The Analysis of Time Dependent Computer Experiments

Thesis submitted for the degree “Doctor of Philosophy”

By

Sigal Levy

Prepared under the supervision of Prof. David Steinberg

Submitted to the Senate of Tel Aviv University

April 2008
אנדרטת תל אביב

הפקולטה למדעי מדעיים ע"ש רימונד ורורלי סאקלר
בית הספר למדעי המחשב
הוותק להשתתפותה ודום ביצועים

נירית ישראל ממחשבת תlevanceboom

הוורר לרשת קבלת התואר "דוקטור לפילוסופיה")
נאות
 наруב

העבורה הועברה בבחינתו של פור הפרופ', דוד שטיינברג
העבורה הועברה לפנレストラン של אוניברסיטת תל אביב
ינשה תשמ"ח – אפריל 2008
An old Jewish Proverb says: “Toil, and you shall find”. Meaning, there is no success without effort and toil. And why does it say “you shall find” and not simply “you shall succeed” or “you shall achieve your goal”? According to Jewish belief, only the hard work and effort are the responsibility of Man, while achievement is a gift from God, similar to a treasure that has been found accidently. In this spirit, a person investing time and effort is already considered successful, because the journey in itself is the essence.

I started my journey in small hesitant steps, that grew and strengthened due to the devoted guidance of my supervisor, Prof. David Steinberg. Thank you from the bottom of my heart, David, for the enormous support and infinite patience. Thanks to you I have learned both statistics and independence. Simple words are not enough to convey the deep appreciation I have for you, both professionally and personally. I hope to walk this path in my future academic career and prove myself worthy of your trust, toil and pride.

I would also like to thank:

Dr. Dorit Shweiki, for her help in constructing the circadian rythme model used as a case study in this work,

Prof. Felix Abramovich, for the many hours he dedicated and for great brainstorming,

My family for their support throughout the journey,

And my close friends, who were always there to support, encourage and motivate: Nili, for years of unparalleled friendship, Hovav, who knew how to push me forward when I needed him.

Sigal Levy
April 2008
Abstract

Computer experiments are a convenient substitute to real life experiments, when such experiments are too complex, expensive or time consuming. This work is concerned with computer experiments in which the output from each run is a dense time trace that is a function of some low dimensional set of explanatory variables. The analysis of these experiments has two main goals: one is to create a good predictor for the output at untested experimental points; the other is to identify the input factors that most affect the output. Two approaches for modelling the functional output are considered: one is to model each time point by the explanatory variables and the other is to smooth the time variable first and then model the other (low dimensional) variables.

In this work we focus on this second approach, namely starting by modelling the time variable. We suggest two-stage methods for modelling and predicting such data, separating the model for time from the model for the explanatory variables.

The time dependence is modelled by fitting known basis functions such as splines, as well as data derived, shape-based basis functions. Such basis functions are generated by clustering the data into similarly shaped functions and taking the mean function of each cluster as a basis function. Several methods were tested for modelling the relation to the explanatory variables. Bayesian and other models that used additional information about the data set were considered as a means of improving the fit obtained by the two-stage methods.

Two data sets were analysed using these methods: a simulation of response to chemotherapy, which yields the amount of cancer cells in a patient’s body in response to different chemotherapy treatment protocols, and a circadian rhythm simulation, showing the mRNA production and degradation throughout a sleep-wake cycle. The shape-based estimation methods proved to be efficient in both cases. Usually Kriging predicted the relation to the explanatory variables with the most accuracy, yet was not widely used in this analysis for computational reasons. The clustering method also produced good results for estimating scalar characteristics of the data, such as the cycle length in the mRNA data.
## Contents

1 Introduction .......................................................... 2

2 Prior Research ......................................................... 3

3 Methods ................................................................. 9
   3.1 Smoothing Time and Modelling Coefficients .................. 9
      3.1.1 Smoothing Time .............................................. 9
      3.1.2 Modelling by Shape ......................................... 15
      3.1.3 Modelling Curve Parameters ............................... 22
      3.1.4 Clustering and Fitting Separate Models For shape and Parameters ............................................. 25
      3.1.5 Improving the Fit by Using Constraints .................. 27
      3.1.6 Placing Constraints ......................................... 35
   3.2 Modelling Time Points and Smoothing Coefficients .......... 39
   3.3 Evaluating Goodness of Fit ...................................... 40
      3.3.1 Measuring Shape Similarity ................................. 41
      3.3.2 Estimating Goodness of Fit ................................. 43
   3.4 Identifying Outliers ............................................. 44

4 Application to Chemotherapy Simulator .......................... 48
   4.1 Smoothing the Time Trace and Estimating Coefficients ...... 49
      4.1.1 Smoothing by B - Splines .................................. 51
      4.1.2 Shape Derived Estimation ................................... 63
      4.1.3 Modelling time points and estimating coefficients ..... 87
      4.1.4 Summary ..................................................... 87

5 Application to Circadian Rhythm Simulator\(^1\) .................. 97
   5.1 Smoothing Time .................................................. 99
      5.1.1 Selecting the Number of Clusters ......................... 116
      5.1.2 Estimating Biological Quantities .......................... 117

6 Concluding Remarks ................................................ 121

\(^1\)This model was constructed with the kind help of Dr. Dorit Shweiki from the Bioinformatics department of the Academic College of Tel Aviv Yaffo.
1 Introduction

Computer simulated experiments can be a natural substitute to physical experiments, when the physical experiment is too expensive, too complicated or impossible to perform. When this is the case, a computer simulation would require as input values for all parameters that are included in the experiment, and result in a value for one or more response variables. If the simulated phenomenon is complex, this simulator may be expensive to run, which raises the need for a mathematical model to replace it. Such a model should enable prediction at untried experiment points, as well as learning about what effect, if any, each input variable has on the response variable.

Such a simulator will differ from a physical experiment if it lacks random error. Unless designed to include some stochastic factor, a computer code will be deterministic, so repeated runs with the same input values will always produce the same response value. In this case, the error that exists in computer experiments is the distance between the true and estimated values of the response variable, and is the result of a model that does not fully suit the data.

This work is concerned with experiments in which the input of each run consists of a number of parameter values and the output is a function of time. As opposed to a single scalar output, or even multidimensional output (of relatively small dimension), our aim is to estimate the value of that function at each point of the time domain, given the values of the other parameters. For practical purposes, it can be assumed that the output will have the form of a vector, each coordinate corresponding to a discrete point of time on a pre-determined scale. However, this output vector is bound to be of very high dimension, so modelling each coordinate, or the function value at each discrete time unit, is not practical. Moreover, modelling the output as a function of time will allow deeper understanding of the simulated phenomenon, as will be discussed later in the text regarding a specific
experiment.

An example of such an experiment, discussed later in this work, concerns the effect of various chemotherapy treatment protocols on the cancer cell dynamics. In this example the low dimensional input consists of 4 explanatory variables and the output is a long vector showing the number of cancer cells along time. The data are generated by a set of differential equations that simulate the biological processes of cell reproduction and destruction.

2 Prior Research

Research regarding (computer or real-life) experiments may involve two main issues: the design and the analysis of the experiment. Since this work is only concerned with the latter part, the prior research reviewed in this section includes relevant work done on analysis of computer experiments. In particular, the work done so far describes the analysis of experiments of which the output is a single scalar, or a small number of scalars.

Sacks, Welch, Mitchell and Wynn (1989) modelled the response, a scalar $y$ that depends on a small number of explanatory variables, $x$, as a realization of a random stochastic process that includes a regression model: $Y(x) = \sum \beta_i f_i(x) + Z(x)$. In this model, $Z$ is treated as a random error, whereas it represents the deviation of the regression model from the genuine function describing the simulator. It is assumed to have zero mean and covariance $\sigma^2 R(x_i, x_j)$. Assuming $Z(x)$ to be a Gaussian process, and given a correlation function $R(x_i, x_j)$ for $Z(x)$, MLEs can be found for the regression coefficients, and $Y$ can be predicted at untried factor settings. Since it is known that repeating the experiment at the same point would result in the same output, a suitable correlation function would satisfy the condition $R(x_i, x_i)=1$. Several such functions are offered, each depending on a set of parameters to be estimated with the regression coefficients. This model is based on Kriging.
modelling of spatial data (Matheron, 1963; Cressie, 1993).

Welch, Buck, Sacks, Wynn, Mitchell and Morris (1992) again considered a similar model for one scalar response \( y \), however, in this case with a large number of variables \( \mathbf{x} = (x_1 \ldots x_n) \). Their objective, in addition to predicting the response at untried values of \( \mathbf{x} \), was to detect which variables would best explain the response \( y \). Since the number of parameters in the correlation function may be large, an algorithm is presented to efficiently approximate their values, using maximum likelihood techniques. This algorithm also screens the non-effective explanatory variables.

In detail, the correlation function used was 
\[
R(\mathbf{w}, \mathbf{x}) = \prod e^{-\theta_i |w_i - x_i|^p_i}.
\]

Due to the high dimensionality of the input, estimating \( \theta_i \) and \( p_i \) for each variable requires substantial computational effort. The approximation algorithm suggested forces some of those parameters to have a common value. At each iteration, a single variable is found that affects the likelihood function enough to have its own value. Variables for which \( \theta_i = 0 \) have no effect on the output.

Allen, Bernshtyn and Kabiri-Bamoradian (2003) modelled computer output by second order polynomial regression and Kriging, and used several experimental designs to compare these methods in terms of prediction error and bias. The authors concluded that given an experimental design that addresses bias errors, both methods produce similar prediction errors.

Kennedy and O’Hagan (2000) considered problems in which different grades of computer simulators are available. The "high-grade" code is considered to be a very accurate representation of the physical system, but is expensive and slow. Simultaneously "low-grade" code can be used, which is cheap and fast, but may be less accurate. For example, codes may differ with respect to the resolution of a finite element scheme for solving a set of differential equations. A Bayesian model was used for prediction: the output from the "high-grade" simulator was modelled as a function of the output from
the "low-grade" simulator, with an independent Gaussian process added to the regression equation as an error term. Given the prior distributions of the Gaussian process and a covariance function between outputs for each level of the simulator, the posterior distribution of the high-grade simulator output is shown to be normal, estimates are given for its parameters and so a model is constructed for the high-grade simulator output.

Bayesian forecasting using a computer simulator was modelled by Craig, Goldstein, Rougier and Seheult (2001). They used past observations and expert knowledge in order to model the prior information: past observations and the discrepancy between the simulation and the physical system. Forecasting was done in two ways: one is assigning distributions to the process components, the other is Bayes linear forecasting that adjusts the expectation and variance by past observations.

Often simulators include some parameters whose values are unknown and can be "tuned" so that the simulator output gives a good fit to the existing real life data. This process, known as calibration, will not be addressed in this work. Examples of such models are Molina, Bayarri and Berger (2005) where a traffic simulator is calibrated, and a Bayes linear predictor that was used for calibrated prediction by Goldstein and Rougier (2006). Bayarri, Berger, Paulo, Sacks, Cafero, Cavendish, Lin and Tu (2007) describe a six step procedure for computer validation, that includes modelling the relation of reality to the computer model, statistically modelling the field and computer outputs and tuning parameters while comparing computer runs with field data. Bayarri, Berger, Kennedy, Kottas, Paulo, Sacks, Cafero, Lin and Tu (in press) extend this procedure to handle functional output by taking a finite number of terms from a basis expansion of the data and treating the coefficient estimation as scalar estimation. The issue of hierarchical modelling is addressed to meet the problem of runs under different conditions, where the difference is not always quantifiable. Bayesian estimation is used to model the coefficients, assuming a Gaussian Process prior and using the
data from computer runs to obtain the posterior means. Bayarri, Berger, Cafeo, Garcia-Donato, Liu, Palomo, Parthasarathy, Paulo and Walsh (in press) further explore highly irregular functional data and the need to acknowledge and incorporate uncertainty in the inputs, and use a wavelet basis representation of the data. Wavelet coefficients are estimated by maximum likelihood.

Higdon, Gattiker, Williams and Rightley (2007) combine field observations with results of computer simulation. Interest is taken in the uncertainty in predictions. Output functions are represented by principal components basis function expansion, and the basis coefficients are modelled using a gaussian Process.

A thorough review of design and analysis of computer experiments is given by Santner, Williams and Notz (2003). In particular, the authors present Bayesian methodologies for modelling, with Gaussian processes as priors, as well as hierarchical Gaussian priors. Prediction, based on the Gaussian model, includes several ways for estimating the Gaussian distribution correlation matrix.

Another review of statistical methods that meet the tasks of designing and modelling is given by Chen, Tsui, Barton and Meckesheimer (2006). The authors include seven modelling methods for low-dimensional output. Some of these methods are described in Section 3.1.1 of this thesis: Response Surface Methodology (polynomial response), Kriging and Multivariate Adaptive Regression Splines (MARS). Other methods discussed by Chen et al. (2006) are regression trees, artificial neural networks and radial basis functions. Regression trees is a recursive algorithm, similar to MARS, that partitions the $x-$ space into disjoint sections and creates basis functions from the leaves of a binary tree that represents this partitioning. Artificial Neural Networks represent the simulator by a diagram of nodes and weighted connections between them in different layers. In the input layer the nodes are the input
variables, the output layer consists of nodes that are the response variables. At least one hidden layer exists between the input and output layers, with an activation function that defines transformation between layers thus connecting the input and output variables. Radial Basis Functions produces a model of the form $\hat{y}(x) = \beta_0 + \sum_{i=1}^{n} \beta_i b(||x - x_i||)$, where $||x - x_i||$ is the Euclidean distance between $x$ and $x_i$. Several options for choosing the basis function $b()$ are given. One choice is least interpolating polynomials, in which a polynomial regression is fitted to data from the deterministic simulator using basis functions that have the lowest degree needed to interpolate the data.

The literature reviewed so far refers to low dimensional output. Since our work concerns time traces that are high dimensional, the following section will review modelling of functional data.

Faraway (1997) models the function $y(t)$ by a functional regression model, $y(t) = \beta' \beta (t) + \epsilon (t)$, where $x$ is the vector of covariates and $\beta (t)$ is a vector of functions. These functions are chosen to minimise $\sum_{i=1}^{n} \| y_i (t) - \beta' \beta (t) \|$ with respect to the $L_2$ norm on the time domain, leading to a solution similar to the usual linear regression case. Having estimated $\beta (t)$, these functions can be plotted and replaced by a parametric form.

Govaerts and Noel (2005) discuss three approaches for modelling. The two-step linear modelling approach can be applied when all curves have a common shape. The first step is to approximate all curves by a parametric function, followed by a second step which is to model the parameters obtained for each curve. The pointwise functional regression approach follows Faraway (1997) by fitting individual models for each time point, and the smoothed functional regression approach approximates each curve as a linear combination of some basis functions and then estimates the relevant coefficients. In addition the authors address the issue of significance testing of the experimental factors.
Gervini and Gasser (2004) construct a semiparametric model using warping functions, which allow curve registration without landmark identification. Warping functions are transformations of the time variable that are used for curve registration - that is the alignment of the curve according to some feature or landmark. The curves are modelled as linear combinations of a small number of linear components that are estimated from the data. Each such component is a linear combination of B-Spline basis functions, and is common to all functions in the data. The linear coefficients of the combination are individually estimated for each function.

Ramsay and Dalzell (1991) show how L-Splines can support generalizations of linear modelling and principal components analysis to samples drawn from random functions.

Drignei (2006) uses a two-stage analysis, first modelling a subset of time points together with the explanatory variables as a discrete-time Brownian bridge and then modelling other time points conditioned on the results of the first stage. In more detail, due to the smoothness of the data, the functions are assumed to be approximately locally linear. Therefore each function can be broken into several near-linear functions. Modelling these linear sub-functions requires the exact function value at the endpoints only, thus reducing the number of runs. The discrete time Brownian bridge models then serve as a prior distribution for the complete data in each segment, and so in the second stage of the model MLEs are derived based on a Bayesian model. This method is demonstrated on a geophysical computer simulation.

The aim of our work is to combine ideas from both sections, and model data which are both computer generated and time dependent. The data that we consider are computer generated and lack random error (in the sense that repeated runs with the same input will produce the same output). The functions we analyse are smooth functions with high density in the time dimension, so they are suitable for modelling by regression on smooth basis
functions. We present methods based on multi-stage modelling, incorporating some of the ideas mentioned above with new suggestions for selecting basis functions, registration and parameter estimation.

3 Methods

The aim of this work is to analyse time traces $y(t; x)$ where $x$ is the input to the simulator and the $t_i$ are the time points for which data are obtained. Regarding and treating time as any one of the other explanatory variables is problematic because of its high density compared to the other explanatory variables. Two approaches for modelling the data will be considered, each consisting of the same two stages: one stage is modelling the effect of the time variable and another is modelling the effect of the other, low density, explanatory input variables. The difference between the two approaches is the order by which the models are constructed. The methods presented in section 3.1 begin by smoothing the data over time using a compact representation of the form $y(t) = \sum \beta_i(x) f_i(t)$. They then proceed by modelling the coefficients $\beta_i(x)$ as a function of $x$. The method presented in section 3.2 fits a time dependent model of the form $y(t) = \sum \beta_i(t) x_i$ at each point $t_i$, and proceeds by fitting a curve to each set of coefficients $\beta_i(t)$.

3.1 Smoothing Time and Modelling Coefficients

3.1.1 Smoothing Time

In this section a review is given of known parametric smoothing methods as well as some new suggestions. For most smooth functions that have no unique features there is no obvious choice of an optimal smoother. However, functional characteristics for which each method is preferable will be pointed out. The error terms in these models that will be presented in this section
are not stochastic errors, but approximation errors reflecting the difference between the simulated data and the approximating function. The simplest method of estimation is to fit a model to the data at each $x$ by ordinary least squares regression.

**Polynomial Regression**  (Section 2.1 in Efroymovich, 1999) The simplest form of polynomial regression is the following model: $y(t; x) = \sum_{j=0}^{J} \beta_j(x)t^j + \text{Error}$.

However, in many cases an orthonormal polynomial basis

$$\left\{ \varphi_j(t) = \sum_{i=0}^{j} \gamma_j t^i \mid j = 0, 1, \ldots, J \right\},$$

resulting in the equivalent model

$$y(t; x) = \sum_{j=0}^{J} \beta_j(x)\varphi_j(t) + \text{Error}$$

is preferred for computational reasons. The polynomial basis functions can give good, parsimonious approximations of smooth functions, yet high-order basis functions are required for approximating functions with strong local features. Moreover, even when high-order polynomials are used, local features are still hard to capture. The approximation tends to be more accurate in the center of the interval than near the edges.

**Splines**  A spline basis (section 2.10 in Hastie and Tibshirani, 1990) fits a smooth function to the data. The data are modelled by a piecewise polynomial, with the constraint that the pieces should link smoothly at the breakpoints, called knots. For the popular case of cubic splines, and given $p$ knots $\xi_1 < \ldots < \xi_p$ the model would be:

$$y(t_i; x) = \sum_{m=0}^{3} \beta_m(x)t_i^m + \sum_{j=1}^{p} \theta_j(x)(t_i - \xi_j)_+^3 + \text{Error}.$$ 

Therefore, $p+4$ parameters need to be estimated for each $x$ in order to construct this function. One important difference between simple polynomials
and splines is that splines are suitable for analysing functions with local features, even though the basic spline functions are polynomials.

**Fourier Series**  The *Fourier Transform* (part II in Burke-Hubbard, 1998) is suitable for analysing periodic functions of time. It detects the dominant frequencies that compose the signal by decomposing the function into an infinite sum of its sine and cosine elements. Suppose \( f(t; x) \) satisfies the following conditions:

1. \( \int_{-\infty}^{\infty} |f(t; x)| \, dt \) exists,

2. The number of discontinuities in \( f(t; x) \) is finite,

3. \( f(t; x) \) has bounded variation,

Then \( f(t; x) \) can be represented as follows:

\[
f(t; x) = \frac{1}{2} a_0(x) + \sum_{j=1}^{\infty} (a_j(x) \cos(2\pi j t) + b_j(x) \sin(2\pi j t))
\]

where

\[
a_j(x) = \int_{-\infty}^{\infty} f(t; x) \cos(2\pi j t) \, dt
\]

and

\[
b_j(x) = \int_{-\infty}^{\infty} f(t; x) \sin(2\pi j t) \, dt
\]

When analysing a finite data set, the Fourier Transform will produce a finite set of frequencies, limiting high frequencies to half the sampling frequency. Consider a data set that consists of \( 2N \) observations measured at times \( t_0, \ldots, t_{2N-1} \) over a period of length \( L \). Assuming that the function has the same value at the beginning and at the end of the period, the above representation takes the following form:

\[
y(t_i; x) = \frac{1}{2} a_0(x) + \sum_{j=1}^{N-1} (a_j(x) \cos \frac{2\pi j t_i}{L} + b_j(x) \sin \frac{2\pi j t_i}{L}) + \frac{a_N}{2} \cos \frac{2\pi N t_i}{L}
\]
where the coefficients \( a_j(x) \), \( j = 0, \ldots, N \) and \( b_j(x) \), \( j = 1, \ldots, N - 1 \), are estimated by

\[
a_j(x) = \frac{1}{N} \sum_{i=0}^{2N-1} y(t_i; x) \cos \frac{2\pi j t_i}{L}
\]

and

\[
b_j(x) = \frac{1}{N} \sum_{i=0}^{2N-1} y(t_i; x) \sin \frac{2\pi j t_i}{L}
\]

If the function doesn’t have the same value at the beginning and the end of the period, it could be forced to have that quality by adding a mirror image of the period to the function, thus doubling its length. In this case a basis of cosine functions is suitable for describing the time trace.

In practice \( f(t; x) \) may be estimated by a smaller number of coefficients and frequencies, where small coefficients are considered negligible and are omitted from the model and replaced by an error term that accounts for approximation error.

As mentioned, the Fourier Transform is useful for periodic functions, or for functions that decay fast enough to have finite integrals. In these cases, a number of relatively large coefficients will point out which frequencies are the most essential in reconstructing the signal, leaving a remainder term that holds the approximation error caused by omitting the negligible coefficients. In addition, the Fast Fourier Transform algorithm makes finding these coefficients computationally feasible. The Fourier Transform is less useful when the signal changes suddenly, or has singular points.

**Wavelet Decomposition**  Extending the Fourier Transform, Wavelets (Abramovich, Baily and Sapatanis, 2000; Burke-Hubbard, 1998) are another way of expanding a function using a set of orthogonal basis functions. Each family of these basis functions is generated by two parent wavelets, a scaling (father) wavelet \( \phi(t) \) and a mother wavelet \( \psi(t) \), using the relationship:

\[
\phi_{j_0 k}(t) = 2^{j_0} \phi(2^{j_0} t - k), \psi_{j k}(t) = 2^{j} \psi(2^{j} t - k), j = j_0, j_0 + 1, \ldots; k \in \mathbb{Z}
\]
for some integer $j_0$. The wavelet expansion of a function $f(t; x)$ would then be

$$f(t; x) = \sum_{k \in \mathbb{Z}} c_{j_0 k} \phi_{j_0 k}(t) + \sum_{j=j_0}^{\infty} \sum_{k \in \mathbb{Z}} w_{j_0 k} \psi_{j k}(t)$$

where $c_{j_0 k}$ and $w_{j k}$ are the inner products of $f(t; x)$ with $\phi_{j_0 k}(t)$ and $\psi_{j k}(t)$, respectively. The basis functions generated this way have the same shape, but not the same scale and displacement.

Practical analysis of data requires adaptation to a finite number of discrete data points and a limited number of coefficients to be estimated. Let us assume that the data set consists of $n = 2^J$ points. (Note that if $n$ is not some power of 2, estimating the coefficients becomes computationally costly. It is possible to force the data to have $2^J$ points by artificially continuing the time trace, either by duplicating and mirror imaging the last section, duplicating the beginning of the time trace in a cyclic manner or adding 0’s at the end.) Limiting the number of coefficients can be done either by truncating the series of coefficients or by thresholding and omitting relatively small coefficients. Taking $j_0 = 0$ (an assumption widely used in software implementations), a truncated model has the form:

$$\hat{f}(t; x) = \hat{c}_0(x) \phi(t) + \sum_{j=0}^{M} \sum_{k=0}^{2^j-1} \hat{\omega}_{j k}(x) \psi(t) + \text{Error}$$

for some $M < J$, where the coefficients are estimated by:

$$\hat{c}_0(x) = \frac{1}{n} \sum_{i=1}^{n} \phi(t_i) y(t_i; x)$$

and

$$\hat{\omega}_{j k}(x) = \frac{1}{n} \sum_{i=1}^{n} \psi_{j k}(t_i) y(t_i; x).$$

Wavelet analysis of a function would yield information regarding both the frequency and location, whereas the Fourier Transform only specifies frequencies. Wavelets are more suitable for detecting irregularities in a function’s behaviour than the Fourier Transform. For practical purposes, the Wavelet
decomposition is expected to produce better results than the Fourier Transform when the estimated function has strong local features such as discontinuities or singular peaks, or periodic behaviour limited to a particular range of time. The Fourier Transform will fit well when the underlying function has some periodic qualities along the entire time range, and has smooth low order derivatives. The Wavelet decomposition, on the other hand, may not fit smooth data so well. Because of the unique shape of its basis functions, smooth data may require a large number of non-zero coefficients in order to flatten the singular peaks in the basis functions.

**Some theoretical notes**  Bounds are known for the Fourier and Wavelet approximations of functions that satisfy certain conditions (see chapter 2 in Efromovich, 1999). Smooth functions will be approximated equally well by a sine and cosine basis or by a polynomial basis, otherwise a Wavelet basis is preferable. However, another issue to be considered is the number of coefficients required in order to achieve good approximation. Obviously the approximation is better when more basis functions are used. However, in this work we present methods that require separate modelling and estimation of the coefficients. These models tend to be less accurate when there are more coefficients to estimate, therefore a small basis with fewer coefficients may be in our advantage.

If a function \( f(t) \) defined on the domain \((0, T)\) is twice differentiable then there exists a constant \( C \) such that \( |\theta_j| \leq \frac{C}{j} \int_0^T \left| f^{(2)}(x) \right| dx \), where \( j \geq 1 \) and \( \theta_j \) is the \( j^{th} \) Fourier coefficient. If, in addition, \( f^{(1)}(0) = f^{(1)}(T) = 0 \) then the coefficients may decay even faster than \( O(j^{-2}) \).

A bound exists for the sum of Haar Wavelet basis coefficients that depends on the approximated function’s total variation (see chapter 2 in Efromovich, 1999). The sum of the Wavelet coefficients decreases as the resolution increases, but this property does not give a bound for the number of coefficients.
required for good approximation.

If $f^{(r)}(t)$ is bounded, and $f(t)$ is defined on a finite interval, then the partial sum of a polynomial basis has a bounded approximation error.

### 3.1.2 Modelling by Shape

Two time traces that obtain distinctly different values may look similar if they are plotted on the same, scale-free rectangle. Similarity in shape between curves can serve as a basis for estimation. The method described in this section is based on separating the estimation of a function’s shape from the estimation of its scale.

Let us define two sets of functional characteristics:

- **Scale features**, which provide information about the function’s values and the region where they lie, such as the minimum and maximum values or the function’s value at a given location $t_0$, and

- **Shape features** showing trends of ascent or descent, the location of extreme points etc.

Let $G(f(t), \alpha)$ denote a transformation that changes the scale of the curve $f(t)$ while preserving its shape, where $\alpha$ is the set of scaling parameters on which the transformation depends. For our purposes $f(t)$ could denote any curve $y(t; x)$. Each curve has its unique parameter vector $\alpha$, however it is assumed that $\alpha$ can be estimated using the set of explanatory variables $x$ only, without any prior knowledge of $y(t; x)$. Satisfying this requirement should enable the prediction of the scaling parameters for untested observations. Adopting the above notation, let $G(y(t, \bar{x}), \alpha(\bar{x})) = g(t; \bar{x})$ denote a transformation that rescales a time trace $y(t; \bar{x})$ to given scale features (for example: have given values as minimum and maximum), without changing
its shape features. Such a transformation can be, for example, a linear transformation. Setting \( \alpha = (a, b) \) and \( G(f(t), \alpha) = a \cdot f(t) + b, a > 0 \) we obtain the shape function \( g(t; x) = a(x) \cdot y(t; x) + b(x) \). For each observation \( x \), the parameters \( (a(x), b(x)) \) can be chosen such that the minimum and maximum values of \( g(t; x) \) are given, while keeping the minimum and maximum locations, trends of ascent or descent and other shape features of \( y(t; x) \).

It is assumed that the function \( G^{-1}(g(t; x); \alpha(x)) \) exists, so \( y(t; x) \) can be obtained from \( g(t; x) \) and \( \alpha(x) \).

If all curves are scale-free, then the variability between the curves may be reduced so modelling the rescaled curves is easier than modelling the original ones. In detail, rather than estimate the function \( y(t; x) \) directly, let us separate the shape estimation from the scale estimation. For an untested \( x \), first obtain a function \( \hat{g}(t; x) \) that is similar in its shape to the desired \( y(t; x) \), and then estimate the rescaler \( \alpha(x) \) in order to transform the shape function \( \hat{g}(t; x) \) to a final estimate \( \hat{y}(t; x) \).

Similarity between two time traces can serve as a criterion for assigning weights for kernel estimation as well as choosing an appropriate basis for modeling \( g(t; x) \), as will be shown in detail below. The methods described in the next section are based on the assumption that small distance between two experimental points \( x_1 \) and \( x_2 \) typically implies similar shapes \( f(t; x_1) \) and \( f(t; x_2) \).

Let \( D_f(f_1(t), f_2(t)) \) be a distance measure between two curves \( f_1(t) \) and \( f_2(t) \). Such measures are suggested in section 3.3.1 below. Let \( D(x_1, x_2) \) be an estimator for \( D_f(f(t; x_1), f(t; x_2)) \) - the distance between the curves corresponding to two sites \( x_1 \) and \( x_2 \). Such an estimator is needed for prediction, since the true distance \( D_f \) is not known for untested \( x_1 \) or \( x_2 \).

**Kernel estimation:** Increasing the shape similarity between the curves can be done by transforming either the time variable (see section 3.1.1), the
outcome variable or both, so that all curves lie in a common region. In this case a two stage estimation can be applied:

1. Transform all output curves to have common scale features. Let \( g(t; x_i) \) denote the transformed curve \( y(t; x_i) \), and let \( D(x_1, x_2) \) be an estimator for \( D_f(g(t; x_1), g(t; x_2)) \). Given an untested input site \( x \), estimate the shape function \( g(t; x) \) by \( \hat{g}(t; x) = \sum_{i=1}^{n} w_i(x) \cdot g(t; x_i) \), where \( w_i(x) \propto D(x, x_i)^{-1} \) and \( \sum w_i(x) = 1 \).

2. Rescale \( \hat{g}(t; x) \) to obtain \( \hat{y}(t; x) = \hat{y}(\hat{g}(t; x), \alpha(x)) \). Scale reconstruction methods are described in section 3.1.2.

**Shape derived basis functions:** Let \( A_1 \cup A_2 \cup \ldots \cup A_k \) denote a partition of the design region into \( k \) disjoint sub-regions. Following the assumption that small distances between experimental points implies similar shape functions, our aim is to partition the experimental space such that each section \( A_i \) consists of experimental points close enough to be represented by a single shape function, denoted \( g_i(t) \). The shape function of an untested observation \( x \) should be estimated by \( g_i(t) \), where \( x \in A_i \). However, the assignment of \( x \) to a sub-region, as well as the partition itself, are not known. While the specification of the region boundaries is not an essential part of the prediction, the assignment of \( x \) to a sub-region must be estimated from the data. Rather than specifying the sub-region’s boundaries and assigning \( x \) to a single sub-region, we leave the boundaries uncertain and estimate a weight \( w_i(x) \) reflecting the probability that \( x \) belongs to each \( A_i \). Therefore an estimate of \( g(t; x) \) will not be based on a single function \( g_i(t) \), but on a weighted combination \( \sum w_i(x) g_i(t) \), where \( w_i(x) \) is monotone decreasing with respect to some measure of distance from \( x \) to \( A_i \).

Given observations \( y(t; x_1), \ldots, y(t; x_n) \) at sites \( x_1, \ldots, x_n \), two types of estimators are required:
1. Estimator for $w_i(x)$: the practical requirement is to assign an untested $x$ to the sub-region $A_i$ in order to obtain its shape function $g_i(t)$.

2. Estimator for $g_i(t)$ for all sub-regions $A_i$, $i = 1, \ldots, k$.

Using some similarity measure $D(g_1(t), g_2(t))$ (see 3.3.1), estimate $\hat{y}(t; \vec{x})$ as follows:

1. Cluster the observed shape functions $g(t; x_1), \ldots, g(t; x_n)$ into $k$ disjoint groups $C_1, \ldots, C_k$, maximising similarity within clusters and minimising similarity between clusters. See section 3.1.4 for clustering methods.

2. Regarding the curve in each cluster $C_i$ as observations sampled from the same region $A_i$, estimate $g_i(t)$ by $\hat{g}_i(t) = \frac{1}{|C_i|} \sum_{x_j \in C_i} g(t; x_j)$.

The functions $\hat{g}_i(t)$ will serve as a set of basis functions for the models described below. Two methods are suggested for obtaining $\hat{y}(t; \vec{x})$ from $\hat{g}(t; \vec{x}_i)$. The first method is a two-stage process, shown in steps 1-4 below, where $\hat{g}(t; \vec{x}_i)$ serves to estimate $g(t; \vec{x}_i)$, and then $g(t; \vec{x}_i)$ is rescaled to obtain $\hat{y}(t; \vec{x})$. The second method is a single-stage estimation of $y(t; \vec{x}_i)$ from the model $\hat{y}(t; \vec{x}) = \sum \beta_i(\vec{x}) \hat{g}_i(t)$, where $\beta_i(\vec{x})$ are obtained using OLS and then estimated for $\vec{x}$.

1. Estimate $\pi_i(\vec{x})$, a weight reflecting a subjective probability of a new observation $\vec{x}$ being in region $A_i$. This probability can be estimated in several ways, such as:

- Discriminant analysis.
- Multiple logistic regression.
- Ordinary least squares, from a linear model $\hat{g}(t; \vec{x}) = \sum_{i=1}^k \pi_i(\vec{x}_j) \cdot \hat{g}_i(t)$. 

18
2. For an untested site \( \mathbf{x} \), estimate \( g(t; \mathbf{x}) \) by \( \widehat{g}(t; \mathbf{x}) = \sum_{i=1}^{k} w_i(\mathbf{x}) \cdot \widehat{g}_i(t) \), where \( w_i(\mathbf{x}) = \pi_i(\mathbf{x}) \).

3. Estimate \( \alpha(\mathbf{x}) \) as part of the scale reconstruction procedure that is described in more detail in section 3.1.2.

4. Rescale \( \widehat{g}(t; \mathbf{x}) \) to obtain \( \widehat{y}(t; \mathbf{x}) = G^{-1}(\widehat{g}(t; \mathbf{x}), \widehat{\alpha}(\mathbf{x})) \).

**Scale Reconstruction**  Shape based modelling may require that all functions lie on the same region. When this is not the case the observed curves \( y(t; \mathbf{x}_i) \) should be transformed into corresponding shape functions \( g(t; \mathbf{x}_i) \). Using the above notation, let \( G(y(t; \mathbf{x}), \alpha(\mathbf{x})) \) denote the transformation \( y \to g \), where \( \alpha(\mathbf{x}) \) is a set of parameters on which \( G() \) depends. The parameters \( \alpha(\mathbf{x}_i) \) can be obtained from the observed data and estimated for an untested \( \mathbf{x} \) using the methods discussed in section 3.1.3. \( \widehat{y}(t; \mathbf{x}) \) is then obtained by \( G^{-1}(\widehat{g}(t; \mathbf{x}), \widehat{\alpha}(\mathbf{x})) \). It is necessary to choose \( G() \) such that \( G^{-1}() \) exists.

Any features that the functions \( y(t; \mathbf{x}) \) have, and that can be estimated from the data with sufficient precision, can be used in rescaling \( g(t; \mathbf{x}) \). Such features may be:

- \( \min_t(y(t; \mathbf{x})) \) and \( \max_t(y(t; \mathbf{x})) \).

- \( y(t_i; \mathbf{x}), i = 1, \ldots, C \) - the function values at a given set of specific time points.

- A combination of the above.

Let \( t_1, \ldots, t_C \) denote time points at which the function \( y(t; \mathbf{x}) \) is estimated, so \( y(t_i; \mathbf{x}), i = 1, \ldots, C \) are the scale features of the function \( y(t; \mathbf{x}) \). The function values at \( t_i \), \( i = 1, \ldots, C \) are scalars that can be estimated for all \( \mathbf{x} \) in the experiment domain by any estimator suitable for modelling low
dimensional data. This estimator does not necessarily yield the same values as the curve estimated by any of the smoothing methods described in section 3.1.1. Let \( h_1(x), \ldots, h_C(x) \) denote the respective estimators of \( y \) at these time points. If \( C = 1 \) or \( C = 2 \) then it is possible to choose a linear rescaling estimator \( \hat{y}(t; x) = a_1(x) \cdot \hat{g}(t; x) \) or \( \hat{y}(t; x) = a_1(x) \cdot \hat{g}(t; x) + a_0(x) \) that satisfies: \( \hat{y}(t_i; x) = h_i(x) \) for all \( i = 1, 2 \) and for all \( x \). This way the scale reconstruction problem is reduced to estimating \( a_0(x) \) and \( a_1(x) \) from the data. Such an estimator has the advantage of keeping the original shape of the function. Note that the methods for estimating coefficients described in section 3.1.3 are suitable for obtaining both \( h_i(x) \) and \( a_i(x) \).

Interpolating \( h_i(x) \) for all \( i \) can only be done using a non-linear rescaling function, which is likely to result in loss of shape. A more complex estimator is needed for \( C > 2 \). Interpolating \( h_i(x) \) for all \( i \) may result in strong bias and loss of shape. However, a linear rescaler can still be used if the demand for interpolating \( h_i(x) \) is replaced by a weaker condition such as minimal distance between \( h_i(x) \) and \( \hat{y}(t_i; x) \). For example, select \( a_0(x) \) and \( a_1(x) \) that minimise \( \sum_{i=1}^{C} (h_i(x) - (a_0(x) \cdot g(t_i; x) + a_1(x)))^2 \).

**Transformation of the Time Variable** In some experiments the length of the output is determined by the input variables, so the data set consists of curves that have different lengths. For example, in the chemotherapy simulation that is described later, a patient is exposed to chemotherapy for a number of hours and the number of cancer cells that are present in the patient’s body is recorded during that period. The resulting output is a curve that lies in the interval from \( t = 0 \), when the drug is administered, to \( t = T(x) \), when the next dose of chemotherapy is given. \( T(x) \) is not a constant - it is one of the input parameters on which the output depends. When modelling such data it might be useful to register all curves on a common interval \([0, T_0]\).
In other experiments the time points marking the beginning and the end of the data of interest are not absolute time coordinates, but events in time. Referring to the previous example, if chemotherapy is simulated until the cells reach a certain level, then each protocol will end at a different time which is not known in advance.

Another reason for transforming the time variable is that fitting the time dependent curve may become more difficult when the data are not of equal length. Some problems that arise in such cases are ambiguity and fewer experimental points to model the curve for large \( t \).

A possible solution to this problem is to re-scale the time variable so that all time traces have equal lengths. Let us denote by \( L_i \) the length of time trace \( i \). Then replacing \( y_i(t_j) \) by \( y'_i(t'_j) = y\left(\frac{t'_j}{L_i}\right) \) would yield a set of observations all lying in the interval \([0, 1]\). However, the problem of different numbers of time points at which data are observed still remains. This obstacle can be overcome by estimating each time trace at a set of pre-determined points over the interval \([0, 1]\). Specifically, suppose each time trace will be estimated at the points \( t'_j = \frac{j}{N} \), \( j = 0, \ldots, N \). If estimation is done by interpolation, then

\[
y'_i(t'_j) = y(\lfloor \frac{j}{N} \cdot L_i \rfloor) + (y(\lfloor \frac{j}{N} \cdot L_i \rfloor + 1) - y(\lfloor \frac{j}{N} \cdot L_i \rfloor)) \cdot \left[ \frac{\frac{j}{N} \cdot L_i - \lfloor \frac{j}{N} \cdot L_i \rfloor}{N} \right].
\]

It should be noted that such interpolation is in itself some form of smoothing of the original data \( y_i(t_j) \). If data are taken over a dense grid the resulting function values \( y'_i(t'_j) \) are expected to be almost identical to the original function values \( y_i(t_j) \). Since the experiments that are discussed in this work yield high density curves over time, such interpolation is acceptable. In particular, since the methods described in this work are intended for analysing time traces in which the time variable has very high density, the interpolation suggested above is expected to maintain the time traces’ original features.

Transforming the time variable makes sense in cases where the curve’s shape does not depend on its length. If this is not the case (for example,
if we consider chemotherapy treatment protocols that are identical in all
parameters but the treatment duration) then clustering can be done within
groups of curves that have the same or similar lengths.

3.1.3 Modelling Curve Parameters

Having modelled the data by one of the time dependent functions mentioned
in the previous sections, we now have a set of parameters $\beta_0, \ldots, \beta_s$ for
each experimental point $x$. These parameters can be the basis function’s
coefficients as well as scale features or other curve characteristics that are
required for the estimation process. The next stage in modelling the data is
therefore modelling these parameters as a function of the simulator input $x$.
Methods for this stage of estimation are reviewed in this section. Throughout
this section it is assumed that each parameter $\beta_j$ is modelled separately and
independently of the other parameters.

**Polynomial Models** *Linear regression* using the simulation parameters,
possibly with transformations, as explanatory variables, would be the naïve
method of modelling the coefficients. A model of the form

$$
\beta_j(x) = \gamma_{j,0} + \sum_{i=1}^{k} \gamma_{j,i}x_i + Error
$$

is constructed for each $\beta_j$, $j = 0, \ldots, s$, producing the estimators $\hat{\beta}_j$.

A natural expansion to this model is a polynomial model of a chosen
degree $D$ that may include interactions. The degree $D$ depends on the num-
ber of observations. Thus, the estimation of the coefficients $\beta_j(x)$ takes the
following form:
Model selection techniques exist that enable the choice of a suitable subset of factors for the model, thus reducing its complexity. See, for example, section 11.3.2 in Draper and Smith (1981). It should be noted that these techniques are mostly based on significance testing and assume the errors to be independent and normally distributed, assumptions that are not valid for computer experiments. In spite of that, simpler models can be achieved if model selection techniques are applied on computer experiments, even if the interpretation of their results is different.

Kriging (Matheron, 1963) regards the output (in this case - the parameters \( \beta_j \)) as a stochastic trend component with noise, \( \beta_j(\mathbf{x}) = f_j(\mathbf{x})^T b_j + z(\mathbf{x}) \), where \( f_j(\mathbf{x}) \) are known functions and \( z(\mathbf{x}) \) is a Gaussian process that satisfies the following properties:

1. \( E(z(\mathbf{x})) = 0. \)
2. \( Var(z(\mathbf{x})) = \sigma_z^2. \) The process variance \( \sigma_z^2 \) is unknown.
3. \( R(z(\mathbf{x}_i), z(\mathbf{x}_j)) = R(\mathbf{x}_i - \mathbf{x}_j). \) A correlation exists between the process values at input sites \( \mathbf{x}_i \) and \( \mathbf{x}_j \) that depends only on the vector difference between the sites. The correlation function may depend on unknown parameters. Lack of random measurement error implies that \( R(0) = 1. \)

\( \beta_j(\mathbf{x}) \) is estimated as a linear combination of the observed parameters:
\[
\hat{\beta}_j(\mathbf{x}) = \sum_{i=1}^n c_i(\mathbf{x}) \beta_j(\mathbf{x}_i),
\]
where \( \mathbf{x}_1, \ldots, \mathbf{x}_n \) are the input sites where data were
obtained, and \( x \) is an untested site. This linear combination is constrained to be unbiased for \( E[\beta_j(x)] \). The coefficients \( c_i \) minimize \( E(\hat{\beta}_j(x) - \beta_j(x))^2 \), and can be found given the form of the correlation function \( R(\cdot) \).

The detailed models can be found in Welch, Buck, Sacks, Wynn, Mitchell and Morris (1992), as well as several options for correlation functions, and a maximum likelihood algorithm for estimating the correlation parameters.

**ACE** (Breiman and Friedman, 1985) performs nonparametric transformations \( g_j(\beta_j), f_{ij}(x_i) \) for \( j = 0, \ldots, s \) and \( i = 1, \ldots, k \), on both the input and output variables, to maximize \( R(g_j(\beta_j), \sum f_{ij}(x_i)) \), the correlation between the transformed output variable and the sum of the transformed input variables. The estimation is done for each coefficient \( \beta_j \) separately, and the transformations have to be estimated in order to allow prediction.

**Projection Pursuit Regression** PPReg (Friedman and Stuetzle, 1981) estimates the outcome variable by a sum of transformations, each made on a different linear combination of the explanatory variables:

\[
\beta_j(x) = \sum_{i=1}^{d} h_i(\sum_{l=1}^{k} a_{il}x_l) + Error
\]

The estimation is performed by an iterative algorithm. In each step of the algorithm a transformed linear combination is selected and added to the model, such that it maximises some fit criterion. The coefficients for those linear combinations are estimated and provided by the algorithm, but the transformations made on them do not have an analytic form and are provided numerically.
3.1.4 Clustering and Fitting Separate Models For shape and Parameters

For all the models described so far in the text, the same type of model was used to describe all curves in the data set. Let us now consider local models that fit different types of models to different values of \( \mathbf{x} \) in the data set. The models can differ either in the type of function that describes the time trace or in the models that estimate the function’s parameters. Local models may improve the modelling process if the variability within the curves is high and one model cannot correctly describe all the curves. It should be taken into account that some loss of information may occur in the process of separately modelling each set, since each one of these models will be based on a smaller data set.

This method requires several stages of estimation. The initial stage is to divide the experiment domain into distinct regions, each one having its unique model. A possible method for partitioning is to cluster the observed data by some common feature, and regard the curves in each cluster as observations from the same region. Once the data are clustered a model is needed that can assign a new observation \( \mathbf{x} \) to a suitable cluster and estimate its parameters accordingly. Following that, separate models should be fit to different regions.

How should the data be clustered? Possible criteria for clustering are:

- **Clustering by the coefficient vector \( \mathbf{\beta} \):** Use the same set of basis functions to describe all curves, but different models for their coefficients. In practice, perform an initial phase of estimating all coefficients from the full data set, and then separate the data set such that data points with similar coefficients will be modelled together.

- **Clustering by shape:** Shape estimation is described in section 3.1.2, as well as clustering the data by shape characteristics. Once the data are
clustered by shape, use the same set of basis functions to describe all
curves and fit a different model for the vector $\beta$ of each cluster.

**Clustering Methods**

- **K-Means Clustering**: this method was designed for clustering low di-
mensional vectors, and not functional data. It is, however, the basis for
the functional clustering described below. Given observations $y_1, \ldots, y_n$
and the desired number $k$ of clusters

1. Select $k$ arbitrary points $m_1, \ldots, m_k$ as cluster centroids. Let
   $C_1, \ldots C_k$ be the respective clusters.
2. Assign each observation $y_i$ to the cluster $j$ that minimises a dis-
tance $|y_i - m_j|$. 
3. Update cluster centroids such that $m_j$ is the centroid of $\{y_i | i \in C_j\}$
4. Repeat steps 2-3 until the centroids remain unchanged.

- **Functional Clustering**: The following algorithm, based on the method
described by Tarpey (2003), produces $k$ clusters of functions from a
given set of functions $f_1, \ldots, f_n$, all defined on the same domain. Let
$D_f(f_1, f_2)$ be some distance measure between the functions $f_1$ and $f_2$,
for example $D_f(f_1, f_2) = \int (f_1 - f_2)^2$.

1. Select $k$ arbitrary functions as cluster centroids. Denote those
functions by $m_1, \ldots, m_k$, and denote by $C_j$ the respective sub-
set of functions assigned to cluster $j$. Different choices of initial
centroids may result in different solutions to the clustering prob-
lem. However, the different solutions are expected to be of similar
quality.
2. Assign each function $f_i$ to the cluster $C_j$ whose mean function $m_j$
has the minimal distance $D_f(f_i, m_j)$. 

26
3. Update the cluster means: 

\[ m_j = \frac{1}{|C_j|} \sum_{i \in C_j} f_i. \]

4. Repeat steps 2-3 until cluster assignments remain unchanged.

Prediction from models that are based on clustering requires good identification of what cluster a new observation \( x \) belongs to. Once the training data are clustered it is possible to model cluster assignment by \( x \) by discriminant analysis or other methods.

### 3.1.5 Improving the Fit by Using Constraints

A different approach for estimating the function’s parameters \( \beta_0, \ldots, \beta_s \) at untested protocols was considered, which used more of the information produced by the simulations. This approach was motivated by the chemotherapy simulation that is described in more detail later in the text. A time dependent function \( y(t; x) \) was estimated by the smoothing methods mentioned in section 3.1, at a set of new input values \( x \), yielding estimates \( \hat{y}(t; x) \). The estimates were then compared to the true functions \( y(t; x) \). It was noticed that in some cases the shape of the function is reconstructed quite successfully, but some shift in the function’s values places the entire estimate lower or higher than the true function. In these cases forcing the estimated function to have the same value as the real function at a certain time point \( t^* \) could place the entire estimate closer to the real function, and reduce the overall error. The true value of the function is of course unknown, but it can be estimated using any one of the methods mentioned in section 3.1.3. In particular, \( y(t^*; x) \) is a scale feature and can be estimated by some estimator \( h(x) \) as described in section 3.1.2. The use of constraints enables incorporating information about the function’s values at given points, regardless of the basis functions selected or method used for estimating the model’s parameters.

Consider the two estimates \( \hat{y}(t^*; x) \) (obtained as part of the overall curve estimation) and \( h(t^*; x) \) (which estimates the value at the single time point
the value of an untested protocol $x$ at $t^*$, regardless of the entire curve) of the value of an untested protocol $x$ at $t^*$. The benefit of using $h(t^*; x)$ as a constraint depends on its quality as an estimator of $y(t^*; x)$. If this estimate is closer than $\hat{y}(t^*; x)$ to the true function value, the function estimation at $t^*$ can be improved. In addition, if the estimate $\hat{y}(t; x)$ is shifted from the true function by a constant, then forcing the constraint $\hat{y}(t^*; x) = h(t^*; x)$ will shift the entire estimated function, for all $t$, towards the true values. As the true value at $t^*$ is not known, and it is not possible to verify that $h(t^*; x)$ is indeed closer to the true value than $\hat{y}(t^*; x)$, it is impossible to guarantee that this constraint will indeed improve the estimation. However, some preliminary tests can be performed in order to assess the benefit of using the estimate at $t^*$:

- Compare the overall fit of the training data set with and without using the constraint. Goodness of fit measures are suggested in section 3.3.2.

- Use the training data set to compare the correlation between the true value at $t^*$ and its estimated value without the constraint, to the correlation yielded by the suggested regression. Namely, compare $\text{Corr} (y(t^*; x), h(t^*; x))$ and $\text{Corr} (y(t^*; x), \hat{y}(t^*; x))$ where $t^*$ is fixed and the correlation is measured for all $x$ in the training data. Alternatively, in order to avoid errors caused by a large constant difference between $y(t^*; x)$ and $h(t^*; x)$, compare the sum of squared differences $\sum (y(t^*; x) - \hat{y}(t^*; x))^2$ and $\sum (y(t^*; x) - h(t^*; x))^2$.

- Compute the response function on a test set of $x'$s, not used in the initial modelling, and apply the criteria mentioned above on the estimated functions with and without constraints.

A constraint metric Let us assume that the $s+1$ characteristic coefficients of the function used to describe the data, $\beta_{i,0}, \ldots, \beta_{i,s}$, were retrieved from the data for $i = 1, \ldots, n$ and that estimators $\hat{\beta}_I(x) = \hat{\gamma}_{0,I} + \sum_{j=1}^{k} \hat{\gamma}_{j,I}g_j(x)$ were
constructed for \( l = 0, \ldots, s \) using the following multivariate linear regression model:

\[
\begin{pmatrix}
\beta_{1,0} & \cdots & \beta_{1,s} \\
\vdots & & \vdots \\
\beta_{n,0} & \cdots & \beta_{n,s}
\end{pmatrix}
= 
\begin{pmatrix}
1 & g_1(x_1) & \cdots & g_k(x_1) \\
\vdots & \vdots & & \vdots \\
1 & g_1(x_n) & \cdots & g_k(x_n)
\end{pmatrix}
\begin{pmatrix}
\gamma_{0,1} & \cdots & \gamma_{0,s} \\
\vdots & & \vdots \\
\gamma_{k,1} & \cdots & \gamma_{k,s}
\end{pmatrix}
+ 
\begin{pmatrix}
\text{Error}_{1,0} & \cdots & \text{Error}_{1,s} \\
\vdots & & \vdots \\
\text{Error}_{n,0} & \cdots & \text{Error}_{n,s}
\end{pmatrix}
\]

where each \( \beta_l \) is estimated separately using ordinary least squares estimation. Given some untested value of \( x \) and estimated coefficients \( \hat{\beta}_l(x) \), our aim is to find \( \beta^* \) values that are the closest to the \( \hat{\beta} \) values estimated by the linear regression subject to matching the constraint at \( t^* \). This raises the question how distance between two \( \beta \) vectors should be measured. A logical solution is to use the Mahalanobis distance between \( \hat{\beta}(x) \) and \( \mu(x) \). Let \( \epsilon_i \) be the vector of residuals for coefficient \( i \), such that \( \epsilon_{i,j} = \beta_{i,j} - \hat{\beta}_{i,j}, i = 1, \ldots, n, j = 0, \ldots, s \). The estimated covariance matrix \( S \), where \( S_{i,j} = \text{Cov}(\epsilon_i, \epsilon_j) \), defines a metric on which a Mahalanobis region, surrounding the original estimates, can be based. This metric has desirable properties for our purposes: relatively large values on the main diagonal of \( S \) suggest poor estimation of the corresponding coefficient, so modifying these coefficients should be liberally enabled. In addition, large positive or negative values outside the main diagonal imply that simultaneous modification of the corresponding coefficients could be more suitable than modifying each of the individual coefficients. This Mahalanobis region for \( \mu \), the vector of coefficients, has the form:

\[
(\mu - \hat{\mu})^T S^{-1} (\mu - \hat{\mu}) \leq \text{const}
\]

So we define the required coefficients \( \beta^* \) as the solution to the following optimization problem

\[
\min_{\hat{\beta}} (\beta - \hat{\beta})^T S^{-1} (\beta - \hat{\beta})
\]
s.t.
\[ \hat{y}(t^*; x; \hat{\beta}) = h(t^*; x) \]

where \( \hat{y}(t^*; x; \hat{\beta}) \) is the function estimating the data using the parameters \( \beta(x) \) and \( h(t^*; x) \) is the direct predictor of the response at \( t^* \) that was defined earlier in this section. The notation \( \hat{y}(t^*; x; \hat{\beta}) \) is used here instead of the previous notation \( \hat{y}(t^*; x) \) in order to emphasize the dependence on the specific estimates \( \hat{\beta} \).

Let us consider the case \( \hat{y}(\beta; t) = \beta'P(t) \), where \( \beta \) are linear coefficients of some basis \( P(t) \). Using Lagrange multipliers, we obtain the optimal solution

\[ \beta = \lambda SP(t^*) + \hat{\beta} \]

where

\[ \lambda = \frac{h(t^*; x) - P'(t^*) \hat{\beta}}{P'(t^*) SP(t^*)} \]

**Using multiple constraints** One way of expanding this method is to use more than one constraint, i.e. force the estimated function to have given values at more than one point of time. Another is using soft constraints - modifying the optimization process so that the objective function will have some penalty for large discrepancies between the estimated function and the direct estimate of the function value at \( t^* \). The optimization problem with \( p \) soft constraints takes the following form

\[
\min_{\hat{\beta}} (\beta - \hat{\beta})' S^{-1} (\beta - \hat{\beta}) + \sum_{j=1}^{p} \lambda_j (\hat{y}(\beta; t^*_j) - h(t^*_j; x))^2
\]

where \( \lambda_j \) is a weight assigned to constraint \( j \). These weights determine both the relative importance of the constraint (the higher the weight - the bigger the penalty for large distances in the objective function), and the relative importance of the main term, determining the permitted distance from the
original coefficients. Selection of the constraint sites $t_j^*$ and the weights $\lambda_j$ will be discussed in section 3.1.6.

**Gradient Projection Optimisation** Gradient Projection (Bertsekas, 1999) is a greedy algorithm for finding optimal solutions to (possibly constrained) convex objective functions like those presented for adding constraints to the function estimation. Consider the optimisation problem

$$\min_{\mathbf{x}} f(\mathbf{x})$$

s.t.

$$\mathbf{x} \in C$$

where $f(\mathbf{x})$ is convex and $C$ is a compact and convex set. Under these conditions $f(\mathbf{x})$ is guaranteed to have a (not necessarily unique) minimum. The minimum is found by an iterative search. In each iteration of this search the suggested solution is modified by a predetermined factor. The direction of this reduction is the gradient - the fastest descent direction. However, if this decrease in $\mathbf{x}$ results in violating the constraint ($\mathbf{x} \notin C$), the new solution will be projected on $C$. Therefore, step $k + 1$ of the search is:

$$\mathbf{x}_{k+1} = P_C(\mathbf{x}_k - \delta \cdot \nabla f(\mathbf{x}_k))$$

where $P_C(\mathbf{x})$ is the projection of $\mathbf{x}$ on $C$.

This optimisation method was used to find coefficients in the Mahalanobis region that minimise the distance between $\widehat{y}(\beta; t^*)$ and $h(t^*; \mathbf{x})$. The optimisation problem is:

$$\min_\beta (\widehat{y}(\beta; t^*) - h(t^*; \mathbf{x}))^2$$

s.t.

$$(\beta - \hat{\beta})' S^{-1} (\beta - \hat{\beta}) \leq \gamma$$

31
where $\gamma$ is the constant corresponding to the required volume of that region.

Some duality exists between the two optimisation methods. Let us replace the strong constraint in the metric constraint problem by a weaker constraint $(\tilde{y}(\beta; t^*) - h(t^*; \mathbf{x}))^2 \leq \epsilon$. This way, the two optimisation problems are stated as follows:

$$\min_\beta (\beta - \hat{\beta})' S^{-1}(\beta - \hat{\beta})$$

s.t.

$$(\tilde{y}(\beta; t^*) - h(t^*; \mathbf{x}))^2 \leq \epsilon$$

and,

$$\min_\beta (\tilde{y}(\beta; t^*) - h(t^*; \mathbf{x}))^2$$

s.t.

$$(\beta - \hat{\beta})' S^{-1}(\beta - \hat{\beta}) \leq \gamma$$

The problems are dual since each problem’s constraint is the other’s objective function. Therefore, for any value of $\epsilon$ it is possible to select $\gamma$ such that both problems will result in the same optimal solution $\beta_{\text{opt}}$.

Expanding this method to the case of function estimation at $p$ points of time can be done by summing the relevant distances in the objective function:

$$\min_\beta \sum_{j=1}^{p} \lambda_j (\tilde{y}(\beta; t_j^*) - h_j(t_j^*; \mathbf{x}))^2$$

s.t.

$$(\beta - \hat{\beta})' S^{-1}(\beta - \hat{\beta}) \leq \gamma$$

Both the Mahalanobis region and Gradient Projection optimisation methods require knowledge of the weights $\lambda_j$. Two methods for determining these weights are suggested:
1. Weights reflecting the relative importance of a constraint: use the training set to test how knowing the function’s value at a certain \( t^* \) improves the overall fitted function. Substitute \( h_j (t_j; \mathbf{x}) \) in the above model by the true values \( y(t_j; \mathbf{x}) \). \( \lambda_j \) is proportional to the improvement in the overall fit, compared to using no constraints.

2. Weights reflecting the accuracy of \( h_j (t_j; \mathbf{x}) \) as an estimator of \( y(t_j; \mathbf{x}) \):
   \( \lambda_j \) is proportional to a goodness of fit measure for \( h_j (t_j; \mathbf{x}) \); for example,
   \[ \lambda_j \propto \frac{1}{S^2_j}, \]
   where \( S^2_j \) is the cross-validation estimator of the prediction error for \( h_j (t_j; \mathbf{x}) \).

**A Bayesian Model for Constraints**  As the computer simulations discussed in this chapter lack random error, the conventional theoretical justification to describe them with a probabilistic model does not hold. However, presenting parts of the data as a realisation of some random process enables the use of statistical and probabilistic tools that could lead to better description and understanding of the data. This idea has been successfully used for modelling data from computer experiments. Examples for Kriging models for computer experiments are Currin, Mitchell Morris and Ylvisaker (1991), modelling a simulated thermal energy storage device and Sacks, Welch, Mitchell and Wynn (1989), modelling the clock asynchronisation in a circuit simulator. In these and other examples the deterministic output is modelled using a stochastic formulation. More examples where Kriging proved to be a good estimation method can be found in Neumann Ben-Ari and Steinberg (in press).

Let \( \hat{\beta}(\mathbf{x}) \) denote an estimator for \( \beta(\mathbf{x}) \) obtained by any one of the methods described in section 3.1.3, so that the real data \( y(t; \mathbf{x}) \) almost exactly equals \( \hat{\beta}(\mathbf{x})^T P(t) \), and \( \hat{\gamma}(t; \mathbf{x}) = \hat{\beta}(\mathbf{x})^T P(t) \) are the estimates of the time traces. Let \( h(\mathbf{x}) \) be a scale feature - an estimator for \( y(t_i; \mathbf{x}) \) at some given time point \( t_i \), as defined in section 3.1.2. The following model regards the es-
Estimator \( \hat{\beta}(x) \) as an observation from a normal distribution with mean \( \mu(x) \) and some covariance matrix \( A^{-1} \). Specifically, let us assume the following conditional distributions:

1. \( \hat{\beta} \mid \beta \sim N(\beta(x), A^{-1}) \), where \( A^{-1} \) can be estimated from the data, given the true \( \beta(x) \) values for the training set by the sample covariance matrix.

2. \( h(x) \mid \beta \sim N(\beta(x)^T P(t_i), \lambda^{-1}) \), where \( \lambda^{-1} \) reflects the amount of certainty we have in \( h(x) \) as a direct measure of \( \beta(x)^T P(t_i) \). A small value for \( \lambda^{-1} \) means high certainty that \( h(x) \) and \( \beta(x)^T P(t_i) \) are close. It follows that \( \lambda \) measures the strength of the constraint at \( t_i \): higher values for \( \lambda \) imply a stronger constraint.

3. In addition let us assume an improper prior distribution for \( \beta(x) \) so that \( P(\beta(x)) \propto 1 \), and that \( \hat{\beta} \mid \beta \) and \( h(x) \mid \beta \) are independent.

A Bayesian estimator for \( \beta \) is the posterior mean \( E(\beta(x) \mid \hat{\beta}, h(x)) \). Standard results for combining information from normal distributions give

\[
E(\beta(x) \mid \hat{\beta}, h(x)) = \left( A + \lambda P(t_i) P(t_i)^T \right)^{-1} \left( A^{-1} \hat{\beta}(x) + \lambda h(x) P(t_i)^T \right)^T
\]

**Equivalence Between Constraints and Bayesian Model** The Mahalanobis metric used in modelling the constraints implies setting quadratic constraints on the coefficients. The constraints can be regarded as representing prior knowledge about the coefficients, and expressed as a normal distribution. Thus adding the constraints can be viewed in the context of Bayesian estimation.

The Bayesian model described above leads to a Normal posterior density function. The Bayes estimator - the posterior mean - is in this case also the posterior mode and can be obtained by maximising the posterior
density with respect to $\beta$. Equivalently, it can be obtained by minimising $Q(\beta)$, where the posterior density is written in the form $c \cdot \exp\{-2Q(\beta)\}$.

Combining the distributional terms from (1) and (2) above gives

$$Q(\beta) = \left(\beta(x) - \hat{\beta}(x)\right)^T S^{-1} \left(\beta(x) - \hat{\beta}(x)\right) + \lambda \left(\beta(x) - \beta^T P(t)\right)^2$$

which is equivalent to the constraint model.

### 3.1.6 Placing Constraints

The use of constraints raises the question of how the points $t_j^*$ and the corresponding weights $\lambda_j$ should be chosen. Let us define a constraint as a $2 \cdot k$ array $(t^*, \Delta)$, where $k$ denotes the number of points at which the function is constrained. Our aim is to find constraints that will improve the overall fit of the test set as much as possible.

The following local search algorithm was used in order to approximate the optimal constraint, using the training set for which the true function is known.

**Local Search Algorithms - Threshold Acceptance** Two commonly used local search methods are simulated annealing (Aarts, Korst, and van Laarhoven, 1997) and threshold acceptance (Dueck and Scheuer, 1995). Consider the optimisation problem:

$$\max_{s \in S} \text{fitness}(s),$$

where $\text{fitness}(s)$ denotes the function to be maximised (replace $\text{fitness}(s)$ by $-\text{fitness}(s)$ for a minimisation problem), and $S = \{s_1, s_2, \ldots\}$ is the set of all feasible solutions to the problem. It is assumed that $S$ is finite or countable, a reasonable assumption since in this application both the time and the level of constraint are sampled from a discrete grid. Local search algorithms are iterative approximation methods for optimisation problems. Let $N(s) = \{s_{i1}, s_{i2}, \ldots\}$ denote the *neighbourhood* of a solution $s$ - a subset
of \( S \) that consists of solutions that are slight modifications of \( s \). Finally, let \( s_{\text{best}} \) denote the solution that has the best fitness value of all solutions tested by the algorithm. Note that \( s_{\text{best}} \) is not necessarily the optimal solution, but serves as an approximation to it. The local search algorithm follows these steps:

1. Select an initial solution \( s_0 \) at random. Set \( s_{\text{best}} = s_0 \)
2. Select a solution \( s_{\text{temp}} \in N(s_{\text{best}}) \) at random
3. If \( \text{fitness}(s_{\text{temp}}) > \text{fitness}(s_{\text{best}}) \) set \( s_{\text{best}} = s_{\text{temp}} \).
4. Repeat steps 2-3 until no improvement in the fitness has been obtained for \( M \) consecutive iterations.
5. Approximate the optimal solution by \( s_{\text{best}} \)

Step 3 of the algorithm guarantees that the estimated solution improves the fitness at each iteration. Once a solution is found such that its entire neighbourhood produces lower fitness values, it will be set as the approximate optimal solution. The algorithm described above has the fault of finding local optima and not necessarily global ones. Some variations of the basic algorithm exist that are designed to amend this drawback by allowing the occasional acceptance of solutions that lower the fitness value, thus enabling different search directions that may lead to a better optimal solution approximation. Threshold Acceptance is one of these variations: let \( th_0 > th_1 > \ldots > 0 \) be a descending set of positive thresholds.

1. Select an initial solution \( s_0 \) at random. Set \( s_{\text{best}} = s_0 \).
2. Set \( th = th_0 \).
3. Select a solution \( s_{\text{temp}} \in N(s_{\text{best}}) \) at random.
4. If $\text{fitness}(s_{\text{temp}}) > \text{fitness}(s_{\text{best}}) - th$ set $s_{\text{best}} = s_{\text{temp}}$.

5. Update threshold: $th = th_{i+1}$ every $k$ iterations.

6. Repeat steps 3-5 until no improvement in the fitness has been obtained for $M$ consecutive iterations.

7. Approximate the optimal solution by $s_{\text{best}}$.

Adapting the threshold acceptance algorithm to a specific optimisation problem requires the definition of the following elements:

- The form of a solution.
- The neighbourhood of a solution.
- The fitness function.

The algorithm was implemented in order to approximate the optimal placing of constraints. The search was limited to a discrete grid by allowing the constraint to be placed only at time points that lie in an interval $[t_{\text{min}}, t_{\text{max}}]$ and have the form $t_{\text{min}} + i \cdot \Delta t$, where $t_{\text{min}}$, $t_{\text{max}}$ and $\Delta t$ were pre-defined parameters. Similarly, $\lambda$ was limited to an interval $[\lambda_{\text{min}}, \lambda_{\text{max}}]$ and to the form $\lambda_{\text{min}} \cdot (\Delta \lambda)^i$ for pre-defined parameters $\lambda_{\text{min}}$, $\lambda_{\text{max}}$ and $\Delta \lambda$. Both $t$ and $\lambda$ had upper bounds for practical computational purposes.

The solution to the constraint problem is defined as a 2 by $k$ array $(\ell^*; \lambda)$, where $k$ denotes the maximum number of points at which the function is constrained. Limiting $k$ prevents constraints from leaving no degrees of freedom and practically disregarding the original unconstrained solution. The algorithm, as implemented in this case, doesn’t force $k$ active constraints, but regards any solution that has at most $k$ constraints as feasible. Because of that the risk of over-constraining is low. In the examples that will be
presented later in this text $k$ was set as half the number of estimated parameters, yet the algorithm suggested using less constraints. The motivation for this initial choice was that the constraints should not affect the estimates more than the original estimation procedure, so at most half of the model’s "degrees of freedom" are assigned to them.

The **neighbourhood** of a solution $s$ consists of all solutions that are minor changes in the location or strength of a constraint, as well as the addition or omission of a constraint. It is obtained by performing one of the following actions on $s$:

- Add a constraint - allowed if the number of constraints is smaller than $k$.
- Omit a constraint.
- Increase $t$ by $\Delta t$ for some constraint - allowed if no constraint in $s$ is set at $t + \Delta t$ and $t + \Delta t \leq t_{\text{max}}$.
- Decrease $t$ by $\Delta t$ for some constraint - allowed if no constraint in $s$ is set at $t - \Delta t$ and $t - \Delta t \geq t_{\text{min}}$.
- Increase $\lambda$ by a factor of $\Delta \lambda$ for some constraint - allowed if $\lambda \cdot \Delta \lambda \leq \lambda_{\text{max}}$.
- Decrease $\lambda$ by a factor of $\Delta \lambda$ for some constraint - allowed if $\lambda / \Delta \lambda \geq \lambda_{\text{min}}$.

The **fitness function** can be based on any one of the similarity measures presented in section 3.3.2, calculated between the observed and estimated functions. The selected measure should be calculated for each function in the test set, and the mean, median or any other central location measure can serve as the fitness value for a given solution.
Repeated runs of threshold acceptance can produce different results, because of the random choice of solutions at each iteration. Moreover, it is quite clear that different functions have different optimal solutions, in the sense that the location and strength of the optimal constraint are not global but curve-specific. In addition, the solution that this procedure reaches is not necessarily the *optimal* solution, but an approximation that depends on the choice of the initial solution, as well as on the functions used as training and test sets. The approximation can be improved by repeated runs only if these runs differ in one or more of these aspects. Therefore, instead of finding a single approximate optimal solution for the entire training set, let us find several approximations: either by using different subsets of the training set for the search and comparing them, or by repeating the search on the same training set - the randomised nature of the search may lead to different solutions. In the chemotherapy example the search was repeated several times, some on the entire dataset and some on a subset of 80% of the curves. In this example there was small variability between optimal solutions for different subsets or different runs, so the same solution was obtained on several subsets.

3.2 Modelling Time Points and Smoothing Coefficients

The models described in the previous section assumed some known form for the dependence of the output $y(t; \mathbf{x})$ on the time variable $t$, and required the estimation of a relatively small number of parameters in order to model the dependence on $\mathbf{x}$. In this section we assume some known form for modelling $y(t; \mathbf{x})$ by $\mathbf{x}$, and estimate the relevant parameters for each $t$. One simple, but often useful, model is $y(t_j) = a_0 + \sum a_i(t_j) \cdot x_i$ (see Govaerts and Noel, 2005). In practice this means fitting a separate model for each time point $t_j$, thus obtaining sets of time dependent coefficients $a_i(t_j)$, and then modelling each coefficient as a function of time. This method requires the use of a very large
number of models in the first stage. In addition, time traces of variable length can present a problem in the modelling process. One problem is that models for the late time points may be based on a smaller number of time traces than models for the early time points, and therefore the resulting models are less accurate. This problem may affect the second stage (smoothing coefficients across time), because the different samples may lead to discontinuities in the functions $a_i(t)$. In addition to that, the time traces that are omitted are not necessarily a random subset of the experimental design, which may cause some estimation bias. This problem can be overcome by mapping the time traces so that all time traces have equal lengths and consist of the same number of time points (see section 3.1.1). This method may be less suitable than other methods shown in this work, yet it is presented in order to provide a complete overview of possible estimation methods.

This method can be expanded to include estimates of the form $y(t_j) = a_{0,j} + \sum a_{ij}(t_j) \cdot g_i(x)$, where $g_i(x)$ are functions of the explanatory variable vector $x$.

### 3.3 Evaluating Goodness of Fit

Any estimation method requires some measure of accuracy. In case of low-dimensional data the Mean Square Error usually provides this indication of how close the fitted values are to the original data. The MSE can still be used when the data are high-dimensional, such as the time traces that are analysed in this work. For example, if we consider the pointwise estimation of the function’s value at each time point, then MSE can be calculated for each $t$. However, more complex definitions of fit should be considered in this case, that take into account the fact that each function is estimated as a whole unit and not necessarily as many separate values. This way features like the function’s shape or scale reconstruction can be evaluated independently of each other. Such measures for goodness of fit are suggested in the following
Shen and Xu (2007) suggest several diagnostic measures for linear models with functional response, for detecting outliers and influential cases. These diagnostics include functional modifications on studentised residuals, Jackknife residuals and Cook’s distance. In this section we focus on goodness of fit measures for single curves, taking into account the fit in shape alone or both in shape and scale.

3.3.1 Measuring Shape Similarity

Measuring how similarly shaped two curves are serves several purposes: To identify sets of curves in the data that can be modelled individually (see section 3.1.4), and to evaluate the quality of the shape reconstruction by comparing the shape of the original time trace with the shape of the estimated one.

The following measures are suggested to compare two time traces $y_1(t)$ and $y_2(t)$. Some of these measures reflect similarity both in shape and scale, while others measure shape similarity only.

1. $D_{f1} = \frac{1}{T} \sum_{i=1}^{T} (d(t_i) - \bar{d}(t))^2$, where $d(t)$ denotes the pointwise distance between $y_1(t)$ and $y_2(t)$, such that $d(t_i) = y_1(t_i) - y_2(t_i)$, and $\bar{d}(t) = \frac{1}{T} \sum_{i=1}^{T} d(t_i)$ . This measure requires that both time traces should be of identical length $T$. It will produce low values for pairs of curves that maintain a fixed distance along time.

2. $D_{f2} = \max\{ \frac{1}{T} \sum_{i=1}^{T} (r_1(t_i) - \bar{r}_1(t))^2, \frac{1}{T} \sum_{i=1}^{T} (r_2(t_i) - \bar{r}_2(t))^2 \}$, where $r_1(t)$ denotes the pointwise ratio such that $r_1(t_i) = \frac{y_1(t_i)}{y_2(t_i)}$, $r_2(t_i) = \frac{y_2(t_i)}{y_1(t_i)}$ and $\bar{r}_k(t) = \frac{1}{T} \sum_{i=1}^{T} r_k(t)$ for $k = 1, 2$. This measure also requires equal lengths of the time traces. It cannot be used if one of the time traces
reaches 0 at some point. This measure will produce low values for pairs of curves that satisfy: \( y_1(t) \cong \text{const} \cdot y_2(t) \).

3. Let us denote by \( R_k^2 \) the proportion of variance explained by the regression model \( y_k(t) = \beta_{0k} + \beta_{1k} \cdot t + \varepsilon \). In fact, \( R_k^2 = \text{Corr}^2(y_k(t), t) \) (the squared Pearson correlation coefficient).

Let \( D_{f3} = \max \{ R_1^2/R_2^2, R_2^2/R_1^2 \} \). This measure compares the fits of both time traces modelled by a straight line.

If \( y_1(t) \cong a \cdot y_2(t) + b \), then

\[
R_1^2 = \text{Corr}^2(y_1(t), t) \cong \text{Corr}^2(a \cdot y_2(t) + b, t)
= \text{Corr}^2(y_2(t), t)
= R_2^2.
\]

Therefore \( D_{f3} \cong 1 \) for curves that are highly correlated. Otherwise, \( R_1^2/R_2^2 \ll 1 \) or \( R_2^2/R_1^2 \gg 1 \), so \( D_{f3} \gg 1 \). As opposed to \( D_{f1} \) and \( D_{f2} \), this measure does not require that \( y_1(t) \) and \( y_2(t) \) should be of equal lengths. It will not detect the difference between curves that are mirror images of each other.

4. Let \( R_{\text{diff}}^2 \) denote the proportion of variance explained by the model

\[
y_1(t) - y_2(t) = \beta_{0\_diff} + \beta_{1\_diff} \cdot t + \varepsilon.
\]

\( D_{f4} = R_{\text{diff}}^2 \) will detect graphs that differ from each other by a straight line, which means that their first derivatives differ by a constant.

If \( \frac{d}{dt} y_1(t) - \frac{d}{dt} y_2(t) \cong \text{const} \) then \( y_1(t) - y_2(t) \cong a \cdot t + b \), and

\[
D_{f4} = \text{Corr}^2(y_1(t) - y_2(t), t)
\cong \text{Corr}^2(a \cdot t + b, t)
= 1.
\]
Therefore $0 \leq D_{f4} \leq 1$, and $D_{f4} \cong 1$ if $y_1(t)$ differs from $y_2(t)$ by a straight line. This measure requires that both graphs should have equal lengths.

When the data itself ranges on a relatively small interval, it seems that $D_{f1}$ is the most informative and least complicated of the four suggested measures.

### 3.3.2 Estimating Goodness of Fit

Any one of the measures presented in section 3.3.1 can be used to evaluate the fit of $\hat{y}(t)$ and $y(t)$. One possible measure is $\text{Corr}(y(t), \hat{y}(t))$ - the correlation coefficient between the two curves. However, this measure will not detect a constant difference between $\hat{y}(t)$ and $y(t)$. Define the Relative Lack of Fit as an alternative measure that takes the distance into account. For curve $k$, let

$$ LOF_k = \sum_{i=1}^{T_k} (\hat{y}_k(t_i) - y_k(t_i))^2 $$

where $T_k$ is the length of the curve. $LOF$ can be used to compare the fits obtained by several different methods, yet some way of standardising it is required (such as $R^2$ for regression models). Comparing the model $LOF$ to $\sum_{i=1}^{T_k} \left( y_k(t_i) - \overline{y}_k(t) \right)^2$, where $\overline{y}_k(t) = \frac{1}{n} \sum_{i=1}^{T_k} y_k(t_i)$, as in $R^2$ does not seem adequate since $\overline{y}(t)$ is not the natural estimator for $y_k(t_i)$ when $\hat{y}_k(t_i)$ is not used. Two possible substitutes for $\overline{y}_k(t)$ are suggested:

1. The "average curve": $\overline{y}(t_i) = \frac{1}{n} \sum_{k=1}^{n} y_k(t_i)$ where $k$ is a curve index.

2. Consider an estimator for $y(t)$ that has the form: $\hat{y}(t) = \sum a_j P_j(t)$, where $P_j(t)$ is a set of basis functions. The coefficients $a_j$ should be modeled as a function of $x$ in order to enable prediction. However, if $x$ is not included in the model, an estimate for $a_j$ can be obtained such that the overall sum $\sum (y(t_i) - \sum a_j P_j(t_i))^2$ is minimised. The resulting estimated curve $\hat{y}(t) = \sum \hat{a}_j P_j(t)$ can be used as a reference curve for standardising the observed $LOF$. 

43
Let \( y_T(t) \) denote the chosen reference curve \((y(t) \text{ or } \widehat{y}(t))\), and let \( ALOF_k \) the lack of fit of a model that uses the alternative estimator \( y_T(t) \). The Relative Lack of Fit is defined as follows: for curve \( k \), \( RLOF_k = \frac{LOF_k}{ALOF_k} \).

Several options for obtaining a goodness of fit measure for the entire data set exist: \( RLOF = \text{mean}(RLOF_k) \). This measure is less robust, as the mean is sensitive to extreme values. The median or the trimmed mean can serve as alternative, robust measures. It should be noted that none of the suggested measures are guaranteed to be smaller than or equal to 1.

Clearly different applications suggest different features that are important in estimating the curve (such as extreme points, exact range or curve length etc.). For example, when clustering curves that have already been rescaled the selected metric should account for any deviation between the curves. The variety of measures suggested in this section allows for adaptation that will reflect the precision of estimating these properties.

### 3.4 Identifying Outliers

Outlier curves can be identified by the following criteria:

- Extreme values that a curve reaches. The scale of the estimated coefficients may shift in order to allow estimation of such extreme values. Such curves can be detected by examining the minimum and maximum values that each curve reaches and singling out extreme values, for example by the \( 1.5 \cdot IQR \) criterion (where IQR is the interquartile range), or any other similar criterion. The \( 1.5 \cdot IQR \) criterion is derived from the normal distribution, where it is approximately equivalent to an observation further than 2 standard deviations from the mean.

- Low similarity in coefficients between a curve and the rest of the data set. Having smoothed the time variable and estimated the characteris-
tic coefficients, Mahalanobis distances can be calculated between those coefficients in order to detect outliers.

- Low shape similarity with most other curves in the data set. Using one of the shape dissimilarity measures suggested in section 3.3.1, curves that have high mean dissimilarity with the rest of the data set can be considered as outliers.

- Curves for which the estimated model produces poor fits (as can be estimated by the measures mentioned in section 3.3.2), compared to the complete data set.

When data are divided into clusters, either by shape similarity or coefficient similarity, outliers are likely to be linked together in the same cluster. Their effect on the model can be tested by comparing fits between the model obtained by the full data set and that obtained by non-outliers only. Outliers may present two main obstacles. Firstly, they add bias to the estimated coefficients, shifting them towards the outlier coefficient values. Secondly, if the estimated coefficients are not suited or scaled to estimate the outlier curves, than the resulting estimated outlier curves may not be sufficiently similar to the original ones. In both cases it seems reasonable to eliminate the outliers from the data set or to use robust methods when estimating the coefficients. In order to minimise the error in prediction it is also necessary to predict if an untested observation $\bar{x}$ is likely to produce an outlier curve. Logistic regression models can make that distinction. If a region within the experimental sample space that produces outliers can be identified, an additional sample of several time traces can be taken from it and a unique model can be constructed to fit the outlier curves. Such methods are suggested in 3.1.4. Otherwise, it should be noted that prediction is restricted to non-outlier values of $\bar{x}$ only.
Figure 1: An outlier curve in the chemotherapy data (strong line).

An example of such an outlier is shown in Figure 1, which shows the entire chemotherapy scheduling dataset in its original scale. The solid, black line marks an outlier curve, as can be seen when all curves are rescaled to the same domain, as can be seen in Figure 2. Clustering the chemotherapy data resulted in the highlighted curve being a cluster that contains a single curve.
Figure 2: Outlier curve in chemotherapy data, shown on mapped curves.
4 Application to Chemotherapy Simulator

The data analysed in this section are simulated chemotherapy protocols (treatment plans). The simulated process starts by exposing a patient to some chemotherapeutic drug. When the drug is present it destroys a fraction of the cancer cells, which at the same time multiply as a part of their life cycle. The drug decomposes with time, thus reducing its effect. Three parameters (set by the doctor) define the protocol: the initial dosage of the drug given, the duration of the current chemotherapy pulse, and the rate at which the drug decomposes in the patient’s body (the third parameter is indirectly set by the doctor, as it is a characteristic of the type of drug used for chemotherapy). A fourth explanatory variable is the cell growth rate, which is a characteristic of the type of tumour that is treated. The output is the amount of cells as a fraction of the initial count at each point of time throughout the treatment duration. The mathematical model that describes cell behaviour is presented in Shochat, Hart and Agur (1999), and the simulation is described in Agur, Hassin and Levy (2006). Throughout the text, each user defined set of explanatory variables (including the growth rate) will be referred to as a treatment protocol. In this example, 100 protocols were simulated and used as a training set, and another set of 20 experimental points was produced from the same domain and used as a test set. The experimental design used was a uniform design as presented in Fang, Lin, Winker and Zhang (2000): each variable was tried at 20 points, uniformly spaced over a pre-determined range, with very low correlations between the different variables. In practice, the time trace data produced are not continuous, but present the amount of cells at close, discrete points in time. An example of the output is shown in Figure 3.

In detail, let $y(t; \vec{x})$ be the fraction of cells at time $t$, given a protocol $\vec{x} = (x_1, \ldots, x_4)$, as computed by the simulation. The experimental domain is described in Table 1.
Figure 3: Chemotherapy simulated output. The chemotherapy treatment destroys about 2% of the cancer cells by time 20, after which there is a gradual increase in the number of cells.

Throughout this section, the protocol $\mathbf{x}$ will serve as the set of explanatory variables for the various models. The results and figures presented in the section refer to the test set of 20 experimental points.

4.1 Smoothing the Time Trace and Estimating Coefficients

The methods presented in section 3.1 are applied to the chemotherapy data and compared. In section 4.1.1 time is modelled by B-splines and the coefficients are estimated by linear regression, ACE, Projection Pursuit Regression (PPReg) and kriging. Local models are then used on clustered coefficients, which are again estimated by linear regression, ACE and PPReg.
<table>
<thead>
<tr>
<th>Variable name</th>
<th>Biological role</th>
<th>minimum</th>
<th>maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>Treatment duration</td>
<td>10 hrs</td>
<td>40 hrs</td>
</tr>
<tr>
<td>$X_2$</td>
<td>Dose</td>
<td>2 mg</td>
<td>15 mg</td>
</tr>
<tr>
<td>$X_3$</td>
<td>Drug decomposition rate</td>
<td>0.1 mg/(cm$^3$·hr)</td>
<td>2 mg/(cm$^3$·hr)</td>
</tr>
<tr>
<td>$X_4$</td>
<td>Cell growth rate</td>
<td>0.01</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 1: Chemotherapy simulation parameters

In section 4.1.2 time is modelled by shape derived basis functions, with the number of clusters ranging between 2 and 11. Parameters are estimated using the abovementioned methods and also tested with a constraint added to the model. Two stage estimation is tested, separating the shape estimation from the scale estimation. Finally, section 4.1.3 presents the results of modelling the data at each time point and smoothing the coefficients as mentioned in section 3.2, with coefficients modelled using linear regression and ACE.

In this section the modelling and prediction of the data are done in two stages. The first stage is describing the 100 discrete time traces produced by the simulation by some continuous analytic function, as described in section 3.1.1. The second stage, essential for prediction purposes, is to model that function’s parameters using the four explanatory variables on which the simulated time traces depend. The estimation methods that will be used are described in section 3.1.3.

The first step in analyzing the simulation data is to estimate $y(t; \mathbf{x})$ by $F \left( f_i(t); \beta_j(\mathbf{x}) \right)$ where $f_i(t)$ is a set of known functions and $\beta_j(\mathbf{x})$ are the related coefficients. It is assumed that $\beta_j(\mathbf{x})$ can be accurately obtained for the training set, and can serve as the data set for estimating $\beta_j(\mathbf{x})$ for any protocol $\mathbf{x}$ in the experimental domain.

Smoothing the observed time traces was done while considering the fact that they show no periodic or strong local features.
4.1.1 Smoothing by $B$–Splines

The time traces were modelled by a cubic $B$ Spline with a single knot. Following visual inspection the knot location was chosen to be $t = 1.5$ hours, where the functions seemed to change rapidly. It should be noted that the knot is placed earlier than the shortest curve length, thus requiring a set of 5 coefficients to be estimated for each run. However, changing the knot location had little impact on the model’s goodness of fit. This model requires the estimation of five coefficients. The following results compare various methods for modelling the $B$ Spline coefficients. In all cases the final model was used to predict and reconstruct the 20 time traces in the test set. The goodness of fit measure for the prediction is mentioned in section 3.3.2: define $\overline{y(t_i)} = \frac{1}{n} \sum_{j=1}^{n} y_j(t_i)$ as the mean function value at time $t_i$, and let $\hat{y}(t_i; \bar{x}_j)$ denote the predicted function value of curve $j$ in the test set at time point $t_i$. The fit of curve $j$ of the test set is then $RLOF_j = \frac{\sum_{i=1}^{T} (y_j(t_i; \bar{x}_j) - \hat{y}(t_i; \bar{x}_j))^2}{\sum_{i=1}^{T} (y_j(t_i; \bar{x}_j) - y_j(t_i))^2}$.

In these graphs the original curve is plotted in black and the fitted curve in blue.

Figures 4 through 7 show the resulting curves, when the $B$ Spline coefficients were estimated by linear regression, Alternating Conditional Expectations (ACE), Projection Pursuit Regression (PPReg) and Kriging with power exponential correlation, respectively. In each one of these methods, the four previously mentioned variables were taken as the set of explanatory variables.

The constraint location algorithm was applied on the chemotherapy data. The search region was restricted to time points in $[0, 100]$, since the shortest curves end at $t = 100$ and this way the constraint would affect all curves. The constraint strength was taken from an interval that for practical purposes ranged from placing no constraint to placing a hard constraint. The
Figure 4: Chemotherapy test set prediction: Time modelled by $B$ Spline, coefficients modelled by Linear Regression. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
Figure 5: Chemotherapy test set prediction: Time modelled by $B$ Spline, coefficients modelled by ACE. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
Figure 6: Chemotherapy test set prediction: Time modelled by B Spline, coefficients modelled by Projection Pursuit Regression. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
Figure 7: Chemotherapy test set prediction: Time modelled by $B$ Spline, coefficients modelled by Kriging. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
search resulted in adding a single, hard constraint at \( t = 100 \). The fact that only a single constraint was recommended by the algorithm can be related to the smoothness of the curves and the estimators - curve values at different time points are rather highly correlated, so the prediction of the function’s value at two (or more) time points adds little information over that obtained by a single time point at which the curve value is predicted. The estimates that were used for \( y(100; x) \) were those obtained by linear regression, as this method was used in the search algorithm. The model \( R^2 \) for \( y(100; x) \) is 0.9. Even though Kriging estimators proved to perform better than linear regression, it was not used in this procedure for computational reasons. Figures 8 through 11 show the estimated curves.

**Local Models** The \( B - Spline \) coefficients were divided into three clusters. The experimental points of each cluster separately were then used to re-model the coefficients, this time allowing different models to be fitted to each cluster. The models were fit using linear regression, ACE and PPReg. Finding the optimal numbers of clusters is a problem that needs further research, taking into account the need for a substantial number of observations in each cluster. The choice of the number clusters in this case did not follow any given method or rule. Three clusters were chosen since it seemed that further division of the data would not produce clusters that are vastly different from the existing ones, yet a reasonable number of data points had to be maintained in each cluster. A discriminant analysis model was fitted to predict cluster assignment for new observations. Let us denote by \( C_i \) the experiment region where \( B - Spline \) coefficients should be assigned to cluster \( i, \ i = 1, 2, 3 \). Let \( \pi_i(x) \) denote the weight assigned to the region \( C_i \) at the experimental point \( x \). These weights were predicted by the discriminant analysis model. Denote by \( \hat{\beta}_i(x) \) the coefficients predicted for \( x \) using the model that was fit to cluster \( i \). The coefficients predicted for \( x \) by this local model are \( \hat{\beta}(x) = \sum_{i=1}^{3} \pi_i(x) \cdot \hat{\beta}_i(x) \). Figures 12 through 14 show the resulting predicted curves.
Figure 8: Chemotherapy test set prediction: Time modelled by B Spline, coefficients modelled by Linear Regression with constraint added at $t = 40$. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
Figure 9: Chemotherapy test set prediction: Time modelled by $B$ Spline, coefficients modelled by ACE with constraint added at $t = 40$. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
Figure 10: Chemotherapy test set prediction: Time modelled by $B$ Spline, coefficients modelled by Projection Pursuit Regression with constraint added at $t = 40$. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
Figure 11: Chemotherapy test set prediction: Time modelled by $B$ Spline, coefficients modelled by Kriging with constraint added at $t = 40$. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
Figure 12: Chemotherapy test set prediction: Time modelled by B Spline, coefficients clustered and then modelled by Linear Regression. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
Figure 13: Chemotherapy test set prediction: Time modelled by B Spline, coefficients clustered and then modelled by ACE. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
4.1.2 Shape Derived Estimation

The training set curves were divided into clusters using the functional clustering algorithm described in 3.1.4. A discriminant analysis model was used to predict cluster assignment for an untested observation \( x \). Prior to clustering, all curves were transformed such that they had the same minimum and maximum values, so that the similarity measures between the transformed curves will show how close in shape two functions are, regardless of their scales. Let \( g_i(t) \) denote the mean curve of cluster \( i \), and let \( \pi_i(x) \) denote the predicted weight assigned at \( x \) to cluster \( i \).

Two stages of prediction will be discussed in this section: shape estimation and scale estimation. The shape estimation stage consists of selecting the appropriate number of clusters and predicting coefficients or weights. The scale prediction requires estimating a relatively small number of features. The quality of the overall prediction depends on both stages, yet the two stages are independent of each other so the quality of each stage can be evaluated separately. Scale reconstruction is usually done using known procedures. Therefore, the focus in this section will be on the shape prediction stage.

Functional Basis The functions \( g_i(t) \) that were used as basis functions for modelling the training set curves and coefficients were obtained for each curve using linear regression. The key issue in this procedure is selecting the number of clusters: too few basis functions will not provide good representation of the different shapes of the various curves, whereas too many clusters will force the estimation of a large number of coefficients, leading to large prediction error. As mentioned above, the scale reconstruction stage does not depend on the initial choice of clusters, but only on the choice of scale features and on the quality of the scale estimators. For that reason, when addressing the task of selecting the number of clusters, only the fit in
Figure 14: Chemotherapy test set prediction: Time modelled by $B$ Spline, coefficients clustered and then modelled by Projection Pursuit Regression. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
shape will be considered. In practice that means that fit will be measured with regard to the mapped curves and not the originally scaled curves. An example of the means is shown in Figure 15, where the chemotherapy data were split into 4 clusters.

One of the basic assumptions in this work is that a functional basis exists that can describe a given curve with very high precision. In order for this assumption to be valid, a sufficiently large basis should be selected so that the training set is reconstructed with very good fit. The chemotherapy training set was modelled using 2 to 11 basis functions. These functions can be used as a basis for estimating two types of curves: one type is the original curves. The
other type is the mapped curves, rescaled to the same length, minimum and maximum. In both cases the next step is to model the functions’ coefficients from the training set and predict them for the test set, thus reconstructing the test set (mapped or original) curves. Note that all curves must have the same length in order to share the same basis functions, so in this section it is assumed that all curves were mapped accordingly.

The first model uses direct estimation of the curve, without eliminating scale features. The predictor is \( \tilde{y}(t; x) = \sum \tilde{\beta}_i(x) \cdot g_i(t) \), which was mentioned in section 3.1.2. Since this discussion focuses on the shape estimation, one method only was tested for estimating \( \tilde{\beta}_i(x) \), using from 2 to 11 basis functions.

Table 2 shows the mean goodness of fit for the training and test sets when modelled by each number of clusters. The measures used in this section are as follows: the model R-squared is the known goodness of fit measure that evaluates any linear regression model. The model fit is calculated for each curve in the training set, and the measure presented in the tables is the mean R-squared over the entire training set. The other measures were calculated over the test set, and reflect the mean, median and trimmed mean of the lack of fit measure RLOF that was presented in section 3.3.2. Examples of the original and fitted curves, for 2 and 11 basis functions, are shown in Figures 16 and 17. It can be seen that the number of clusters has little effect on the fit quality.

The second model used the same basis functions to model the shape functions. The predictor is \( \tilde{g}(t; x) = \sum \tilde{\beta}_i(x) \cdot g_i(t) \), where \( g(t; x) \) is the shape function corresponding to the original curve \( y(t; x) \). The scale features selected were the minimum and maximum of each curve. Table 3 shows fit indicators for this estimator. It should be noted that these fit measures do not account for lack of fit caused by the scale reconstruction stage. The original and predicted curves for 2 and 11 clusters are shown in Figures 18.
Figure 16: Chemotherapy test set prediction: Regression on 2 basis functions, coefficients obtained for modelling from training set curves in their original scale (shape and scale estimated in a single stage).
Figure 17: Chemotherapy test set prediction: Regression on 11 basis functions, coefficients obtained for modelling from training set curves in their original scale (shape and scale estimated in a single stage). In all panels, \( t \) runs from 0 to 400 and \( y(t) \) from \( \text{min} \) to 1, where \( \text{min} \) depends on the particular simulator run.
Following that, models for estimating the coefficients were fitted using the methods described in section 4.1.1. The basis consisted of 3 cluster means. Figures 20 through 22 show the results of estimating the curves, when the coefficients are estimated by linear regression, ACE, PPReg and Kriging.

A hard constraint was added to the model. The location search algorithm described in section 3.1.6 was repeated several times with different subsets and different (randomly selected) starting points. As expected, several options for constraint locations were obtained. The differences in fitness among the various suggested solutions were not significant, which can be explained by the fact that the models of the individual time points had similar qualities - $R^2$ of around 0.9. An example of the effect of a hard constraint, placed at $t = 40$, is shown in Figures 23 through 25.

The coefficients obtained by the various methods were also clustered to obtain a local model, as shown in Figures 26 through 28. The true coefficients

<table>
<thead>
<tr>
<th>No. of Clusters</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model R-Squared</td>
<td>0.9810</td>
<td>0.9949</td>
<td>0.9977</td>
<td>0.9986</td>
<td>0.9989</td>
</tr>
<tr>
<td>Mean</td>
<td>0.5945</td>
<td>0.5911</td>
<td>0.5908</td>
<td>0.5906</td>
<td>0.5905</td>
</tr>
<tr>
<td>Median</td>
<td>0.2543</td>
<td>0.2542</td>
<td>0.2542</td>
<td>0.2542</td>
<td>0.2542</td>
</tr>
<tr>
<td>Trimmed mean</td>
<td>0.3843</td>
<td>0.3798</td>
<td>0.3794</td>
<td>0.3791</td>
<td>0.3790</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No. of Clusters</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model R-Squared</td>
<td>0.9992</td>
<td>0.9994</td>
<td>0.9994</td>
<td>0.9995</td>
<td>0.9996</td>
</tr>
<tr>
<td>Mean</td>
<td>0.5905</td>
<td>0.5905</td>
<td>0.5905</td>
<td>0.5905</td>
<td>0.5905</td>
</tr>
<tr>
<td>Median</td>
<td>0.2542</td>
<td>0.2542</td>
<td>0.2542</td>
<td>0.2542</td>
<td>0.2542</td>
</tr>
<tr>
<td>Trimmed mean</td>
<td>0.3790</td>
<td>0.3790</td>
<td>0.3790</td>
<td>0.3790</td>
<td>0.3790</td>
</tr>
</tbody>
</table>

Table 2: Goodness of fit for different numbers of clusters - regression estimation of original curves. The model R-Squared measure presented here is the mean R-Squared over all curves in the training set. The other measures relate to the Relative Lack of Fit over all curves in the test set.

and 19.
Figure 18: Chemotherapy test set shape prediction: Regression on 2 basis functions, coefficients obtained for modelling from training set curves rescaled to the same length, minimum and maximum values (shape and scale estimated in two separate stages).
Figure 19: Chemotherapy test set shape prediction: Regression on 11 basis functions, coefficients obtained for modelling from training set curves rescaled to the same length, minimum and maximum values (shape and scale estimated in two separate stages). In all panels, $t$ runs from 0 to 400 and $y(t)$ from 1 to 2.
Figure 20: Chemotherapy test set prediction: 3 basis functions, coefficients modelled by Linear Regression. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
Figure 21: Chemotherapy test set prediction: 3 basis functions, coefficients modelled by ACE. In all panels, $t$ runs from 0 to 400 and $y(t)$ from $\min$ to 1, where $\min$ depends on the particular simulator run.
Figure 22: Chemotherapy test set prediction: 3 basis functions, coefficients modelled by Projection Pursuit Regression. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
Figure 23: Linear Regression with constraint at $t = 40$. In all panels, $t$ runs from 0 to 400 and $y(t)$ from $\min$ to 1, where $\min$ depends on the particular simulator run.
Figure 24: ACE with constraint at $t = 40$. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
Figure 25: Projection Pursuit Regression with constraint at $t = 40$. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
<table>
<thead>
<tr>
<th>No. of Clusters</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model R-Squared</td>
<td>0.9776</td>
<td>0.9952</td>
<td>0.9980</td>
<td>0.9988</td>
<td>0.9992</td>
</tr>
<tr>
<td>Mean</td>
<td>0.6217</td>
<td>0.5826</td>
<td>0.5823</td>
<td>0.5813</td>
<td>0.5812</td>
</tr>
<tr>
<td>Median</td>
<td>0.5254</td>
<td>0.4874</td>
<td>0.4887</td>
<td>0.4891</td>
<td>0.4893</td>
</tr>
<tr>
<td>Trimmed mean</td>
<td>0.5789</td>
<td>0.5356</td>
<td>0.5356</td>
<td>0.5346</td>
<td>0.5345</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No. of Clusters</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model R-Squared</td>
<td>0.9994</td>
<td>0.9995</td>
<td>0.9996</td>
<td>0.9997</td>
<td>0.9998</td>
</tr>
<tr>
<td>Mean</td>
<td>0.5812</td>
<td>0.5812</td>
<td>0.5812</td>
<td>0.5812</td>
<td>0.5812</td>
</tr>
<tr>
<td>Median</td>
<td>0.4893</td>
<td>0.4894</td>
<td>0.4894</td>
<td>0.4894</td>
<td>0.4894</td>
</tr>
<tr>
<td>Trimmed mean</td>
<td>0.5345</td>
<td>0.5345</td>
<td>0.5345</td>
<td>0.5345</td>
<td>0.5344</td>
</tr>
</tbody>
</table>

Table 3: Goodness of fit for different numbers of clusters - regression estimation of shape. The model R-Squared measure presented here is the mean R-Squared over all curves in the training set. The other measures relate to the Relative Lack of Fit over all curves in the test set.

of the training set were assigned to one of three clusters, each cluster was modelled by Linear Regression, ACE or PPReg. Prediction for the test data was done using the weighted mean of the predictions from each cluster. The weights reflected the probability of an observation belonging to each cluster. The centroid functions that are derived from the centroid coefficients are shown in Figure 29. These functions are obtained by considering each of the three cluster centroids as coefficients for a \textit{B-Spline} basis. It can be seen that the centroids are highly correlated, yet maintain a significant constant distance between them, which can explain the improvement in prediction results as compared to the global estimation methods. These results are summarised in Table 5.

**Shape Prediction** The mean functions \(g_i(t)\) were used to predict the characteristic shape function of an untested observation \(\bar{x}\). Using the above notation, the predicted shape function at \(\bar{x}\) is \(\hat{g}(t; \bar{x}) = \sum_{i=1}^{k} \pi_i(\bar{x}) \cdot g_i(t)\), where \(k\) is the number of clusters (2 to 11 as in the functional basis section). The
Figure 26: Clustered coefficients modelled using linear regression. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
Figure 27: Clustered coefficients modelled using ACE. In all panels, \( t \) runs from 0 to 400 and \( y(t) \) from \( \text{min} \) to 1, where \( \text{min} \) depends on the particular simulator run.
Figure 28: Clustered coefficients modelled using Projection Pursuit Regression. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
Figure 29: Three functions describing the three centroids of the clustered coefficients.
<table>
<thead>
<tr>
<th>No. of Clusters</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model R-Squared</td>
<td>0.7431</td>
<td>0.8570</td>
<td>0.8918</td>
<td>0.9131</td>
<td>0.9312</td>
</tr>
<tr>
<td>Mean</td>
<td>0.7628</td>
<td>0.5242</td>
<td>0.4238</td>
<td>0.4424</td>
<td>0.4206</td>
</tr>
<tr>
<td>Median</td>
<td>0.7566</td>
<td>0.4871</td>
<td>0.4078</td>
<td>0.3872</td>
<td>0.3964</td>
</tr>
<tr>
<td>Trimmed mean</td>
<td>0.7607</td>
<td>0.5076</td>
<td>0.403</td>
<td>0.421</td>
<td>0.3904</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No. of Clusters</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model R-Squared</td>
<td>0.9417</td>
<td>0.9419</td>
<td>0.9427</td>
<td>0.09233</td>
<td>-0.3171</td>
</tr>
<tr>
<td>Mean</td>
<td>0.3973</td>
<td>0.3956</td>
<td>0.3628</td>
<td>2.203</td>
<td>1.706</td>
</tr>
<tr>
<td>Median</td>
<td>0.3613</td>
<td>0.3594</td>
<td>0.3128</td>
<td>1.152</td>
<td>1.434</td>
</tr>
<tr>
<td>Trimmed mean</td>
<td>0.3784</td>
<td>0.3764</td>
<td>0.3384</td>
<td>1.857</td>
<td>1.661</td>
</tr>
</tbody>
</table>

Table 4: Goodness of fit for different numbers of clusters - shape prediction by weighted cluster means. The model R-Squared measure presented here is the mean R-Squared over all curves in the training set. The other measures relate to the Relative Lack of Fit over all curves in the test set.

results shown in Table 4 include the mean R-Squared measure for goodness of fit for the training set (as seen already in the functional basis section), as well as the mean, median and 10% trimmed mean of the Lack of Fit measure for the test set.

It can be seen from the table that 3 clusters are enough to provide precise fit for the training set, and that almost all lack of fit measures show that 3 to 9 clusters enable good prediction. However, using more than 9 clusters results in overfitting the training set, as can be seen from the goodness of fit measure for the model. As expected, the resulting predictions for the test set are poor when more than 9 basis functions are used. The true and predicted curves for the test set, modelled using 3, 9 and 11 clusters, are shown in Figures 30, 31 and 32 respectively.

In order to transform the estimated shape function into its true scale, models were constructed to predict $a(x)$ and $b(x)$, the coefficients of the linear transformation that was used to convert the curves to a common scale. Reconstructing the scale by the minimum and maximum values of each curve
Figure 30: Shape estimated by 3 basis functions
Figure 31: Shape estimated by 9 basis functions. In all panels, $t$ runs from 0 to 400 and $y(t)$ from 1 to 2.
Figure 32: Shape estimated by 11 basis functions. In all panels, \( t \) runs from 0 to 400 and \( y(t) \) from 1 to 2.
resulted in the predictions shown in Figures 33 and 34. The curves rescaled by predicting the values at $t = 5$ and $t = 30$ are shown in Figures 35 and 36. It can be seen that when scaling was done by estimating the minimum and maximum by linear regression, 3 predicted curves became ascending and not descending - a result of the estimated minimum being greater than the estimated maximum, which in itself is a result of the separate estimation for the minimum and maximum, as opposed to a single model that could enable adding the constraint $\min < \max$. This seems to happen for curves that reach a minimum that is close to the boundary point 0, where prediction is inaccurate. These outlier curves can account for the extremely poor mean Lack of Fit score that these methods achieved. On the other hand, reconstructing scale using ACE produced good results, even when the individual curves are examined and not just the mean fit.

4.1.3 Modelling time points and estimating coefficients

Following the model described in section 3.2, the coefficients $a_i(t_j)$ were modelled using linear regression, as shown in Figure 37. The coefficients were also modelled using ACE, resulting in the relatively non-smooth curve presented in Figure 38. This curve was smoothed by a running median, resulting in the curve in Figure 39. A summary comparing the goodness of fit of the various models is given in section 4.1.4.

4.1.4 Summary

Table 5 is a summary of the results that were presented above. The values are the mean Relative Lack of Fit defined in section 3.3.2.

Looking at these results more closely it can be seen that modelling the time traces by B-Spline followed by Kriging estimation of the resulting coefficients yielded the best fits (Figure 7). This method achieved impressive
Figure 33: Scale reconstruction my min-max, $a(x)$ and $b(x)$ modelled by linear regression. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to max, where min and max depend on the particular simulator run.
Figure 34: Scale reconstruction by min-max, $a(x)$ and $b(x)$ modelled by ACE. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
Figure 35: Scale reconstruction by predicted values at $t = 5$ and $t = 30$, $a(x)$ and $b(x)$ predicted using linear regression. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to max, where min and max depend on the particular simulator run.
Figure 36: Scale reconstruction by predicted values at $t = 5$ and $t = 30$, $a(x)$ and $b(x)$ predicted using ACE. In all panels, $t$ runs from 0 to 400 and $y(t)$ from $\min$ to 1, where $\min$ depends on the particular simulator run.
Figure 37: Time points modelled by linear regression. In all panels, \( t \) runs from 0 to 400 and \( y(t) \) from \( \text{min} \) to 1, where \( \text{min} \) depends on the particular simulator run.
Figure 38: Time points modelled using ACE. In all panels, \( t \) runs from 0 to 400 and \( y(t) \) from min to 1, where min depends on the particular simulator run.
Figure 39: Time points modelled by ACE, then smoothed. In all panels, $t$ runs from 0 to 400 and $y(t)$ from min to 1, where min depends on the particular simulator run.
### B-Spline basis

<table>
<thead>
<tr>
<th></th>
<th>Regular</th>
<th>With constraint</th>
<th>Clustered coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear regression</td>
<td>0.498</td>
<td>0.5161</td>
<td>0.478</td>
</tr>
<tr>
<td>ACE</td>
<td>0.811</td>
<td>0.5599</td>
<td>0.418</td>
</tr>
<tr>
<td>PPReg</td>
<td>0.871</td>
<td>0.5827</td>
<td>0.462</td>
</tr>
<tr>
<td>Kriging</td>
<td>0.123</td>
<td>0.4482</td>
<td></td>
</tr>
</tbody>
</table>

### Shape derived basis

<table>
<thead>
<tr>
<th></th>
<th>Regular</th>
<th>With constraint</th>
<th>Clustered coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear regression</td>
<td>0.486</td>
<td>0.446</td>
<td>0.567</td>
</tr>
<tr>
<td>ACE</td>
<td>3.152</td>
<td>2.118</td>
<td>0.672</td>
</tr>
<tr>
<td>PPREG</td>
<td>0.645</td>
<td>0.5433</td>
<td>0.46</td>
</tr>
</tbody>
</table>

### Shape prediction

<table>
<thead>
<tr>
<th></th>
<th>by min &amp; max</th>
<th>by t=5 &amp; t=30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear regression</td>
<td>3.021</td>
<td>2.949</td>
</tr>
<tr>
<td>ACE</td>
<td>1.083</td>
<td>0.779</td>
</tr>
</tbody>
</table>

### Modelling by time

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear regression</td>
<td>0.483</td>
</tr>
<tr>
<td>ACE</td>
<td>0.674</td>
</tr>
<tr>
<td>ACE and smoothing</td>
<td>0.687</td>
</tr>
</tbody>
</table>

Table 5: Chemotherapy simulation results - goodness of fit measured by Lack of Fit
fit values due to the precise prediction of curves 3 and 15 which are outliers - most other methods failed to place these curves in the right domain. However, even though Kriging proved to be an excellent method for low dimensional prediction, it was not used for scale prediction in every example for computational reasons - it required running programs interactively on two separate systems and manually adjusting the output to fit both systems. Since the main goal of this work is the modelling of time variable effect and not the low dimensional variables effect, Kriging was used on a smaller set of runs than other methods.

The factors affecting the curve coefficients: All 4 factors affect the intercept, with the dose and growth rate having the strongest impact. The basis function’s coefficients are not affected by the treatment duration, but are affected by all others with the drug decomposition rate having the strongest effect. As for modelling specific time points (for example, for the purpose of adding constraints), all factors except the treatment duration are found to be effective, with the dose and growth rate showing stronger effects for early time points, and drug decomposition rate and growth rate showing the stronger impact for later time points. Looking at scale features, all but treatment duration affect the minimum and the maximum, which is equivalent to the intercept, is affected by all 4 factors, especially the dose and growth rate.

The effect of the factors on the cluster assignment was tested using Analysis of Variance with the factors as dependent variables and cluster assignment as the grouping variable. It was found that the treatment duration and the drug decomposition rate control the cluster assignment when the number of clusters is relatively small (5 or less). With 6 clusters or more, no significant difference can be found between different clusters in any of the experimental factors. The fact that the clustering here is not tied to the input factors supports the conclusion that a large number of clusters significantly reduces the predictive ability of the model.
5 Application to Circadian Rhythm Simulator²

This model, fully described by olde Scheper, Klinkenberg, Pennartz and van Pelt (1999), simulates the production and degradation of protein and mRNA during a sleep-wake cycle. It is based on a set of differential equations that depend on a set of 7 biological parameters and describe the mRNA and protein levels along time. Normally, if the biological parameters lie in a realistic domain, these equations produce robust, periodic functions. An example of these functions is shown in Figure 40. In this figure, as in the figures that follow, the horizontal axis marks time where the time unit is 0.1 hours.

Since the two output functions have similar forms, only the mRNA data were analysed. It should be noted that the underlying cycle length of such a function depends on the biological parameters. One particular set of parameters characterises a free-run state - when a person is not exposed to any stimulations that indicate the true time of the day, thus creating their own sleep-wake cycle, the length of which is estimated to be 24.6 hours. The experimental domain chosen for this simulation, shown in Table 6, is a region surrounding this special set of parameters. Biologically reasonable cycle lengths range between 21 and 29 hours.

The training set simulation initially consisted of 200 experimental points. However, 22 points were found to be biologically unrealistic, producing no sleep-wake cycle at all, and were therefore dropped from the data set. Similarly, two out of 20 experimental points were omitted from the test set for the same reason. Later in this section we address the problem of predicting what factor combinations generate such curves. Other experimental points

²This model was constructed with the kind help of Dr. Dorit Shweiki from the Bioinformatics department of the Academic College of Tel Aviv Yaffo.
that produced extremely short or extremely long cycles were left in the experiment. Given these data, several aspects of the mRNA are of interest, for example:

- Modelling the mRNA as a cyclic function of time.
- Identifying biologically feasible and unfeasible combinations of the 7 parameters.
- Estimating cycle length.
<table>
<thead>
<tr>
<th>Variable</th>
<th>biological role</th>
<th>free-run</th>
<th>min</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>X₁</td>
<td>mRNA production rate constant</td>
<td>1.0 hr⁻¹</td>
<td>0.5</td>
<td>1.5</td>
</tr>
<tr>
<td>X₂</td>
<td>Protein production rate constant</td>
<td>1.0 hr⁻¹</td>
<td>0.5</td>
<td>1.5</td>
</tr>
<tr>
<td>X₃</td>
<td>mRNA degradation rate constant</td>
<td>0.21 hr⁻¹</td>
<td>0.13</td>
<td>0.3</td>
</tr>
<tr>
<td>X₄</td>
<td>Protein degradation rate constant</td>
<td>0.21 hr⁻¹</td>
<td>0.13</td>
<td>0.3</td>
</tr>
<tr>
<td>X₅</td>
<td>Hill coefficient</td>
<td>2</td>
<td>1.2</td>
<td>2.8</td>
</tr>
<tr>
<td>X₆</td>
<td>Non-linearity in protein synthesis cascade</td>
<td>3</td>
<td>2.6</td>
<td>3.4</td>
</tr>
<tr>
<td>X₇</td>
<td>Duration of protein synthesis cascade</td>
<td>4 hr</td>
<td>2.4</td>
<td>5.6</td>
</tr>
</tbody>
</table>

Table 6: mRNA simulation parameters

5.1 Smoothing Time

Given the cyclic nature of the data, the obvious choices for the time smoothing basis were Fourier decomposition and periodic B-splines. Both methods yielded good fits, yet required a relatively large number of basis functions to do so. For example, modelling by Fourier decomposition with 11 coefficients to estimate (an intercept term, 5 real parts and 5 imaginary parts) resulted in a mean R-squared of less than 90% for the training set, whereas the methods that will be presented later in this section achieved a mean R-squared of over 98% with only 3 coefficients to estimate. Using the Fourier decomposition with 3 coefficients to model resulted in a mean R-squared of about 84%. For that reason, modelling and prediction for these data focused on shape based, data-derived basis functions.

As can be seen from Figure 40 the time traces consist of stable cycles. Therefore, each time trace can be represented by a single cycle. In order to increase the similarity between the curves all cycles were traces between two consecutive local minima. An example of a cycle is shown in Figure 41.

Functional basis and shape prediction  It was mentioned in section 3.1.2 that linear transformations can be applied in order to map the data to a common region, thus limiting the required estimation to the shape features.
only. The circadian rhythm data (single cycle format) were mapped to have the same cycle length, minimum and maximum. The mapped curves were clustered by shape.

As the main purpose of this section is to study the shape prediction and the effect that the number of clusters has on it, coefficient prediction was done by linear regression only, and results were compared across models. The models that were used included 2 to 11 clusters. Good results were obtained for the shape function even with a small number of clusters. The scale features of the time traces must be estimated from the data: unlike the chemotherapy data, the cycle length is not given as one of the explanatory
variables, so it is a scale feature that must be estimated. The conventional
methods for low-dimensional data are suitable here, and they were used as
well as a data-driven method that will be described later in the text.

Three estimation methods will be used in this section. All three methods
are based on basis functions that are obtained by clustering scale-free curves.

1. Weighted means - the predicted curve is a weighted average of the
cluster means. The weights assigned to the different means are the
predicted probabilities of belonging to the respective clusters, as ob-
tained by a discriminant analysis model. The predicted curves are also
scale-free and require scale reconstruction.

2. Regression of non-rescaled curves on cluster means - the original-scale
training set is modelled by linear regression on the cluster means, the
regression coefficients are modelled and the predicted curves provide
both the shape and the scale.

3. Regression of rescaled curves on cluster means - the scale-free training
set is modelled by linear regression on the cluster means, the regression
coefficients are modeled and the predicted curves are scale-free, and
require scale reconstruction.

Table 7 presents the goodness of fit measures for the training and the test
sets. The measures used in this section are as follows: the model R-squared
is the known goodness of fit measure that evaluates any linear regression
model. The model fit is calculated for each curve in the training set, and
the measure presented in the tables is the mean R-squared over the entire
training set. The other measures were calculated over the test set, and reflect
the mean, median and trimmed mean of the lack of fit measure $RLOF$ that
was presented in section 3.3.2.
Table 7: Goodness of fit for different numbers of clusters - shape prediction by weighted cluster means (mRNA data). The model R-Squared measure presented here is the mean R-Squared over all curves in the training set. The other measures relate to the Relative Lack of Fit over all curves in the test set.

The results in Table 7 were obtained when modelling was done by weighted means, therefore the resulting predicted curves are scale-free. The weights were the cluster assignment probabilities predicted by a discriminant analysis model. A small number of clusters is sufficient in order to reconstruct the training set with good accuracy; yet the reconstruction accuracy, as well as the test set prediction, dropped significantly when 10 or more clusters were used. An example of the basis function is shown in Figure 42, where 4 clusters were used. The original and fitted curves for 3 and 11 cluster means are shown in Figures 43 and 44, respectively. In these graphs the original curve is plotted in black and the fitted curve in blue.

Table 8 shows goodness of fit measures for modelling the original curves by 2 to 11 cluster means as basis functions. It can be seen that the number of clusters has little effect on the predicted curves’ goodness of fit. Examples of true and predicted curves using 3 and 11 clusters can be seen in Figures 45 and 46 respectively.
Figure 42: Cluster means for mRNA data - data split into 4 clusters by shape
Figure 43: Shape reconstruction using weighted cluster means, 3 clusters (mRNA data). In all panels, $t$ runs from 0 to 400 and $y(t)$ from 1 to 2.
Figure 44: Shape reconstruction using weighted cluster means, 11 clusters (mRNA data). In all panels, $t$ runs from 0 to 400 and $y(t)$ from 1 to 2.
Figure 45: Non-rescaled mRNA data predicted using regression on 3 basis functions. In all panels, $t$ runs from 0 to 400 and $y(t)$ from 1 to 2.
Figure 46: Non-rescaled mRNA data predicted using regression on 11 basis functions. In all panels, $t$ runs from 0 to 400 and $y(t)$ from 1 to 2.
<table>
<thead>
<tr>
<th>No. of Clusters</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model R-Squared</td>
<td>0.9875</td>
<td>0.9979</td>
<td>0.9985</td>
<td>0.9997</td>
<td>0.9999</td>
</tr>
<tr>
<td>Mean</td>
<td>0.4715</td>
<td>0.4565</td>
<td>0.4558</td>
<td>0.4558</td>
<td>0.4559</td>
</tr>
<tr>
<td>Median</td>
<td>0.4364</td>
<td>0.4301</td>
<td>0.4274</td>
<td>0.4264</td>
<td>0.4263</td>
</tr>
<tr>
<td>Trimmed mean</td>
<td>0.4632</td>
<td>0.4366</td>
<td>0.4357</td>
<td>0.4354</td>
<td>0.4354</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No. of Clusters</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model R-Squared</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Mean</td>
<td>0.4559</td>
<td>0.4559</td>
<td>0.4559</td>
<td>0.4559</td>
<td>0.4559</td>
</tr>
<tr>
<td>Median</td>
<td>0.4263</td>
<td>0.4263</td>
<td>0.4263</td>
<td>0.4263</td>
<td>0.4263</td>
</tr>
<tr>
<td>Trimmed mean</td>
<td>0.4354</td>
<td>0.4354</td>
<td>0.4354</td>
<td>0.4354</td>
<td>0.4354</td>
</tr>
</tbody>
</table>

Table 8: Goodness of fit for different numbers of clusters - shape prediction by regression on cluster means with no adjustment for scale - mRNA data). The model R-Squared measure presented here is the mean R-Squared over all curves in the training set. The other measures relate to the Relative Lack of Fit over all curves in the test set.

Table 9 shows goodness of fit measures for modelling scale-free curves by 2 to 11 cluster means as basis functions. It can be seen that the number of clusters has little effect on the predicted curves’ goodness of fit. Examples of true and predicted curves using 3 and 11 clusters can be seen in Figures 47 and 48 respectively.

**Scale reconstruction** Weighted means and regression on cluster means with mapped curves require the reconstruction of a curve’s scale. Regression on cluster means with non-rescaled curves still requires estimation of the curve length. As mentioned, the scale features that need to be estimated in this case are the cycle’s length, minimum and maximum. Since our main interest is the comparison between shape prediction methods, the scale estimation was done using two types of rescaling: one is predicting scale features by linear regression and using these predictors for rescaling, the other is using the "best possible estimator" - the true test set scale features. True data
Figure 47: Scale-free mRNA data, predicted by regression on 3 basis functions. In all panels, $t$ runs from 0 to 400 and $y(t)$ from 1 to 2.
Figure 48: Scale-free mRNA data, predicted by regression on 11 basis functions. In all panels, $t$ runs from 0 to 400 and $y(t)$ from 1 to 2.
Table 9: Goodness of fit for different numbers of clusters - shape prediction regression on cluster means (mapped mRNA data). The model R-Squared measure presented here is the mean R-Squared over all curves in the training set. The other measures relate to the Relative Lack of Fit over all curves in the test set.

were used as a baseline for comparison: since scale reconstruction depends on the quality of a low-dimensional data predictor, this gives an indication as to the results that would be obtained by the "optimal" predictor available. In addition, since the scale reconstruction is independent of the number of clusters used for shape modelling, comparison will be made for models with 3 clusters only. Since in practice time is measured by discrete units, length predictions of all types were rounded to the nearest unit for computational reasons. A comparison between different predictions is given in Table 10. It can be seen that using separate models for shape and scale prediction produces better results than direct estimation of both the shape and the scale, even when the scale predictors are not very accurate.

Looking deeper into the estimation of scale features raises the question of their relative importance in reconstructing the scale. We examined this question by comparing 3 models, each using two estimated and one true scale features. Table 11 shows that using the true maximum brings the fit
### Table 10: Shape estimated by 3 basis functions. Scale reconstruction by cycle length, minimum and maximum - Lack of Fit criterion. Scale features are estimated by Linear Regression and then taken at their true values for comparison. Goodness of fit measure is taken over all curves in the training set.

<table>
<thead>
<tr>
<th>Shape estimation</th>
<th>Scale estimation</th>
<th>Mean</th>
<th>Median</th>
<th>Trimmed mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weighted means</td>
<td>Linear regression</td>
<td>0.2487</td>
<td>0.2235</td>
<td>0.2383</td>
</tr>
<tr>
<td></td>
<td>True value</td>
<td>0.1526</td>
<td>0.1346</td>
<td>0.1472</td>
</tr>
<tr>
<td>Scaled regression</td>
<td>Linear regression</td>
<td>0.2413</td>
<td>0.2218</td>
<td>0.2261</td>
</tr>
<tr>
<td></td>
<td>True value</td>
<td>0.1423</td>
<td>0.1273</td>
<td>0.1422</td>
</tr>
<tr>
<td>Non-scaled regression</td>
<td></td>
<td>0.4565</td>
<td>0.4301</td>
<td>0.4366</td>
</tr>
</tbody>
</table>

### Table 11: Scale reconstruction by cycle length, minimum and maximum - ALOF criterion, with one true value at a time (other features estimated by Linear Regression). Feature R-Squared is the goodness of fit of the linear regression model used to estimate this feature.

<table>
<thead>
<tr>
<th>True feature</th>
<th>Mean</th>
<th>Median</th>
<th>Trimmed mean</th>
<th>Feature R²</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>0.1748</td>
<td>0.1694</td>
<td>0.1729</td>
<td>0.909</td>
</tr>
<tr>
<td>Minimum</td>
<td>0.2319</td>
<td>0.1762</td>
<td>0.2129</td>
<td>0.822</td>
</tr>
<tr>
<td>Cycle length</td>
<td>0.2280</td>
<td>0.2382</td>
<td>0.2142</td>
<td>0.967</td>
</tr>
</tbody>
</table>

measures close to those obtained by using true values for all 3 scale features, whereas using the true minimum or length leaves the fits closer to those obtained by using estimates for all scale features. Table 11 also shows that the linear regression estimator for the maximum is quite accurate, reaching an $R^2$ of almost 91%, and yet substituting it for the true value has a noticeable effect on the fit measures. Therefore it seems that the maximum is the most essential feature in reconstructing the mRNA cycle scale.

**Non-rescaled Cycles** So far the complete curves served as a basis for predicting curves, and a small amount of information (say, the maximum value of each curve) was used when scale features were estimated. In this section we use the entire curve as a basis for the estimation of low-dimensional
scale features. The suggested method is to repeat the shape estimation, this time not mapping the curves to the same length but keeping them at their original lengths. Since the shape estimation procedure requires that all curves have the same length, a section was taken from each curve that starts at a beginning of a cycle (that is, at a local minimum) and has some constant length, therefore possibly including more than one cycle. The length of the time trace was chosen so that all curves are represented by at least one complete cycle. Such a curve is shown in Figure 49. These curves were clustered by shape, as before. The resulting estimated curves usually consisted of several unstable cycles, as shown in Figure 50, the first of which most resembles the original data. This estimation was done using 4 clusters, since this number of clusters is not too high, on the one hand, and proved to yield good estimation, on the other hand.
Figure 50: The estimated curve corresponding to Figure 49
Table 12: Length estimation - correlation and RMSE between estimated length and true length. Linear Regression included main effects only.

<table>
<thead>
<tr>
<th>Estimation method</th>
<th>Training set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Correlation</td>
<td>RMSE</td>
</tr>
<tr>
<td>Linear regression</td>
<td>0.9832</td>
<td>0.7237</td>
</tr>
<tr>
<td>Kriging</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PPreg</td>
<td>0.9913</td>
<td>0.5218</td>
</tr>
<tr>
<td>ACE</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Clustering</td>
<td>0.9572</td>
<td>1.581</td>
</tr>
</tbody>
</table>

Although this curve is a poor shape estimator, it is a good estimator of the curve’s length. Specifically, the estimator is the length of the first cycle in this curve. Table 12 shows the correlation between the true curve lengths and lengths estimated by different methods. When the estimated cycle lengths are incorporated in the overall curve estimation problem, some modification is required regarding the curve goodness of fit assessment. The measures presented in section 3.3.2 can only be used for equally long curves. However, since the cycle length itself is estimated and so bound to differ from the original curve’s length, evaluating the estimation fit cannot be done directly. This problem was solved by continuing the shorter curve in a cyclic manner to have the same length as the longer one, and then using the goodness of fit measure. Let $l_{\text{true}}$ and $l_{\text{est}}$ denote the true curve’s and the predicted curve’s lengths, respectively. Similarly, let $y(t)$ and $\tilde{y}(t)$ denote the true and predicted curves. Then add points to one of the curves following the rule,

$$
\begin{align*}
(y(l_{\text{true}} + 1), \ldots, y(l_{\text{est}})) &= (y(1), \ldots, y(l_{\text{est}} - l_{\text{true}})) & l_{\text{est}} > l_{\text{true}} \\
(\tilde{y}(l_{\text{est}} + 1), \ldots, \tilde{y}(l_{\text{true}})) &= (\tilde{y}(1), \ldots, \tilde{y}(l_{\text{true}} - l_{\text{est}})) & l_{\text{true}} > l_{\text{est}}
\end{align*}
$$

This combined clustering method raises the question of estimating scale features using clustered curves. This option makes sense if, in general, estimating a curve is more accurate than estimating low-dimensional data. The distance function that is used in the clustering process can be modified in order to show higher similarity between curves if their scale features that are
5.1.1 Selecting the Number of Clusters

The prediction error of the two stage models includes two components: one is an insufficiently accurate basis, the other is inaccurate estimation of the basis function’s coefficients. Since the coefficients depend on the selected basis functions, these errors are not independent. This dependence causes some trade-off between the two errors. If the number of basis functions is too small then it may not span a large enough space to accurately describe the untested time traces. On the other hand, a large number of basis functions enables flexible prediction for a larger span of the basis functions, yet requires predicting a large number of coefficients, resulting in large cumulative prediction error. The optimal number of clusters balances the two errors.

The contribution of each error component can be estimated by the following decomposition. In this decomposition the prediction error is compared to that of an oracle that produces optimal basis functions and optimal predictors for the coefficients of that basis.

Let $y \equiv y(t; x)$ denote the true function for a given untested observation $x$. Similarly, let $\hat{y}(\hat{g}(t); \hat{a}(\hat{g}(t))) \equiv \hat{y}(t; x; \hat{g}(t); \hat{a}(\hat{g}(t)))$ denote the estimated function for a certain basis $\hat{g}(t)$ with corresponding estimated coefficients $\hat{a}$. In addition, let $\hat{g}^*(t)$ and $\hat{a}^*(\hat{g}(t))$ denote the optimal predicted basis and corresponding coefficients, respectively.

$$y - \hat{y}(\hat{g}; \hat{a}(\hat{g})) = [y - \hat{y}(g^*; \hat{a}^*(g^*))] + [\hat{y}(g^*; \hat{a}^*(g^*)) - \hat{y}(\hat{g}; \hat{a}^*(\hat{g}))] + [\hat{y}(\hat{g}; \hat{a}^*(\hat{g})) - \hat{y}(\hat{g}; \hat{a}(\hat{g}))]$$

Let us examine each component of the prediction error separately.

The term $[y - \hat{y}(g^*; \hat{a}^*(g^*))]$ is a lower bound on the prediction error, as it is the result of a hypothetical oracle predictor.

The term $[\hat{y}(g^*; \hat{a}^*(g^*)) - \hat{y}(\hat{g}; \hat{a}^*(\hat{g}))]$ estimates the error caused by a
non-optimal basis. In order to evaluate this term we need to define the optimal basis \( g^* \) that can be obtained from the training data. Obviously the widest span of basis functions that the training data can generate consists of every single function in the training data. In practice, however, it can be seen in Tables 8 and 9 that excellent fit values of \( R^2 \) close to 1 were obtained using a much smaller set of basis functions. In this case it seems reasonable to define \( g^* \) as the set of basis functions for which a large enough fit value can be obtained for the training set.

The optimal coefficients \( a^* (g^*) \) can be obtained by allowing the oracle to use the test data as well as the training data. Given a least squared error criterion, estimating \( a^* (g^*) \) by using ordinary least squares estimation on the test data with \( g^* \) as basis would yield the optimal coefficients for that basis.

The term \( \hat{y} (\hat{g}; \hat{a} (\hat{g})) - \hat{y} (\hat{g}; \hat{a} (\hat{g})) \) reflects the lack of fit caused by inaccurate estimation of the coefficients. It can be estimated using the oracle predictors mentioned before and the estimated function \( \hat{y} \).

The optimal number of clusters is that for which the sum of errors caused by using a sub-optimal basis and sub-optimal coefficients, compared to the oracle, is minimal.

\subsection*{5.1.2 Estimating Biological Quantities}

In this section we estimate two biological quantities that characterised each curve. One is the dichotomous quality of whether a certain experimental point produced a cyclic curve, the other is the proportion of time that a curve remained above a given value. We compare our estimates to those obtained by known methods that do not rely on estimating the entire curve.

\textbf{Valid Cycle Domain} As mentioned, some runs of the simulation resulted in non-cyclic functions that are biologically unrealistic. olde Schepers,
Klinkenberg, Pennartz, and van Pelt (1999) showed that each of the 7 variables that create the simulation has a range of values that lies around the free-run value and produces cyclic output. These ranges were found for each parameter holding the rest at their free-run values. In this section we aim to construct a predictor for the generation of a cyclic function given the input variables, while taking into account their mutual effect on the output.

Let us denote the part of the experimental domain that produces cyclic curves by \( D \), so the part that does not produce such curves is denoted \( \overline{D} \). In section 5.1 we presented scale feature predictors based on all observations \( x \in D \) in the training set, while \( \overline{D} \) was excluded from the dataset. The predictor we suggest in this section is based on the assumption that using the scale feature predictors on \( x \in \overline{D} \) as opposed to \( x \in D \) would produce results that lie on distinct ranges. Given this fact, the scale prediction of an observation \( x \) can point out whether \( x \in D \) or \( x \in \overline{D} \). Using the above notation, let \( h_i(x) \) be the scale feature estimator. We seek a predictor of the form:

\[
\delta(x) = \begin{cases} 
\text{True} & x \in D \\
\text{False} & x \in \overline{D} 
\end{cases}
\]

In practice, three scale features were selected: the predicted minimum, maximum and length of a curve. These features were predicted for \( D \cup \overline{D} \), using a linear regression model that was based on \( D \) alone. An independent samples t-test was used to determine that there is a significant and large difference in the mean predicted values between \( D \) and \( \overline{D} \), in all three features. Each feature was assigned a cutpoint \( c_i \), defining three indicators \( R_i = \{h_i(x) \leq c_i\} \) or \( R_i = \{h_i(x) > c_i\} \). The cutpoint \( c_i \) was determined as the value that minimises the number of prediction errors over the training set, when the predictor is \( \delta(x) = R_i \). The following indicators were found to be best, where \( \min \), \( \max \) and \( \text{length} \) denote the predicted feature for a given observation \( x \):

\( R_1 = \{\min \leq 0.57\} \) with 2 false negative and 8 false positive predictions
out of 200 observations.

$R_2 = \{\text{max} > 0.73\}$ with 8 false positive predictions.

$R_3 = \{\text{length} > 176\}$ with 21 false negative and 9 false positive predictions.

If we use a single indicator, then $R_2$ is the best choice. However, a combination of rules can improve prediction. The predictor that produced the best results is $\delta(x) = R_1 \cap R_2$ with 3 false positive and 2 false negative predictions. This is illustrated in Figure 51.
The results obtained by this predictor were compared to those obtained by conventional methods. A discriminant analysis model was tested, as well as stepwise logistic regression. In addition, the means of the min, max and length were set as centers for each one of the groups, and prediction was done according to the nearest center by Mahalanobis distance. A summary of the results is given in Table 13.

### Proportion of time above a given value

The biological characteristic that is estimated in this section is the proportion of time that a curve obtains a value greater than 1. This value was chosen as it is an overall mean of the dataset’s minima and maxima. We predict this proportion for an untested observation \( x \) by first obtaining the predicted curve \( \hat{y}(t; x) \) and then calculating the proportion of time that the predicted curve remains above 1. The curve was predicted by regression on 3 cluster means. This estimator was compared to common estimation methods: linear regression, Ace, Projection Pursuit regression and Kriging. The estimates were compared by two measures: the correlation between the estimate and the true value, and the RMSE of the difference between the estimate and the true value. The results are shown in Table 14. It can be seen that prediction using the curve estimation produced the best results by both criteria.

To conclude this case study, it was found that all factors have a significant effect on at least one of the coefficients of the basis functions, therefore affecting the curve’s shape. All factors but the protein production and de-

---

<table>
<thead>
<tr>
<th>Method</th>
<th>False Positive</th>
<th>False Negative</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discriminant Analysis</td>
<td>9</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>8</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Mahalanobis Distance</td>
<td>1</td>
<td>33</td>
<td>34</td>
</tr>
<tr>
<td>( \delta = R_1 \cap R_2 )</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 13: Number of miscalssifications in predicting validity of curves
### Table 14: Proportion of time above 1 - comparison of estimates

<table>
<thead>
<tr>
<th>Method</th>
<th>Correlation</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted curve</td>
<td>0.9709</td>
<td>0.1269</td>
</tr>
<tr>
<td>Linear regression</td>
<td>0.9438</td>
<td>0.1676</td>
</tr>
<tr>
<td>Ace</td>
<td>0.5608</td>
<td>0.4331</td>
</tr>
<tr>
<td>PPReg</td>
<td>0.9012</td>
<td>0.2287</td>
</tr>
<tr>
<td>Kriging</td>
<td>0.9519</td>
<td>0.1793</td>
</tr>
</tbody>
</table>

Hill’s coefficient, and both Hill’s coefficient and the protein degradation rate affect the extreme values of the curve.

The factors affecting the cluster assignment were tested using Analysis of Variance, similarly to what was done in section 4.1.4. It was found the for a small number of clusters all factors but Protein degradation rate constant and Non-linearity in protein synthesis cascade, are effective. However, when modelling by a large number of clusters there is little consistency in the effectiveness of the factors.

## 6 Concluding Remarks

In this work we examined several methods for modelling time traces. Most of the models discussed in this work consist of two stages: high-dimensional modelling of the time variable by selecting an appropriate set of basis functions, and low-dimensional modelling of the coefficients of the selected bases or other related parameters. We tested methods for both stages.

In selecting a set of basis functions we address two issues: the form of the basis (for example, polynomial, B Spline etc.) and the number of functions in the basis. We tested two main types of basis functions: known or custom-made. A custom made basis, based on clustering the data by their shape, has the advantage of capturing the data’s unique shape properties and therefore
is likely to fit the data more accurately than a known basis. In the examples presented in this work, a small number of custom made basis functions was sufficient for accurately modelling the data.

A few points of interest are left for future research, regarding the selection of a proper basis. The main issue is finding more methods of selecting the family of functions from which the basis will be taken. For example, Functional Principal Components Analysis can provide a basis for modelling time. Another issue is selecting the number of functions that will serve as the basis. A large number of functions is likely to fit the training data better, yet will require estimating a large number of coefficients, which will eventually reduce the model’s prediction accuracy.

It should also be noted that the data used as examples in this work were quite smooth and similarly shaped. Further research can include data sets with higher variability between functions. The number of basis functions that are used may well depend on such a measure of data variability and smoothness.

Low-dimensional modelling was applied for two types of parameters: one is the basis coefficients, the other is additional data that can be derived from the training set, such as the estimated value of the time trace at one or more time points, the minimum or maximum of the function. Those *scale features* were estimated from the data independently of the curve estimation process. We used known methods, such as linear regression, ACE etc., as well as custom made estimation by clustering. Of the known methods, modelling by Kriging seemed to produce the best results., and the estimation by clustering also proved to be accurate.

Two main issues need to be addressed when considering low-dimensional modelling: One is what estimation method to use, the other is what additional data should be estimated and incorporated in the final model. The second question arises strongly when estimation is separated to *shape* and
scale. Results show that shape estimation is usually an easy task, whereas scale reconstruction, based on low-dimensional scalar estimation, is by far more challenging. Good fits are achieved by this method if scale features are wisely selected and accurately estimated. In our case studies, using good estimators for the minimum or maximum values of the function resulted in better fits than information about the function’s value at given time points. Future research will be done in order to determine what part of the data should be assigned an independent estimator, and what estimation method will provide best results. For example, clustering was used to estimate some biological features of the data; it may be used as an estimation method for other scalars or even the function’s coefficients.


Drignei, D. "Empirical Bayesian Analysis for High-Dimensional Com-


Govaerts, B., Noel, J. "Analysing the Results of a Designed Experiment When the Response Is a Curve: Methodology and Application in Metal Injection Moulding", Quality and Reliability Engineering International, 21 (5), (2005), 509-520.


Kennedy, M.C., O'Hagan, A. "Predicting the Output from a Complex Computer Code when Fast Approximations Are Available", Biometrika, 87


