.14}$$

The weights are now updated as the product of the old weight and the new one computed from the cost in a specific scenario. At the end of the "for" loop, it is required by the Kitagawa resampling that the sum of the weights must be equal

to one. This is required due to the fact that the weights represent a discrete probability density function. For that reason, the weights are scaled in line 12. This is done using the following equation:

$$\sigma_{(z)} \leftarrow \frac{\sigma_{(z)}}{\sum_{z=1}^{\chi} \sigma_{(z)}} .$$

Also, it is important to reset the values at line 15 due to the new weight updating from equation (7.14).

Taking into account different realisations of the random variables will turn out to be a more conservative solution of the initially proposed algorithm because of step 9. It means that for a certain sequence of arrivals and workloads, particles which do not fulfill the conditions on maximum temperature and QoS will not be resampled at next iteration. In algorithm 4, this condition was checked only once for the sequence obtained from the mean values of the distributions. However, in algorithm 5, it is necessary to fulfill this condition in every scenario (which is more restrictive).

7.4.2 Parallel implementation

In order to lighten the computational burden of the proposed algorithms, they are implemented in a parallel manner when it is possible. For this purpose, a Graphics Processor Unit (GPU) is used. The use of GPUs to accelerate general purpose computations has been growing since the initial release of CUDA in 2007. CUDA stands for *Compute Unified Device Architecture* and it is the main programming platform for general purpose programming in GPUs [186]. There are also other alternatives like OpenCL, but CUDA is still more popular due to its better performance [187].

The main advantage of the GPU computing is the decrease of the computation time [188] provided that the tasks considered fit in the CUDA programming model which is based on a Single Instruction Multiple Thread (SIMT) scheme [189].

Because of its nature, the proposed particle algorithms are highly parallelizable in some of their steps such as the prediction of the system evolution, feasibility verification, cost calculation and weight updating. Others can not be completely parallelizable (i.e. the resampling process) due to the nature of the Kitagawa resampling [185]. The reason of this is that the computation of the cumulative probability density function requires that a certain thread z knows the value of the thread $z-1$ to do its job. Thus, this resampling must be done in a pure sequential manner.

For simplicity reasons, the simulations presented in this section have been coded in CUDA C kernels called from Matlab scripts. Although not the most efficient way of using GPU computing with CUDA, it allows an easy integration with Matlab CPU code.

7.5 Bounds on the constraint violation rate

Once reached to this point, it is easy to see that the algorithms proposed in this chapter have some design parameters that have to be chosen, i.e. the control horizon (N_c) and the number of scenarios (S). Depending on these parameters, a different controller will be obtained. In this section, the results of chapter 6 are used to compute the constraint violation level for a given controller in order to be able to compare their performance.

For the sake of simplicity, only the QoS constraint will be considered in the following although in the numerical results both constraints are considered.

Using theorem 6.2 or 6.3, the bounding scheme will be carried out using the following steps *for each controller* i to be validated:

1. Determine a number of closed loop simulations (H). Note that higher H implies a higher number of completed tasks (N) and thus lower $\Delta\rho$. In practice, H should be chosen as the highest affordable value.
2. Run H closed loop simulations with different values of the parameters in the probability distributions in the data center model to reflect different operating conditions.
3. Denote N_H as the number of tasks executed during the H simulations and s_H as the number of QoS violations in the N_H. Then compute ρ as

$$\rho = \frac{s_H}{N_H}.$$

4. Once done that, $\Delta\rho$ can be calculated for each controller with equation (6.1) or (6.3) (depending on the chosen bound) for the specified confidence $1 - \delta$.
5. Choose the best controller according to the desired criteria. For example, minimum constraint violation performance, trade-off between constraint satisfaction level and operating costs, etc.

7.6 Numerical results

The proposed management schemes and the probabilistic validation method will be illustrated by means of examples. The improvements from the parallel implementation will be also shown.

Four controllers with different parameters will be compared by means of simulations. These simulations have a fixed time step of 2500 time units. A number of $H = 100$ simulations were enough to have a large number of tasks to obtain the desired bounds. In each one of these simulations, to reflect different operating conditions, the variables L and W were given different expected values taking

into account the maximum workload and number of tasks that a data center of $M = 25$ servers can handle with respect to equation (7.4).

On the other hand, the integration step is $t_s = 1s$, the sampling time of the controller $k_m = 100$ time units (i.e. $100s$) and the prediction horizon is set to $N_p = 10$ sampling times (i.e. 1000 time units). Also, the maximum temperature and QoS are set to $80°C$ and 50 time units respectively. On the other hand, the time constant of the CRAC unit is $\tau = 180$ time units. The values of the power consumption are $a_1 = 180\,W$ and $a_2 = 120\,W$. Other thermal parameters like the server thermal capacity and the product of the air heat capacity times the air flow are $K_t = 160\,\frac{J}{K}$ and $c_p\,q_a = 5\,\frac{W}{K}$. The weighting factors in (19) are $\kappa_u = 200$ and $\kappa_{T_r} = 100$. Finally, table 7.1 shows the parameters of the four controllers considered and the line pattern for every controller in the following figures.

C_1	S-Pbo, $N_c = 1$, $S = 25$	Solid line	Orange
C_2	S-Pbo, $N_c = 5$, $S = 25$	Dotted line	Blue
C_3	Pbo, $N_c = 1$	Dashed-dotted line	Purple
C_4	Pbo, $N_c = 5$	Dashed line	Yellow

Table 7.1: Controllers considered and line patterns. Pbo refers to algorithm 4 whereas S-Pbo refers to algorithm. 5

Figure 7.3 shows the mean queue length, number of "on" servers and number of idle servers. It can be seen how the controller without scenarios and longest control horizon N_c (C_4) has the largest mean queue length in the experiments. This implies longer waiting times and thus the elapsed time to be expected for a certain task with this controller will be higher. Taking into account multiple scenarios, as in C_2, results in a lower queue length. This comes from the fact that considering multiple scenarios yields a more robust and conservative management. The other controllers (C_1 and C_3) shows the effect of working with minimal control horizons. In that case, the controller is forced to satisfy the constraints along a large prediction horizon with few degrees of freedom, resulting in a more conservative management. In the case of C_1, it keeps the queue length very small at the cost of a greater power consumption. On the other hand, C_3 for C_4 have very similar performance. That is, without taking into account scenarios, the operation of the controllers is practically not affected by the control horizon. A more conservative management also implies a higher number of "on" servers, as shown in the middle subplot, and also more idle servers (bottom subplot) when the real operating conditions are less demanding than the predicted ones. Having more servers "on" but idle makes sense in order to be able to deal with the unknown upcoming tasks, due to the delay time k_{on} needed to get the servers active from the "off" condition.

Figure 7.4 shows the mean values of the terms of the cost function (7.11) for each

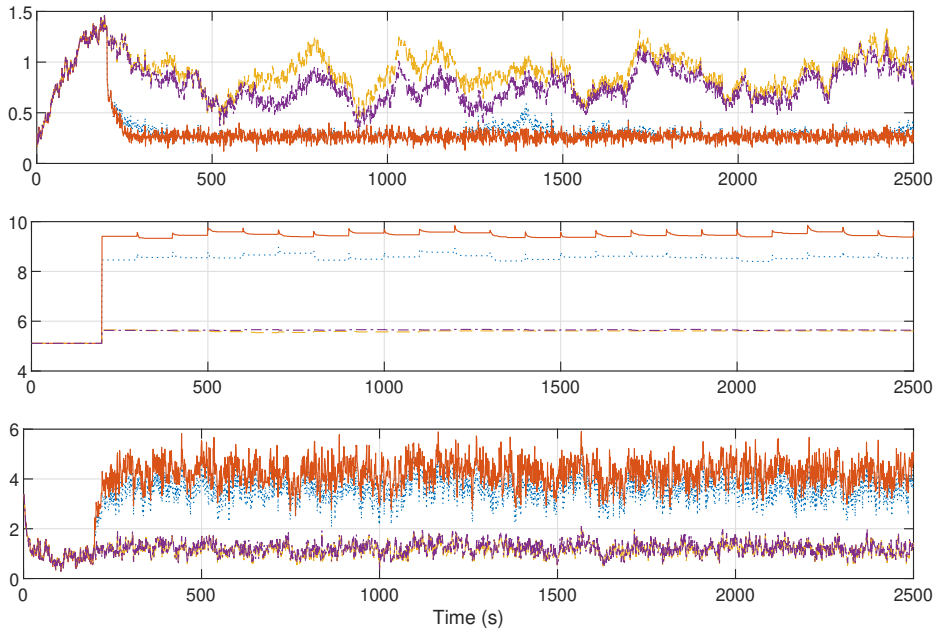


Figure 7.3: From top to bottom, Mean Queue Length, Mean Number of On Servers and Mean Number of Idle Servers

controller. The top subplot shows the power consumption term in Watts (W). As it can be guessed, the controllers with larger number of active servers will have greater power consumption, due to the consumption of the servers themselves but also for the greater consumption of the CRAC unit (which following (7.9) is assumed proportional to that of the servers).

The lower subplots are referred to the control efforts of the control actions. Here, it can be seen that the controllers with $N_c = 5$ tend to change more their control actions which leads to greater control efforts, but also facilitates a more tuned application of the control action that results in lower overall costs as seen in Figure 7.5. In this figure, the instantaneous total cost for each controller is shown. As expected, the more conservative controllers are the most expensive ones in terms of performance cost.

7.6.1 QoS violation rate

Once all simulations are carried out, the bounds can be computed for each one of the four controllers. Table 7.2 shows the results for the QoS constraint. The final result is shown in the $\rho + \Delta\rho$ columns which are the upper bounds (with confidence $1 - \delta$) of the probability of not meeting the agreed service time D for the theorems 2.2 and 2.3 respectively. Lower numbers imply better probabilistic

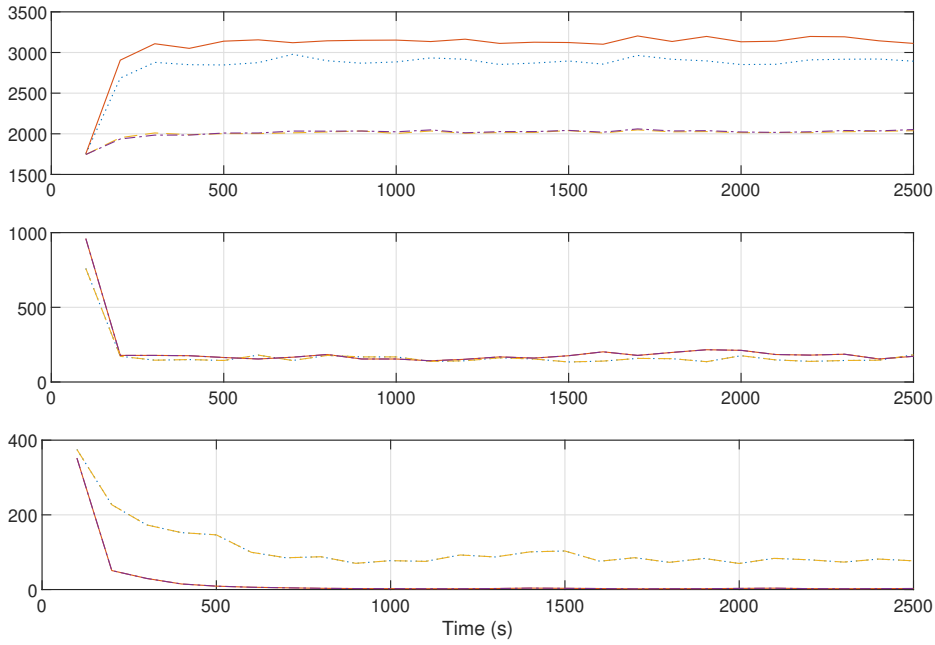


Figure 7.4: From top to bottom: Mean Power Consumption, Mean Control Effort for booting servers and Mean Control Effort for changing the CRAC temperature reference

guarantees, which as expected corresponds to the more conservative controllers (C_1 and C_2).

		ρ	$\rho + \Delta\rho(2.2)$	$\rho + \Delta\rho(2.3)$	N
C_1	S-Pbo, $N_c = 1$	0.0032	0.0050	0.0045	65960
C_2	S-Pbo, $N_c = 5$	0.0040	0.0060	0.0054	65926
C_3	Pbo, $N_c = 1$	0.0278	0.0328	0.0314	65708
C_4	Pbo, $N_c = 5$	0.0337	0.0392	0.0376	65678

Table 7.2: Violation rate of the proposed controllers for the QoS constraints with $\delta = 10^{-6}$

7.6.2 Thermal constraint violation rate

Bounds have also been obtained for the temperature constraint, with table 7.3 summarizing the results. In this case, N stands as the number of total time instants for all the simulations H (which were all of the same length), and the upper bound is on the probability of reaching a temperature higher than 80°C. Again the more conservative a controller has the better probabilistic guarantees, but at the expense of a higher cost.

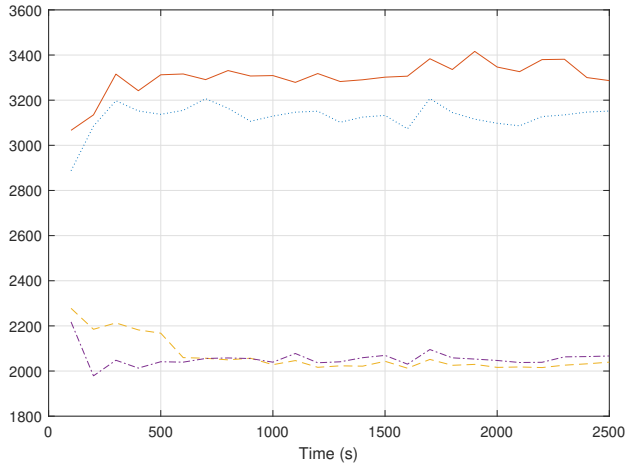


Figure 7.5: Instantaneous total cost

		ρ	$\rho + \Delta\rho(2.2)$	$\rho + \Delta\rho(2.3)$	N
C_1	S-Pbo, $N_c = 1$	0.0000	0.0055	0.0058	2500
C_2	S-Pbo, $N_c = 5$	0.0000	0.0055	0.0058	2500
C_3	Pbo, $N_c = 1$	0.0001	0.0071	0.0067	2500
C_4	Pbo, $N_c = 5$	0.0000	0.0055	0.0058	2500

Table 7.3: Violation rate of the proposed controllers for the maximum temperature constraint with $\delta = 10^{-6}$

7.6.3 Parallel computation improvement

In order to prove the speed-ups obtained with a parallel implementation, the algorithms have been implemented in the CPU and also in a CUDA capable GPU. The CPU version is coded completely in Matlab while the GPU version uses Matlab code for the serial operations and C code for the CUDA kernels. Table 7.4 shows the execution time of the controllers for an increasing number of particles. Cells with a hyphen mean that the time exceeded the sampling time. Finally, figure 7.6 shows the relative speed-up for different numbers of particles.

	S-Pbo GPU	S-Pbo CPU	Pbo GPU	Pbo CPU
10	17.0931	23.4282	0.6974	1.0001
100	17.8046	-	0.7230	9.4192
1000	30.2881	-	1.2267	93.6546
10000	91.4658	-	3.7208	-

Table 7.4: Mean computation time of the controllers in seconds.

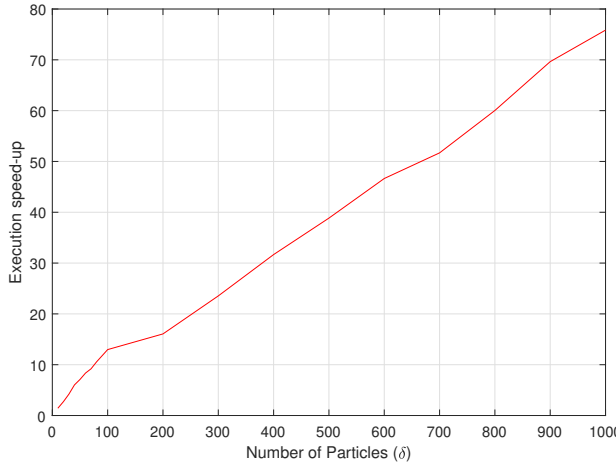


Figure 7.6: Speed-up obtained with the GPU computing for the controller execution.

As it can be expected, at a higher number of particles, the execution time within the CPU grows strongly whereas the computation time within the GPU yields almost constant in comparison, which leads to the increasing slope shown in figure 7.6. It also should be noted that once reached a number of particles greater than 1000, the execution times of the CPU become unmanageable and the simulations are extremely costly.

7.6.4 Computation time analysis

Also, an analysis of the computation time of the algorithm with respect to different number of servers has been done in order to study its behavior for the problem of large data centers. The results are shown in figure 7.7. A polynomial of order $n = 2$ is also added to the figures, confirming a quadratic complexity.

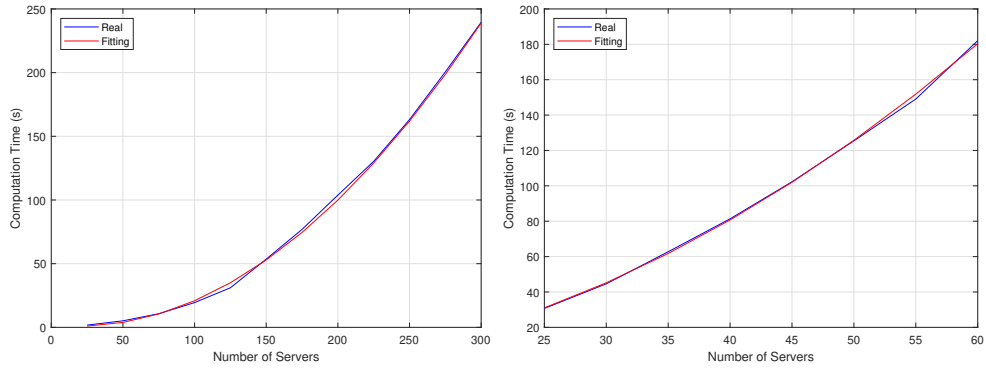


Figure 7.7: Computation times of the algorithms for an increasing number of servers. Left: Algorithm without scenarios. Right: Algorithm with scenarios

Thus, the algorithm is feasible from a computational point of view and it can be

used with larger data centers. Nevertheless, for higher numbers of servers, besides adding more computational power, one could consider multiple controllers each one for each cold aisle or consider that the control actions handle clusters of servers instead of individual ones.

7.7 Conclusions

This chapter presented a complex model of a data center and proposed a methodology to deal with the problem of minimizing the power consumption of the facilities while maintaining the quality of service at acceptable levels. The proposed MPC-inspired controller simulates the whole data center model in order to compute the optimal inputs to be applied to the system. For this reason, a particle-based algorithm was used to solve the optimization problem at each time instant k. The bounds proposed in chapter 6 were used to tune the hyperparameters of the controllers.

Chapter 8

Conclusions and future work

This chapter briefly summarizes the contributions of this dissertation and presents some future lines of work in each field.

8.1 Contributions

As it was stated in the introduction, the contributions of this thesis are related to three different fields: system identification, probabilistic forecasting and model predictive control. In what follows, the contributions of each chapter are stated:

- Chapter 2 presented the notion of dissimilarity and the dissimilarity function to be used throughout this dissertation. It was shown that the proposed dissimilarity function has many advantageous properties. Also, by means of some forecasting examples, it was shown that the dissimilarity function can be used as a predictor, obtaining better results than those obtained with a neural network in the field of stock forecasting.

- In chapter 3, the case of uncertainty predictions was tackled. Instead of providing just the expected value of the forecasting, a prediction region is obtained. First, a method oriented to obtain interval predictions of a univariate system or time series was proposed. This method was computationally expensive because it required the integration of the empirical conditioned probability density function for every time instant. In any case, the numerical examples showed that the performance was better in comparison to set-membership and quantile regression methods.

  Then, a method to bypass the numerical integration of the empirical conditioned probability density function was proposed. Besides this great advantage, which reduces considerably the execution time of the algorithm, multivariate prediction regions could be obtained. Also, it was shown that ellipsoidal approximations of such regions can be easily obtained. Finally,

by means of a numerical example, it was shown that the proposed approach and the ellipsoidal approximations are smaller than those regions obtained with quantile regression, gaussian processes or inverse regression methods while fulfilling the probabilistic specifications.

- Chapter 4 presented the state-space kriging method for autonomous systems. By using this technique, a model of the system can be obtained using only historical data of the process. Two different variations are proposed, one obtained by weighting the locality within the dissimilarity function and another one where the nonlinearity is modeled by means of kernel functions. Also, it was shown that the kalman filter can be used to improve the quality of the predictions. Finally, two numerical forecasting examples were presented, where it was shown that the proposed approaches obtain better results than other machine learning techniques.

- On the other hand, chapter 5 presented the state-space kriging method for non-autonomous systems. Thanks to the modifications made to the constraints in the dissimilarity function, it is possible to include the effect of an external input. As the models obtained in this chapter are suitable for MPC, a MPC formulation using the K-SSK model was proposed. Finally, a simulation example and a real experiment were conducted in order to show the good performance of the controller.

- In chapter 6, by means of the Chernoff's bound [128, 159] and the results by Alamo et al. in [61], two different sharp bounds of the constraint violation rate of a certain controller were obtained. No assumptions on the probability distributions of the disturbances were made in order to prove the obtained results. It is only required to obtain i.i.d. samples, which is a standard assumption in the field. Finally, a simple MPC example was proposed to show how the proposed bounds should be computed.

- Chapter 7 showcased a more complicated example where the proposed bounds can be used. The chosen system was a data center. This kind of facilities have an extremely large electrical consumption due to both IT equipment and infrastructure facilities. To tackle this problem, an energy-efficient management approach was proposed. Here, the bounds proposed in chapter 6 were used to tune the parameters of the controller.

8.2 Future work

The future lines of work for each one of the parts of this dissertation can be found in the following:

- As the obtained interval predictions and prediction regions of part I performed good enough in a forecasting example, it would be interesting to apply these uncertain predictions in a control scheme, i.e. developing new

stochastic MPC controllers whose prediction model is given by the predictors obtained in chapter 3.

The most approachable way would be to synthesize a controller for a unidimensional system where the proposed interval predictor is used to obtain uncertain predictions of the system. These predictions would be used within the optimization problem to ensure the constraint satisfaction in a chance-constrained manner.

- The proposed MPC formulation in part II presented nominal stability and robust asymptotic stability, but lacked robust constraint satisfaction. For this reason, it would be interesting to improve the controller formulation to tackle this case. Also, the integration of a real time optimizer and the consideration of economic objectives in the controller is another pending task.

  On the other hand, the proposed SSK models hardly accept new data because this would mean that the state of the system is constantly increasing. It is necessary to find a good method that would allow the user to introduce new data to the SSK model without increasing the complexity.

- When some of the proposed bounds in part III are used, it is possible to lose the recursive feasibility of the MPC controller unless the controller formulation is relaxed somehow. For that reason, a solution to this phenomenon should be proposed after a careful analysis of its origin.

  On the other hand, the i.i.d. assumption, although usual in the literature, is hard to be fulfilled because samples drawn from a dynamical system are inherently correlated. It would be interesting to develop methods which allow us to obtain samples that are as i.i.d. as possible.

Bibliography

[1] L. Ljung, *System identification: Theory for the user*. Pearson Education, 1998.

[2] K. Ogata, *Modern control engineering*, vol. 5. Prentice hall Upper Saddle River, NJ, 2010.

[3] P. Young, "Parameter estimation for continuous-time models—a survey," *Automatica*, vol. 17, no. 1, pp. 23–39, 1981.

[4] N. R. Kristensen, H. Madsen, and S. B. Jørgensen, "Parameter estimation in stochastic grey-box models," *Automatica*, vol. 40, no. 2, pp. 225–237, 2004.

[5] F. Ding, G. Liu, and X. P. Liu, "Parameter estimation with scarce measurements," *Automatica*, vol. 47, no. 8, pp. 1646–1655, 2011.

[6] P. Geladi and B. R. Kowalski, "Partial least-squares regression: a tutorial," *Analytica chimica acta*, vol. 185, pp. 1–17, 1986.

[7] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of fluid mechanics*, vol. 656, pp. 5–28, 2010.

[8] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

[9] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic mode decomposition with control," *SIAM Journal on Applied Dynamical Systems*, vol. 15, no. 1, pp. 142–161, 2016.

[10] J. Schoukens and L. Ljung, "Nonlinear system identification: A user-oriented road map," *IEEE Control Systems Magazine*, vol. 39, no. 6, pp. 28–99, 2019.

[11] O. Nelles, "Nonlinear dynamic system identification," in *Nonlinear System Identification*, pp. 547–577, Springer, 2001.

[12] K. Liu, "Identification of linear time-varying systems," *Journal of Sound and Vibration*, vol. 206, no. 4, pp. 487–505, 1997.

[13] P. L. Dos Santos, T. P. A. Perdicoulis, C. Novara, J. A. Ramos, and D. E. Rivera, *Linear parameter-varying system identification: New developments and trends*, vol. 14. World Scientific, 2011.

[14] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE transactions on systems, man, and cybernetics*, no. 1, pp. 116–132, 1985.

[15] K. Zeng, N.-Y. Zhang, and W.-L. Xu, "A comparative study on sufficient conditions for Takagi-Sugeno fuzzy systems as universal approximators," *IEEE Transactions on fuzzy systems*, vol. 8, no. 6, pp. 773–780, 2000.

[16] E.-W. Bai, "A blind approach to the Hammerstein-Wiener model identification," *Automatica*, vol. 38, no. 6, pp. 967–979, 2002.

[17] A. Wills, T. B. Schön, L. Ljung, and B. Ninness, "Identification of Hammerstein-Wiener models," *Automatica*, vol. 49, no. 1, pp. 70–81, 2013.

[18] J. K. Gruber, D. R. Ramirez, T. Alamo, and C. Bordons, "Nonlinear min-max model predictive control based on volterra models. Application to a pilot plant," in *2009 European Control Conference (ECC)*, pp. 1112–1117, IEEE, 2009.

[19] J. Gruber, D. R. Ramirez, D. Limon, and T. Alamo, "Computationally efficient nonlinear min–max model predictive control based on Volterra series models—Application to a pilot plant," *Journal of Process Control*, vol. 23, no. 4, pp. 543–560, 2013.

[20] S. Chen and S. A. Billings, "Representations of non-linear systems: the NARMAX model," *International journal of control*, vol. 49, no. 3, pp. 1013–1032, 1989.

[21] S. A. Billings, *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[22] Q. Zhang and A. Benveniste, "Wavelet networks," *IEEE transactions on Neural Networks*, vol. 3, no. 6, pp. 889–898, 1992.

[23] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. MIT Press, 2006.

[24] S. Eleftheriadis, T. Nicholson, M. Deisenroth, and J. Hensman, "Identification of Gaussian process state space models," *Advances in neural information processing systems*, vol. 30, 2017.

[25] J. Umlauft, A. Lederer, and S. Hirche, "Learning stable Gaussian process state space models," in *2017 American Control Conference (ACC)*, pp. 1499–1504, IEEE, 2017.

[26] J.-P. Calliess, "Lipschitz optimisation for Lipschitz interpolation," in *2017 American Control Conference (ACC)*, pp. 3141–3146, IEEE, 2017.

[27] J. M. Manzano, D. Limon, D. Muñoz de la Peña, and J.-P. Calliess, "Robust learning-based MPC for nonlinear constrained systems," *Automatica*, vol. 117, p. 108948, 2020.

[28] J. M. Zamarreño and P. Vega, "State space neural network. Properties and application," *Neural networks*, vol. 11, no. 6, pp. 1099–1112, 1998.

[29] I. Rivals and L. Personnaz, "Black-box modeling with state-space neural networks," in *Neural Adaptive Control Technology*, pp. 237–264, World Scientific, 1996.

[30] B. Lim and S. Zohren, "Time-series forecasting with deep learning: a survey," *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2194, p. 20200209, 2021.

[31] S. H. Rudy, J. N. Kutz, and S. L. Brunton, "Deep learning of dynamics and signal-noise decomposition with time-stepping constraints," *Journal of Computational Physics*, vol. 396, pp. 483–506, 2019.

[32] S. Lawrence and C. L. Giles, "Overfitting and neural networks: conjugate gradient and backpropagation," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks.*, vol. 1, pp. 114–119, IEEE, 2000.

[33] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach," *Physical review letters*, vol. 120, no. 2, p. 024102, 2018.

[34] K. Nakajima and I. Fischer, *Reservoir computing*. Springer, 2021.

[35] D. Li, M. Han, and J. Wang, "Chaotic time series prediction based on a novel robust echo state network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 5, pp. 787–799, 2012.

[36] D. Goswami, A. Wolek, and D. A. Paley, "Data-driven estimation using an echo-state neural network equipped with an ensemble Kalman filter," in *2021 American Control Conference (ACC)*, pp. 2549–2554, IEEE, 2021.

[37] R. Babuška and H. Verbruggen, "Neuro-fuzzy methods for nonlinear system identification," *Annual reviews in control*, vol. 27, no. 1, pp. 73–85, 2003.

[38] J.-S. Jang and C.-T. Sun, "Neuro-fuzzy modeling and control," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 378–406, 1995.

[39] A. Mauroy, Y. Susuki, and I. Mezić, *Koopman operator in systems and control*. Springer, 2020.

[40] P. Bevanda, S. Sosnowski, and S. Hirche, "Koopman operator dynamical models: Learning, analysis and control," *Annual Reviews in Control*, vol. 52, pp. 197–212, 2021.

[41] G. Mogk, T. Mrziglod, and A. Schuppert, "Application of hybrid models in chemical industry," in *Computer aided chemical engineering*, vol. 10, pp. 931–936, Elsevier, 2002.

[42] J. Sansana, M. N. Joswiak, I. Castillo, Z. Wang, R. Rendall, L. H. Chiang, and M. S. Reis, "Recent trends on hybrid modeling for industry 4.0," *Computers & Chemical Engineering*, vol. 151, p. 107365, 2021.

[43] T. Alamo, J. M. Bravo, and E. F. Camacho, "Guaranteed state estimation by zonotopes," *Automatica*, vol. 41, no. 6, pp. 1035–1043, 2005.

[44] T. Alamo, J. M. Bravo, M. Redondo, and E. Camacho, "A set-membership state estimation algorithm based on DC programming," *Automatica*, vol. 44-1, pp. 216–224, 2008.

[45] R. Thabet, T. Raïssi, C. Combastel, D. Efimov, and A. Zolghadri, "An effective method to interval observer design for time-varying systems," *Automatica*, vol. 50, no. 10, pp. 2677 – 2684, 2014.

[46] S. Chebotarev, D. Efimov, T. Raïssi, and A. Zolghadri, "Interval observers for continuous-time LPV systems with l_1/l_2 performance," *Automatica*, vol. 58, pp. 82 – 89, 2015.

[47] S. Raka and C. Combastel, "Fault detection based on robust adaptive thresholds: A dynamic interval approach," *Annual Reviews in Control*, vol. 37, no. 1, pp. 119 – 128, 2013.

[48] F. Xu, V. Puig, C. Ocampo-Martinez, F. Stoican, and S. Olaru, "Actuator-fault detection and isolation based on set-theoretic approaches," *Journal of Process Control*, vol. 24, no. 6, pp. 947 – 956, 2014.

[49] M. Milanese and C. Novara, "Set membership identification of nonlinear systems," *Automatica*, vol. 40, no. 6, pp. 957–975, 2004.

[50] M. Milanese and C. Novara, "Unified set membership theory for identification, prediction and filtering of nonlinear systems," *Automatica*, vol. 47, no. 10, pp. 2141–2151, 2011.

[51] R. Fernandez-Canti, J. Blesa, S. Tornil-Sin, and V. Puig, "Fault detection and isolation for a wind turbine benchmark using a mixed bayesian/set-membership approach," *Annual Reviews in Control*, vol. 40, pp. 59 – 69, 2015.

[52] M. Milanese, J. Norton, H. Piet-Lahanier, and E. Walter, *Bounding Approaches to System Identification*. Plenum Press, New York, 1996.

[53] M. Milanese and C. Novara, "Set membership prediction of nonlinear time series," *IEEE Transactions on Automatic Control*, vol. 50, no. 11, pp. 1655–1669, 2005.

[54] E. Bai, Y. Ye, and R. Tempo, "Bounded error parameter estimation: A sequential analytic center approach," *IEEE Transactions on Automatic control*, vol. 44, no. 6, pp. 1107–1117, 1999.

[55] L. Jaulin, "Interval constraint propagation with application to bounded-error estimation," *Automatica*, vol. 36, pp. 1547–1552, 2000.

[56] J. M. Bravo, T. Alamo, M. Vasallo, and M. Gegundez, "A general framework for predictors based on bounding techniques and local approximation," *IEEE Transactions on Automatic Control*, vol. 62, no. 7, pp. 3430–3435, 2017.

[57] J. Roll, A. Nazin, and L. Ljung, "Nonlinear system identification via direct weight optimization," *Automatica*, vol. 41, no. 3, pp. 475–490, 2005.

[58] J. M. Bravo, T. Alamo, M. Gegundez, and D. Marin, "Combined stochastic and deterministic interval predictor for time-varying systems," in *23th Mediterranean Conference on Control and Automation (MED)*, pp. 833–839, IEEE, 2015.

[59] C. Combastel, "Zonotopes and kalman observers: Gain optimality under distinct uncertainty paradigms and robust convergence," *Automatica*, vol. 55, pp. 265 – 273, 2015.

[60] B. Efron and R. Tibshirani, "Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy," *Statistical science*, pp. 54–75, 1986.

[61] T. Alamo, R. Tempo, A. Luque, and D. R. Ramirez, "Randomized methods for design of uncertain systems: Sample complexity and sequential algorithms," *Automatica*, vol. 52, pp. 160–172, 2015.

[62] T. Alamo, J. Manzano, and E. Camacho, "Robust design through probabilistic maximization," in *Uncertainty in Complex Networked Systems*, pp. 247–274, Springer, 2018.

[63] V. Mirasierra, M. Mammarella, F. Dabbene, and T. Alamo, "Prediction error quantification through probabilistic scaling," *IEEE Control Systems Letters*, vol. 6, pp. 1118–1123, 2022.

[64] K. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[65] R. Koenker and G. Bassett, "Regression quantiles," *Econometrica: journal of the Econometric Society*, pp. 33–50, 1978.

[66] C. Davino, M. Furno, and D. Vistocco, *Quantile Regresssion. Theory and Applications*. Wiley, 2014.

[67] G. Bassett and H. Chen, "Portfolio style: Return-based attribution using quantile regression," in *Economic Applications of Quantile Regression*, pp. 293–305, Springer, 2002.

[68] A. Papoulis and S. Pillai, *Probability, Random Variables and Stochastic Processes*. Mc Graw Hill, 2002.

[69] J. Navarro, "A very simple proof of the multivariate Chebyshev's inequality," *Communications in Statistics-Theory and Methods*, vol. 45, no. 12, pp. 3458–3463, 2016.

[70] B. Stellato, B. Van Parys, and P. J. Goulart, "Multivariate Chebyshev inequality with estimated mean and variance," *The American Statistician*, vol. 71, no. 2, pp. 123–127, 2017.

[71] S. Portnoy and R. Koenker, "The Gaussian hare and the Laplacian tortoise: computability of squared-error versus absolute-error estimators," *Statistical Science*, vol. 12, no. 4, pp. 279–300, 1997.

[72] R. Tempo, E. Bai, and F. Dabbene, "Probabilistic robustness analysis: explicit bounds for the minimum number of samples," *Systems & Control Letters*, vol. 30, pp. 237–242, 1997.

[73] H. Lütkepohl, *Introduction to multiple time series analysis*. Berlin, Springer, 1991.

[74] M. Wolf and D. Wunderli, "Bootstrap joint prediction regions," *Journal of Time Series Analysis*, vol. 36, no. 3, pp. 352–376, 2015.

[75] J. H. Kim, "Bias-corrected bootstrap prediction regions for vector autoregression," *Journal of Forecasting*, vol. 23, no. 2, pp. 141–154, 2004.

[76] J. H. Kim, "Asymptotic and bootstrap prediction regions for vector autoregression," *International Journal of Forecasting*, vol. 15, no. 4, pp. 393–403, 1999.

[77] P. Vidoni, "Improved multivariate prediction regions for Markov process models," *Statistical Methods & Applications*, vol. 26, no. 1, pp. 1–18, 2017.

[78] G. Fonseca, F. Giummole, and P. Vidoni, "A note about calibrated prediction regions and distributions," *Journal of Statistical Planning and Inference*, vol. 142, no. 9, pp. 2726–2734, 2012.

[79] C. Lagazio and P. Vidoni, "Calibrated prediction regions for Gaussian random fields," *Environmetrics*, vol. 29, no. 3, p. e2495, 2018.

[80] D. J. Olive, "Applications of hyperellipsoidal prediction regions," *Statistical Papers*, vol. 59, no. 3, pp. 913–931, 2018.

[81] W. Lin, Y. Zhuang, S. Zhang, and E. Martin, "On estimation of multivariate prediction regions in partial least squares regression," *Journal of Chemometrics*, vol. 27, no. 9, pp. 243–250, 2013.

[82] E. Devijver and E. Perthame, "Prediction regions through Inverse Regression.," *Journal of Machine Learning Research*, vol. 21, pp. 1–24, 2020.

[83] F. Golestaneh, P. Pinson, R. Azizipanah-Abarghooee, and H. B. Gooi, "Ellipsoidal prediction regions for multivariate uncertainty characterization," *IEEE Transactions on Power Systems*, vol. 33, no. 4, pp. 4519–4530, 2018.

[84] G. Shafer and V. Vovk, "A tutorial on conformal prediction.," *Journal of Machine Learning Research*, vol. 9, no. 3, 2008.

[85] K. Karydis, I. Poulakakis, J. Sun, and H. G. Tanner, "Probabilistically valid stochastic extensions of deterministic models for systems with uncertainty," *The International Journal of Robotics Research*, vol. 34, no. 10, pp. 1278–1295, 2015.

[86] E. F. Camacho and C. Bordons Alba, *Model predictive control*. Springer London, second ed., 2007.

[87] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, WI, second ed., 2017.

[88] J. Richalet, A. Rault, J. Testud, and J. Papon, "Model predictive heuristic control: Applications to industrial processes," *Automatica*, vol. 14, no. 5, pp. 413–428, 1978.

[89] C. R. Cutler, *Dynamic matrix control: an optimal multivariable control algorithm with constraints*. University of Houston, 1983.

[90] D. W. Clarke, C. Mohtadi, and P. S. Tuffs, "Generalized predictive control—part i. the basic algorithm," *Automatica*, vol. 23, no. 2, pp. 137–148, 1987.

[91] D. W. Clarke, C. Mohtadi, and P. S. Tuffs, "Generalized predictive control—part ii extensions and interpretations," *Automatica*, vol. 23, no. 2, pp. 149–160, 1987.

[92] J. H. Lee, M. Morari, and C. E. Garcia, "State-space interpretation of model predictive control," *Automatica*, vol. 30, no. 4, pp. 707–717, 1994.

[93] K. R. Muske and J. B. Rawlings, "Model predictive control with linear models," *AIChE Journal*, vol. 39, no. 2, pp. 262–287, 1993.

[94] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[95] P. Krupa, I. Alvarado, D. Limon, and T. Alamo, "Implementation of model predictive control for tracking in embedded systems using a sparse extended ADMM algorithm," *IEEE Transactions on Control Systems Technology*, 2021.

[96] P. Krupa, D. Limon, and T. Alamo, "SPCIES: Suite of Predictive Controllers for Industrial Embedded Systems," 2020.

[97] P. O. Scokaert, D. Q. Mayne, and J. B. Rawlings, "Suboptimal model predictive control (feasibility implies stability)," *IEEE Transactions on Automatic Control*, vol. 44, no. 3, pp. 648–654, 1999.

[98] G. Pannocchia, J. B. Rawlings, and S. J. Wright, "Conditions under which suboptimal nonlinear MPC is inherently robust," *Systems & Control Letters*, vol. 60, no. 9, pp. 747–755, 2011.

[99] M. M. Seron, J. A. De Dona, and G. C. Goodwin, "Global analytical model predictive control with input constraints," in *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No. 00CH37187)*, vol. 1, pp. 154–159, IEEE, 2000.

[100] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[101] D. R. Ramirez and E. F. Camacho, "Piecewise affinity of min–max MPC with bounded additive uncertainties and a quadratic criterion," *Automatica*, vol. 42, no. 2, pp. 295–302, 2006.

[102] D. Muñoz de la Peña, D. R. Ramírez, E. F. Camacho, and T. Alamo, "Explicit solution of min–max MPC with additive uncertainties and quadratic criterion," *Systems & control letters*, vol. 55, no. 4, pp. 266–274, 2006.

[103] E. Zafiriou, "On the closed-loop stability of constrained QDMC," in *1991 American Control Conference*, pp. 2367–2372, IEEE, 1991.

[104] J. A. Primbs and V. Nevistic, "A framework for robustness analysis of constrained finite receding horizon control," *IEEE Transactions on Automatic Control*, vol. 45, no. 10, pp. 1828–1838, 2000.

[105] D. Limon, T. Alamo, D. M. Raimondo, D. Muñoz de la Peña, J. M. Bravo, A. Ferramosca, and E. F. Camacho, "Input-to-state stability: a unifying framework for robust model predictive control," in *Nonlinear model predictive control*, pp. 1–26, Springer, 2009.

[106] G. Grimm, M. J. Messina, S. E. Tuna, and A. R. Teel, "Examples when nonlinear model predictive control is nonrobust," *Automatica*, vol. 40, no. 10, pp. 1729–1738, 2004.

[107] P. J. Campo and M. Morari, "Robust model predictive control," in *1987 American control conference*, pp. 1021–1026, IEEE, 1987.

[108] D. R. Ramirez, T. Alamo, E. F. Camacho, and D. Muñoz de la Peña, "Min-max MPC based on a computationally efficient upper bound of the worst case cost," *Journal of Process Control*, vol. 16, no. 5, pp. 511–519, 2006.

[109] P. O. Scokaert and D. Q. Mayne, "Min-max feedback model predictive control for constrained linear systems," *IEEE Transactions on Automatic control*, vol. 43, no. 8, pp. 1136–1142, 1998.

[110] D. Q. Mayne, E. C. Kerrigan, E. Van Wyk, and P. Falugi, "Tube-based robust nonlinear model predictive control," *International journal of robust and nonlinear control*, vol. 21, no. 11, pp. 1341–1353, 2011.

[111] D. Limon, I. Alvarado, T. Alamo, and E. F. Camacho, "Robust tube-based MPC for tracking of constrained linear systems with additive disturbances," *Journal of Process Control*, vol. 20, no. 3, pp. 248–260, 2010.

[112] R. Gonzalez, M. Fiacchini, T. Alamo, J. L. Guzman, and F. Rodríguez, "Online robust tube-based MPC for time-varying systems: A practical approach," *International Journal of Control*, vol. 84, no. 6, pp. 1157–1170, 2011.

[113] D. Chatterjee, P. Hokayem, and J. Lygeros, "Stochastic receding horizon control with bounded control inputs: A vector space approach," *IEEE Transactions on Automatic Control*, vol. 56, no. 11, pp. 2704–2710, 2011.

[114] D. Chatterjee and J. Lygeros, "On stability and performance of stochastic predictive control techniques," *IEEE Transactions on Automatic Control*, vol. 60, no. 2, pp. 509–514, 2014.

[115] G. C. Calafiore and M. C. Campi, "The scenario approach to robust control design," *IEEE Transactions on automatic control*, vol. 51, no. 5, pp. 742–753, 2006.

[116] M. C. Campi, S. Garatti, and M. Prandini, "Scenario optimization for MPC," in *Handbook of Model Predictive Control*, pp. 445–463, Springer, 2019.

[117] B. Karg, T. Alamo, and S. Lucia, "Probabilistic performance validation of deep learning-based robust nmpc controllers," *International Journal of Robust and Nonlinear Control*, vol. 31, no. 18, pp. 8855–8876, 2021.

[118] M. Alamir, "On probabilistic certification of combined cancer therapies using strongly uncertain models," *Journal of theoretical biology*, vol. 384, pp. 59–69, 2015.

[119] R. Amrit, J. B. Rawlings, and D. Angeli, "Economic optimization using model predictive control with a terminal cost," *Annual Reviews in Control*, vol. 35, no. 2, pp. 178–186, 2011.

[120] A. Ferramosca, J. B. Rawlings, D. Limon, and E. F. Camacho, "Economic MPC for a changing economic criterion," in *49th IEEE Conference on Decision and Control (CDC)*, pp. 6131–6136, IEEE, 2010.

[121] M. Diehl, R. Amrit, and J. B. Rawlings, "A Lyapunov function for economic optimizing model predictive control," *IEEE Transactions on Automatic Control*, vol. 56, no. 3, pp. 703–707, 2010.

[122] P. D. Christofides, R. Scattolini, D. Muñoz de la Peña, and J. Liu, "Distributed model predictive control: A tutorial review and future research directions," *Computers & Chemical Engineering*, vol. 51, pp. 21–41, 2013.

[123] J. M. Maestre, R. R. Negenborn, *et al.*, *Distributed model predictive control made easy*, vol. 69. Springer, 2014.

[124] D. Limon, I. Alvarado, T. Alamo, and E. F. Camacho, "MPC for tracking piecewise constant references for constrained linear systems," *Automatica*, vol. 44, no. 9, pp. 2382–2387, 2008.

[125] D. Limon, A. Ferramosca, I. Alvarado, and T. Alamo, "Nonlinear MPC for tracking piece-wise constant reference signals," *IEEE Transactions on Automatic Control*, vol. 63, no. 11, pp. 3735–3750, 2018.

[126] N. Cressie, "Kriging nonstationary data," *Journal of the American Statistical Association*, vol. 81, no. 395, pp. 625–634, 1986.

[127] J. P. Kleijnen, "Kriging metamodeling in simulation: A review," *European journal of operational research*, vol. 192, no. 3, pp. 707–716, 2009.

[128] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *The Annals of Mathematical Statistics*, vol. 23, no. 4, pp. 493–507, 1952.

[129] G. Alfonso, A. D. Carnerero, D. R. Ramirez, and T. Alamo, "Stock forecasting using local data," *IEEE Access*, vol. 9, pp. 9334–9344, 2020.

[130] A. D. Carnerero, D. R. Ramirez, and T. Alamo, "Probabilistic interval predictor based on dissimilarity functions," *IEEE Transactions on Automatic Control*, 10.1109/TAC.2021.3136137, 2021.

[131] A. D. Carnerero, D. R. Ramirez, and T. Alamo, "State-space kriging: A data-driven method to forecast nonlinear dynamical systems," *IEEE Control Systems Letters*, vol. 6, pp. 2258 – 2263, 2022.

[132] A. D. Carnerero, D. R. Ramirez, D. Limon, and T. Alamo, "Particle based optimization for predictive energy efficient data center management," in

2020 59th IEEE Conference on Decision and Control (CDC), pp. 2660–2665, IEEE, 2020.

[133] A. D. Carnerero, D. R. Ramirez, T. Alamo, and D. Limon, "Probabilistically certified management of data centers using predictive control," *IEEE Transactions on Automation Science and Engineering*, 2021.

[134] G. Alfonso, A. D. Carnerero, D. R. Ramirez, and T. Alamo, "Receding horizon optimization of large trade orders," *IEEE Access*, vol. 9, pp. 63865–63875, 2021.

[135] A. Goshtasby, *Image Registration. Principles, Tools and Methods*. Springer, 2012.

[136] S. T. Wierzchoń and M. Kłopotek, *Modern algorithms of cluster analysis*. Springer, 2018.

[137] J. R. Salvador, D. R. Ramirez, T. Alamo, and D. Muñoz de la Peña, "Offset free data driven control: application to a process control trainer," *IET Control Theory & Applications*, vol. 13, no. 18, pp. 3096–3106, 2019.

[138] J. R. Salvador, D. Muñoz de la Peña, T. Alamo, and A. Bemporad, "Data-based predictive control via direct weight optimization," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 356–361, 2018.

[139] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[140] A. Beck, *First-order methods in optimization*. SIAM, 2017.

[141] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[142] Y. Nesterov, *Lectures on convex optimization*, vol. 137. Springer, 2018.

[143] L. Ljung, *System Identification*. Prentice Hall, Upper Saddle River, NJ, 1999.

[144] V. P. Upadhyay, S. Panwar, R. Merugu, and R. Panchariya, "Forecasting stock market movements using various kernel functions in support vector machine," in *Proceedings of the International Conference on Advances in Information Communication Technology —& Computing*, AICTC '16, (New York, NY, USA), Association for Computing Machinery, 2016.

[145] M. Thomason, "The practitioner methods and tool," *Journal of Computational Intelligence in Finance*, vol. 7, no. 3, pp. 36–45, 1999.

[146] M. Mammarella, V. Mirasierra, M. Lorenzen, T. Alamo, and F. Dabbene, "Chance-constrained sets approximation: A probabilistic scaling approach," *Automatica*, vol. 137, p. 110108, 2022.

[147] T. Alamo, A. Cepeda, and D. Limon, "Improved computation of ellipsoidal invariant sets for saturated control systems," in *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 6216–6221, IEEE, 2005.

[148] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[149] M. Erhard and H. Strauch, "Control of towing kites for seagoing vessels," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 5, pp. 1629–1640, 2012.

[150] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[151] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[152] H. K. Khalil, *Nonlinear control*, vol. 406. Pearson New York, 2015.

[153] D. E. Seborg, T. F. Edgar, D. A. Mellichamp, and F. J. Doyle, *Process dynamics and control*. John Wiley & Sons, 2016.

[154] G. Pannocchia and J. B. Rawlings, "Disturbance models for offset-free model-predictive control," *AIChE journal*, vol. 49, no. 2, pp. 426–437, 2003.

[155] U. Maeder, F. Borrelli, and M. Morari, "Linear offset-free Model Predictive Control," *Automatica*, vol. 45, no. 10, pp. 2214–2222, 2009.

[156] G. Pannocchia, "Offset-free tracking MPC: A tutorial review and comparison of different formulations," in *2015 European Control Conference (ECC)*, pp. 527–532, 2015.

[157] P. M. Oliveira and J. D. Hedengren, "An APMonitor temperature lab PID control experiment for undergraduate students," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 790–797, IEEE, 2019.

[158] T. Alamo, R. Tempo, and E. F. Camacho, "Randomized strategies for probabilistic solutions of uncertain feasibility and optimization problems," *IEEE Transactions on Automatic Control*, vol. 54, no. 11, pp. 2545–2559, 2009.

[159] M. Okamoto, "Some inequalities relating to the partial sum of binomial probabilities," *Annals of the institute of Statistical Mathematics*, vol. 10, no. 1, pp. 29–35, 1959.

[160] H. Cramér, "Les sommes et les fonctions de variables aléatoires," *Actualités Scientifiques et Induestrielles. Conférences Internationales de Sciences.* Paris Hermann, vol. 3, 1938.

[161] A. Klenke, *Probability theory: a comprehensive course*. Springer Science & Business Media, 2013.

[162] L. Parolini, B. Sinopoli, B. H. Krogh, and Z. Wang, "A cyber–physical systems approach to data center modeling and control for energy efficiency," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 254–268, 2011.

[163] J. Scaramella, "Worldwide server power and cooling expense 2006-2010 forecast," *Market analysis, IDC Inc*, 2006.

[164] A. S. Andrae and T. Edler, "On global electricity usage of communication technology: trends to 2030," *Challenges*, vol. 6, no. 1, pp. 117–157, 2015.

[165] P. X. Gao, A. R. Curtis, B. Wong, and S. Keshav, "It's not easy being green," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 211–222, 2012.

[166] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 732–794, 2015.

[167] W. Van Heddeghem, S. Lambert, B. Lannoo, D. Colle, M. Pickavet, and P. Demeester, "Trends in worldwide ICT electricity consumption from 2007 to 2012," *Computer Communications*, vol. 50, pp. 64–76, 2014.

[168] N. Lazic, C. Boutilier, T. Lu, E. Wong, B. Roy, M. Ryu, and G. Imwalle, "Data center cooling using model-predictive control," in *Advances in Neural Information Processing Systems*, pp. 3814–3823, 2018.

[169] H. Endo, S. Suzuki, H. Kodama, T. Hatanaka, H. Fukuda, and M. Fujita, "Development of predictive control system using just-in-time modeling and enthalpy-aware control in air conditioners for large-scale data center," in *2018 18th International Conference on Control, Automation and Systems (ICCAS)*, pp. 1278–1283, IEEE, 2018.

[170] M. Ogawa, H. Fukuda, H. Kodama, H. Endo, T. Sugimoto, T. Kasajima, and M. Kondo, "Development of a cooling control system for data centers utilizing indirect fresh air based on model predictive control," in *2015 7th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pp. 132–137, IEEE, 2015.

[171] L. Fu, J. Wan, J. Yang, D. Cao, and G. Zhang, "Dynamic thermal and it resource management strategies for data center energy minimization," *Journal of Cloud Computing*, vol. 6, no. 1, p. 25, 2017.

[172] C. Bash and G. Forman, "Cool job allocation: Measuring the power savings of placing jobs at cooling-efficient locations in the data center.," in *USENIX Annual Technical Conference*, vol. 138, pp. 140–149, 2007.

[173] J. D. Moore, J. S. Chase, P. Ranganathan, and R. K. Sharma, "Making scheduling cool: Temperature-aware workload placement in data centers.," in *USENIX annual technical conference*, pp. 61–75, 2005.

[174] S. Li, H. Le, N. Pham, J. Heo, and T. Abdelzaher, "Joint optimization of computing and cooling energy: Analytic model and a machine room case study," in *2012 IEEE 32nd International Conference on Distributed Computing Systems*, pp. 396–405, 2012.

[175] L. Parolini, B. Sinopoli, and B. H. Krogh, "Model predictive control of data centers in the smart grid scenario," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 10505–10510, 2011.

[176] M. Ogura, J. Wan, and S. Kasahara, "Model predictive control for energy-efficient operations of data centers with cold aisle containments," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 209–214, 2018.

[177] Q. Fang, J. Wang, and Q. Gong, "QoS-driven power management of data centers via model predictive control," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 4, pp. 1557–1566, 2016.

[178] J. Yao, Z. Pan, and H. Zhang, "A distributed render farm system for animation production," in *Entertainment Computing – ICEC 2009* (S. Natkin and J. Dupire, eds.), pp. 264–269, Springer Berlin Heidelberg, 2009.

[179] B. Bixby, "The Gurobi optimizer," *Transp. Research Part B*, vol. 41, no. 2, pp. 159–178, 2007.

[180] A. L. Visintini, W. Glover, J. Lygeros, and J. Maciejowski, "Monte Carlo optimization for conflict resolution in air traffic control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 4, pp. 470–482, 2006.

[181] N. Kantas, J. Maciejowski, and A. Lecchini-Visintini, "Sequential Monte Carlo for model predictive control," in *Nonlinear model predictive control*, pp. 263–273, Springer, 2009.

[182] G. Andreadis, L. Versluis, F. Mastenbroek, and A. Iosup, "A reference architecture for datacenter scheduling: design, validation, and experiments," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, p. 37, 2018.

[183] E. Chromy, T. Misuth, and M. Kavacky, "Erlang c formula and its use in the call centers," *Advances in Electrical and Electronic Engineering*, vol. 9, no. 1, pp. 7–13, 2011.

[184] J. P. De Villiers, S. Godsill, and S. Singh, "Particle predictive control," *Journal of Statistical Planning and Inference*, vol. 141, no. 5, pp. 1753–1763, 2011.

[185] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *Journal of computational and graphical statistics*, vol. 5, no. 1, pp. 1–25, 1996.

[186] J. Sanders and E. Kandrot, *CUDA by example: an introduction to general-purpose GPU programming, portable documents*. Addison-Wesley Professional, 2010.

[187] K. Karimi, N. G. Dickson, and F. Hamze, "A performance comparison of CUDA and OpenCL," *arXiv preprint arXiv:1005.2581*, 2010.

[188] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, and K. Skadron, "A performance study of general-purpose applications on graphics processors using CUDA," *Journal of parallel and distributed computing*, vol. 68, no. 10, pp. 1370–1380, 2008.

[189] P. Bialas and A. Strzelecki, "Benchmarking the cost of thread divergence in CUDA," in *International Conference on Parallel Processing and Applied Mathematics*, pp. 570–579, Springer, 2015.