Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Ferdy Odin Wessing

System Identification and Spatio-Temporal Kalman Filtering for Muscular Pressure Data

Master's thesis in Cybernetics and Robotics
Supervisor: Damiano Varagnolo
June 2020

NTNU
Norwegian University of Science and Technology

Ferdy Odin Wessing

System Identification and Spatio-Temporal Kalman Filtering for Muscular Pressure Data

Master's thesis in Cybernetics and Robotics
Supervisor: Damiano Varagnolo
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Abstract

This thesis serves as an initial step towards smoothed estimation of the pressure caused by the pelvic floor muscles. The method used to achieve the smoothing can be divided into two steps. First, system identification is done to create a polynomial model of the dynamics. This is done by approximating the brain signals which control the muscles as a binary input with a change detection algorithm. This input must be time-shifted back in time to occur before the excitation in the pressure. The collected pressure data from the FemFit vaginal pressure sensor is combined with this input to create a discrete-time ARX model for each of the 8 sensors of the FemFit. This is done by assuming that each of the pressures measured have the same distribution, but with a scaling factor on the input. The scaling factors, ARX parameters and time-shifts are jointly optimized for different orders of ARX models. All the ARX models achieve a simulation fit of approximately 45% on the validation set, indicating that they are likely limited by the simple input estimation. The simplest model, the ARX211, is converted into a state space model. This state space model is used in the second step for spatio-temporal gaussian regression using a Kalman filter. This Kalman filter is adapted, for the purpose of muscular pressure data, from Todescato et al. [2020]. As opposed to the original paper, the method in this thesis arrives directly at a discrete-time model. Because of this, the initial covariance of the Kalman filter must be estimated, and the noise covariances must be tuned. A tuning which adequately models the dynamics was not found, but the smoothing properties of the method are presented. Some improvements are suggested in future work, together with some possible applications of the method.

Sammendrag

Denne masteroppgaven er et første skritt mot å oppnå ett glattet estimat av trykket som skapes av bekkenbunnsmuskulaturen. Metoden som presenteres kan deles inn i to skritt. Først gjøres system identifikasjon for å finne en polynomisk modell av systemets dynamikk. Signalene fra hjernen som styrer musklene approksimeres som et binært inngangssignal ved hjelp av en change detection algoritme. Dette inngangssignalet må tidsskiftes tilbake i tid, slik at inngangssignalet eksiteres før trykket. Trykkdataen som er samlet av FemFit, en vaginal trykksensor, kombineres med det estimerte inngangssignalet for å finne en diskret-tids ARX modell for hver av de 8 sensorene som FemFit består av. Dette gjøres ved å anta at hver av trykkmålingene har samme fordeling, med unntak av en skaleringsfaktor på inngangssignalet. Skaleringsfaktoren, ARX parametrene og tidsskiftet optimeres samtidig for ARX modeller med forskjellig orden. Alle de evaluerte ARX modellene oppnådde et simulerings-fit på omtrent 45% på valideringsdata, noe som indikerer at nøyaktigheten begrenses av den simple estimeringen av inngangssignalet. Den enkleste modellen, ARX211, konverteres til en tilstandsromrepresentasjon. Tilstandsromrepresentasjonen brukes i det andre skrittet til gaussisk regresjon i tid og rom ved hjelp av et Kalman filter. Kalman filteret er tilpasset fra Todescato et al. [2020] for bruk med muskeltrykk som data. I motsetning til den originale algoritmen, blir et system funnet i diskret-tid direkte. På grunn av dette må den initielle kovariansen estimeres, og støy kovariansene må stilles inn. Verdier for kovariansene som modellerer dynamikken tilstrekkelig ble ikke funnet, men den glattende egenskapen – som glatter ut støy – blir presentert. Til slutt blir noen forbedringer foreslått som fremtidig arbeid, og noen potensielle bruksområder nevnes.

Preface

This thesis was conducted during the spring of 2020, and concludes a Master of Science in Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU).

I would like to thank my supervisor, prof. Damiano Varagnolo, for his guidance and for taking the time to regularly discuss the thesis with me. A special thanks to my family and friends for the motivation and feedback they have provided, and thanks to OV for providing me with the energy to keep working.

Ferdy Wessing

Trondheim, June 25, 2020

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis adapts spatio-temporal Gaussian regression to a biomedical application in which measurements of muscular pressure fields from noisy and not entirely reliable sensors shall be filtered considering both spatial and temporal correlations. Spatio-temporal processes are, as the name implies, processes which are dependent on both space and time. Such processes emerge in many different scientific fields such as weather forecasting [Das and Ghosh, 2014], Epidemiology [Meliker and Sloan, 2011], a field concerned with the emergence and spread of diseases, and detection and tracking of vehicles from video [Chen et al., 2003]. There are many approaches to model such processes, but because of their inherent complexity some simplifying assumptions are often made. These assumptions are mostly about the form of the covariance function. Common assumptions are temporal stationarity, spatial stationarity, separability, and/or a combination of these. Stationarity means that the covariance is independent of the values of the variable, and only depends on the distance between them. Therefore, the covariance is unchanged when the two values are shifted equally in time. separability signifies that the covariance is a sum of two independent functions. One which depends only on the spatial variables, and another which only depends on the temporal ones. Stationarity is defined by equation (1.1a) while separability is defined by equation (1.1b).

$$K(t,t') = K(\tau), \qquad \text{where } \tau = t - t' \tag{1.1a}$$

$$K(x,x',t,t') = K_s(x,x')h(t,t') \tag{1.1b}$$

Here, only processes that are separable in time and space, as well as stationary in time are considered. Furthermore, the process is assumed to be Gaussian. Gaussian processes (GP) inherit powerful tools from Gaussians while producing good results for many different problems. These Gaussian process methods have become the standard approach for many different applications. While much work has been done on problems with spatial locations as input, recent research has expanded to spatio-temporal processes. The most basic approach is to bundle time together with space as simple another input feature [Williams and Rasmussen, 2006]. However, this non-iterative

method is memory and computationally intensive. Other works reduce the problem space by using sparse approximations [Williams and Rasmussen, 2006] or finite memory implementations [Xu et al., 2011]. Other research has focused on creating state-space representations for temporal GP regression, which can be combined with Kalman filtering [Hartikainen and Särkkä, 2010; Todescato et al., 2020]. This thesis attempts to leverage the spatio-temporal regression using a Kalman filter presented by Todescato et al. [2020] for biomedical applications where there is a need is for filtering noisy measurements of muscular pressure fields.

In order to use the spatio-temporal Kalman filtering, a model which explains the behaviour of each individual state is required. In this thesis, these models are found by polynomial model regression. Since the brain waves acting as the input to the musculature is unknown, an approximation is suggested. The polynomial models are converted to state space models, which are then stacked to represent the full system.

1.1 Motivation and Goal

The motivation for this thesis is the need for a systematic, data-driven approach to treat signals from muscular pressure sensors (in our specific case, from pelvic floor muscle monitoring systems). Improved physiological treatments often require good understanding of the physiological and psychological status of patients. Improved modelling of how well the musculature can respond to stimuli, how well it can develop by following training exercises, and how fast it fatigues allow the physicians to give more appropriate feedback, can improve the individualization of treatments, and can simplify, and make more accurate, medical diagnoses.

In this thesis we consider a specific case, i.e., data from pelvic floor muscles. The medical importance of such musculature derives from the fact that these muscles commonly become weaker due to causes such as pregnancy, overuse of the muscles, being overweight and advancing age. Issues related to the weakened muscles, such as urinary incontinence, are estimated to affect one in three women [Walker and Gunasekera, 2011]. The standard treatment of these conditions is called pelvic floor muscle treatment PFMT, or Kegel exercises. However, these methods often lack feedback, and results may not be seen for several months. Many women report that they are not confident in their ability to correctly perform the exercises. Therefore, Kask et al. [2019] suggest gamification, together with the FemFit measurement device, as a biofeedback tool to give feedback on how the exercise is performed quickly, as well as improving retention. For this application, it is important to filter the raw data. There is a need to remove noisy and fault data, which can stem from faulty sensors, or imperfect insertion of the FemFit. Additionally, a smooth estimate of the underlying function can improve the results of many methods. Their initial study indicates that general models may not generalize well from person to person. This introduces a need to individualize the models, for example by tuning parameters of the models themselves or by tuning hyperparameters associated with the various algorithms that are coupled to medical diagnoses. When these parameters and hyperparameters can be re-tuned online during sessions, they may also become an indication of improvements in the patients. Moreover, good models may also be able to evaluate whether the correct pelvic floor muscles are used, or if the person is overusing their abdominal or hip muscles.

The goal of this thesis is to be an initial step towards an improved modelling approach of the pelvic floor muscles by using data from the FemFit. More specifically for the purpose of estimating fatigue levels, personalization and identifying whether the Kegel exercises are done correctly. The steps to be taken towards achieving this are summarized below.

- Estimate the brain-force which is responsible for the observed data
- Create simple linear models based on the data and the brain-force estimate
- Determine which model best approximates the dynamics while keeping complexity low
- Use the optimal linear model as a starting point for spatio-temporal estimation
- Modify a Spatio-temporal estimation method for use with muscular pressure data

1.2 Thesis Structure

The next chapter (chapter 2) provides the required background for the thesis. This includes the algorithms used for input estimation, System identification using polynomial models, Kalman filtering, and Gaussian processes. Additionally, the algorithm suggested by Todescato et al. [2020] is presented. Chapter 3 describes how the specific parts are implemented, as well as introducing the dataset used. The results are presented in chapter 4, which are subsequently discussed in chapter 5. Finishing thoughts and future work are presented in chapter 6.

Chapter 2

Background Theory

This chapter presents the theoretical background of the thesis, as well as the recent research on which it is based. First, relevant methods for system identification and hyperparameter optimization are presented. Secondly, the basic Kalman filter is explained. The third subsection introduces the basics of gaussian processes and gaussian process regression. The fourth subsection shows how these concepts can be combined to create powerful spatio-temporal estimators, while the last section introduces the medical application.

2.1 System Identification

System identification is a field which uses statistical methods together with a dataset to create mathematical models of time-dependent systems. Most of these models can be classified as one of three types: black-box, white-box or grey-box. A black-box model assumes that the system is completely unknown. For this type of modelling, it is often necessary to estimate parameters of several model structures so that the results can be compared. Typically, one begins with a simple linear model, and weighs the improvement gained by using more complex models against the increase in complexity. White-box models are in a sense the opposite of black-box models. The model is created while all the necessary information about the system is known. Everything in between these extremes is known as grey-box modelling. Typically, more *a priori* information, information acquired before modelling, about the system results in higher accuracy of the model. Most commonly, one has some intuition or prior knowledge about the type of function or the smoothness of the output.

Mathematical models can also be divided into whether the identification is done with both the input and the output as a dataset, or if only the output is used. Having access to the input typically improves the accuracy, however such data is not available in all cases. Since the mathematical

modelling is data-driven the results will depend heavily on the quality of the dataset. in particular it will depend on how well the dataset manages to capture the essential qualities of the system. Therefore, creating experiments that optimally captures the system is a field of study in itself, see for example the overview presented by Pronzato [2008].

Ljung [2007] describes the field of system identification as consisting of 4 main parts. The dataset, a set of candidate models, a criterion of fit and validation. One wishes to find the model among the set of candidates that best represents the dataset, as indicated by the criterion of fit. When one model is chosen, it must subsequently be evaluated and validated.

2.1.1 Change Detection Algorithms

Change detection is a field of study concerned with detecting abrupt changes for monitoring, segmentation and detection of failure. These abrupt changes are not necessarily large, but signify a shift in the underlying properties as indicated by a change in the average value or stationary value. If the system has two different states separated by abrupt changes, one can make use of change detection algorithms to estimate a binary signal which indicates the state of the system. For some systems, this binary signal can be seen as a simplified estimate of the input which causes the abrupt changes seen in the output.

Change detection algorithms are most commonly used on three classes of problems: on-line detection of a change, off-line hypotheses testing and off-line estimation of the change time [Basseville et al., 1993]. The performance is evaluated differently depending on both the class and the application. However, some intuitive performance indexes, paraphrased from Basseville et al. [1993], are:

- mean detection delay
- probability of nondetection
- mean time between false detections
- probability of false detections
- accuracy of the change time and magnitude estimates

On-line Change Detection

On-line change detection considers a sequence of random variables $\{y_k\}$ with a probability density $p_\theta(y)$. One then wishes to detect a change in θ as soon as possible after it occurs. In the simplest case, θ is scalar and is one of two values, in other words is $\theta = \theta_0$ before the change, and $\theta = \theta_1$ after the change. One way to estimate the probability of a value of theta is the logarithm of the likelihood ratio, given by equation (2.1). A change in the sign of this value indicates a change in

the parameter θ. Two algorithms to detect this change are the geometric moving average, and the finite moving average.

$$s(y) = \ln \frac{p_{\theta_1}(y)}{p_{\theta_0}(y)} \tag{2.1}$$

The geometric moving average (GMA) algorithm is based on the behaviour of the log likelihood ratio defined in equation (2.1) as well as exponential weighting. Exponential weighting ensures that recent observations have high weights, while past ones have lower weights which rapidly decrease towards zero. The distributions of the weights for two different tunings are shown in figure 2.2. A simplification can be made for the case of two levels. Since a threshold is used to define when the change occurs, one can simply take the difference from the initial equilibrium as s_k and tune the threshold appropriately. the algorihtm is defined recursively by equation (2.2a) for the one-sided case. The alarm time t_k is then given by the first k for which g_k is above a chosen threshold h_{GMA}, as in equation (2.2b). The value α is the forgetting factor which indicates how quickly a measurement should be forgotten, and is the only tunable hyperparameter of the method.

$$g_k = (1-\alpha)g_{k-1} + \alpha(y_k - \mu_0), \qquad \text{with } g_0 = 0 \tag{2.2a}$$

$$t_k = min\{k : g_k > h_{GMA}\} \tag{2.2b}$$

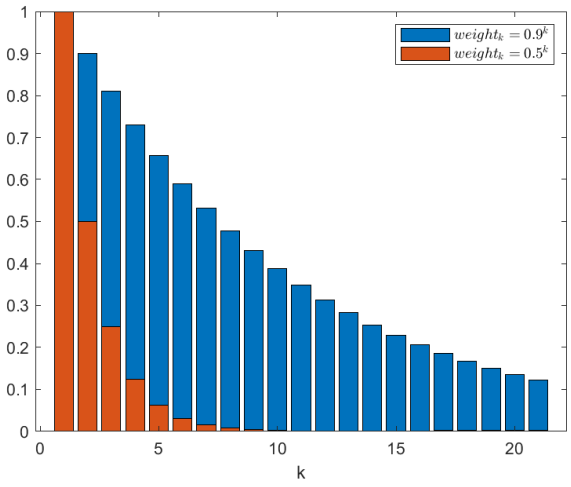


Figure 2.1: Distribution of exponential weights for two different forgetting factors α.

An alternative approach to GMA is th finite moving average algorithm (FMA). Instead of using an exponential forgetting factor, it uses a finite set of weights. To use this method both the window length and the weights need to be designed. Given the weights γ_i and the window length N the algorithm to detect a change in mean is given by equation (2.3a). The stopping rule, given by

equation (2.3b), is similar to GMA, where the change time is given by the first k for which the value of g_k goes above a threshold h_{FMA}. The threshold will depend on the chosen weights. A possible weighting method is to use linear weights which sum to one. This ensures that the threshold can be set relative to the strength of the signal itself. The weights of this method are illustrated in figure 2.1.

$$g_k = \sum_{i=0}^{N-1} \gamma_i (y_{k-i} - \mu_0) \tag{2.3a}$$

$$t_k = min\{k : g_k > h_{FMA}\} \tag{2.3b}$$

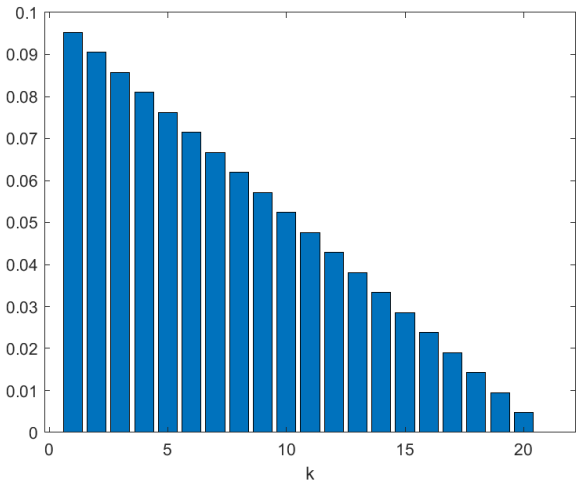


Figure 2.2: Distribution of normalized linear weights with a window size of 20.

2.1.2 Polynomial Model Regression

The following section is based largely on the works of Lennart Ljung. In particular, Ljung [2007] provides an overview of the field, while Ljung [2012] describes the theory behind, and implementation of, the system identification toolbox for matlab. A polynomial model expresses the discrete-time relationship between the input $u(k)$, noise $e(k)$ and output $y(k)$ by defining polynomials of the time-shift operator q^{-1}. This q corresponds to the z of a Z-transform, and is used to represent difference equations as polynomials. It does this with the notation $q^{-r}y(t) = y(t - rT_s) = y(k - r)$, where T_s is the sampling interval. Since these models do not require prior knowledge about the system, they are considered black-box models. The general form of a polynomial model with a single input is given by equation (2.4). where $u(k)$ is the input, while $e(k)$ is white noise. The general form includes the polynomials $A(q)$, $B(q)$, $C(q)$, $D(q)$ and $F(q)$ as well as an integrator in the noise. Usually, the general model is simplified by setting some of the matrices to identity

or zero. This results in simpler forms such as AR, ARX, ARMAX or Output-Error. These forms differ by which polynomials are included, and can have variants which include the integrator. In addition to choosing the type of polynomial model to use, one also needs to choose the vector which specifies the model order, defines as the orders of the individual matrices. The values in this vector specify how many coefficients there are for each of the polynomials A, B, C, D and F as well as an additional value n_k which gives the delay between input and a reaction in the output. This delay n_k, often called the dead time, increases the smallest delay in $B(q)$ to q^{-n_k}, resulting in an order of $n_b + n_k$ of the polynomial. The model order also uniquely defines the number of poles and zeros for both the transfer function between input and output and between noise and output.

$$A(q)y(k) = \frac{B(q)}{F(q)}u(k - n_k) + \frac{C(q)}{D(q)}\frac{1}{1 - q^{-1}}e(k) \tag{2.4}$$

AR and ARX Models

Autoregressive (AR) models are used for time-series data with a single output and no input. The model order is then the scalar n_a which defines the polynomial order of $A(q)$. This means that the output $y(k)$ is dependent on $y(k - 1)$, ..., $y(k - n_a)$. ARX stands for Autoregressive with Exogenous input. An exogenous variable is simply a variable whose value is determined outside of the model, and acts on the model. Unlike the AR model it incorporates an input and therefore also includes the polynomial $B(q)$. The model order of an ARX model is defined by the vector $[n_a, n_b, n_k]$. The model type is often described by appending the model type and the model orders, for example ARX211 for an arx model of order [2, 1, 1]. The equation for an AR model is given in equation (2.5a), while equation (2.5b) shows the equation for an ARX model.

$$A(q)y(k) = e(k) \tag{2.5a}$$
$$A(q)y(k) = B(q)u(k - n_k) + e(k) \tag{2.5b}$$

An ARX model is a good solution when the noise is affected directly by the system dynamics. This is because the noise model is given by $1/A(q)$ when rearranging to have $y(k)$ alone on the left hand side. Said another way, the previous noises affect the current measurement only by propagation through previous measurements, therefore being multiplied by the coefficients of $A(q)$. This model works well with a good signal-to-noise ratio while keeping the model complexity low.

ARMA and ARMAX Models

An Autoregressive Moving Average (ARMA) model extends AR models by adding the polynomial $C(q)$, giving it the model order $[n_a, n_c]$. This allows for the measurement to be affected by a moving average of the noise disturbances. The ARMAX model extends the ARX model in the same way. the ARMA and ARMAX difference equations are given by equation (2.6a) and equation (2.6b)

respectively. Because of the added polynomial, the ARMAX has an additional model order parameter to choose; n_c, resulting in the model order vector $[n_a, n_b, n_c, n_k]$. The ARMAX model works particularly well when the disturbances enter at the input.

$$A(q)y(k) = C(q)e(k) \tag{2.6a}$$
$$A(q)y(k) = B(q)u(k - n_k) + C(q)e(k) \tag{2.6b}$$

Determining model order and Regression

The optimal model order and model type are in most cases unknown and must be determined. Usually, one begins with a simple model structure such as an ARX. Then several ARX models are estimated, and the results can be compared. A subjective evaluation is then done by weighing performance against model complexity. At this point, one can either choose to continue with an ARX model, or use the chosen model orders as a starting point for a more complex model such as ARMAX.

A method to evaluate models more rigorously is called the Akaike Information Criterion. The estimator, introduced by Akaike [1974], estimates the prediction error, giving an estimate of the relative quality of a model. It does so by weighing the goodness of fit of the model against the complexity. Therefore, it both punishes overfitting and underfitting. The basic equation is given by equation (2.7), where k is the number of parameters, and $\hat{L}$ the maximum value of the likelihood function.

$$\mathrm{AIC} = 2k - 2\ln(\hat{L}) \tag{2.7}$$

Optimal parameters can be found for an ARX model non-iteratively. This is done by using QR factorization to solve the linear equations constituting the least-squares estimation problem. The model parameter vector θ can be estimated by solving the normal equation

$$(J^T J)\theta = J^T y \tag{2.8}$$

Where y is the measured output and J is a regressor matrix, also called the design matrix. Using least-squares estimation allows the equations to be in analytic form, and ensures a unique solution for each optimization goal. Common optimization goals are to maximize the fit for k-step ahead predictions or for simulation. Simulation means that one only uses the input data and some initial conditions. The whole response is then calculated based on those initial values. k-step ahead prediction, on the other hand, uses both the measured signal and the input, and predicts the values k steps ahead at each point. The optimal parameters for an ARMAX can only be found by using an iterative approach.

Pre-processing

Treating and pre-processing the dataset before use is in many instances essential for system identification. In the case of a dataset from a typical use-case, rather than a designed scenario, it is often not ideal for system identification. Such a dataset may contain level changes, disturbances and outliers. For these cases it is beneficial to choose a part of the dataset which seems representative of the overall system dynamics. This dataset should be further split into an estimation set and a validation set to ensure that the validation is not affected by overfitting the model to the training data.

An important consideration when pre-processing a dataset is whether one should detrend the data. Detrending means that one removes the mean or linear trends from a signal. This often helps create more accurate predictions, since linear models can't capture arbitrary differences between input and output. Such a scenario can for example occur when there is drift during the experiment. On the other hand, one should never detrend the data using the mean or similar measures when the measurements are relative to a value that corresponds to a physical equilibrium. In this case, one can detrend using the equilibrium level.

2.1.3 Numerical Optimization

Numerical optimization is the act of finding the optimal value of an objective function given that it depends on some characteristics, or variables, of the system. Additionally, the variables can be constrained in some way. Identifying the best objective, variables and constraints is called modelling. After this, an optimization algorithm is used to find the solution. It is important to choose an algorithm which is appropriate for the problem structure. After an answer has been acquired, it can be validated. This can indicate whether the solution is optimal, how the estimates can be improved and which variables the solution is sensitive to.

Mathematically, the general optimization problem describes above can be written as following

$$\begin{aligned} \min_{x \in R^n} \quad & f(x) \\ \text{subject to} \quad & c_i(x) = 0, \quad i \in \mathcal{E} \\ & c_i(x) \leq 0, \quad i \in \mathcal{I} \end{aligned} \tag{2.9}$$

Where $f(x)$ is a scalar objective function to minimize, c_i are the constraint functions and $\mathcal{E}$ and $\mathcal{I}$ are the set of indices for equality and inequality constraints respectively. The solution algorithms can be divided into some classes. The following division of algorithms is based on the introduction chapter of Nocedal and Wright [2006]. If the variables are integers, the problem is referred to as an integer programming problem. If some of the variables are not restricted to integer or binary values, they are referred to as mixed integer programming problems. More generally, discrete programming contains values contained in a set. Discrete problems often have a limited, but large, set of possible solutions for the variable x. The feasible set for continuous optimization, on the other hand, is

usually uncountably infinite. Yet, continuous problems are often easier to solve because one can exploit the smoothness of the functions.

The problem space is often also divided into constrained and unconstrained optimization. Constrained problems are when the constraint functions affect the solution. If both the objective function and the constraints are linear functions of x, the problem is referred to as a linear programming problem. A separation is also made based the quality of the solution. For easier problems, a global solution may be found. However, for many real applications one only arrives at a local solution. Different local solutions may then be found by varying the initial point.

Regularization

Regularization is a technique to avoid overfitting to the training data, as well as limiting the values of the variables. This is also exploited by a set of methods called penalty methods. here, one replaces the constraints with a penalty on being far away from satisfying the constraint. Ideally, the solution of this unconstrained problem will then converge to the constrained problem.

Regularization is done by adding a term to the objective function. Therefore, the optimization seeks to achieve a balance between the objective itself, and keeping the regularization term small. In general, a cost function with regularization is of the form

$$f(x) = f^*(x) + \lambda R(x) \tag{2.10}$$

where $f^*(x)$ is the objective without regularization, $R(x)$ is the regularization term itself and λ is the relative weighting parameter. Regularization may be thought of as imposing Occam's razor. This means that it attempts to find those among approximately equally good solutions that introduces the least unnecessary complexity.

Two common regularization techniques are called L1 regularization and L2 regularization. For these techniques the regularization terms are given by the following two equations respectively

$$R(x) = |x - x_0|$$
$$R(x) = (x - x_0)^2$$

L1 regularization shrinks the coefficients of the less important features to zero, thereby reducing the problem complexity. L2 on the other hands retains nonzero weights on all variables, therefore accounting even for the less important ones.

2.2 Kalman Filtering

Kalman filtering is an algorithm which takes in the known inputs to a system together with the outputs of the system to produce estimates of internal, unknown variables. The algorithm uses these measurements over time to produce improved estimates. The basic interaction with the system is shown in figure 2.3. The standard Kalman filter contains a linear model of the system, defined by equation (2.12). where $\boldsymbol{x}(k) \in \mathbb{R}^n$, and $\boldsymbol{y}(k) \in \mathbb{R}^m$. A complete overview of the vector and matrix sizes is given by table 2.1. $A(k)$ defines the state transition from the previous state, $B(k)$ is a model applied to the input, while $C(k)$ maps out how the observed state space relates to the true states. $\boldsymbol{w_k} = \boldsymbol{w}(k)$ is the process noise and $v_k = v(k)$ is the measurement noise. Both of these are assumed to be drawn from a multivariate gaussian with zero mean. The covariance of $\boldsymbol{w_k}$ is given by Q_k, while the covariance of $\boldsymbol{v_k}$ is given by R_k, as given by equation (2.11). All noise vectors are assumed to be independent.

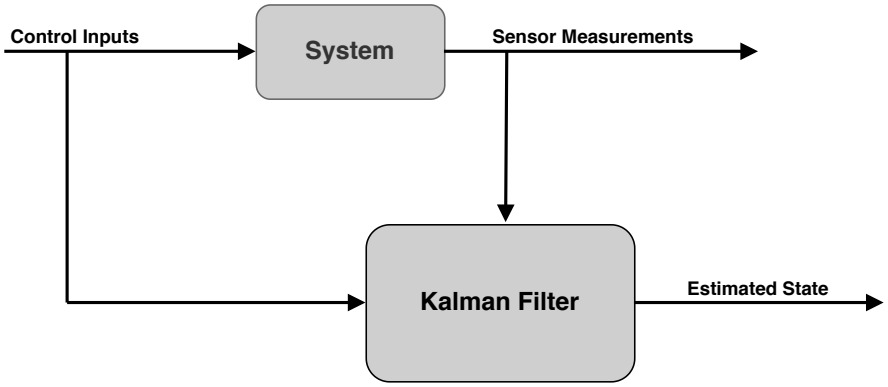


Figure 2.3: Block diagram showing the interaction between the system and the Kalman filter.

$$\boldsymbol{w_k} \sim \mathcal{N}(0,\, Q_k) \tag{2.11a}$$
$$\boldsymbol{v_k} \sim \mathcal{N}(0,\, R_k) \tag{2.11b}$$

$$\boldsymbol{x}(k) = A(k)\boldsymbol{x}(k) + B(k)\boldsymbol{u}(k) + \boldsymbol{w}(k) \tag{2.12a}$$
$$\boldsymbol{y}(k) = C(k)\boldsymbol{x}(k) + \boldsymbol{v}(k) \tag{2.12b}$$

The Kalman filter algorithm can be thought of as consisting of two stages. A prediction stage and an update stage. Alternatively, they are often referred to as the propagation and correction phase. The prediction stage is given by the following equations

$$\boldsymbol{x}(k+1 \mid k) = A(k)\boldsymbol{x}(k|k) + B(k)\boldsymbol{u}(k) \tag{2.13a}$$
$$\Sigma(k+1 \mid k) = A(k)\Sigma(k \mid k)A(k)^T + Q(k) \tag{2.13b}$$

The predicted state is found using the current input together with the estimate from the previous time step. Σ is the estimated covariance matrix, the equation above estimates how accurate the predicted state is. These predicted values are referred to as *A priori* estimates, since no information from the current output is used. In the update stage the observation from the current time step is incorporated. This observation is combined with the previous prediction to create a more accurate estimate. The update equations are given by

$$\begin{aligned} L(k+1) = \Sigma(k+1 \mid k)C(k+1)^T \\ \left(C(k+1)\Sigma(k+1 \mid k)C(k+1)^T + R(k)\right)^{-1} \end{aligned} \tag{2.14a}$$
$$\begin{aligned} \boldsymbol{x}(k+1 \mid k+1) = \boldsymbol{x}(k+1 \mid k) + L(k+1) \\ (\boldsymbol{y}(k+1) - C(k+1)\boldsymbol{x}(k+1 \mid k)) \end{aligned} \tag{2.14b}$$
$$\Sigma(k+1 \mid k+1) = (I - L(k+1)C(k+1))\,\Sigma(k+1 \mid k) \tag{2.14c}$$

The matrix $L(k+1)$ is the Kalman gain, and is multiplied by the difference in predicted state and measured state to provide the correction term. It can be thought of as weighing the uncertainty of the prediction against the uncertainty of the measurement, based on the estimates of noise and covariance. $\boldsymbol{x}(k+1 \mid k+1)$ and $\Sigma(k+1 \mid k+1)$ are the *a posteriori* estimates. One can observe from the minus sign in equation (2.14c) that adding additional information about the system, in this case the observed output, decreases the covariance estimate from the predict stage. The procedure is shown as a block diagram in figure 2.4, where the updates of estimated covariance and Kalman gain are omitted for clarity.

Table 2.1: Sizes of all variables and matrices which are part of the Kalman filter and system model.

Variable	**Size**
$\boldsymbol{x}(k)$, $\boldsymbol{w}(k)$	$n \times 1$
$\boldsymbol{y}(k)$, $\boldsymbol{v}(k)$	$m \times 1$
$\boldsymbol{u}(k)$	$p \times 1$
$A(k)$, Q_k, $\Sigma(k)$	$n \times n$
$B(k)$	$n \times p$
$C(k)$	$m \times n$
R_k	$m \times m$
$L(k)$	$n \times m$

For most applications, the hidden states one wishes to estimate is much larger than the observable outputs. As one can see from the equations, the Kalman filter is recursive. This means that only values from the previous timestep are required to produce the next iteration. The Kalman filter

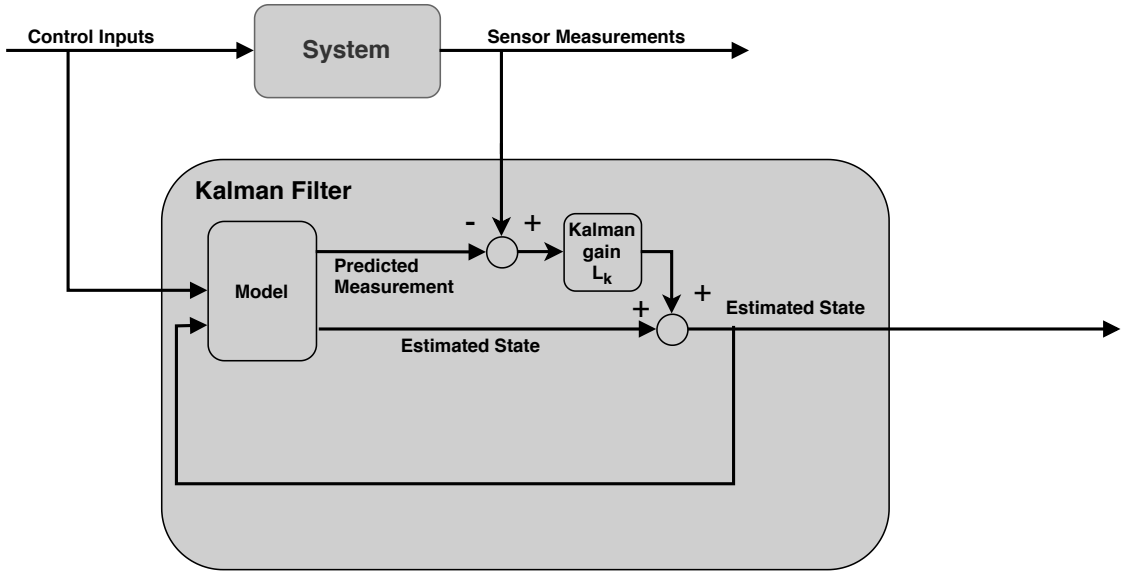


Figure 2.4: Block diagram showing the interactions of the Kalman filter and the system and a simplified view on the estimation procedure.

was shown by Kalman [1960] to be the optimal linear filter given some conditions. Mainly that the model perfectly matches the real system, the noise is white and uncorrelated and the covariances of the noise are known accurately.

2.3 Gaussian Process Regression

A Gaussian Process (GP) is a type of stochastic process. Stochastic processes are a collection of random variables often characterized by some mutual properties. Gaussian processes have the property that all finite collections of the random variables have a multivariate gaussian distribution, sometimes called a multivariate normal or simply a gaussian distribution. Therefore, we will first look at the properties of a gaussian distribution before looking closer at gaussian processes.

2.3.1 Univariate Gaussian Distribution

A Gaussian or normal distribution is characterized by the probability density function given in equation (2.15). The μ parameter defines the expected value, or mean, of the distribution, while σ^2 defines the variance. Since a Gaussian distribution is uniquely defined by the expected value and the variance, the following notation is used to denote that a random variable is normally distributed with parameters μ and σ^2: $X \sim \mathcal{N}(\mu, \sigma^2)$. The gaussian distribution is always symmetric, and shaped like a bell-curve as shown in figure 2.5.

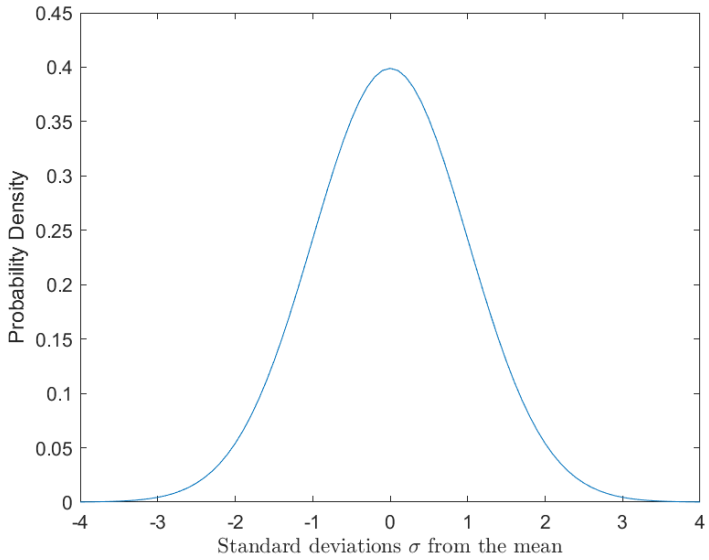


Figure 2.5: The shape of the standard normal distribution.

The gaussian distribution is essential in statistics because of some useful properties. One of things that makes the gaussian so powerful is the central limit theorem. This states that, given some conditions, the normalized sum of any independent random variables tends towards a normal distribution. This occurs even when the original variables are not normally distributed. This is a powerful theorem, as the Gaussian has many desirable properties which make it easier to work with. Notably, the family of normal distributions is closed under linear transformations. This means that given a normally distributed X with mean μ and variance σ^2 then $Y = aX + b$, with $a, b \in \mathbb{R}$ is also normally distributed with mean $a\mu + b$ and variance $a^2\sigma^2$. Furthermore, any linear combination of normal variables is also a normal variable. Other properties of a gaussian is symmetry around the mean and an infinitely differentiable density.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \tag{2.15}$$

2.3.2 Multivariate Gaussian Distribution

The Multivariate gaussian is a generalization of the above univariate gaussian distribution. To make this generalization, The degenerate case of the univariate gaussian is often included. A degenerate distribution of a single variable is deterministic in that it only has one possible value. This case occurs when the variance tends towards zero while the probability density functions height goes to infinity at the mean. When expanding the degenerate case to multivariate gaussians, this occurs

when one or more of the random variables are deterministic functions of the others. When this is the case, the determinant of the covariance is zero.

A random variable $\boldsymbol{X} \in \mathbb{R}^n$ has a multivariate Gaussian distribution if any linear combination of its components is a univariate gaussian. In other words if $a^T\boldsymbol{X} = \sum_{i=0}^{N} a_i X_i$, for $\forall a \in \mathbb{R}^n$ is a univariate gaussian. The multivariate gaussian is denoted as $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is a covariance matrix. In the non-degenerate case, the probability density function is given by equation (2.16). The covariance matrix is always symmetric, and is positive semi-definite in the general case and positive definite in the non-degenerate case. The value Σ_{ij} is a measure of how correlated the variables X_i and X_j are. They are uncorrelated for the special case $\Sigma_{ij} = 0$. This means that if each component of $\boldsymbol{X}$ is independent from the others and normally distributed, then the covariance matrix is a diagonal matrix with the individual variances on the diagonal. It does not hold in general that a vector consisting of univariate gaussians is a multivariate gaussian.

$$f_{\boldsymbol{X}}(x_1, \ldots, x_k) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right) \tag{2.16}$$

Conditional distributions

The conditional distribution of a multivariate-gaussian is essential for GP regression. This is found by partioning the N-dimensional vector $\boldsymbol{x}$ so that we have the following:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x_a} \\ \boldsymbol{x_b} \end{bmatrix}, \text{ where } \boldsymbol{x_a} = \begin{bmatrix} x_1 \\ \vdots \\ x_l \end{bmatrix} \text{ and } \boldsymbol{x_b} = \begin{bmatrix} x_{l+1} \\ \vdots \\ x_n \end{bmatrix} \tag{2.17}$$

This results in the following partition of the mean and covariance:

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix}, \text{ with size } \begin{bmatrix} l \times 1 \\ (N-l) \times 1 \end{bmatrix} \tag{2.18a}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{bmatrix}, \text{ with size } \begin{bmatrix} l \times l & l \times (N-l) \\ (N-l) \times l & (N-l) \times (N-l) \end{bmatrix} \tag{2.18b}$$

One can then take the distribution of $\boldsymbol{x_a}$ on $\boldsymbol{x_b} = \boldsymbol{c}$, this results in a multivariate normal $(\boldsymbol{x_a}|\boldsymbol{x_b} = \boldsymbol{c}) = \mathcal{N}(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}})$, where the parameters are given by equation (2.19).

$$\bar{\boldsymbol{\mu}} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{c} - \boldsymbol{\mu}_2) \tag{2.19a}$$

$$\bar{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} \tag{2.19b}$$

Note that having information about $\boldsymbol{x}_b$ decreases the overall uncertainty of $\boldsymbol{x}_a$ as seen by the expression for the covariance. Further one sees that new covariance does not depend at all on the value $\boldsymbol{x}_b = \boldsymbol{c}$, but only on the different covariance matrices. The mean, on the other hand, is shifted by a matrix times the offset between the expected mean and the values that were observed.

2.3.3 Gaussian Processes

Gaussian processes can be viewed as a generalization of the multivariate Gaussian distribution that allows for infinitely many variables. This means that instead of being defined by a mean vector $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$, it can be characterized by some functions over $x \in S$ for any set S. The mean function is denoted by $m(x)$ while the covariance function, often referred to as the kernel, is denoted by $K(x,\, x')$. The distribution of the process can then be written as in equation (2.20).

$$f(x) \sim \mathcal{GP}\left(m(x),\, K(x, x')\right) \tag{2.20}$$

When evaluating the gaussian process at a limited set of input locations $x_1, ..., x_n \in S$ the gaussian process boils down to a multivariate gaussian with the covariance defined as $\boldsymbol{\Sigma} = \{\Sigma_{ij}\}$ for $i, j \in [1, ..., n]$ and with each component given by $\Sigma_{ij} = K(x_i, x_j)$. This is useful since one often has a set of input and output pairs $D_n = (X_n, Y_n)$, and one then wishes to find out what function could have generated those observed values. This means that we are looking for the conditional distribution $Y(x)|D_n$ with $Y(x) \in \mathcal{GP}$.

To find equations for this conditional distribution we can use the theory presented in section 2.3.2. Similarly as before, we divide the input vectors and matrices to arrive at the following system.

$$\begin{bmatrix} Y(x) \\ \boldsymbol{Y}_n \end{bmatrix}, \text{ with sizes } \begin{bmatrix} 1 \times 1 \\ n \times 1 \end{bmatrix} \tag{2.21a}$$

$$\begin{bmatrix} \Sigma(x,x) & \boldsymbol{\Sigma}(x, \boldsymbol{X}_n) \\ \boldsymbol{\Sigma}(\boldsymbol{X}_n, x) & \boldsymbol{\Sigma}_n \end{bmatrix}, \text{ with sizes } \begin{bmatrix} 1 \times 1 & 1 \times n \\ n \times 1 & n \times n \end{bmatrix} \tag{2.21b}$$

Furthermore a zero mean is assumed for the distribution of Y, as such a transformation can always be done. Then, plugging our system into the equations as before one arrives at the following parameters

$$\mu(x) = \boldsymbol{\Sigma}(x, X_n)\boldsymbol{\Sigma}_n^{-1}\boldsymbol{Y}_n \tag{2.22a}$$

$$\sigma^2(x) = \Sigma(x,x) - \boldsymbol{\Sigma}(x, \boldsymbol{X}_n)\boldsymbol{\Sigma}_n^{-1}\boldsymbol{\Sigma}(\boldsymbol{X}_n, x) \tag{2.22b}$$

These equations can also be expanded to many predictive locations at once. For that the location x can simply be replaced with the vector of locations $\boldsymbol{X} \in \mathbb{R}^n$. This produces better results than

doing it individually for each point, as the covariance matrices give a more complete picture. Using this prediction strategy to predict the underlying function produces distribution quantiles which are close near the points in D_n, while the uncertainty increases further away from the observed points. However, the uncertainty does not continue to increase indefinitely, but will eventually be bounded. The quantiles are illustrated by figure 2.6.

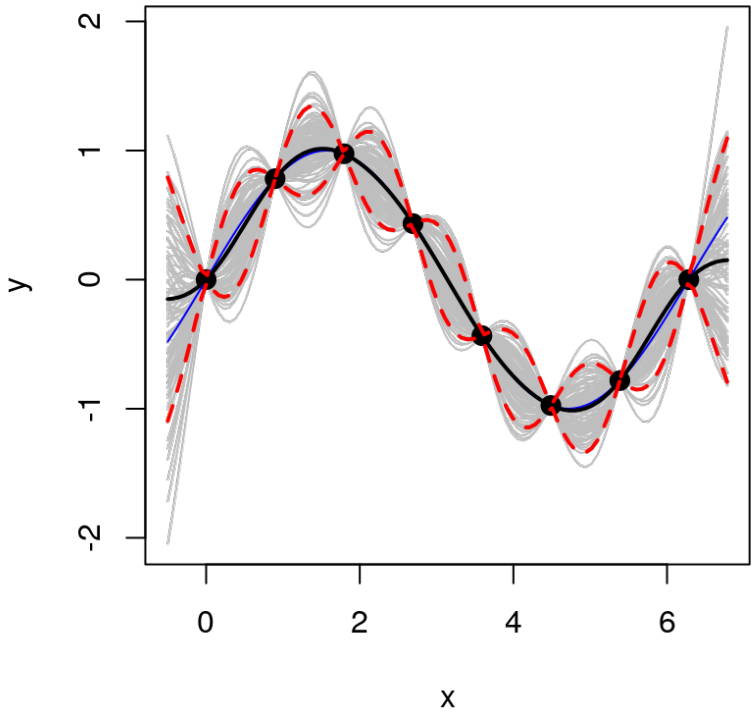


Figure 2.6: The posterior predictive distribution. The black bar shows the predictive mean, the blue line shows the true function from which the values are taken. the red dotted line show the 90% quantiles and the gray lines are a set of random predictions. Taken from `https://bookdown.org/rbg/surrogates/chap5.html#chap5hyper`.

Designing the Kernel

The kernel $K(x_i, x_j)$ can be thought of as specifying the similarity between the values of the underlying function for the given points. In this way, it determines the properties of the model. one particular kernel, the squared exponential kernel, given by equation (2.23), is used to model functions which are very smooth. The parameter σ_k^2 specifies the signal variance. This says something about how much the values deviate from their mean value. The α parameter describes the smoothness of the signal. A large α means that the functions value can change quickly, while a small one means the function changes slowly and is smoother. This also implies that a small α

means that values can be extrapolated accurately further away from the training data, because of the smoothness of the data.

$$K(x_i, x_j) = \sigma_k e^{\left(-\alpha \|x_i - x_j\|^2\right)} \tag{2.23}$$

2.4 Spatio-Temporal Gaussian Regression

Spatio-temporal gaussian regression is a type of gaussian regression where the state variable x consists of both a spatial vector $s \in \mathbb{R}^n$ as well as time $t \in \mathbb{R}_+$. It is possible to solve this by viewing time as just another input feature. However, the resulting method is memory and computationally intensive, and is non-iterative in nature. This means that a solution must be found over the whole of the problem space, often implemented with batches of time. This does not lend itself to real-time applications. Solutions that speed up the computations are often based on truncated observations, therefore losing accuracy. Such methods include sparse approximations [Williams and Rasmussen, 2006] and finite memory implementations [Xu et al., 2011]. An alternative approach, which is the focus of the rest of this section, is suggested by Todescato et al. [2020]. A state-space representation is found for the Gaussian process so that it can be combined with a Kalman filter.

2.4.1 System Model and Assumptions

The algorithm suggested by Todescato et al. [2020] is exact in the sense that the exact minimum variance estimate is found for any prediction location. Furthermore, the restrictions placed on the kernel are mild, allowing it to be used on many systems. The key step to achieve the method lies in the representation of a type of processes as a state-space model. Given a stationary random process $f(t)$ with covariance $h(\tau)$, $\tau = t - t'$, the power spectral density (PSD) is given by $S_r(\omega) = W(\boldsymbol{i}\omega)W(-\boldsymbol{i}\omega)$, where

$$W(\boldsymbol{i}\omega) = \frac{b_{r-1}(\boldsymbol{i}\omega)^{r-1} + b_{r-2}(\boldsymbol{i}\omega)^{r-2} + \ldots + b_0}{(\boldsymbol{i}\omega)^r + a_{r-1}(\boldsymbol{i}\omega)^{r-1} + \ldots + a_0} \tag{2.24}$$

where $\boldsymbol{i}$ is the imaginary unit. Rational functions of this form can be rewritten as a controllable canonical form state-space, given by

$$\dot{\boldsymbol{s}}(t) = F\boldsymbol{s}(t) + G\boldsymbol{w}(t) \tag{2.25a}$$
$$\boldsymbol{z}(t) = H\boldsymbol{s}(t) \tag{2.25b}$$

with matrices

$$F = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ & & & \ddots & \\ 0 & 0 & 0 & \dots & 1 \\ -a_0 & -a_1 & -a_2 & \dots & -a_{r-1} \end{bmatrix}, \quad G = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$
$$H = \begin{bmatrix} b_0 & b_1 & b_2 & \dots & b_{r-1} \end{bmatrix}$$

Noise vector $\boldsymbol{w}(t) \sim \mathcal{N}(0,\, I)$ and initial state $\boldsymbol{s}(0) \sim \mathcal{N}(0,\, \Sigma_0)$. Where Σ_0 is the solution of the Lyapunov equation given by $FX + XF^T = -GG^T$.

The problem to be solved is as following. One has noisy measurements of the form given by equation (2.26), where $\boldsymbol{v_i}$ defines the measurement noise. In general, the discrete-time instants t_k may be non-uniformly distributed. The true function operates on the spatial set $\mathcal{X}$. The spatial measurement locations are defined as the subset $\mathcal{I} \subseteq \mathcal{X}$. Not all spatial locations need to be present at all time instants t_k, meaning that the measurements may, in general, come from a time-varying set of input locations; $x_i \in \mathcal{M}(k) \subseteq \mathcal{I}$. The different spatial sets are visualized in figure 2.7.

$$y_i(t_k) = f(x_i, t_k) + v_i(t_k), \qquad \text{where } v_i(t_k) \sim \mathcal{N}(0,\, \sigma^2) \tag{2.26}$$

The process $f(x,t)$ is assumed to have a kernel function K which is stationary in time, and separable in time and space. This means that the kernel is given by

$$K(x, x', t, t') = K_s(x, x')h(\tau), \qquad \tau = t - t'$$

Furthermore, the power spectral density is assumed to be decomposable as described above, with $W(\boldsymbol{i}\omega)$ given by equation (2.24).

The paper suggests a solution which can be divided into two steps. Estimating the process $\boldsymbol{f}$ over the possible input locations $\mathcal{I}$, and secondly showing how this can be extended to an arbitrary spatio-temporal location $\boldsymbol{x} \in \mathcal{X}$, $t \in \mathbb{R}_+$. The first step is summarized in the following section.

2.4.2 Kalman Regression on the Measurement Locations

To implement the Kalman filter equations 2.13 and 2.14, the process over $\mathcal{I}$ is defined by the vector

$$\boldsymbol{f}(t) = \begin{bmatrix} f(x_1, t), \dots, f(x_M, t) \end{bmatrix}^T$$

The equation for each $f(x_i, t)$ is given by equation (2.25). Then, the complete system can be defined by using the Kronecker product, denoted by $\otimes$. The kronecker product is defined by

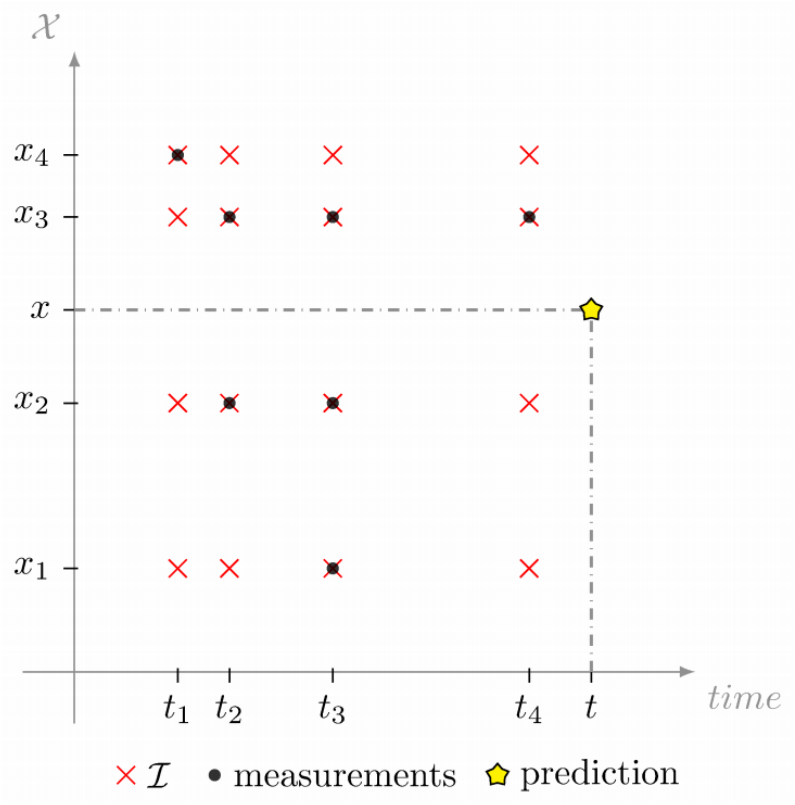


Figure 2.7: The spatio-temporal problem space. The x-axis represents discrete time instants, the y-axis is the $\mathcal{X}$ Domain. The red crosses are the possible measurement locations in $\mathcal{I}$, while the black circles are the actual measurement locations $\mathcal{M}(k)$. The star represents any spatio-temporal prediction location. Figure by Todescato et al. [2020].

equation (2.27). The individual systems can be combined together by creating block diagonal matrices where each block consists of the corresponding matrix. The continuous-time system is then defined by equation (2.28). $\bar{K}_s$ is the spatial kernel K_s over the space $\mathcal{I}$, meaning $[\bar{K}_s]_{ij} = K_s(x_i, x_j)$, $x_i, x_j \in \mathcal{I}$. Defining the size of each s_i as r, and given M measurement locations in $\mathcal{I}$, one has that $\boldsymbol{s} \in \mathbb{R}^{rM}$.

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \dots & a_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & \dots & a_{mn}\boldsymbol{B} \end{bmatrix} \tag{2.27}$$

$$\dot{\boldsymbol{s}}(t) = (I \otimes F)\boldsymbol{s}(t) + (I \otimes G)\boldsymbol{w}(t) \tag{2.28a}$$
$$\boldsymbol{f}(t) = \bar{K}_s^{1/2}(I \otimes H)\boldsymbol{s}(t) \tag{2.28b}$$

Since the Kalman equations are given in discrete time, system 2.28 must be discretized to the standard discrete form

$$\boldsymbol{s}(k+1) = A(k)\boldsymbol{s}(k) + \boldsymbol{w}(k) \tag{2.29a}$$
$$\boldsymbol{y}(k) = C(k)\boldsymbol{s}(k) + \boldsymbol{v}(k) \tag{2.29b}$$

By using equations 2.30, given that the times between measurements is given by $T_k = t_k - t_{k-1}$.

$$A(k) = e^{(I\otimes F)T_k} \in \mathbb{R}^{rM\times rM} \tag{2.30a}$$
$$C(k) = I_k \bar{K}_s^{1/2}(I \otimes H) \tag{2.30b}$$

Where $I_k \in \mathbb{R}^{M_k \times M}$ is a selection matrix which selects which input locations are available at time t_k. M_k is the number of available measurements at time k. This results in a $\boldsymbol{y}$ vector which varies in size according to how many input locations are available. The noises are defined by

$$\boldsymbol{w} \sim \mathcal{N}(0,\, Q(k)) \in \mathbb{R}^{rM} \tag{2.31a}$$
$$\boldsymbol{v} \sim \mathcal{N}(0,\, R(k)) \in \mathbb{R}^{M_k}, \qquad \text{where} \tag{2.31b}$$
$$Q(k) = I \otimes \bar{Q}(k), \tag{2.31c}$$
$$\bar{Q}(k) = \int_0^{T_k} (e^{F\tau})GG^T(e^{F\tau})^T d\tau \tag{2.31d}$$
$$R(k) = \sigma^2 I \in \mathbb{R}^{M_k \times M_k} \tag{2.31e}$$

Given that the sampling has uniform timesteps and a constant set of locations, the matrices $A(k)$, $C(k)$, $Q(k)$ and $R(k)$ become constant. Then, the Kalman filter gain $L(k)$ converges to a constant value which can be pre-computed. This significantly reduces the computational burden.

Given that the above equations hold, the estimate $\hat{\boldsymbol{f}}(t)$ of the process $\boldsymbol{f}(t)$ as well as the error covariance $\Sigma^{\boldsymbol{f}}(t)$ for $t \geq t_k$ are found using algorithm 1. As shown, the algorithm has the regular prediction and update stages from the Kalman filter. However, there is an additional state regression

for times between measurements, and a continuous-time process estimation stage after each update.

Algorithm 1: Kalman Regression on $\mathcal{I}$. Restated from Todescato et al. [2020], algorithm 1.

Require: (F, G, H) state-space representation for $S_r(\omega)$. σ^2 measurement noise variance. Input location space $\mathcal{I}$. Spatial kernel $K_s(x, x')$ and temporal kernel $h(\tau)$. Σ_0 solution of lyapunov equation $FX + XF^T = -GG^T$.

Initialize $\hat{s}(0 \mid -1) = 0$, $\Sigma(0 \mid -1) = I \otimes \Sigma_0$.
Store the variable $\hat{s}(t)$, where $\hat{s}(0) = 0$.

for $t \in \mathbb{R}_+$ **do**
  `// State Regression`
  if $t \in [t_k, t_{k+1}]$ **then**
    `// Open-loop prediction between measurements`
    $\hat{\boldsymbol{s}}(t) = e^{(I \otimes F)\tau} \hat{\boldsymbol{s}}(k \mid k), \qquad \tau = t - t_k$
    $\Sigma^{\boldsymbol{s}}(t) = \left(e^{(I \otimes F)\tau}\right) \Sigma(k \mid k) \left(e^{(I \otimes F)\tau}\right)^T$
  else if $t = t_{k+1}$ **then**
    `// Kalman Estimation`
    Compute $A(k)$, $C(k)$, $Q(k)$, $R(k)$ using equation (2.30)

    `// Prediction-stage`
    $\hat{\boldsymbol{s}}(k+1 \mid k) = A(k)\hat{\boldsymbol{s}}(k \mid k)$
    $\Sigma(k+1 \mid k) = A(k)\Sigma(k \mid k)A(k)^T + Q(k)$

    `// Update-stage`
    $L(k+1) = \Sigma(k+1 \mid k)C(k+1)^T \left(C(k+1)\Sigma(k+1 \mid k)C(k+1)^T + R(k+1)\right)^{-1}$
    $\hat{\boldsymbol{s}}(k+1 \mid k+1) = \hat{\boldsymbol{s}}(k+1 \mid k) + L(k+1)\left(\boldsymbol{y}(k+1) - C(k+1)\hat{\boldsymbol{s}}(k+1 \mid k)\right)$
    $\Sigma(k+1 \mid k+1) = \left(I - L(k+1)C(k+1)\right)\Sigma(k+1 \mid k)$

    `// Update continuous-time estimates`
    $\hat{\boldsymbol{s}}(t) = \hat{\boldsymbol{s}}(k+1 \mid k+1)$
    $\Sigma^{\boldsymbol{s}}(t) = \Sigma(k+1 \mid k+1)$

  `// Process Estimate`
  $\hat{\boldsymbol{f}}(t) = \bar{K}_s^{1/2}(I \otimes H)\hat{\boldsymbol{s}}(t)$
  $\Sigma^{\boldsymbol{f}}(t) = \bar{K}_s^{1/2}(I \otimes H)\Sigma^{\boldsymbol{s}}(t)(I \otimes H)^T \bar{K}_s^{1/2}$

2.5 Pelvic Floor Muscles

The pelvic floor is a layer of muscles that spans the bottom of the pelvis. Its purpose is to support the pelvic organs, mainly the bladder and bowel, but also the uterus in women. Normally, these muscles are firm, thick and able to stretch. The pelvic floor muscles have three passages. One for the urethra, one for the vagina and one for the anus [Herschorn, 2004]. An illustration of the pelvic floor muscles and the pelvic floor is shown in figure 2.8.

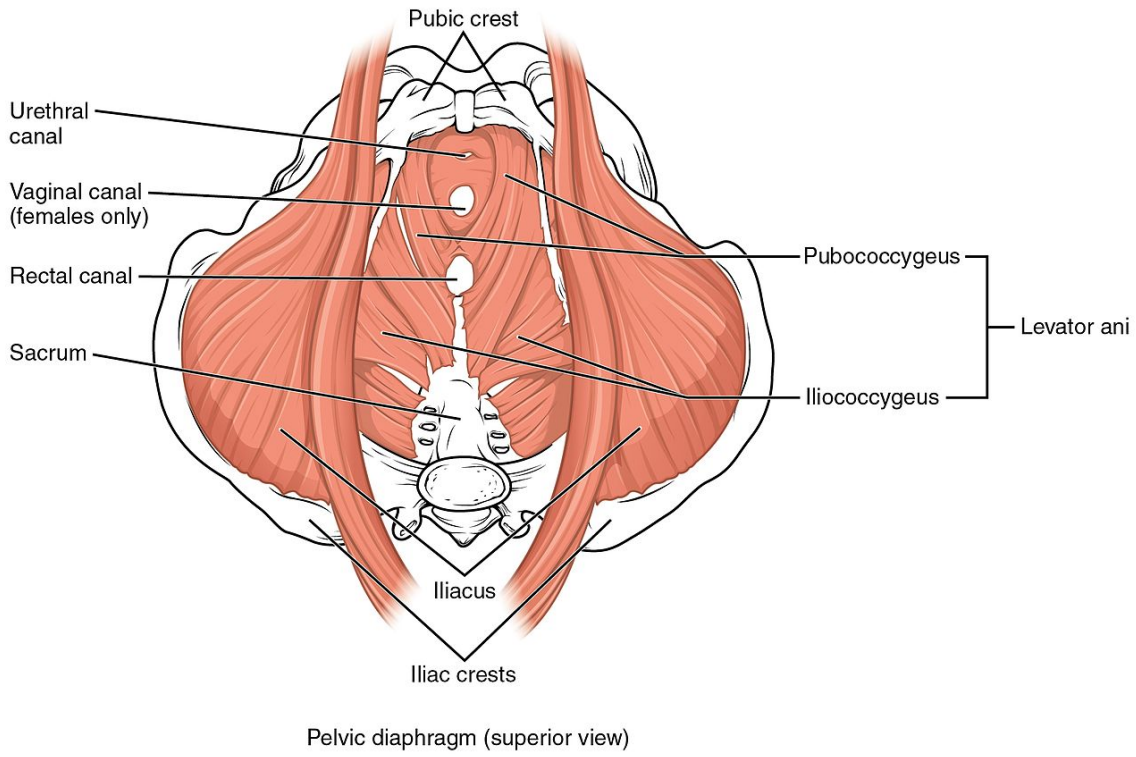


Figure 2.8: Illustration of the pelvic floor and the pelvic floor muscles. Taken from Betts et al. [2013].

The pelvic floor muscles are essential for providing support for the pelvic organs. They also provide conscious control over the bladder and bowel. Therefore, weakening of these muscles may result in urinary and faecal incontinence. Incontinence refers to the involuntary voiding of urine and faeces through the urethra and rectum respectively. The muscles are also key during pregnancy, where they provide support for the fetus, and need to be relaxed while giving birth. Furthermore, the PFM plays a key role in sexual sensation and arousal. Pleasure is derived from contracting the pelvic floor, and decreased muscular elasticity and tonicity hinder the capability of the vaginal duct to be lubricated and the clitoral system to be vasocongested. Vasocongested refers to vascular blood flow

which causes the bodily tissue to swell. Tonicity describes the osmotic pressure gradient, or how well fluids can pass through. These factors can lead to pain during penetrative sexual intercourse, also called dyspareunia.

Pelvic floor dysfunction is a condition where one is unable to correctly relax and control the pelvic floor muscles. Common causes of this condition include pregnancy, traumatic injuries, overuse of the muscles, being overweight and advancing age. The pelvic floor muscles can become strained during pregnancy, especially if labor was difficult. Urinary incontinence is estimated to affect about every third women [Walker and Gunasekera, 2011], and pelvic floor prolapses might occur in more serious cases.

2.5.1 Treatment of Weakened Pelvic Floor Muscles

Pelvic floor exercises are most commonly known as Kegel exercises. To do them, one repeatedly contracts and relaxes the muscles in the pelvic floor. Often, one tightens the muscles for 3-5 seconds at a time, and then relaxes them for the same amount of time before repeating. These exercises may be done daily or multiple times a day, but it may take several months for the results to become apparent [Tian et al., 2018].

The main issues with Kegel exercises is the lack of feedback and the time it takes to see results. A study of women that had recently given birth by Chiarelli et al. [2003] found that as much as 1 of 3 women were not confident in their ability to do the exercise correctly. A possible reason for this is that it may be hard to identify if one is using the correct muscles. Using abdominal and hip muscles is a common mistake. Additionally, because results may take months to become apparent, many women do not exercise enough, or quit entirely. This comes down to both the lack of confidence in if they are performing the exercise correctly, as well as social taboos about sexual health [Kask et al., 2019]. This was also noted by Chiarelli et al. [2003], where 91% intended to continue performing Kegel exercises at least every second day after birth. However, only 58% of women actually did them often enough three months after giving birth.

An approach to improve the results of Kegel exercises is biofeedback. Biofeedback aims to reduce the time between the exercise and feedback on how the exercise is performed. Thereby training the patient to have greater control of their physical response. The most basic tool for biofeedback with Kegel exercises is a perineometer, developed by Dr. Kegel. It assisted women by measuring the vaginal air pressure. More modern methods often employ electromyograph (EMG) perineometers. These measure electrical activity, and the measurements may be visualized on a screen. Kask et al. [2019] proposes gamification as a biofeedback tool to encourage regular exercising, and have access to biofeedback even when doing the exercises at home.

2.5.2 The Importance of Spatio-Temporal Filtering

A crucial part to biofeedback is to be able to interpret the measurements and estimate the relevant quantities. For Kegel exercises, there are some special considerations that must be taken. Firstly, the pressure profile measured in the vaginal duct is very different when the person does the exercise

while standing or while lying down [Knorn et al., 2020]. When relaxed and lying down, there is no clear muscular activation. When standing however, there is a broad peak, with the maximum pressure zone located approximately where the pelvic floor muscles lie. When pressing, a narrow peak is observed in both cases. However, the maximum pressure is much larger when standing, because of the effects of the abdominal muscles (that are involuntarily involved in the body balancing mechanisms). Because of these differences, it is essential to first identify whether the person is lying down or standing, before further interpreting the measurements. Additionally, Knorn et al. [2020] showed that an estimator for one person does not generalize well to other people. Therefore, there is a need for individualized muscular pressure models. These models should include the spatial features mentioned above (e.g., the location, extension and shape of the muscular pressure field in the spatial dimension). However, when exercising over time, the muscles will fatigue and the detection of these features may become more or less prominent. Over time, one would expect that, with training, the patients improve both their proprioception capabilities (i.e., the ability to control their musculature) and their overall strength. Therefore, one can expect that when Kegel exercises are performed by a well-trained person one will measure narrower and taller muscular pressure peaks (from the spatial dimension) and shorter transients in the temporal one. Therefore it in general important, when considering muscular fitness models, to consider spatio-temporal modelling and filtering.

Chapter 3

Methodology

This chapter describes how the estimation method was found. Firstly, the dataset is described, then the input estimation is explained. The third section goes through the system identification to find an ARX model, and finally the spatio-temporal regression using a Kalman filter is explained. The following is implemented using MATLAB and the System Identification Toolbox [Ljung, 2012].

3.1 Dataset

The dataset used for this thesis is collected by the FemFit vaginal pressure sensor shown in figure 3.1. The FemFit is a wirelessly connectable, wearable vaginal pressure sensor. It is developed by the Auckland Biomedical Institute (ABI) in New Zealand. The device consists of 8 pressure sensors, giving it the capability to measure and transmit the vaginal pressure at 8 different locations within the vaginal duct in real time. The dataset consist of a compilation of measurements from different sessions with the device, resulting in approximately one hour of measurements. Since it is compiled from different sessions, this introduces some unwanted variations. Such variations stem from differences in how the device is inserted each time, possible deterioration of the sensors and differences in the fatigue levels of the participant. Therefore, a subset of the dataset it selected where the signals characteristics are consistent.

The selected dataset for system identification is shown in figure 3.2. As can be seen from the figure, the entire dataset is de-trended so that the bodies resting pressure is set to 0kPa. As shown in the figure, the data is captured over 4 minutes. The spatio-temporal gaussian regression uses the 4 minutes following the system identification dataset, with very similar characteristics. A sampling interval of $T_k = 0.12$ seconds was used. These slices were selected because of the consistency of the peak height, as well as the shape of the pressure. Figure 3.3 shows a zoomed in view of some of the peaks in pressure, which more clearly illustrates the shape of the pressure response. When the

Figure 3.1: The vaginal pressure sensur, called FemFit, used to collect the dataset.

participant tightens their muscles, it leads to a sharp increase in pressure, which then gradually decays. The selected dataset for system identification is further divided into a training set and a validation set, using a 50% split. One thing to note is that one of the endpoint sensors, sensor 1, was faulty during the entire session, outputting only zero.

3.2 Input Estimation

To create a model of the system which can react to inputs – which can't be measured and occur at unknown times – the inputs must be estimated from the signal itself. Creating an accurate estimate of the brain signals which cause the tightening of muscles is outside of the scope of this thesis. Instead, the assumption is made that the brain signal is binary. Either, the patient tightens their muscles, or they are relaxed. Such an input signal can be estimated using change detection algorithms. Instead of stopping when the signal is far enough from the equilibrium, the algorithm continues to set the input to one, until the moving average goes below the detection threshold.

Independent of the specific change detection algorithm used, this results in a relation between output and input that looks similar to figure 3.4. Observe that the estimated signal is shorter than the pulse in pressure. This is because the methods react to a large enough difference between equilibrium and the current value, therefore the initial escalation and final de-escalation are not captured. The implementation used to create the figure is based on moving averages. This means that the signal value used to estimate the input is a weighted average of the current and previous measurements. This accounts for the additional delay observed in the image. Clearly, this input-output relationship is a poor model for a physical system, as the response occurs before the estimated input. Therefore, additional work must be done to find the optimal number of measurements to shift the input back in time. Note that this introduces a delay in the identification, meaning there is an added minimal delay introduced for output prediction.

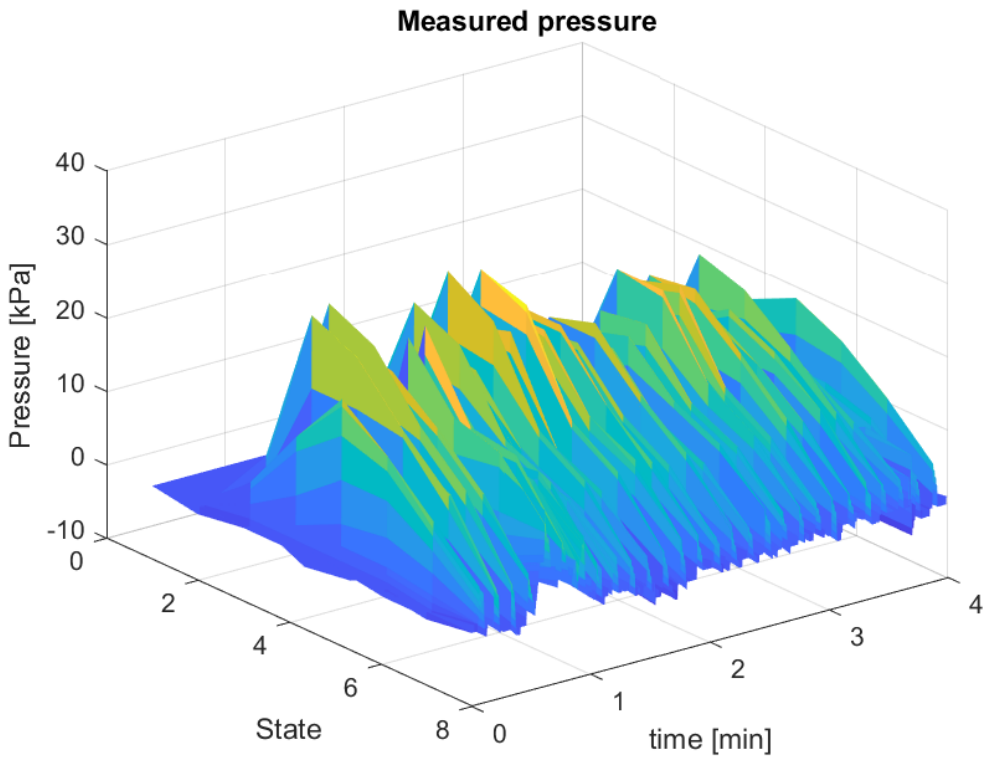


Figure 3.2: Measured pressure surface of the chosen subset of data.

Both the GMA and FMA algorithm are considered for input estimation. The GMA algorithm has a theoretical infinite memory, although the exponential decrease in weights makes the long-term effect miniscule. The FMA has a finite window, but adds complexity in terms of designing the weights. Because of the relative stability of the resting pressure, a low threshold is chosen. This ensures as small of a detection delay as possible, as well as capturing most of the signal. The threshold is set to 10% of the maximum average signal value detected. The resting pressure is known to be zero since the dataset has been de-trended, therefore μ_0 can be set to zero. For the design of the weights of the FMA, they are chosen so that they sum to one. This ensures that the size of the threshold can be scaled relative to the average pressure. Evenly spaced weights, decreasing in time, were found to work well.

3.3 Model Estimation

To estimate models for each pressure sensor individually, the observation is made that each set of measurements has the same general shape, except for the heights of the peaks. This means that a

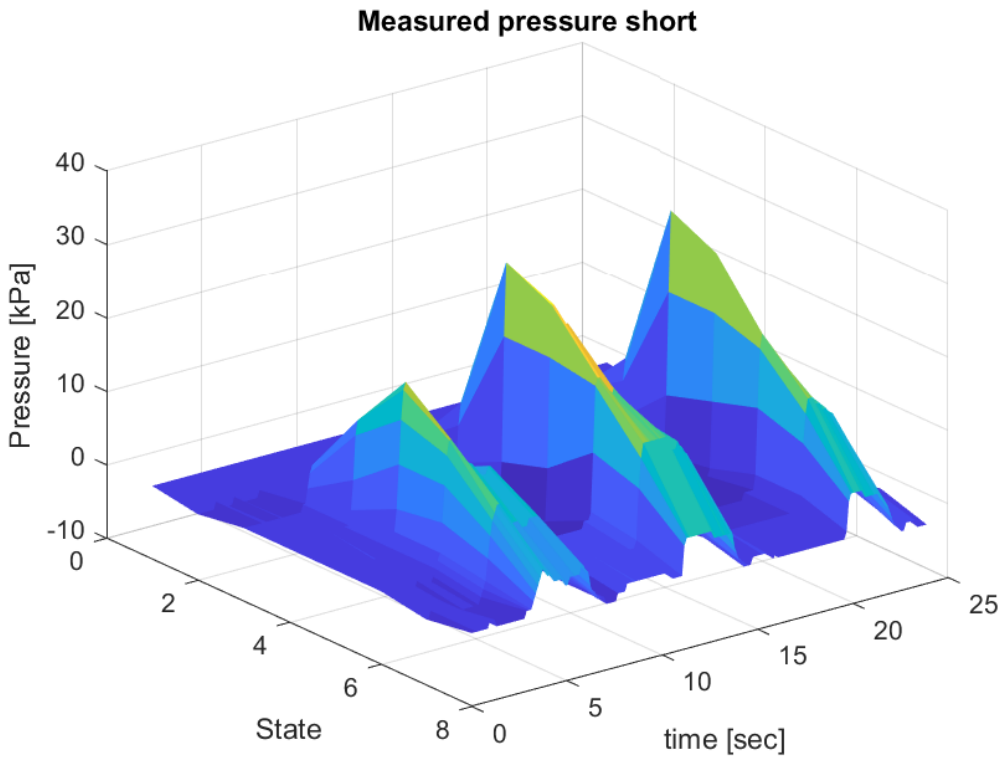


Figure 3.3: Some peaks of the Measured pressure surface.

common model can be found, except for a scaling factor on the input. Because of this observation, the model estimation can be divided into two steps:

1. Estimate a representative model from the signals with the most prominent shape. These signals are referred to as the representative set $\mathcal{T}$.
2. Modify this representative model with a multiplication factor for each remaining sensor: $\beta_i, \quad i \in 1, \ldots, 8 \setminus \mathcal{T}$

The best signals to use for the initial model estimation are selected by choosing those with the most prominent peaks. These signals are appended, and the time-shifted input is repeated to match the number of signals. The model structure chosen is the ARX model. To estimate the optimal arx model in Matlab, given a model order, the command `arx(trainingData, modelOrder)` is used. `trainingData` simply contains both the appendend outputs and inputs. The `modelOrder` parameter is 3-dimensional and defines the number of poles, zeros, and the dead time.

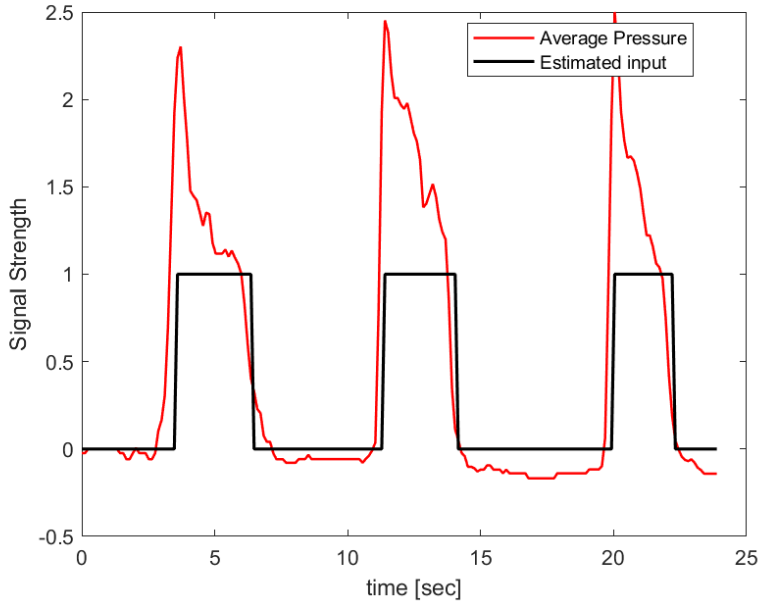


Figure 3.4: Relation between estimated input and output when using change detection algorithms based on moving averages.

3.3.1 Finding the Optimal Model Parameters

In the sections above, several sets of parameters are mentioned which need to be optimized. The optimization goal is to minimize the difference between the true measurement and the simulated values for each sensor. The simulated values are chosen for evaluation, as they more clearly highlight issues such as drift with the system as opposed to k-step ahead predictions. Clearly, the value of the time shift Δ will affect the coefficients of the ARX model, $\boldsymbol{\theta}$. These will both in turn have an effect on the optimal vector $\boldsymbol{\beta}$. Therefore, it is beneficial to formulate this as a joint optimization. As a cost function, the mean squared error, equation (3.1), is used. M represents the number of states, while N is the number of time steps.

$$J(\Delta, \boldsymbol{\theta}, \boldsymbol{\beta}) = \sum_{i=0}^{N} \sum_{j=0}^{M} (y_{measured,ij} - y_{sim,ij})^2 \tag{3.1}$$

Notice that the time shift Δ must be an integer, as it represents the number of sample times to shift the signal. Meanwhile, both $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ contain continuous values. Therefore, the optimization problem is a mixed integer programming problem. This adds complexity, as gradient-based methods cannot be used directly. The problem can be circumvented by using prior knowledge about the optimization problem. Logically, the sharp peaks of the signal mean that only a small shift is necessary for the input. When constraining the time shift to be in a limited range, such as $\Delta \in$

$[0, 15]$, it is possible to solve a continuous problem for each Δ. Although this solution seems sub-optimal, it actually performs well, since solutions for a continuous problem are found using more efficient algorithms.

Furthermore, a penalty must be placed on large time shifts Δ in the cost function. This is because the extra margin before a reaction in the output can be negated by a larger n_k value of the ARX model. Therefore, when no penalty is placed, the cost will be equal if the time shift and n_k are both increased by the same number. However, since a large time shift introduces larger delays into the system, this is clearly an inferior solution. The modified cost function is then

$$J^*(\Delta, \boldsymbol{\theta}, \boldsymbol{\beta}) = J(\Delta, \boldsymbol{\theta}, \boldsymbol{\beta}) + \gamma\Delta^2 \tag{3.2}$$

where γ is a hyperparameter which must be tuned. Additional, the betas can also be constrained, since the other sensors shouldn't have much larger peaks than the chosen reference signals. This results in the optimization problem as formulated in equation (3.3).

$$\begin{aligned} \underset{\Delta, \boldsymbol{\theta}, \boldsymbol{\beta}}{\arg\min} \quad & J(\Delta, \boldsymbol{\theta}, \boldsymbol{\beta}) + \gamma\Delta^2 && \text{(3.3a)} \\ \text{Subject to:} \quad & \Delta \in [0, 1, \ldots, 15] && \text{(3.3b)} \\ & \boldsymbol{\beta} \in [0, 1.5] && \text{(3.3c)} \end{aligned}$$

3.3.2 Model Order Selection

As previously stated in section 2.1.2 determining the best model order is often a subjective assessment. More rigorous methods exist, such as the Akaike information criterion, however they more often serve as guidelines than an absolute rule. For this use case, it is hard to find a likelihood function for the entire model as it consists of optimization of more than just a polynomial model. Therefore, the assessment is mostly based on weighing the complexity against the fit of the predictions on the validation set. Furthermore, the AIC score for the representative model against the representative signals can be evaluated. However, this value is not necessarily a good indication of the overall performance. The above evaluations result in the choice of an ARX model with order [2, 1, 1], as explained in chapter 4.

3.4 Spatio-Temporal Gaussian Regression using Kalman Filters

This section goes through the steps required to transform the system found in the previous section to a state-space model for use with a Kalman filter, and then looks at how algorithm 1 can be modified for use with a discrete-time system. Note that the systems considered are all single-input single-output (SISO) systems.

3.4.1 ARX to State Space for SISO Systems

In general, the realization of a linear time invariant system in its state space form starting from its transfer function representation is immediate. In other words, starting from a generic ARMA representation

$$y(k) = \frac{b_1 q^{-1} + \ldots + b_n q^{-n}}{1 + a_1 q^{-1} + \ldots + a_n q^{-n}} u(k) \tag{3.4}$$

one can immediately transform such ARMA representations into the equivalent input-output relation

$$\begin{cases} \boldsymbol{x}(k+1) & = A\boldsymbol{x}(k) + Bu(k) \\ y(k) & = C\boldsymbol{x}(k) + Du(k) \end{cases} \tag{3.5}$$

where the matrices can be given as the controllable canonical form matrices:

$$A = \begin{bmatrix} -a_1 & -a_2 & \ldots & \ldots & -a_n \\ 1 & 0 & \ldots & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ & \ddots & \ddots & \ddots & \\ & & 0 & 1 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad C = \begin{bmatrix} b_1 & b_2 & \ldots & b_n \end{bmatrix} \qquad D = 0 \tag{3.6}$$

Or, equivalently, by their observable canonical form as

$$A = \begin{bmatrix} -a_1 & -a_2 & \ldots & \ldots & -a_n \\ 1 & 0 & \ldots & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ & \ddots & \ddots & \ddots & \\ & & 0 & 1 & 0 \end{bmatrix}^T \qquad B = \begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & \ldots & 0 \end{bmatrix} \qquad D = 0 \tag{3.7}$$

Note that the presence of the matrix D in equation (3.6) is connected to the presence of a direct term in the transfer function (3.4). More precisely, if that transfer function is not strictly proper, and thus expressable as

$$y(k) = \frac{b'_0 + b'_1 q^{-1} + \ldots + b'_n q^{-n}}{1 + a_1 q^{-1} + \ldots + a_n q^{-n}} u(k) \tag{3.8}$$

then it would be immediately rewritable as

$$y(k) = d_0 u(k) + \frac{b_1 q^{-1} + \ldots + b_n q^{-n}}{1 + a_1 q^{-1} + \ldots + a_n q^{-n}} u(k) \tag{3.9}$$

and in this way D in (3.6) and (3.7) would in both cases become $D = d_0$.

The aim is then to extend the ARMA to state space representation procedure above, to be able to convert SISO ARMAX models of the form

$$y(k) = \frac{b_1 q^{-1} + \ldots + b_n q^{-n}}{1 + a_1 q^{-1} + \ldots + a_n q^{-n}} u(k) + \frac{1 + c_1 q^{-1} + \ldots + c_n q^{-n}}{1 + a_1 q^{-1} + \ldots + a_n q^{-n}} e(k) \tag{3.10}$$

into an equivalent state space form. Observe that $e(k)$ enters into play with its "own" d_0 term (equal to 1), and with delay factors $c_1, \ldots, c_n$. Note that these delay factors are zero in the case where the generic ARMAX model can be reduced to an ARX model.

Furthermore, assuming $\boldsymbol{x}(0) = \mathbf{0}$, and working in the z-transform domain, equation (3.5) can be written as

$$\begin{cases} qX &= AX + BU \\ Y &= CX + DU \end{cases}$$

and thus as

$$\begin{cases} (qI - A)X &= BU \\ Y &= CX + DU. \end{cases}$$

Using the observation canonical form, the matrices A and C are defined as in equation (3.7) and thus not dependent on the polynomial $1 + a_1 q^{-1} + \ldots + a_n q^{-n}$.

Considering that the system is linear, the superposition of the effects must hold. This means that the previous expression has to be expandable into

$$\begin{cases} (qI - A)X &= B_u U + B_e E \\ Y &= CX + D_u U + D_e E. \end{cases}$$

This eventually implies that the ARMAX expression (3.10) can be transformed into the equivalent observation-like state space representation given by

$$\boldsymbol{x}(k+1) = A\boldsymbol{x}(k) + B_u u(k) + B_e e(k) \tag{3.11a}$$
$$y(k) = C\boldsymbol{x}(k) + D_u u(k) + D_e e(k) \tag{3.11b}$$

With the matrices

$$A = \begin{bmatrix} -a_1 & -a_2 & \ldots & \ldots & -a_n \\ 1 & 0 & \ldots & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ & \ddots & \ddots & \ddots & \\ & & 0 & 1 & 0 \end{bmatrix}^T \qquad B_u = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \qquad B_e = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \tag{3.12a}$$

$$C = \begin{bmatrix} 1 & 0 & \ldots & 0 \end{bmatrix} \qquad D_u = 0 \qquad D_e = 1. \tag{3.12b}$$

3.4.2 Converting an ARX211 to State Space

Applying the above equations to the ARX211 for which we want a state-space equations is straightforward. Since the ARX model doesn't contain any terms $c_1, \ldots, c_n$ we have $B_e = 0$. The ARX model is given by

$$y(k) = \frac{b_1 q^{-1} + 0q^{-2}}{1 + a_1 q^{-1} + a_2 q^{-2}} u(k) + \frac{1}{1 + a_1 q^{-1} + a_2 q^{-2}} e(k)$$

and is then directly defined, when using the notation from section 2.4, by

$$\boldsymbol{x}(k+1) = F\boldsymbol{x}(k) + Bu(k) \tag{3.13a}$$
$$y(k) = Hx(k) + v(k) \tag{3.13b}$$

where the matrices are defined as

$$F = \begin{bmatrix} -a_1 & 1 \\ -a_2 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} b_1 \\ 0 \end{bmatrix}, \qquad H = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

The white noise $v(k)$ is then the measurement noise and has distribution $v(k) \sim \mathcal{N}(0, \sigma^2)$. Note that in the following form, all noise is considered as measurement noise.

Since the ARX model is already discrete, this method of finding the system dynamics gives a direct discrete representation for each state. This means that the discretization step defined by equation (2.30) and equation (2.31) don't need to be performed. Rather, stacking the matrices into block-diagonal matrices can be done directly in discrete time, as shown in equation (3.14). M denotes the number of states, in this case 8. the matrices $A(k)$, $B(k)$ and $C(k)$ can be seen directly from this equation. Note that no selection matrix is included in $C(k)$, since it is unlikely that data will be missing from sensors at any time. Furthermore, an input matrix is added to take the estimated input into consideration.

$$\boldsymbol{s}(k+1) = (I \otimes F)\boldsymbol{s}(k) + \begin{bmatrix} \beta_1 B \\ \beta_2 B \\ \vdots \\ \beta_M B \end{bmatrix} u(k) \tag{3.14a}$$
$$\boldsymbol{y}(k) = \bar{\boldsymbol{K}}_s^{1/2}(I \otimes H)\boldsymbol{s}(k) + \boldsymbol{v}(k) \tag{3.14b}$$

The noise vector is given by $\boldsymbol{v}(k) \sim \mathcal{N}(0, R(k))$ with $R(k) = \sigma_R^2 I \in \mathbb{R}^{M \times M}$. To use this model with the Kalman filter, a process noise term $\boldsymbol{w}(k) \sim \mathcal{N}(0, Q(k))$ is added back to the system. The assumption is made that this has a similar covariance, defined by $Q(k) = \sigma_Q^2 I_{16 \times 16}$

3.4.3 Changes to the Regression Algorithm

Using the system above as the starting point for the spatio-temporal gaussian regression requires some further alterations to section 2.1.1. Because of the discretization, Line 4 and 5 in of the

algorithm cannot be executed directly as they require information about the continuous time system. Hence, if one wanted to do time interpolation, one would have to find an equivalent continuous time model from the discretized one, and then discretize it again with the varying time step τ for each such time instant. For our purpose, this part of algorithm 1 is omitted. Assuming the matrices given above are used, the remaining algorithm is still valid with some changes. Line 15 and 16 become unnecessary, as we only work within the discrete measurement times. Therefore, these lines can be ignored. In the process estimate equations, line 16 and 17, the continuous time estimates $\hat{\boldsymbol{s}}(t)$ and $\Sigma^{\boldsymbol{s}}(t)$ may be replaced with the discrete $\hat{\boldsymbol{s}}(k+1 \mid k+1)$ and $\Sigma(k+1 \mid k+1)$ respectively. This results in the discrete time process estimates $\hat{\boldsymbol{f}}(k+1)$ and $\Sigma^{\boldsymbol{f}}(k+1)$. Since the discretized system does not contain a matrix G, the lyapunov equation for Σ_0, given in algorithm 1 is not valid. Rather, the initial value is chosen as $\gamma I_{2\times 2}$.

The tuning of the model is done by selection a kernel $K(x_i, x_j)$. For this application, the squared exponential kernel defined by equation (2.23) is used, to control how much smoothing to do in the spatial dimension. Given this kernel, the system has 4 tune-able parameters. The variance of the system noise σ_Q^2, the variance of the measurement noise σ_R^2, and the parameters of the kernel. The parameters of the kernel are given as σ_k^2, the signal variance, and α which defines the spatial smoothness. To simplify the evaluation $\sigma_k = 1$ is used.

Chapter 4

Results

This chapter presents the results obtained for each step of the process. First, the input estimation and system identification is presented. Secondly, the results of the spatio-temporal gaussian regression are shown.

4.1 Input Estimation

Input estimation was performed by both the finite moving average algorithm and the Geometric moving average algorithm which were described in section 2.1.1. For the finite moving average a window size of 10 was used, along with linear weights which sum to one. For the geometric moving average an α value of 0.3 was chosen. For both methods the threshold was set to 10% of the maximum value in the average pressure. The result for both methods over the chosen dataset, described by section 3.1, is shown in figure 4.1. In this zoomed out view the estimation methods seem to produce very similar results. There are no missed detections, and no obvious false positives. A zoomed in view of the first two peaks is shown in figure 4.2. This shows that there are some minor differences in their performance, but one does not stand out as significantly better than the other. For these examples, the difference between their edges is only a single time step of length $T_k = 0.12$. From the figure, the delay between change in the pressure and estimated input is only approximately 5 to 10 time steps.

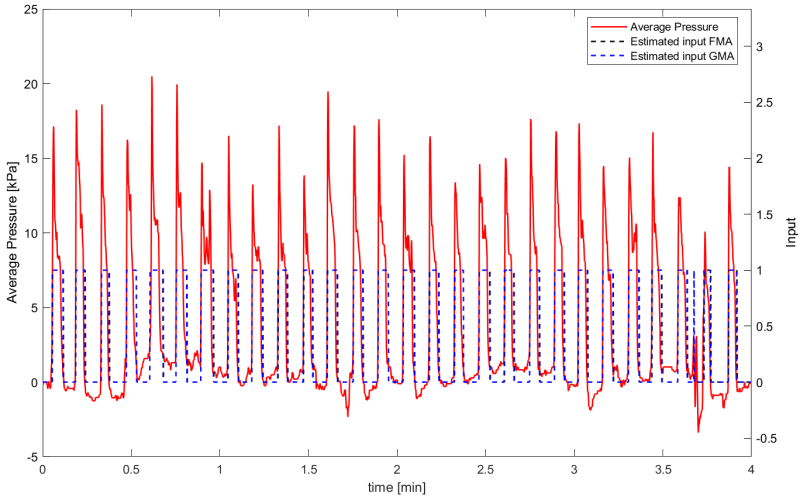


Figure 4.1: Input estimated by both FMA and GMA for the chosen dataset

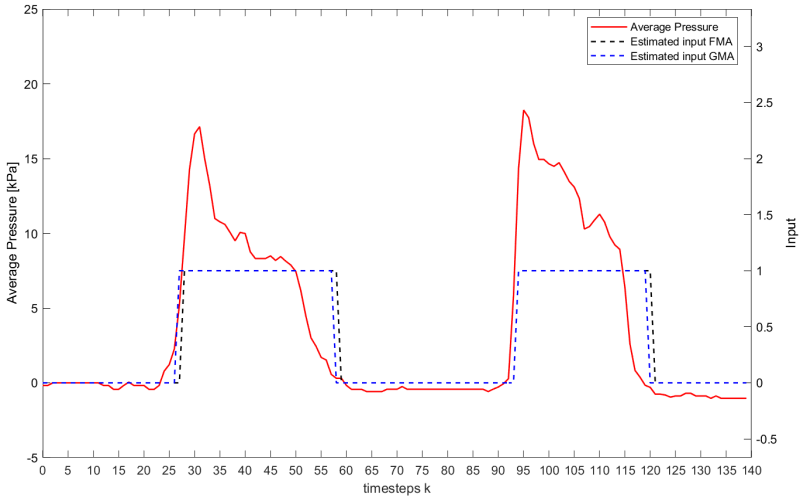


Figure 4.2: Input estimated by both FMA and GMA for the two first peaks of the dataset

Figure 4.3 shows both the finite moving average and the geometric moving average overlayed on the average pressure signal. The figure shows that they match the original signal quite well, but smooth out the peaks. The zoomed in view of the first two peaks in figure 4.4 shows that the moving averages work as expected. Since it is averaged out, there is a delay in escalation and

de-escalation. The peaks are also less steep. Note that the tuning of both algorithms is chosen so that the smoothing is not very severe.

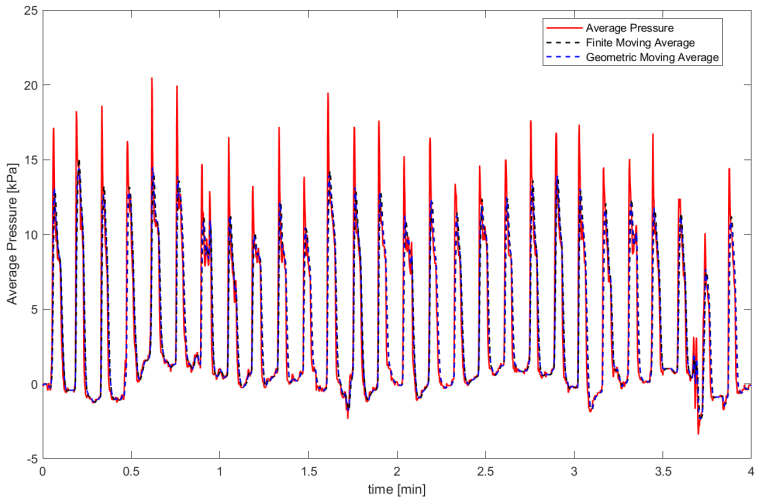


Figure 4.3: The signals produced by the FMA and GMA algorithms over the dataset

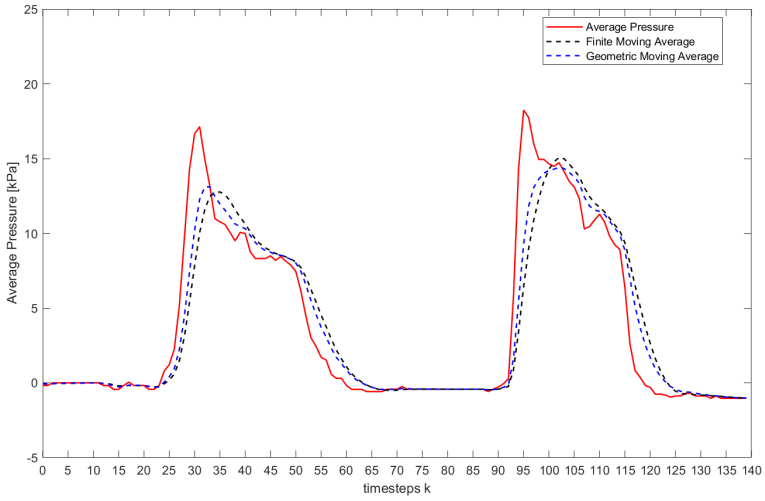


Figure 4.4: The signals produced by the FMA and GMA algorithms by the first two peaks in the dataset

4.2 System Identification

The following section compares the results found with the different model orders by looking at their simulation capabilities and by looking at the reasonableness of their parameters. From this, a model structure is chosen which is evaluated further.

4.2.1 Model Order Selection

The evaluated models are all ARX models with model orders given by $n_a \in [2, 3, \ldots, 5]$, $n_b \in [1, 2, \ldots, 4]$, $n_k \in [1, 2, \ldots, 5]$. This means that a total of $4 \times 4 \times 5 = 80$ different models have been evaluated. The evaluation was done with a regularization factor on the time-shifts with value $\gamma = 20$. The AIC scores described by section 3.3.2 are shown in figure 4.5. As shown, the AIC values vary between 6400 and 6900. There is no clear correlation between model complexity as a whole and AIC value, rather one can observe three separate clusters which depend on the value of n_a to a large degree. In general, models with $n_a = 2$ have the best AIC scores. Figure 4.6 Shows the fit percentage against model complexity. Both the fit percentages on the training set and on the validation set are shown. figure 4.6a shows a zoomed in view of the results, while figure 4.6b shows the whole y-axis from 0% to 100%. In general, the model performs better on the validation data than on the training data. By looking at the signal characteristics this does however seem reasonable as the different sensors have more similar characteristics in the validation data. The training and validation sets can be seen in appendix A. When selecting a model, most trust should be placed in the performance on the validation data as this is not affected by overfitting. figure 4.6b shows that the models perform very similarly as a whole, only varying by less than five percent. Therefore, it is reasonable to choose the simplest model structure, ARX211.

The similarity of the models may also be an indirect measure of how accurate they are. Figure 4.7 shows the distribution of the optimal time-shift, Δ over all models. The values range between 5 and 12 time steps of delay, and are centered around 9. This is consistent with the previous estimate that the input trails by five to ten time steps. Additionally, one can expect there to be an increased delay because of added dead time in the ARX models. The correlation between Δ and n_k is verified by figure 4.9. Although there is some variance, the dead time on average increases the optimal Δ found. Figure 4.8 shows the variance in the different beta values. Since sensor 1 was broken, every model correctly gave that sensor a beta of zero. Beta 3 and 4 were constrained to 1, while beta 5 and 6 were the maximum allowed value for all models, indicating that the limit might not have been high enough. The remaining 3 sensors had some amount of variance, but are quite consistent from model to model.

4.2.2 Selected Model

The selected model to use for regression is the ARX211 found. The optimal parameters found for this model are $\Delta = 6$ and $\boldsymbol{\beta} = [0, 0.03, 1, 1, 1.5, 1.5, 1.4, 0.60]$ and the ARX model before multiplying

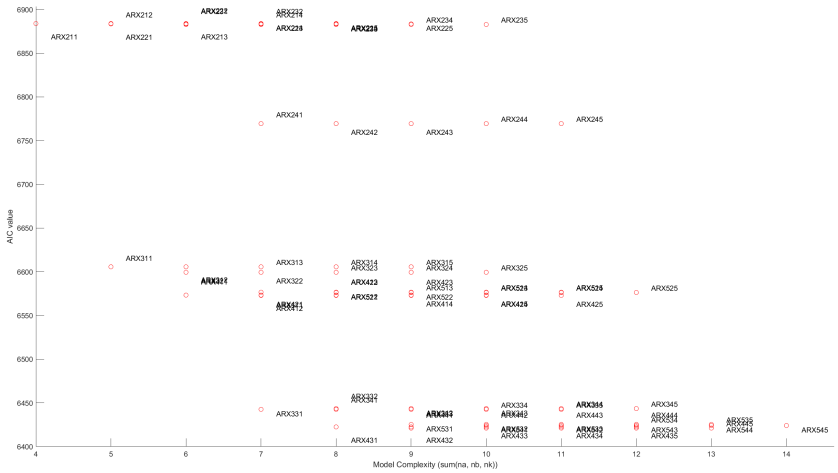


Figure 4.5: AIC values of the evaluated models for the ARX model found on the most prominent signals.

the B with the β_i's is given by

$$y(k) = 1.45y(k-1) - 0.59y(k-2) + 1.1u(k-1) \tag{4.1}$$

or equivalently, given by the matrices

$$A(q) = 1 - 1.45q^{-1} + 0.59q^{-2}$$
$$B(q) = 1.1q^{-1}$$

The model achieved a fit on the training set of 43.8% and a fit of 45.5% on the validation set. The full simulation results on the datasets are shown in appendix A. One can see that the worst performing predicted signals are 5 and 6, as the predicted peaks are smaller than the real magnitudes. Furthermore, the predictions on sensor 8 are a bit larger than the actual validation set, mainly because of some noise in the training set.

4.3 Spatio-temporal Gaussian Regression using a Kalman Filter

The model found by system identification gives, by the procedure described in section 3.4.1, the state space model equation (3.13) with matrices given by (4.2) for each sensor, with β_i's as defined in the previous section.

$$F = \begin{bmatrix} 1.44 & 1 \\ -0.59 & 0 \end{bmatrix}, \qquad B = \beta_i \begin{bmatrix} 1.1 \\ 0 \end{bmatrix}, \qquad H = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{4.2}$$

Although the parameters σ_R, σ_Q and σ_K have not been properly tuned, some results are presented which clearly show their effect on the system. Each figure has the tuneable parameters displayed in a textbox. Additionally, the full pressure profile for each tuning can be found in appendix B. The measured reference signal is shown in figure 4.10. A comparison between a low and high measurement noise variance is shown in figure 4.11. Notice that a small measurement noise variance makes the Kalman filter place more emphasis on the actual signal, resulting in less smoothing than with a high measurement noise. High measurement noise means the measurements are more uncertain, therefore the model with the tuned kernel is trusted more.

Varying the signal noise variance is shown in figure 4.12. Increasing the variance of the signal noise has an effect which is opposite to the measurement noise. This is because this variance represents uncertainty in the model. Therefore, the model, and therefore also the kernel is trusted less when σ_Q is increased.

As discussed previously, alpha represents how spatially smooth the underlying system is. Four different values of α are shown in figure 4.13. From this, it is clear that a large alpha most closely follows the model $(F, \beta_i B, H)$ for each system. When decreasing the alpha, the signal becomes more spatially smooth. This is because of the kernel which is introduced in the equation for C.

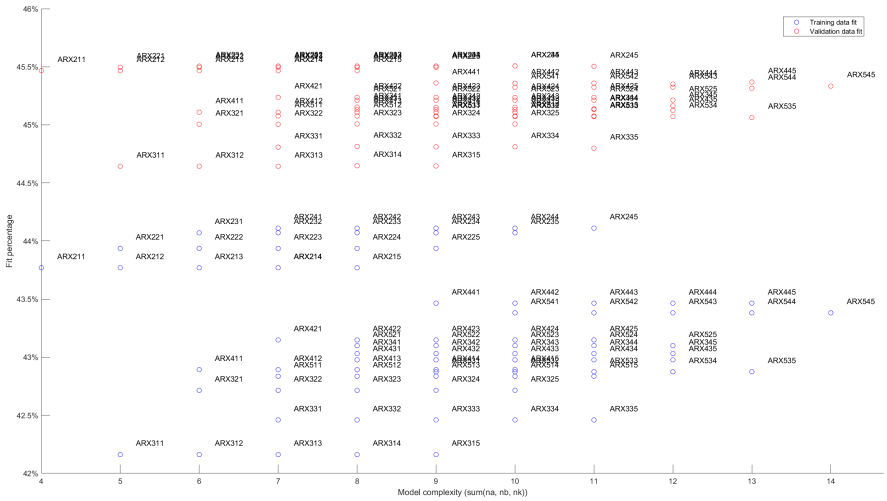
4.3.1 Error Distribution

Since the intent of the smoothing is to reduce the effects of measurement noise, which is assumed to be white and Gaussian, one ideally has that the errors are also white and Gaussian. This is because optimally, the difference between the measurements and the estimated pressure profile is only this noise. Therefore, how white and Gaussian the errors are serves as a measure of how well the regression performs. Note that this is not entirely valid, as there are other sources of noise present in the system, such as a non-functioning sensors, and drift of the basal muscular pressure.

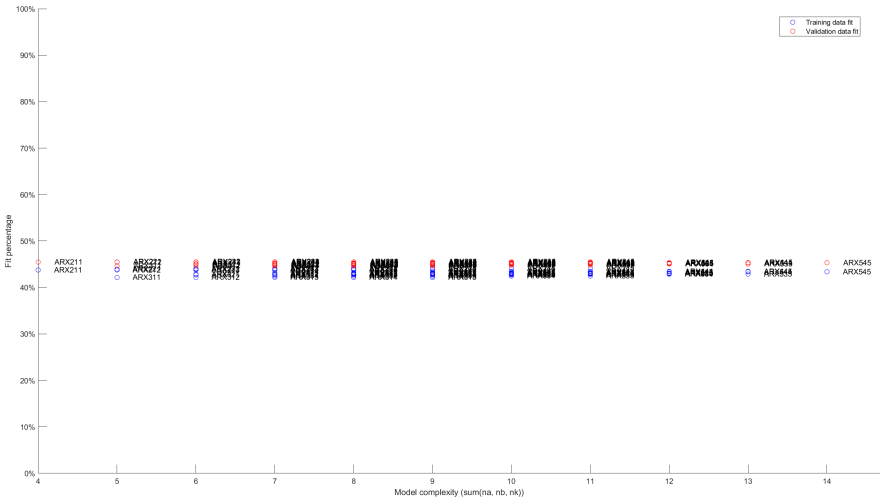
The following plots are the results from a tuning of $\alpha = 0.1$, $\sigma_R = 0.1$ and $\sigma_Q = 0.1$. Figure 4.14 shows the correlation between errors in time by plotting each error against the previous error for each state. Notice that each plot has a nearly Gaussian distribution, with some deviations, particularly at the tails of the empirical marginal distributions. These marginal distributions of the

errors per-sensor are shown in figure 4.15. Here, one ideally should have a Gaussian distribution per marginal, yet many of the error plots have non-null skewness, i.e., a more prominent tail in a particular direction. Larger versions of these two figures can be found in appendix C.

From practical purposes, though, we have been unable to improve the results above. In other words, these are the results we obtained using the best tuned parameters / hyperparameters given the available datasets. We foresee that nonlinear modelling and filtering approaches may lead to improved results (potentially in association with more extensive datasets). These tests are outside the scope of this thesis and we suggest them as potential future works.

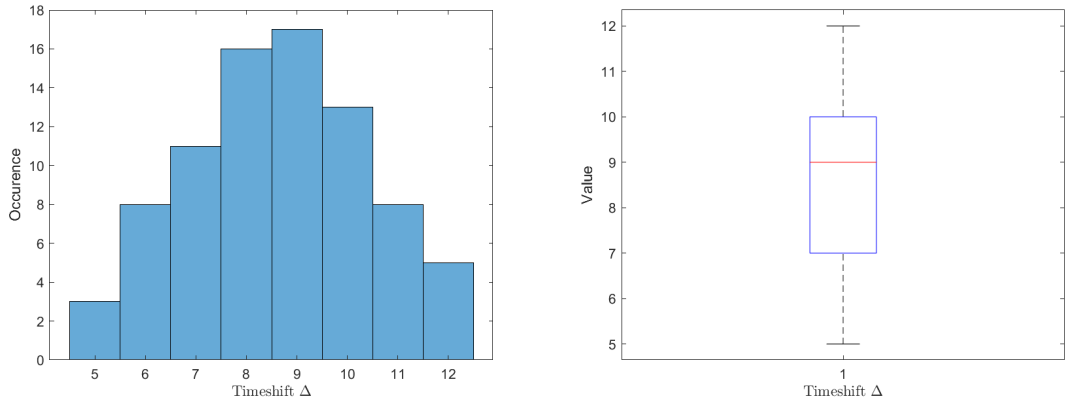


(a) Zoomed in view of fit percentage to model complexity.



(b) Zoomed out view of fit percentage to model complexity.

Figure 4.6: Fit percentage of simulation on both the training set and validation set plotted against the model complexity.



(a) Histogram of variance in the optimal time shift.

(b) Boxplot of variance in the optimal time shift.

Figure 4.7: Variance in timeshifts Δ estimated by different model orders. Visualized as a histogram in (a) and a boxplot in (b).

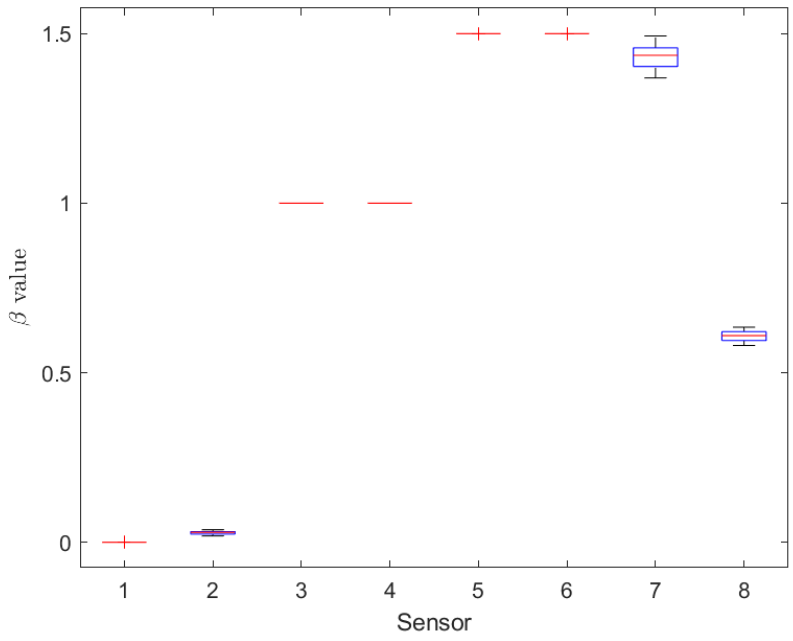


Figure 4.8: Histogram of optimal betas for different model orders.

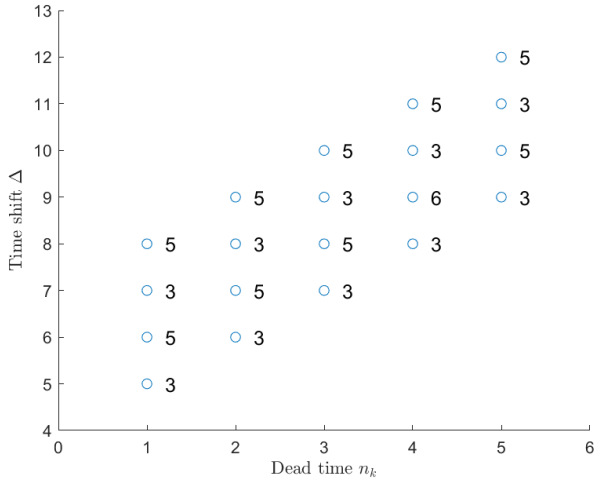


Figure 4.9: Scatterplot showing the clear correlation between the selected dead time and optimal time-shift over all models. The numbers signify the number of models with that particular time-shift and dead time.

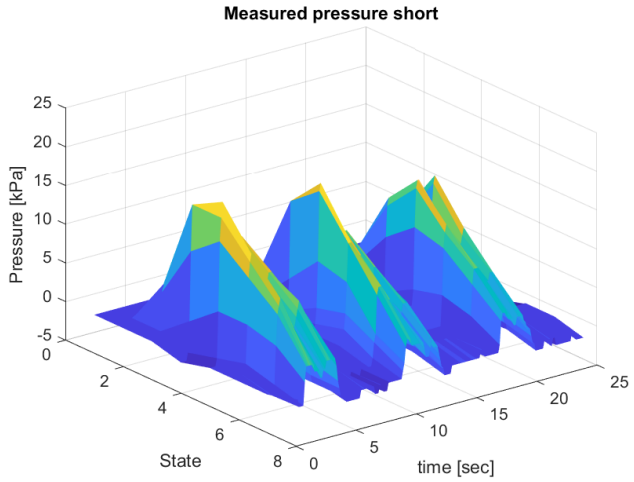
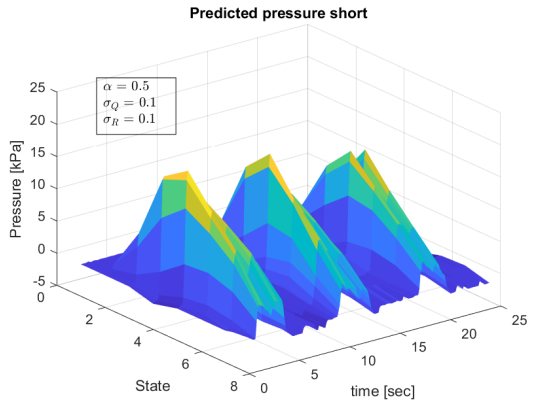
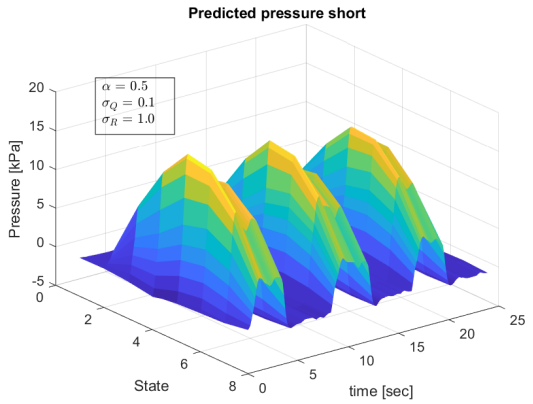


Figure 4.10: The surface of the measured pressure at the time instants used to show the result of different tuning

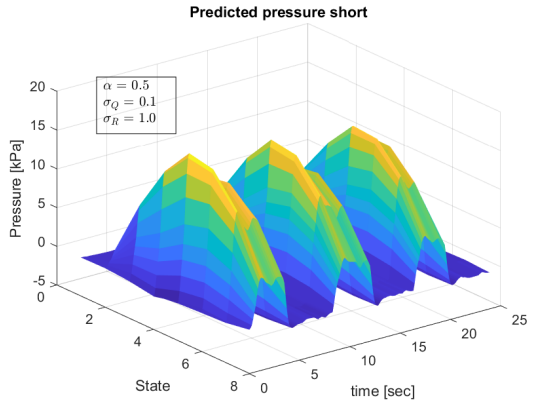


(a) predicted pressure surface with measurement noise $\sigma_R = 0.1$

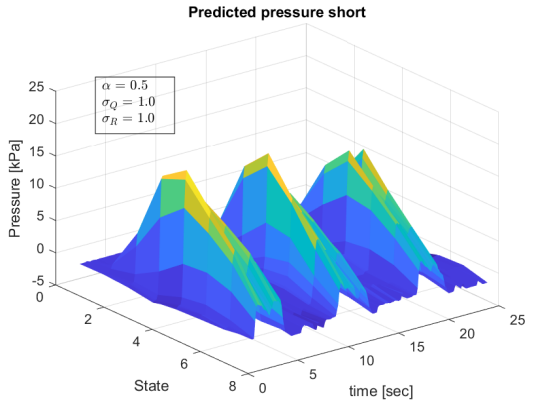


(b) predicted pressure surface with measurement noise $\sigma_R = 1$

Figure 4.11: Predicted pressure surface with varying measurement noise variance.

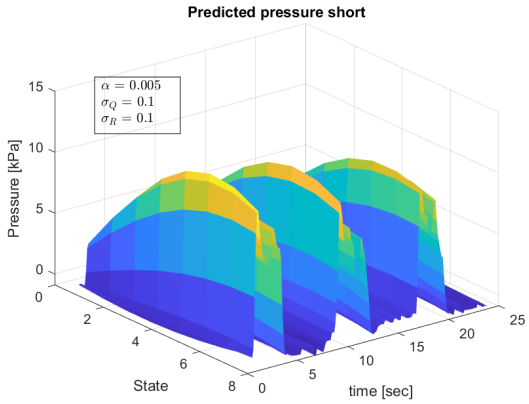


(a) predicted pressure surface with signal noise $\sigma_Q = 0.1$

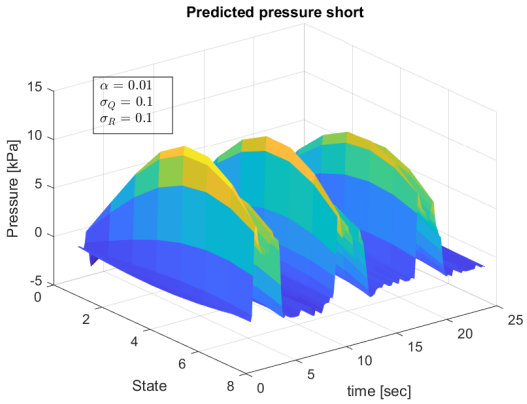


(b) predicted pressure surface with signal noise $\sigma_Q = 1$

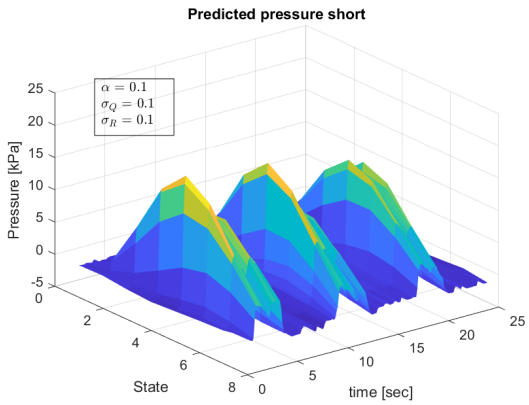
Figure 4.12: Predicted pressure surface with varying signal noise variance.



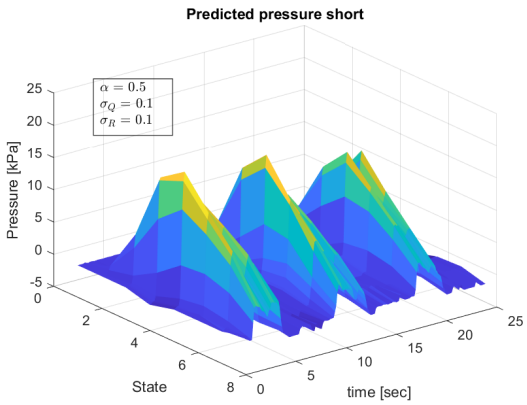
(a) predicted pressure surface with $\alpha = 0.005$



(b) predicted pressure surface with $\alpha = 0.01$



(c) predicted pressure surface with $\alpha = 0.1$



(d) predicted pressure surface with $\alpha = 0.5$

Figure 4.13: Predicted pressure surface with varying smoothing of the kernel.

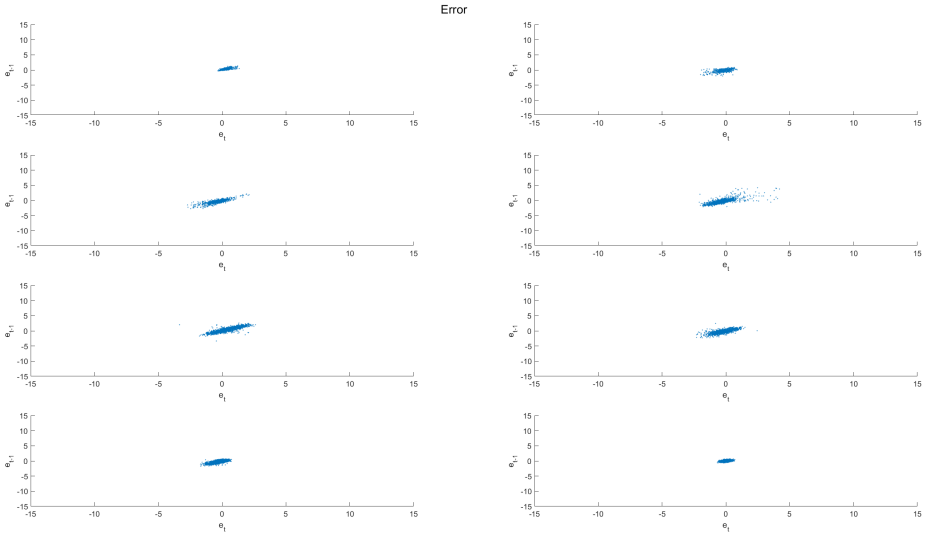


Figure 4.14: Scatterplot showing the correlation of the errors in time.

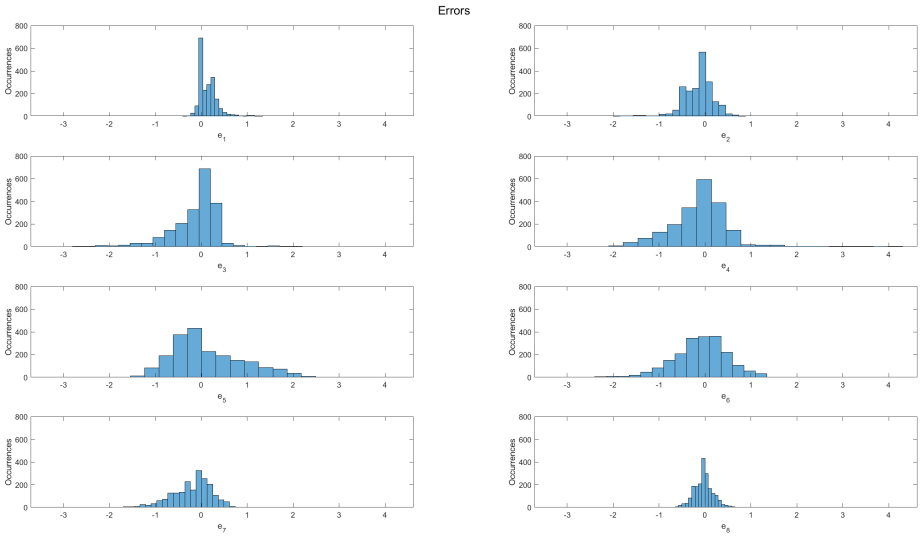


Figure 4.15: Histograms showing the distribution of the errors for each state.

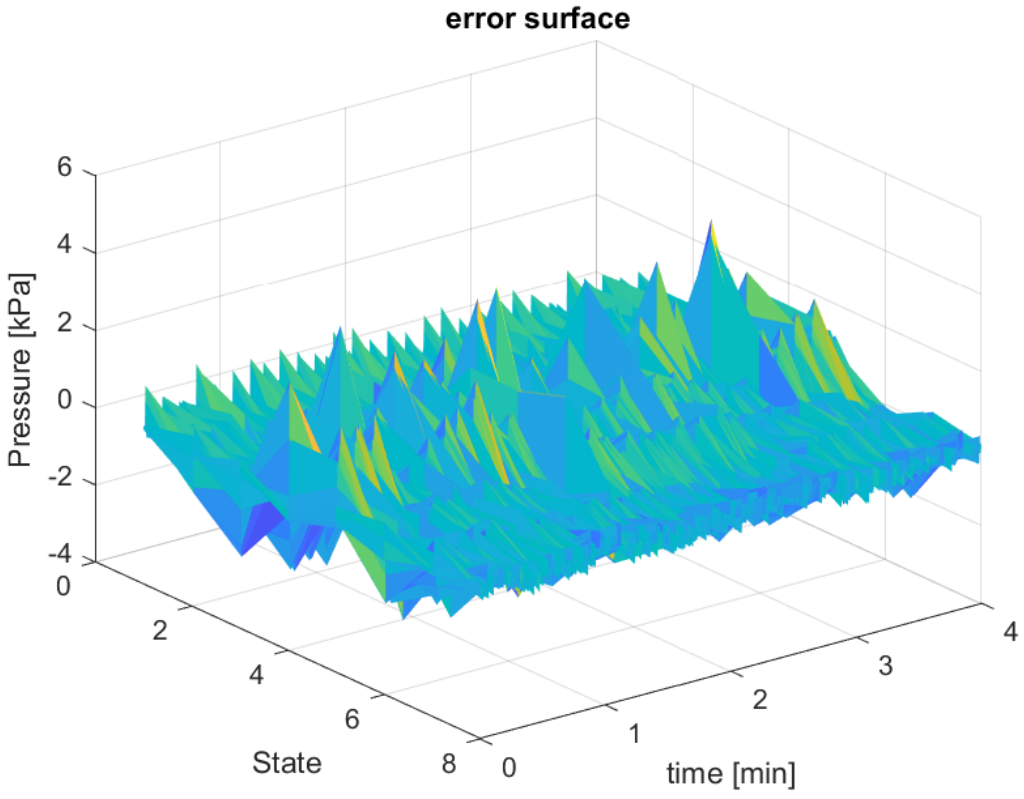


Figure 4.16: The complete error surface showing error distribution in space and time

Chapter 5

Discussion

This chapter discusses the methods presented in chapter 3 and the results presented in chapter 4. The individual methods and results for each part are discussed.

5.1 Dataset

The available dataset as a whole was not entirely suitable for system identification, since information about the input is missing. A blind-identification approach would have been feasible, meaning to estimate both the parameters and the input at the same time, however since the problem of reconstructing the inputs was deemed as simple initially, this was not done. Rather, we followed a two-steps approach: 1) estimate the input as a signal processing problem, 2) estimate the system parameters using such reconstructed input. Additionally, since the dataset was a collection of several sessions, system characteristics were not constant. This was however a non-issue, as several time ranges could be selected for which the measurements had consistent characteristics without significant drift. The selected datasets were 4 minutes long, a relatively short time-period whose length might affect results. During these 4 minutes, there were 28 peaks (i.e., jumps executed by the player) giving information about the dynamics of the muscular pressure from which the system identification step can be done in a meaningful way. However, a longer dataset which is guaranteed to have unchanged characteristics might be hard to achieve since a patient will fatigue quickly (in the order of minutes).

Another issue with the available dataset was that the outer sensor was faulty. However, for the second-most outer sensor the peaks were already very small, to the point that it likely doesn't affect the overall shape of the peaks significantly.

5.2 Input Estimation

A contributing factor in the relatively small delay between the signal change and detection is that the threshold for change could be set to only 10% of the maximum value. This is because the peaks heights are much larger than any noise seen in the measurements, and no noise reached more than $2\% - 3\%$ of the maximum peak height. This might not be the case in general, where wear on the sensors or incorrect insertion might lead to noise with a much larger variance. In this case, the threshold has to be increased, which will add an additional delay between exitation and detection. As mentioned in section 4.1 the tuning of both algorithms was done to provide relatively little smoothing. This is done in the form of a small window size for the finite moving average, and a relatively large α. This is possible because of of the large difference in the two states compared to amplitudes of the noise. This is another factor which speeds up the detection of a change.

Since both methods perform approximately equally well, one can be chosen freely. The finite moving average method is used here because the automatically scaling weights make choosing the threshold easier. The threshold is to a large degree independent from the window size as it is only affected by the increased smoothing caused by larger windows.

In reality, the brain signal we are attempting to estimate is not binary. This estimation results in some information being lost, which likely leads to a decrease in the overall accuracy of the modelled system. Particularly for our datasets, the peak times are quite consistent, while the heights of the peaks vary more. However, since the binary input is estimated using a threshold on the pressure, the difference in peak height is not captured by our input estimates. This leads to estimation with very uniform peak heights.

5.3 System Identification

Our results about the identification of the temporal dynamics of the muscular pressure is discussed in two parts. Firstly, the performance of the different model orders is discussed, and secondly the chosen model is examined in more details.

5.3.1 Model Order Selection

As explained in section 4.2.1, the models were found using ranges on the three model order parameters. Although a grid search could have covered a larger "area" of model orders, given the qualitative outputs that we can measure from the sensors, it is assumed beforehand that the desired model order is quite small. However, some systems with larger orders were evaluated as well to see how large the gain in generalization performance was compared to the added complexity. In figure 4.6 one can clearly see that the improvement for more complex models was minimal, and varied quite a lot for different models of similar size. One reason for this could be the binary input, which limits the possible complexity. Furthermore, little weight was placed on the AIC values since these do not represent the entire system. Nevertheless, a general trend of lower order models per-

forming better was observed. Particularly models with a low order n_a, which seemed to affect the results the most. Since the systems performed better on the validation set than on the training set it could indicate that it would be beneficial for estimation to swap these. Although, conventionally a training set that comes before the validation set in time is chosen.

All model orders have a quite poor fit, with less than 50% on both the training and validation set. Again, some part of this is due to the binary input. Since the peak heights were lost in the model, this decreased the fit substantially. Furthermore, limiting the β's to 1.5 was too severe when using sensor 3 and 4 for the initial model. This can be seen in appendix A, where the peaks are much smaller for the predictions for these particular sensor. Therefore, some improvement of the fit would be seen by either using the signals with largest peaks for the initial estimation, or by relaxing the upper limits on the value of the β's. On the other hand, since the model is the basis for regression using a Kalman filter, there are less strict requirements on its accuracy. The initial model is simply a starting point for a better estimation method.

Overall, the optimization resulted in parameters of reasonable values. Initially, there was an issue where the performance with several different values of Δ had practically the same performance. This resulted in unnecessarily large time shifts in many of the models for no particular reason. Adding regularization on this time shift removed this issue, and kept the values in a more reasonable range for the most part. The largest values of the time shift are mostly a result of unnecessary dead time in the system, as illustrated by figure 4.9. Furthermore, there was very little variance on the values of β_i for $i \in [1, ..., 8]$ [3, 4]. partly because three of them were constrained by the lower and upper limit. However even β_7 with the most variance only varied by about 0.15 or roughly 10% of its value.

5.3.2 Parameters of the Selected System

Although the selected ARX model (with model order $[na, nb, nk] = [2, 1, 1]$) is one of the simplest possible models,it gives roughly the same fit percentage as the other models. Besides having probably a very low variance, as an estimator (but of course a potentially higher bias), one computational advantage of such a low model order is that it allows for a state-space representation with only two states. Furthermore, because of the small delay time n_k selected before, the time shift required is very small, allowing it to estimate with a delay of only 6 samples, or 0.72 seconds, when used in on-line settings. The same issues mentioned in the previous subsection apply to this system. The constraint on the betas decreased the performance on the signals from sensor 5 and 6. The noise on sensor 8 in the training set, makes the model overestimate the peaks, decreasing performance on that signal as well.

5.4 Spatio-Temporal Gaussian Regression using a Kalman Filter

Acquiring a state-space representation by first estimating an ARX model affects the algorithm in that no continuous time representation is available. If necessary, such a representation can be found anyway, but this increases the computations required. Since this isn't implemented, time interpolation can't be done. Nevertheless, the algorithm still provides the main function of smoothing the measured signal in both time and space. This smoothing works to reduce measurement noise and drift over time, and may also partially counter-act faulty sensors.

While figure 4.11 and figure 4.12 give an indication on how the parameters affect the system, these should not be used as a means to tune them. In fact, the optimality of the Kalman filter relies on process noise covariance Q, and measurement noise covariance R which accurately represent the system. If the assumptions made about the structure of these are inadequate, it may lead to filter divergence [Singh and Mehra, 2015]. There is extensive research on estimating these matrices from data. One popular method is the autovariance least-squares (ALS) technique. This uses time-lagged autocovariances during normal operation to estimate the covariances [Rajamani and Rawlings, 2009]. There are also methods, such as the Field Kalman filter presented by Bania and Baranowski [2016], which simultaneously approximate the state and noise covariances.

It is clear from figure 4.13 that the spatio-temporal regression presented allows for fine-tuning of the desired spatial smoothness. Furthermore, the temporal noise is smoothed out by the Kalman Filter by combining measurements with the model found. If tuned correctly, this improved method of acquiring smooth estimates of the true underlying process may allow for better feature extraction of the signal. This can in turn be used to find improved statistics. One application of such statistics is to separate whether the user is standing or lying down, as in Knorn et al. [2020]. This paper uses a similar spatial kernel, but does not exploit the temporal nature of the signal. Rather, single instants of time are analyzed. Therefore it is reasonable to assume that analysis which includes both dimensions might improve results. Such a smoothed signal can also be applied to fatigue estimation and analysis of which muscle groups are being used.

The figures which show the error distributions indicate that the tuning done is not sufficient to adequately model the system. Many of the error distributions have a more prominent tail in one direction, indicating that the true dynamics are not entirely captured.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis has shown a method for doing spatio-temporal gaussian regression via Kalman filtering for muscular pressure data from the pelvic floor muscles. This is done by firstly estimating an input by using change detection algorithms, then creating a set of polynomial models by using polynomial model regression and an optimization algorithm. The models are then converted to a state-space model, which are used in the Kalman filter itself. The Kalman filter for spatio-temporal gaussian regression is adapted from Todescato et al. [2020]. Since discrete-time polynomials are used to model the system, no continuous-time model is available. Because of this, interpolation in time can't be done. The performance of the ARX models found was quite poor independent of model order chosen, achieving less than a 50% fit when simulating the validation data. The main factor for the poor performance is that the estimated input is too simple, and therefore limits the possible complexity. In other words, the binary input estimated does not represent the true brain signals to a sufficient degree. Since there is very little difference in the performance of different models, the simplest model – an ARX211 – is used for the Kalman filter.

Directly estimating a discrete-time model is found to complicate the Kalman filter design. Since the continuous-time system noise matrix is not directly available from our model, both the system noise and the initial covariance must be tuned. In this thesis, no tuning is found for which the model accurately represents the true system. This can be seen from the error distributions for each sensor. These do not approach a gaussian, as expected, but rather have more prominent tails in a particular direction. Nevertheless, several methods for tuning the noise covariances are suggested. Even though no sufficient tuning has been found, the results show that the kernel parameters allow for fine-tuning of the spatial-smoothness of the function. If the Kalman filter is properly tuned, this also ensures smoothing of noise in time. Therefore, this is an initial step towards a smooth estimate which can be used for many applications. Additionally, re-tuning of the model and Kalman filter

is possible on-line, allowing for increased individualization and fatigue-estimation.

6.2 Future Work

Firstly, a method should be found to properly tune the current spatio-temporal gaussian regression. If this is successful, the result serves as an initial method to smooth muscular pressure data. Such a smoothed estimation can be used to extract better statistics to detect if someone is standing or lying down, possibly improving the results of Knorn et al. [2020]. This also allows for evaluation of the pressure spatially and temporally, which may improve detection of which muscles are being used and how the methods should be adjusted for the specific person and game-session. If the tuning is unsuccessful, this may indicate that the model must be improved. Furthermore, an approach where the true input is estimated simultaneously as the other system parameters should be considered. A more sophisticated input may lead to greater differentiation between model structures, and can therefore provide greater insight into what model is most appropriate for the system. Finally, estimates are only available for the measurement locations at the measurement times. Todescato et al. [2020] suggests an algorithm to estimate the function value at arbitrary locations in the problem space and in time. Therefore, such estimates might be possible on the pelvic floor muscle data by adapting the algorithm to the implementation suggested in this thesis.

Bibliography

Akaike, H. (1974). A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723.

Bania, P. and Baranowski, J. (2016). Field kalman filter and its approximation. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 2875–2880. IEEE.

Basseville, M., Nikiforov, I. V., et al. (1993). *Detection of abrupt changes: theory and application*, volume 104. prentice Hall Englewood Cliffs.

Betts, J. G., Young, K. A., Wise, J. A., Johnson, E., Poe, B., Kruse, D. H., Korol, O., Johnson, J. E., Womble, M., and DeSaix, P. (2013). *Anatomy and Physiology*. OpenStax, `https://openstax.org/books/anatomy-and-physiology/pages/1-introduction`.

Chen, S.-C., Shyu, M.-L., Peeta, S., and Zhang, C. (2003). Learning-based spatio-temporal vehicle tracking and indexing for transportation multimedia database systems. *IEEE Transactions on Intelligent Transportation Systems*, 4(3):154–167.

Chiarelli, P., Murphy, B., and Cockburn, J. (2003). Women's knowledge, practises, and intentions regarding correct pelvic floor exercises. *Neurourology and Urodynamics: Official Journal of the International Continence Society*, 22(3):246–249.

Das, M. and Ghosh, S. K. (2014). A probabilistic approach for weather forecast using spatio-temporal inter-relationships among climate variables. In *2014 9th International Conference on Industrial and Information Systems (ICIIS)*, pages 1–6. IEEE.

Hartikainen, J. and Särkkä, S. (2010). Kalman filtering and smoothing solutions to temporal gaussian process regression models. In *2010 IEEE international workshop on machine learning for signal processing*, pages 379–384. IEEE.

Herschorn, S. (2004). Female pelvic floor anatomy: the pelvic floor, supporting structures, and pelvic organs. *Reviews in urology*, 6(Suppl 5):S2.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45.

Kask, N., Budgett, D. M., Kruger, J. A., Nielsen, P. M., Varagnolo, D., and Knorn, S. (2019). Data-driven modelling of fatigue in pelvic floor muscles when performing kegel exercises. In *IEEE Conference on Decision and Control (CDC), Nice*.

Knorn, Varagnolo, Jackson, Budgett, Kruger, and Nielsen (2020). Online, data-driven detection of human position during kegel exercising. *IEEE transactions on automatic control*.

Ljung, L. (1988-2012). System identification toolbox. *The Matlab user's guide*.

Ljung, L. (2007). System identification. *Wiley Encyclopedia of Electrical and Electronics Engineering*.

Meliker, J. R. and Sloan, C. D. (2011). Spatio-temporal epidemiology: principles and opportunities. *Spatial and spatio-temporal epidemiology*, 2(1):1–9.

Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

Pronzato, L. (2008). Optimal experimental design and some related control problems. *Automatica*, 44(2):303–325.

Rajamani, M. R. and Rawlings, J. B. (2009). Estimation of the disturbance structure from data using semidefinite programming and optimal weighting. *Automatica*, 45(1):142–148.

Singh, Y. and Mehra, R. (2015). Relative study of measurement noise covariance r and process noise covariance q of the kalman filter in estimation. *IOSR Journal of Electrical and Electronics Engineering*, 10(6):112–116.

Tian, T., Budgett, S., Smalldridge, J., Hayward, L., Stinear, J., and Kruger, J. (2018). Assessing exercises recommended for women at risk of pelvic floor disorders using multivariate statistical techniques. *International urogynecology journal*, 29(10):1447–1454.

Todescato, M., Carron, A., Carli, R., Pillonetto, G., and Schenato, L. (2020). Efficient spatio-temporal gaussian regression via kalman filtering. *Automatica*, 118:109032.

Walker, G. J. and Gunasekera, P. (2011). Pelvic organ prolapse and incontinence in developing countries: review of prevalence and risk factors. *International urogynecology journal*, 22(2):127–135.

Williams, C. and Rasmussen, C. E. (2006). Gaussian processes for machine learning, vol. 2, no. 3.

Xu, Y., Choi, J., and Oh, S. (2011). Mobile sensor network navigation using gaussian processes with truncated observations. *IEEE Transactions on Robotics*, 27(6):1118–1131.

Appendices

A ARX211 Simulation Results

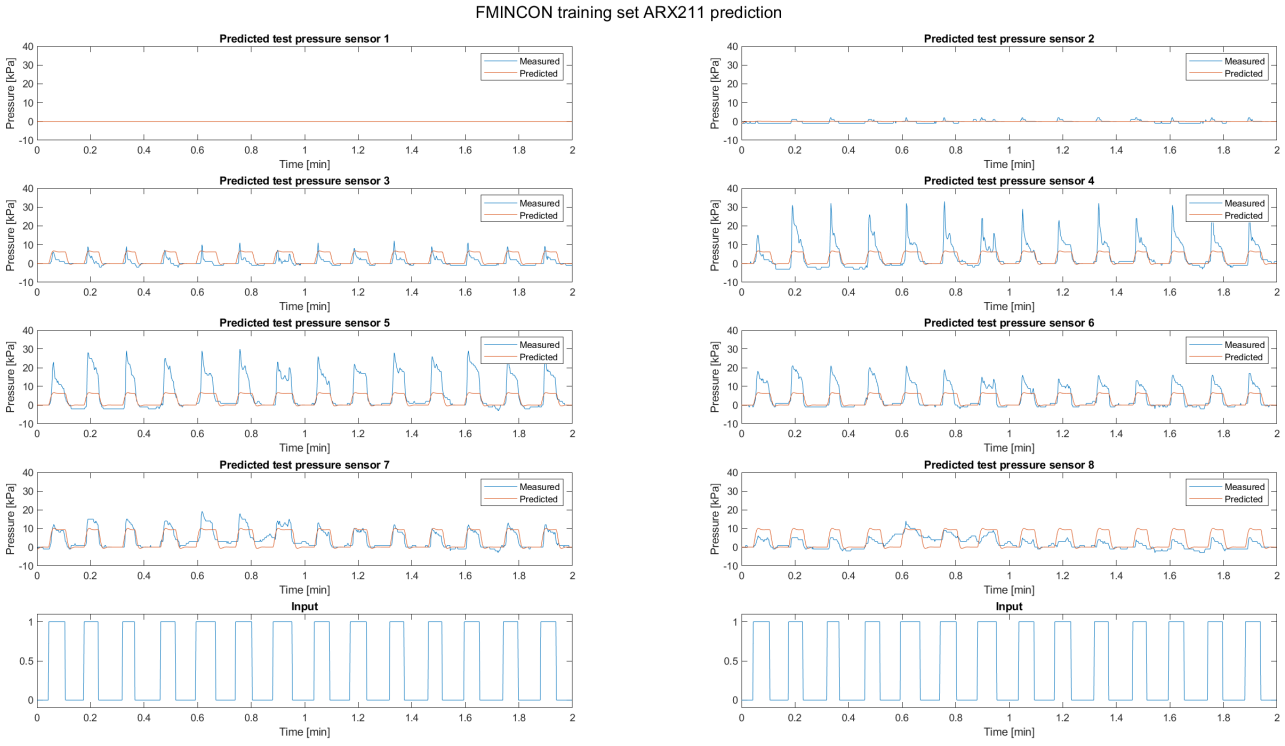


Figure A.1: Results of simulation of ARX211 on the training data against the measured signal.

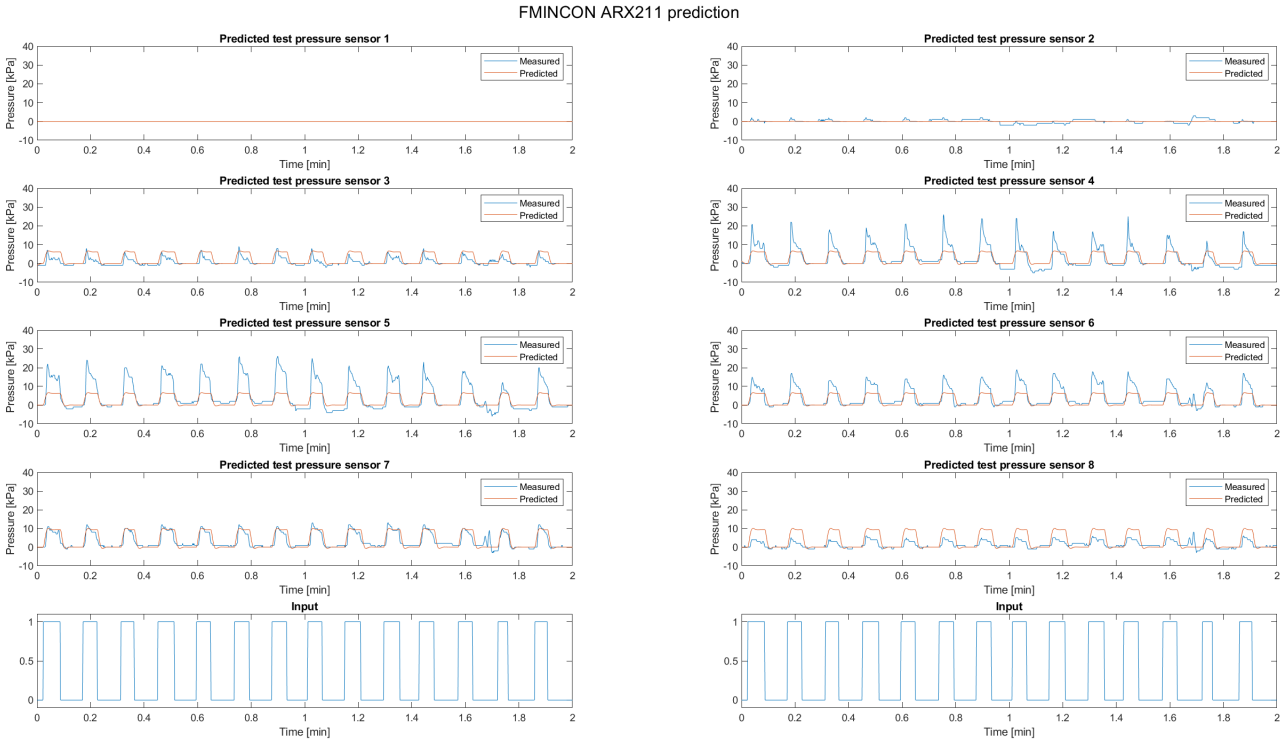


Figure A.2: Results of simulation of ARX211 on the validation data against the measured signal.

B Spatio-Temporal Regression Results

This appendix shows the complete predictions for each tuning mentioned in chapter 4.

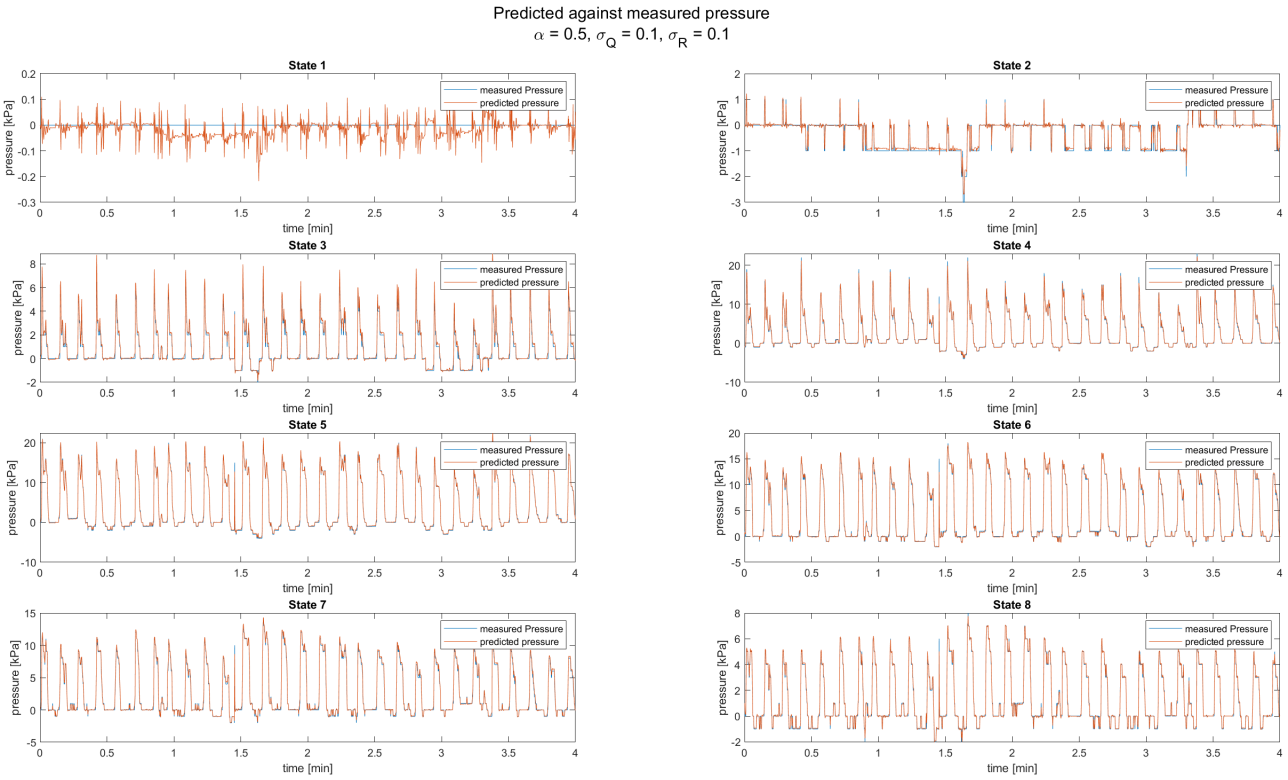


Figure B.1: Predicted pressure with tuning $[\alpha, \sigma_Q, \sigma_R] = [0.5, 0.1, 0.1]$.

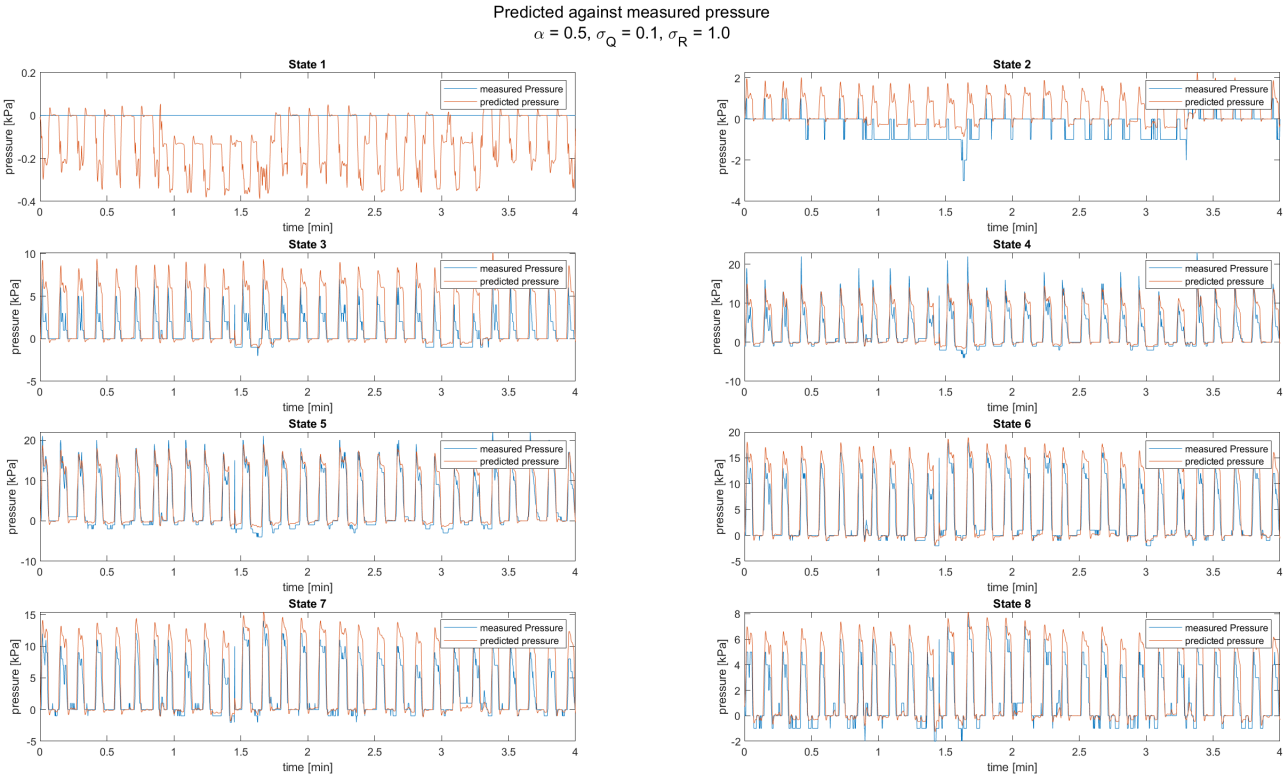


Figure B.2: Predicted pressure with tuning $[\alpha, \sigma_Q, \sigma_R] = [0.5, 0.1, 1.0]$.

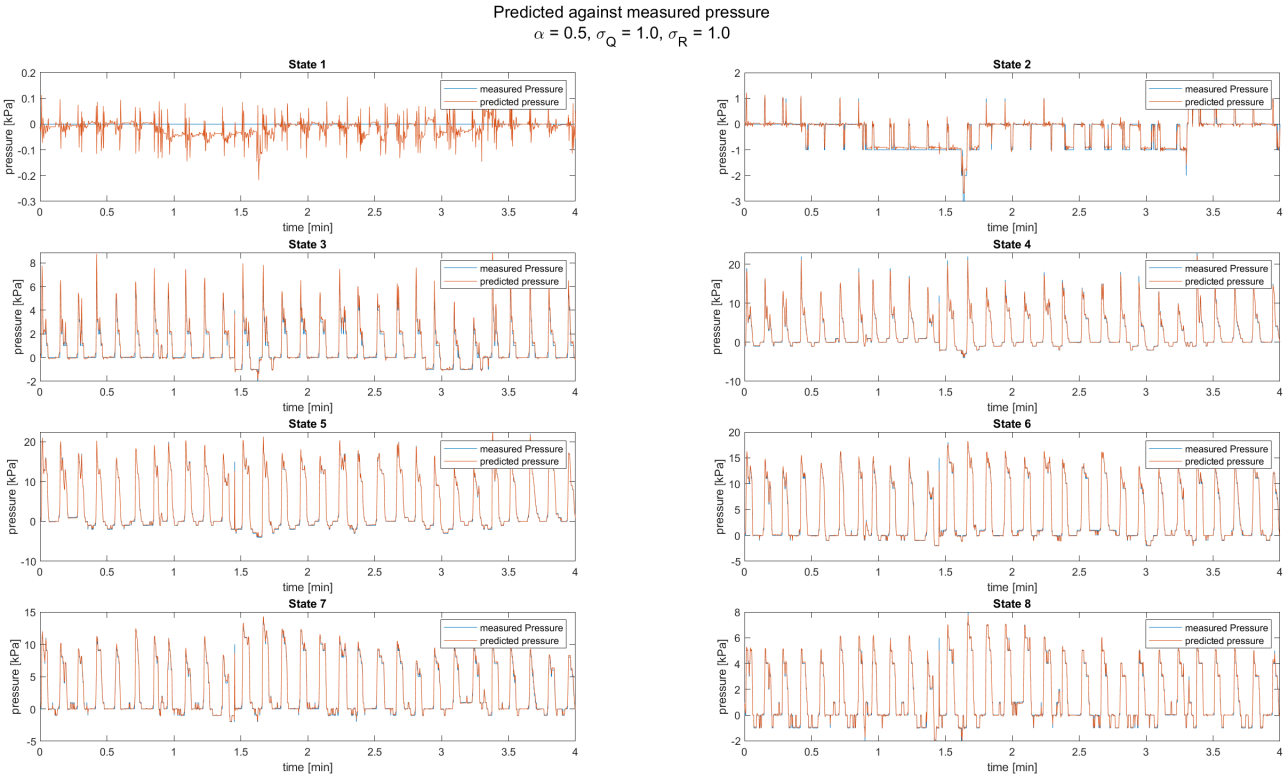


Figure B.3: Predicted pressure with tuning $[\alpha, \sigma_Q, \sigma_R] = [0.5, 1.0, 1.0]$.

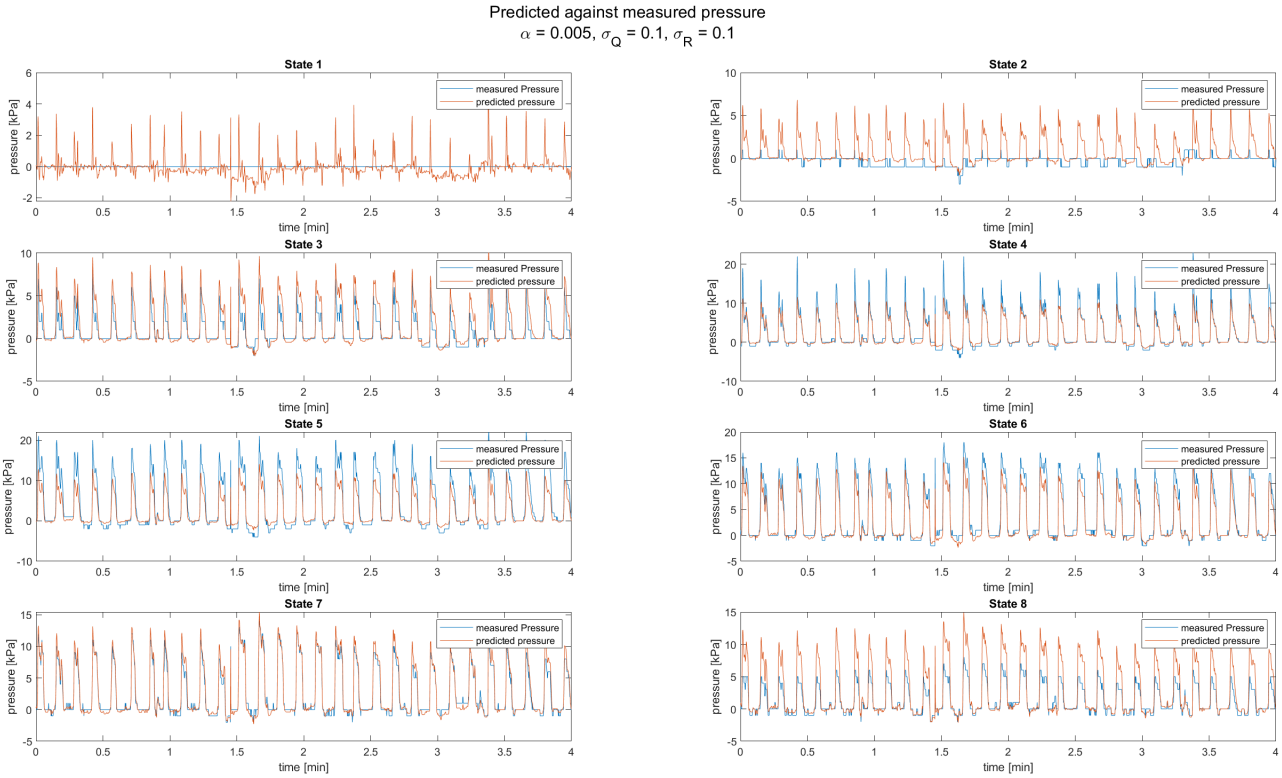


Figure B.4: Predicted pressure with tuning $[\alpha, \sigma_Q, \sigma_R] = [0.005, 0.1, 0.1]$.

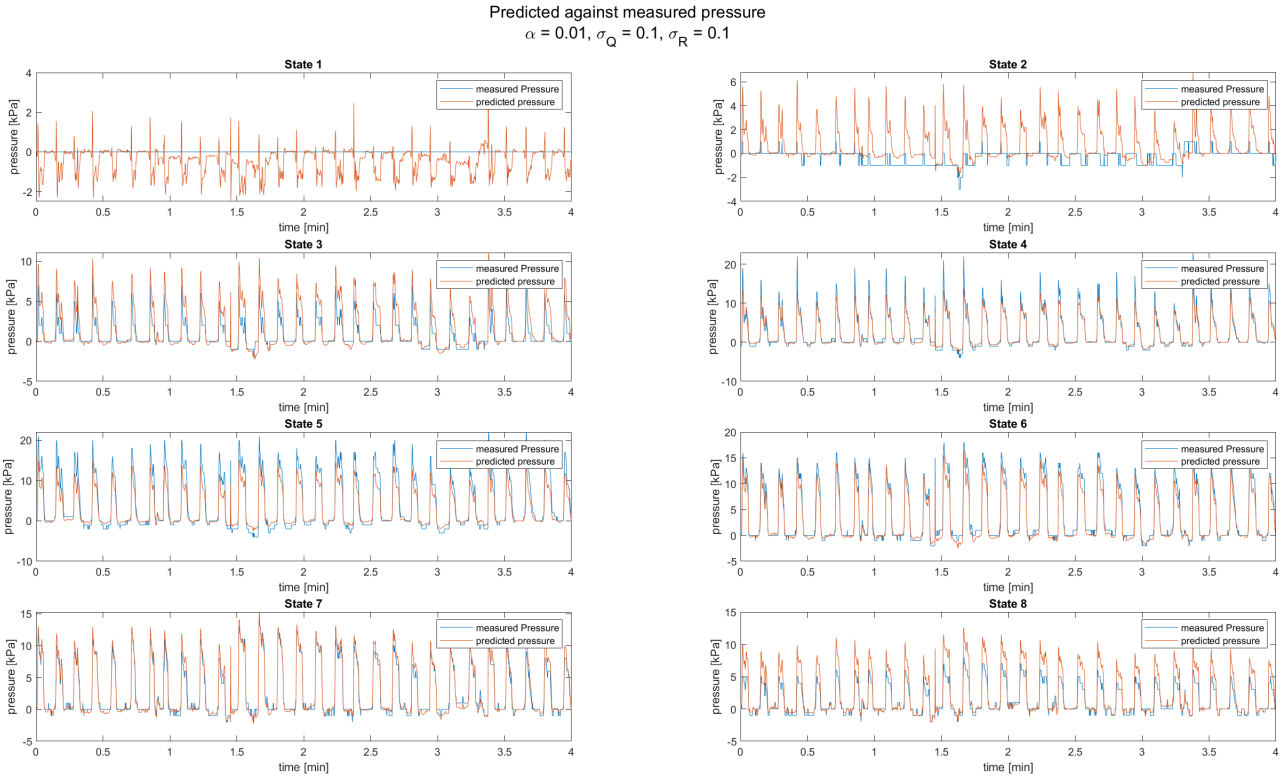


Figure B.5: Predicted pressure with tuning $[\alpha, \sigma_Q, \sigma_R] = [0.01, 0.1, 0.1]$.

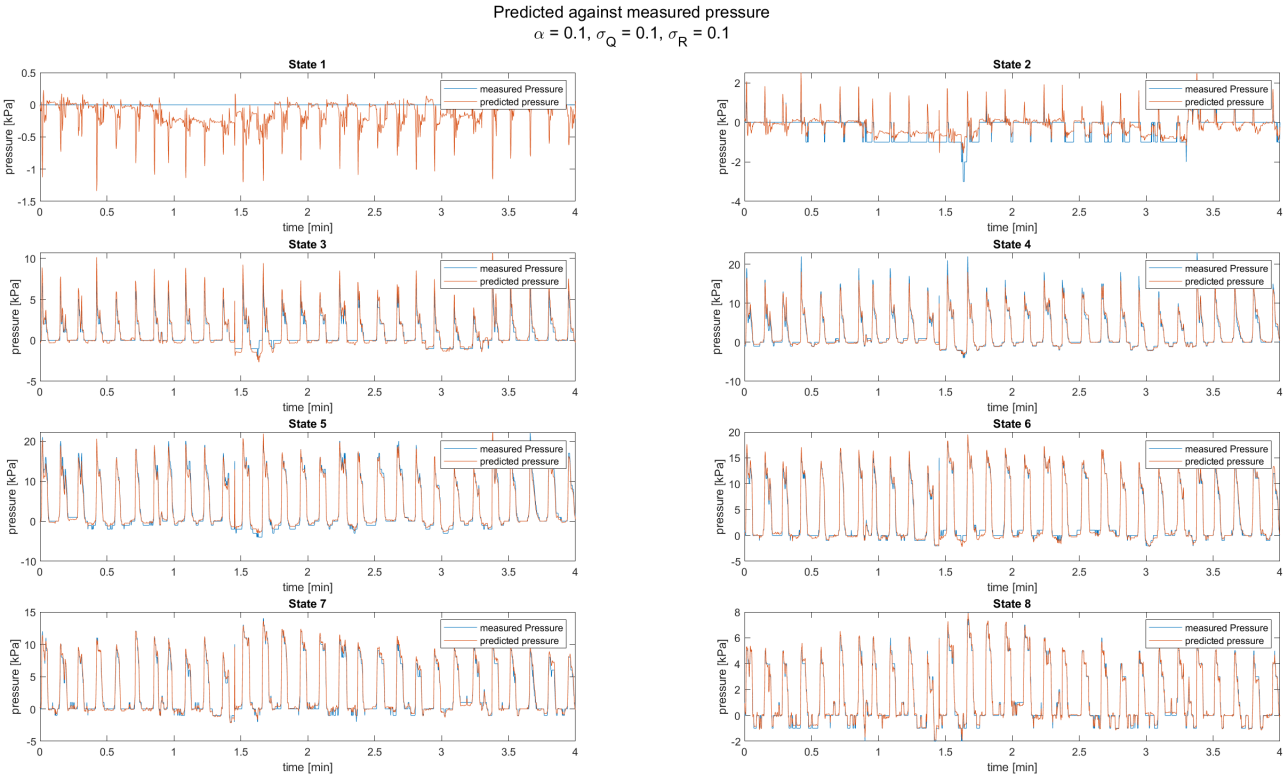


Figure B.6: Predicted pressure with tuning $[\alpha, \sigma_Q, \sigma_R] = [0.1, 0.1, 0.1]$.

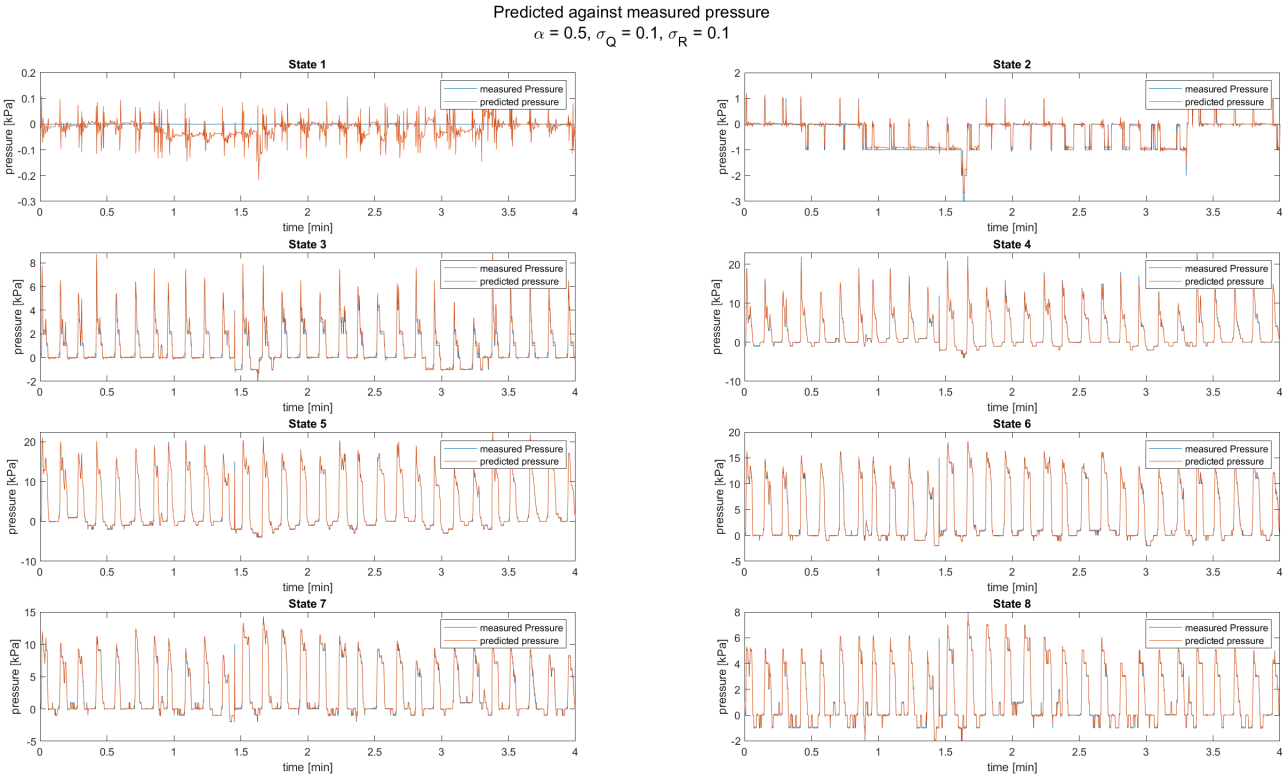


Figure B.7: Predicted pressure with tuning $[\alpha, \sigma_Q, \sigma_R] = [0.5, 0.1, 0.1]$.

C Error Distribution

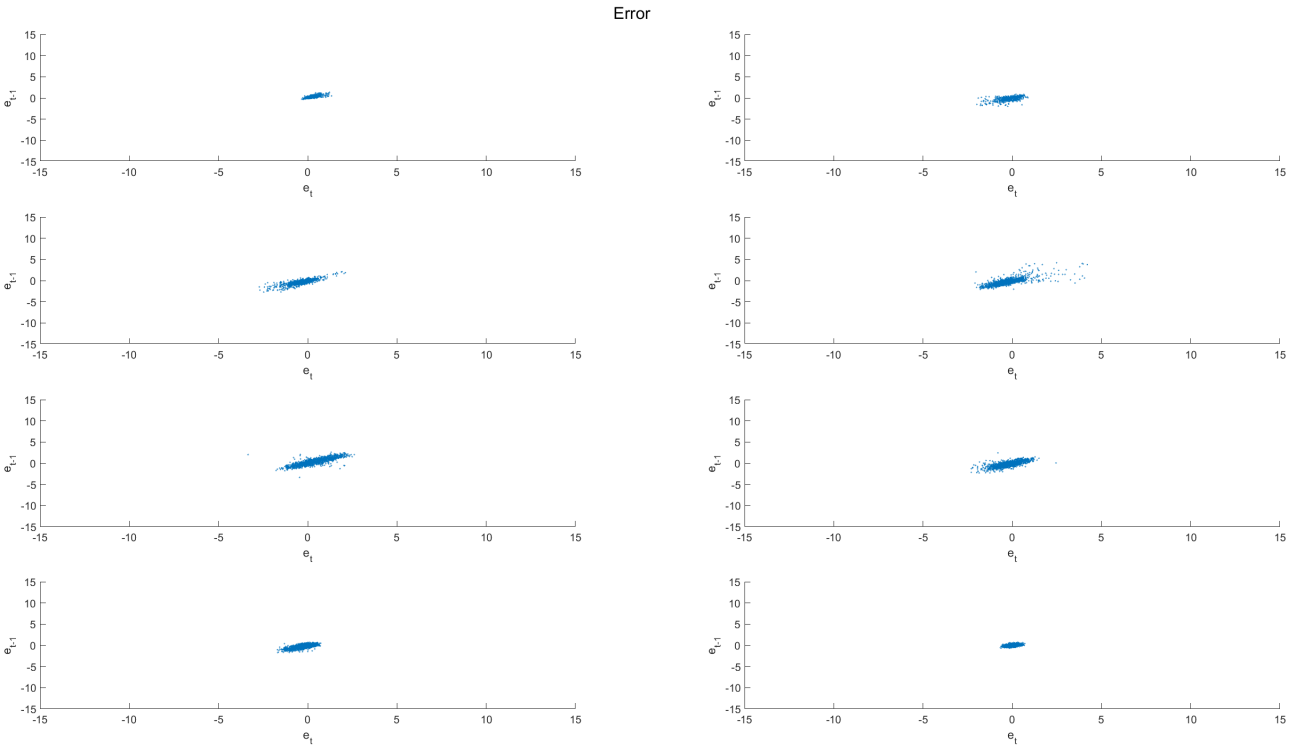


Figure C.1: Scatterplot showing the correlation of the errors in time.

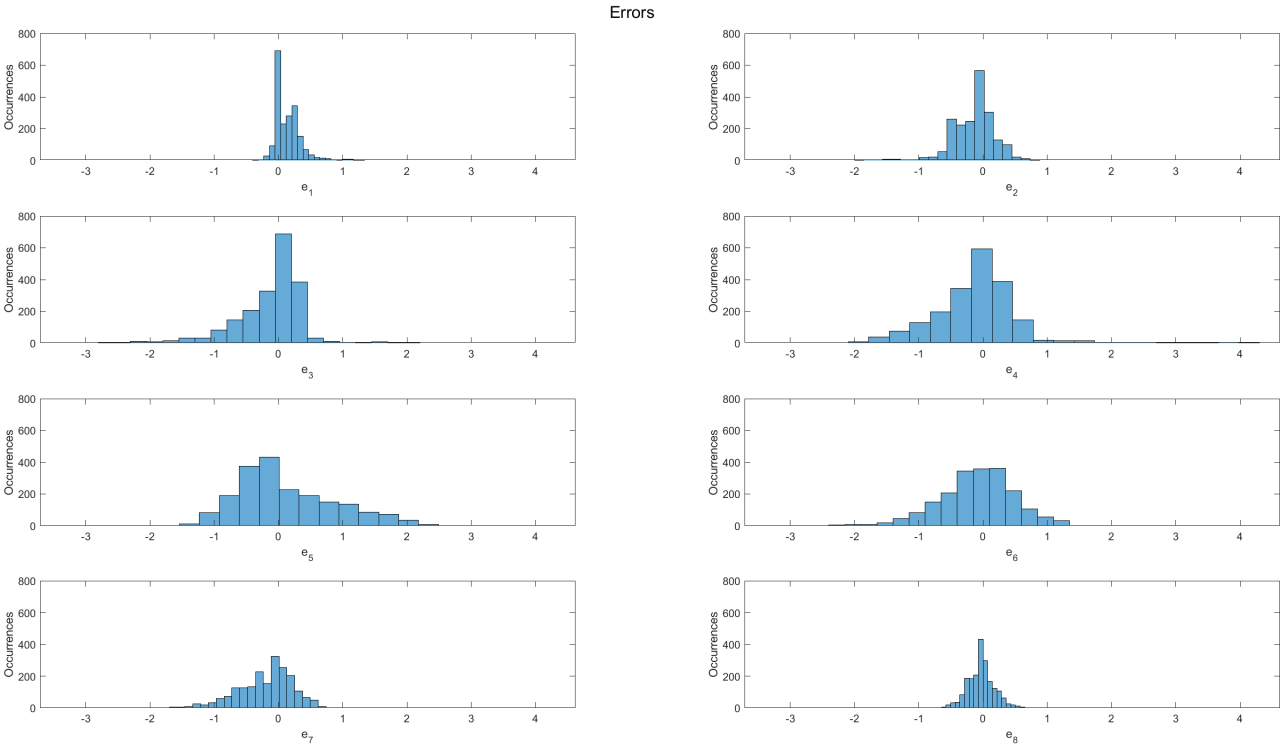
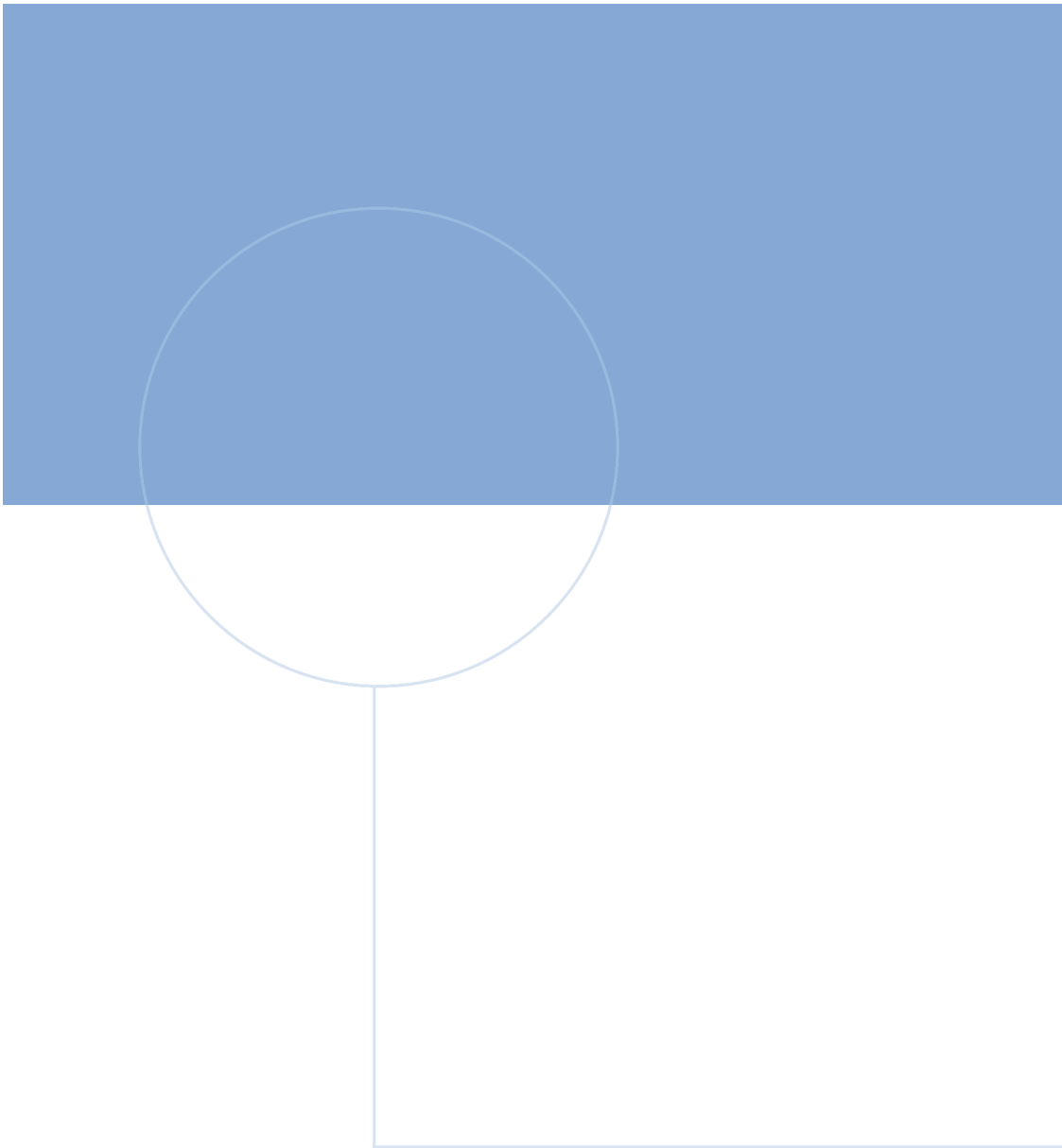


Figure C.2: Histograms showing the distribution of the errors for each state.



NTNU

Norwegian University of
Science and Technology