Functional Regression Analysis and Variable Selection for Motion Data

Yafeng Cheng

Thesis submitted for the degree of
Doctor of Philosophy

School of Mathematics & Statistics
Newcastle University
Newcastle upon Tyne
United Kingdom

January 7, 2016

Acknowledgements

I would like to thank my supervisor Dr. Jian Qing Shi for his valuable guidance and continuous support of my Ph.D study and related research. It is a privilege to be supervised by Dr. Shi during this four years of study. He has helped me grow enormously not only in academic knowledges but also in academic thinking and pursuing ideas. His constantly encouragement, patient and positive guided me to my targets. I am also very thankful to Dr Javier Serradilla and Prof Janet Eyre for their insightful comments, encouragement and supports.

Thanks to my colleagues in the School of Mathematics and Statistics for sharing knowledge, offering helps and enjoying joyful moments. I thank the general office in School of Mathematics and Statistics for their supposes.

I am grateful to my mum, dad, uncle and aunt for their encouragement and support every single day. Finally, I am thankful to Doudou who has loved me and kept me company through all the joys and difficulties.

Abstract

Modern technology offers us highly evolved data collection devices. They allow us to observe data densely over continua such as time, distance, space and so on. The observations are normally assumed to follow certain continuous and smooth underline functions of the continua. Thus the analysis must consider two important properties of functional data: infinite dimension and the smoothness. Traditional multivariate data analysis normally works with low dimension and independent data. Therefore, we need to develop new methodology to conduct functional data analysis.

In this thesis, we first study the linear relationship between a scalar variable and a group of functional variables using three different discrete methods. We combine this linear relationship with the idea from least angle regression to propose a new variable selection method, named as functional LARS. It is designed for functional linear regression with scalar response and a group of mixture of functional and scalar variables. We also propose two new stopping rules for the algorithm, since the conventional stopping rules may fail for functional data. The algorithm can be used when there are more variables than samples. The performance of the algorithm and the stopping rules is compared with existed algorithms by comprehensive simulation studies.

The proposed algorithm is applied to analyse motion data including scalar response, more than 200 scalar covariates and 500 functional covariates. Models with or without functional variables are compared. We have achieved very accurate results for this complex data particularly the models including functional covariates.

The research in functional variable selection is limited due to its complexity and onerous computational burdens. We have demonstrated that the proposed functional LARS is a very efficient method and can cope with functional data very large dimension. The methodology and the idea have the potential to be used to address other challenging problems in functional data analysis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As technology continuous to evolve, the structures of the data we could obtain become more complex. One of the new data type is functional data, where the basic units of the data are functions rather than points. One example of this is the movement data that we will discuss in this thesis, where records are signals with high frequencies. On the other hand, the purpose of data analysis remains the same, and we still interest in the traditional statistical problems, such as regression and classification, etc. New data structures, such as functional data, bring new challenges to the analysis. We will address some of these challenges in this thesis, by using both scalar and functional data.

The thesis consists of two parts. The first will focus on functional variable selection while the second will focus on applying this variable selection method to analyse large, real motion data.

Part I: Variable Selection in Functional Linear Regression

Functional regression is the regression model with functional variables in either response or predictors, or both of them. The model we are interested in has a scalar variable as the response and both scalar and functional variables can be included in the predictors. The focus in the literature is univariate functional regression, i.e., only one functional variable is used in the predictor. However, as data collection technology becomes more powerful, multidimensional functional variables in one data set become more common. Thus, we have to treat each one of the functional variables as a whole in the analysis. The research in this area is still developing. As the number of functional variables increases,

the variable selection problem in the regression analysis becomes more important.

There are two major difficulties associated with functional variable selection. One is that the number of predictors could be larger than the sample size. The other one is the functional structure of the variables and the coefficients should be captured in the analysis. The former difficulty is one of the problems that generally occurs in variable selection algorithms. This is caused by the collinearity from the large number of candidate predictors. The later difficulty is from the smoothness behaviour of the functional objects. We need to consider the smoothness of functional variables in the regression and make sure the estimated functional coefficients are smooth functions. Moreover, from a computational point of view, the functional variables add an additional burden to the computation of the variable selection procedure.

We propose a new functional variable selection algorithm, called functional least angle regression to overcome the difficulties above. It is an extension of the least angle regression processed by Efron et al. (2003) from a multivariate domain to a functional domain. The key in least angle regression is the linear correlation between the residuals and the predictors measured by Pearson's correlation coefficient. We replace this correlation by an extension of functional canonical correlation analysis. We use this correlation analysis to calculate the correlation between a scalar variable and a set of mixed scalar and functional variables. The functional variables are represented as discrete data points, while functional coefficients could be represented by different methods. We introduce three different representation methods for functional coefficients: discrete data points, basis functions and Gaussian quadrature. To the best of our knowledge, the last method has not been used in functional regression analysis before. The smoothness of the functional coefficients is controlled by the roughness penalty, and the tuning parameter for the penalty function is selected by generalized cross validation to avoid heavy computational costs. By using this correlation measure, we can achieve accurate and efficient estimation of the coefficients for the variables in the functional least angle regression. Least angle regression can be modified to give a better estimation of the regression equation. We also propose a modification addressing the same problem by checking the variance of the projection of each selected variable. Because of the difficulty in calculating the degrees of freedom of the regression equation, conventional stopping rules cannot provide accurate information about the stopping point. Therefore we propose two new stopping rules for practical usage. These would allow the algorithm to stop immediately after the correct stopping point and therefore further reduce the computational time.

In Part I, we first briefly introduce and review recent developments in functional data analysis in Chapter 2. We then explore the functional canonical correlation analysis in Chapter 3. Functional least angle regression for functional predictors and mixed scalar and functional predictors are discussed in Chapter 4 and Chapter 5, respectively. Comprehensive simulation studies are included in both of these chapters for different correlation structures.

Part II: Application of the Functional Regression Analysis in Motion Data

The motion data we will analysis in this thesis come from a study of the recovery of stroke patients using the signal data from a video game played by stroke patients at home. We want to build a regression model to represent the dependence level of stroke patients in daily life and do prediction for new patients.

The raw data set contains rich information, but many pretreatment steps are required to better compare the difference between samples. After the pretreatment of the data, we obtain a large number of scalar and functional variables. The total number of these two types of variables is much larger than the sample size. Thus the variable selection is one of the necessary tasks to do. As this is a longitudinal study, the within and between patient variation should also be considered in the analysis. In addition, the change of the dependence level of any patient should be smooth, unless a second stroke occurred during the data collection period.

Many models are tested and a few typical models are listed in detail in the thesis. For selecting a good subset of predictors, least angle regression is used in the model with scalar variables only, while functional least angle regression is used in the model with both scalar and functional variables. The best model is the mixed effects model with functional linear regression in fixed effect and Gaussian process in the random effect. The functional linear regression is used to capture the linear variation while Gaussian process is used to capture the non-linear variation and to introduce smoothness to the change of the upper limb function of individual patients.

In Part II, we first introduce the background of the motion data in detail in Chapter 6. The models will be explained in Chapter 7 with model comparison from repeated k-fold cross validation at the end.

Part I

Functional Variable Selection

Chapter 2

Introduction for Functional Data Analysis

Functional data analysis, or FDA in short, uses functional variables in addition to the conventional multivariate data analysis. The most common characteristic of functional variable is that the observed samples follow some smooth underlying functions. Functional variables can be thought as the observations of an object changes its state on a continua, such as time t. However, different observation qualities may give different types of functional variables. Ideally, we want the functions to be observed on all $t \in [0, T]$. Since t is continuous, the ideal case would give infinite data points, which is not feasible to use practically. Typically, we can have the function observed on a grid of time t. The grid can be dense, such as the well studied children's gait data from Olshen et al. (1989) and Berkeley growth data studied in Ramsay and Silverman (2005). When the observation grid is sparse, the data are more closely related to the longitudinal data. For example, the longitudinal data recording the recovery of a patient can be thought as sparsely observed functional data as the patient should recovery smoothly over time.

In this thesis, we focus on the regression problem in functional data analysis, but there are other aspects in functional data analysis which are equally important. We give an introduction and review for some of the topics in FDA in this chapter. First of all, we briefly introduce smoothing and registration for functional variables in Section 2.1. Secondly, we roughly review the popular representative of functional variables in Section 2.2, i.e., the functional principle component analysis. We then show some recent development in functional regression with brief discussions in Section 2.3.

2.1 Smoothing and Registration for Functional Data

Smoothing can reduce the noise from observation. But more importantly, it can recover the underlying functions from the observed functional data. Registration, on the other hand, can change the phases of the samples of the functional variables slightly to align the important features for future analysis.

2.1.1 Smoothing

Suppose we observe a realization of a functional variable $X(t)$ at K discrete time points $(t_1, t_2, \ldots, t_K)$ with values $\mathbf{x} = (x_1, x_2, \ldots, x_K)$, where $t \in [0, T]$ and $0 \leq t_1 < t_2 < \ldots <\leq T$. Denote the underlying function of this realization as $x(t)$, then we have:

$$x_k = x(t_k) + \epsilon_k.$$

There are many methods that can be used in data smoothing. One of the most commonly used method is to use basis functions to represent the functional object. Suppose we have known basis functions $\Phi_m(t)$, we can express the functional object $x(t)$ by the basis functions:

$$x(t) = \sum_{m=1}^{\infty} c_m \Phi_m(t),$$

where c_m are unknown basis coefficient. Practically, we can use M basis functions instead of an infinite number of basis function to represent the observed data $\mathbf{x}$:

$$\mathbf{x} = x(t) + \boldsymbol{\epsilon} \approx \sum_{m=1}^{M} c_m \Phi_m(t) = \mathbf{c}\boldsymbol{\Phi},$$

where $\boldsymbol{\epsilon}$, $\mathbf{c}$ and $\boldsymbol{\Phi}$ are the vector form of the error, basis coefficient and basis functions, respectively. The basis coefficient is the unknown quantity to estimate. We can use least square or penalized least square to estimate $\hat{\mathbf{c}}$.

For least square estimation, we have:

$$\hat{\mathbf{c}} = \boldsymbol{\Phi}\left(\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}\mathbf{x}.$$

The smoothness of the outcome is controlled by the number and the order of the basis

functions. This can be done by cross validation. If the basis functions have a large M and a high the order, they may overfit the observed data. However, if the basis functions have a small M and a low order, they may give a fitted curve that is too smooth to capture the features of the observed data.

For penalized least square estimation, we have:

$$\hat{\mathbf{c}} = \boldsymbol{\Phi}\left(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda \text{Pen}\right)^{-1}\boldsymbol{\Phi}\mathbf{y}, \tag{2.1}$$

where Pen is the penalty function, and λ is the smoothing parameter. With the penalty function, we can use a set of basis functions with relatively large number and relatively high order in the calculation and let the smoothing parameter λ control the smoothness. As λ increases, the smoothness of the outcome increases, and vice-verse.

The penalty function can be written as $Lx(t)$, where L is the linear differential operator for $x(t)$. The choice of L depends on the choice of the basis functions we use in the smoothing process. We first show two examples for commonly used basis functions, spline basis and Fourier basis. Let us denote $D^n x(t)$ as the n-th derivative of $x(t)$.

The spline basis is a set of basis function of t. Without loosing the generality, the functional object can be expressed as a polynomial of order q, where q is the order of the basis function:

$$x(t) = c_1 + c_2 t + c_3 t^2 + \ldots = \sum_{q=1}^{Q} c_q t^q.$$

The commonly used penalty function is $L = D^2$. We can relate this penalty function to the differential equation $D^2 x(t) = 0$. The general solution of it is

$$A_1 + A_2 t,$$

for any constant A_1 and A_2. Intuitively, when λ increases in the Eqn (2.1), the coefficients corresponding to the basis functions with order higher than one would tends to zero. Therefore, if $\lambda = \infty$, the outcome is the fitted value of the linear regression $\mathbf{x} = A_1 + A_2\mathbf{t} + \boldsymbol{\epsilon}$, where $\mathbf{t}$ is the continuum, and $\boldsymbol{\epsilon}$ is the error.

If we use a set of Fourier basis, the functional object can be written as:

$$x(t) = c_1 + c_2 \sin \omega t + c_3 \cos \omega t + c_4 \sin 2\omega t + \ldots,$$

where $1/q\omega$ is the period of the $(2q)$-th and $(2q+1)$-th Fourier basis functions. One of the popular penalty functions for Fourier basis functions is $L = D^3 + \omega^2 D$. We can also connect it to the differential equation $D^3x(t) + \omega^2 Dx(t) = 0$. The general solution of it is:

$$A_1 + A_2 \cos \omega t + A_3 \sin \omega t,$$

for any constant A_1, A_2 and A_3. Intuitively, the Fourier basis functions with short periods would have less effect to the outcome when λ increases, and the outcome becomes closer to the fitted value of the non-linear regression $\mathbf{x} = A_1 + A_2 \cos \omega \mathbf{t} + A_3 \sin \omega \mathbf{t} + \boldsymbol{\epsilon}$.

Similarly, we can define other linear operators of L. Table 2.1 shows a few more general penalty functions with corresponding components in the general solutions:

	Operator	terms
1	D^q	$\{1, t, ..., t^{q-1}\}$
2	$D^2 + \omega^2$	$\{cos(\omega t), sin(\omega t)\}$
3	$D + A$	$\{exp(-At)\}$

Table 2.1: General solution for low order different differential operator

where A is a constant; q is the order of the derivative. They can be considered for specific data sets and basis functions in practical use.

2.1.2 Registration

Suppose we have two realizations $x_1(t)$ and $x_2(t)$ of the functional variable $X(t)$. The registration process changes the continuum t for each realization by using wrapping functions $h_1(t)$ and $h_2(t)$ such that $x_1(t)$ and $x_2(t)$ becomes $x_1(h_1(t))$ and $x_2(h_2(t))$ respectively, with the important features aligned. This can help to capture the important features when analysing the data.

2.2 Representative of Functional Data

In addition to the basis function expressions, the most commonly used method to represent the functional variables is the functional principle component analysis (FPCA). There is an extensive research on the topic of the FPCA, such as Hall et al. (2006); Ramsay and

Silverman (2005). We briefly review the method of FPCA here as this method is often connected to the regression problem in FDA.

The FPCA is designed for fully observed independent samples from one functional random variable. The idea is to represent the functional variable by a set of orthogonal basis functions estimated from the auto-covariance matrix. This set of basis functions can be used as normal basis functions on any functional object. The original version of the FPCA has a few limitations. In order to address these limitations, many variations of the FPCA have been proposed. Here we show three typical methods targeting problems from sparse observation grid, multi-dimensional functional data and dependent samples, respectively. Yao et al. (2005a) extend the FPCA to sparse observed data, or longitudinal data. The data are collected with sparse observation grid. Due to the small sample size from each subject, conventional smoothing techniques cannot be applied to recover the underlying functions with respect to each of the sample. Thus, a mean function is calculated by smoothing over all the samples from all the subject. The FPCA for the i-th sample is carried out using the corresponding covariance matrix Σ_i estimated from the subset of the auto-covariance matrix of the whole data set. The conditions of this method are weak and easy to satisfy. Ramsay and Silverman (2005) gives details about the FPCA and extends the FPCA to multivariate case. The method they use considers the information in both auto-covariance and cross-covariance of the functional variables. Di et al. (2009) introduced the multi-level FPCA to deal with the longitudinally collected signal data from a large number of subjects. This method targets the variations on the within subjects level and between subjects level at the same time.

2.3 Regression Analysis of Functional Data

The regression analysis in FDA can have functional variables in both response and predictors. We first review functional regression with scalar response, as it is our main focus in the later chapters. We will also look at the functional regression with functional response here.

A large proportion of the literature for functional regression is concerned with univariate functional regression. The univariate functional linear model with scalar response is:

$$y = \mu + \int x(t)\beta(t)dt + \epsilon,$$

where y here is the response; μ is the intercept; $x(t)$ is the functional variable; $\beta(t)$ is the functional coefficient and ϵ is the noise follows a normal distribution.

We can use basis functions to represent the functional objects and turn this model to the straight forward extension of the multivariate linear regression. Similar to smoothing, the choice of the basis functions for the functional variable and the functional coefficient can affect the estimations. One popular choice is to use the same set of orthogonal basis functions for both functional variable and functional coefficients. This can reduce the complexity of the estimation, since the orthogonal basis would cancel out in the calculation. Or we can use spline basis with low order and small numbers to control the smoothness of the functional objects. We can also introduce a roughness penalty to control the smoothness by a smoothing parameter.

As we mentioned above, the basis functions can be provided from the FPCA. The eigenfunctions we obtain from FPCA are orthogonal basis driven by data. Cardot et al. (1999); Reiss and Ogden (2007) study this model. C.Crainiceanu (2009) proposed generalized multilevel functional regression following the work from the multilevel FPCA. Mixed effects model is carried out using the principle components to capture the variation between different levels.

Similar to principle component regression in the multivariate case, regression with the FPCA has an obvious disadvantage, i.e., the principle components are obtained without considering any information from the response variable. Thus the focus is moving away from functional regression with the FPCA to functional regression with other discrete methods such as basis functions representations and even the discrete data points of the functional variables. We will discuss the details of the discrete methods in Chapter 3.

We can see other alternative methods for functional linear regression with scalar response and univariate functional covariates. Müller and Yao (2012) propose a functional additive model. The model uses the components from FPCA to build an additive model. It is more flexible than the original FPCR, and it is able to overcome some problems that FPCR has. Functional partial least square (or FPLS) extends the partial least square method from a multivariate domain to functional domain. Reiss and Ogden (2007) propose FPLS smoothed by either basis functions or roughness penalties. James et al. (2009) propose a method to give a better estimation of the functional coefficient in the functional linear regression model with scalar response. The method targets the problem that when the functional coefficient is exactly zero at some t, the estimation at and near t would be inaccurate. If the functional coefficient is zero in an area on t, the problem would be worse.

Since most of the time the functional coefficients are non-zero on most of $t \in [0, T]$, it would be difficult to locate the zero positions. However, the derivatives of the functional coefficient could be sparse. Instead of estimating the functional coefficient, they then estimate the n-th order derivative of the functional coefficient, which can be sparse and therefore easily estimated by conventional variable selection methods.

Functional generalized linear regression has also been studied. Müller and Stadtmüller (2005) proposed a generalized functional linear model. The functional objects are represented by orthogonal basis. The algorithm to estimate the parameters can work with non-exponential family errors. J.Goldsmith et al. (2011) propose a generalized linear regression model with sparse observed noisy functional variables. The auto-covariance matrix is first calculated using the raw sparse observed functional variable. Instead of smoothing the functional variables, the auto-covariance matrix is smoothed to perform FPCA. They then recover the functional variables by the data driven orthogonal basis and perform the generalized linear regression. This method can also be extended to multivariate functional regression.

Multivariate functional regression has become common recently. Fan and James (2013) propose a functional additive model with scalar response and functional predictors:

$$y = \sum_{p=1}^{P} f_p(x_p(t)) + \epsilon,$$

where $f_p(\cdot)$ could be linear or non-linear and f_p for non-linear additive model could be unknown. They also propose the variable selection and model selection methods for both linear and non-linear additive models. The selection for linear model is done by group lasso with standardization. The selection for non-linear models is done by iterative method. This approach is closely related to that proposed in Fan et al. (2014), where the response variable is functional.

Functional regression with functional response is also a popular area. As this topic is not our focus, we only briefly introduce a few methods proposed in this area. He et al. (2000) use functional canonical correlation between two functional variables to model the relationship between a functional response and a functional predictor. Yao et al. (2005b) extend the FPCA for longitudinal data to the longitudinal regression model. Both response variable and predictor are observed longitudinally and treated as sparsely observed functional variables. The FPCA is carried out individually for both response and predictor, and regression analysis is carried out using the principle components from both func-

tional variables. Shi and Wang (2008) connect concurrent functional regression with a Gaussian process for curve prediction. The functional regressions are used as the mean model and greatly increase the prediction accuracy for the curves. Rosen and Thompson (2009) study a very complex model. Both response and predictor are multivariate functional variables. This model also considers the dependences between the response functional variables.

Our focus is in the area of functional regression with scalar response and multivariate functional variables. But before we look at the regression model, we will study the relationship between a set of functional variables and a scalar variable in Chapter 3.

Chapter 3

Correlation for Functional Data

In order to build a functional regression model and also to perform variable selection for functional data, we need to study the correlation between one functional variable (or one set of functional variables) and one or more other variables. The latter could be functional or scalar. More specifically, we are interested in measuring the correlation between the information contained in two sets of functional or scalar variables.

The choices of correlation measures between random variables are limited. However, there exist many correlation measures between random vectors. Functional random variables can be thought as random vectors with complex structures. Thus we start by learning different correlation measures in this chapter. The correlation we focus on in this chapter is the canonical correlation analysis, which is an extension of commonly used Pearson's correlation. Functional canonical correlation analysis has been studied in the literature for some time (Ramsay and Silverman (2002, 2005)). We are going to extend it to measure a more general relationship for functional data, such as the correlation between two groups of functional variables and the correlation between one scalar variable and one group of functional variables.

This chapter is organized as follows. A number of correlation measures are reviewed in Section 3.1, followed by the introduction of the canonical correlations in Section 3.2. Canonical correlation between functional variables and canonical correlation between scalar and functional variables are discussed in details in Section 3.3 and 3.4, respectively. A simulation study is given in Section 3.5.

3.1 Different measures of correlation

The most commonly used measure for determining the linear correlation between two random variables is Pearson's product-moment correlation coefficient. Normally if not specified, the term "correlation" means Pearson's correlation. The formula for correlation ρ between two random variables X and Y is

$$\rho_{X,Y} = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\mathbb{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}, \tag{3.1}$$

where $\mathrm{Cov}(X,Y)$ is the covariance of X and Y, σ_X is the population standard deviation of X, μ_X is the population mean of X, and $\mathbb{E}[\cdot]$ is the expectation. In practice, the population mean and standard deviation are replaced by the sample mean and standard deviation. When $\rho_{X,Y} = \pm 1$, X and Y are exactly correlated and can be written as:

$$X = aY + b$$

for any constant a ($a \neq 0$) and b. When $\rho_{X,Y} = 0$, X and Y are uncorrelated, but not necessary independent.

The correlation between two groups of random variables or random vectors is more complex. We will now review several commonly used measures. Assume we have two data sets $\mathbf{X}$ and $\mathbf{Y}$ generated from two random vectors with the dimension p_x and p_y respectively. We assume that the sample size for $\mathbf{X}$ and $\mathbf{Y}$ are both n.

1. R_v coefficient

   The R_v coefficient was proposed by Yves Escoufier in the early 70's, and the original articles are in French. Later Robert and Escoufier (1976) relate the R_v coefficient with other multivariate data analysis tools including principle component analysis. This coefficient gives an indication of the linear relationship between two sets of data points. The two sets of data could be high dimensional. Let us denote the dimension of X and Y to be p_x and p_y, respectively. For this particulate correlation measure, we need to have $p_x = p_y$.

   The formula of R_v is:

$$\rho_{\mathbf{X},\mathbf{Y}} = \frac{\mathrm{tr}(\boldsymbol{\Sigma}_{xy}^T \boldsymbol{\Sigma}_{xy})}{\sqrt{\mathrm{tr}(\boldsymbol{\Sigma}_{xx}^T \boldsymbol{\Sigma}_{xx})\mathrm{tr}(\boldsymbol{\Sigma}_{yy}^T \boldsymbol{\Sigma}_{yy})}},$$

where $\boldsymbol{\Sigma}_{\cdot\cdot} = \text{Cov}(\cdot, \cdot)$. More specifically,

$$\boldsymbol{\Sigma}_{xy} = \frac{\mathbf{X}^T\mathbf{Y}}{(n-1)}, \quad \boldsymbol{\Sigma}_{xx} = \frac{\mathbf{X}^T\mathbf{X}}{(n-1)}, \quad \boldsymbol{\Sigma}_{yy} = \frac{\mathbf{Y}^T\mathbf{Y}}{(n-1)}.$$

This is a well studied linear correlation measure. There are many modifications of it. For example, the Procrustes coefficient is:

$$\rho_{\mathbf{X},\mathbf{Y}} = \frac{\sqrt{\text{tr}(\boldsymbol{\Sigma}_{xy}^T\boldsymbol{\Sigma}_{xy})}}{\sqrt{\text{tr}(\boldsymbol{\Sigma}_{xx}^T\boldsymbol{\Sigma}_{xx})\text{tr}(\boldsymbol{\Sigma}_{yy}^T\boldsymbol{\Sigma}_{yy})}},$$

and modified R_v coefficient is:

$$\rho_{\mathbf{X},\mathbf{Y}} = \frac{\text{tr}([\boldsymbol{\Sigma}_{xy} - \text{diag}(\boldsymbol{\Sigma}_{xy})]^T[\boldsymbol{\Sigma}_{xy} - \text{diag}(\boldsymbol{\Sigma}_{xy})])}{\sqrt{\text{tr}([\boldsymbol{\Sigma}_{xx} - \text{diag}(\boldsymbol{\Sigma}_{xx})]^T[\boldsymbol{\Sigma}_{xx} - \text{diag}(\boldsymbol{\Sigma}_{xx})])\text{tr}([\boldsymbol{\Sigma}_{yy} - \text{diag}(\boldsymbol{\Sigma}_{yy})]^T[\boldsymbol{\Sigma}_{yy} - \text{diag}(\boldsymbol{\Sigma}_{yy})])}}.$$

The former one is not scale invariant (Josse and Holmes (2013); Josse (1971)), but it is popular in the areas such as ecology and morphology.

We can further modify it to obtain non-linear correlation measures. This is done by changing the form of the $\boldsymbol{\Sigma}$'s to that of the other dissimilarity matrices. It is referred to as the generalized R_v coefficient and was proposed by Minas et al. (2013). The properties of the generalized R_v coefficient depend on the properties of the new $\boldsymbol{\Sigma}$'s.

2. Congruence coefficient
   Another closely related correlation measure is the congruence coefficient. It was first proposed by Cyril Burt in 1948. The formula of this measure is very simple

$$\rho_{\mathbf{X},\mathbf{Y}} = \frac{\sum(\mathbf{X}^T\mathbf{Y})}{\sqrt{(\sum(\mathbf{X}^T\mathbf{X}))\,(\sum(\mathbf{Y}^T\mathbf{Y}))}}.$$

   where $\sum(\ldots)$ is the summation of all the elements in the matrix.

   This correlation measure uses the data directly rather than the covariance matrix we see in the R_v coefficient. And thus we can have $p_x \neq p_y$ in this case. It is widely used in measuring the similarity between the factors in two matrices, such as Lorenzo-Seva and Ten Berge (2006).

3. Mantel coefficient
   The Mantel coefficient was originally introduced by Mantel (1967). This is a pop-

ular correlation measure especially in ecology. Similar to the R_v coefficient, the Mantel coefficient uses the distance matrices to do the calculation. Consider $\boldsymbol{\Sigma}$ as one kind of distance or dissimilarity matrix, and $\boldsymbol{\Sigma}^u$ as the upper triangular matrix of $\boldsymbol{\Sigma}$. The formula is:

$$\rho_{\mathbf{X},\mathbf{Y}} = \frac{\sum[(\boldsymbol{\Sigma}^u_{xx} - \bar{\boldsymbol{\Sigma}}^u_{xx})(\boldsymbol{\Sigma}^u_{yy} - \bar{\boldsymbol{\Sigma}}^u_{yy})]}{\sqrt{\sum[((\boldsymbol{\Sigma}^u_{xx} - \bar{\boldsymbol{\Sigma}}^u_{xx}))^T((\boldsymbol{\Sigma}^u_{xx} - \bar{\boldsymbol{\Sigma}}^u_{xx}))]\sum[((\boldsymbol{\Sigma}^u_{yy} - \bar{\boldsymbol{\Sigma}}^u_{yy}))^T((\boldsymbol{\Sigma}^u_{yy} - \bar{\boldsymbol{\Sigma}}^u_{yy}))]}},$$

where $\bar{\boldsymbol{\Sigma}}^u$ is the mean of all the entries of the upper triangular distance matrix $\boldsymbol{\Sigma}$.

Similar to the R_v coefficient, there are many modifications to the Mantel coefficient from the extensive practical usage. In addition, many new proposed correlation measures use Mantel coefficient as the target for comparison.

4. Distance correlation coefficient

   Székely et al. (2007) proposed a distance covariance/correlation coefficient to measure the correlation between two random vectors. This is a non-linear measure. In this case, we can have $p_x \neq p_y$. As a non-linear correlation measure, it is 0 if, and only if, the two random vectors are independent. This valuable property has attracted further research from statistical point of view. For example, J. Berrendero (2013) studied functional data analysis and used this correlation to select effective data points. This correlation measure is calculated in the following way. First, we need to transfer the data matrices $\mathbf{X}$ and $\mathbf{Y}$ into some distance matrices $\boldsymbol{\Sigma}^X$ and $\boldsymbol{\Sigma}^Y$. Secondly, $\boldsymbol{\Sigma}^X$ is centred to $\boldsymbol{\Delta}^X$ by:

$$\Delta^X_{i,j} = \boldsymbol{\Sigma}^X_{i,j} - \bar{\boldsymbol{\Sigma}^X}_{i,\cdot} - \bar{\boldsymbol{\Sigma}^X}_{\cdot,j} + \bar{\boldsymbol{\Sigma}^X}_{\cdot,\cdot}.$$

   And $\boldsymbol{\Sigma}^Y$ is centred in the same way. Then the squared distance covariance is:

$$\mathrm{dCov}^2_n(\mathbf{X},\mathbf{Y}) = \frac{1}{n^2}\sum[\boldsymbol{\Delta}^X]^T[\boldsymbol{\Delta}^Y].$$

   The correlation is:

$$\rho_{\mathbf{X},\mathbf{Y}} = \frac{\mathrm{dCov}(\mathbf{X},\mathbf{Y})}{\sqrt{\mathrm{dCov}(\mathbf{X})\mathrm{dCov}(\mathbf{Y})}}.$$

A few articles review the different statistical properties and usages of the correlation coefficients. Josse and Holmes (2013) introduces the history and development of some com-

monly used correlation measures. Abdi (2007) reviews many distance functions, while Abdi (2010) reviews three correlation measures from the statistical test and distribution perspectives.

3.2 Canonical correlation analysis between random vectors

Canonical correlation analysis, or CCA measures the linear correlation between two groups of random variables or two random vectors. It is a natural extension of Pearson's correlation coefficient. We replace the random variables in Eqn (3.1) by the projection of the random vectors with some weights or coefficients. For each pair of given weights or coefficients $\tilde{a}_1$ and $\tilde{a}_2$, there exists one correlation measure between two random vectors $\mathbf{X}_1$ and $\mathbf{X}_2$:

$$\rho(\mathbf{X}_1, \mathbf{X}_2|\tilde{a}_1, \tilde{a}_1) = \frac{cov(\tilde{a}_1^T\mathbf{X}_1, \tilde{a}_2^T\mathbf{X}_2)}{\sqrt{Var(\tilde{a}_1^T\mathbf{X}_1)Var(\tilde{a}_2^T\mathbf{X}_2)}}$$

For each of the random vectors, there exist more than one possible projections $U_a = \tilde{a}^T\mathbf{X}$ when the dimension of $\mathbf{X}$ is larger than one, regardless the scale of $\tilde{a}$. Among all the possible projections U_a for $\mathbf{X}_1$ and U_b for $\mathbf{X}_2$, the first canonical correlation uses the pair that can give the maximum correlation. The first canonical correlation coefficient is obtained by:

$$\rho(\mathbf{X}_1, \mathbf{X}_2) = \max_{\tilde{a}_1, \tilde{a}_2} \frac{cov(\tilde{a}_1^T\mathbf{X}_1, \tilde{a}_2^T\mathbf{X}_2)}{\sqrt{Var(\tilde{a}_1^T\mathbf{X}_1)Var(\tilde{a}_2^T\mathbf{X}_2)}}, \tag{3.2}$$

under the constraint that $Var(\tilde{a}_1^T\mathbf{X}_1) = 1$ and $Var(\tilde{a}_2^T\mathbf{X}_2) = 1$ in order to scale the coefficients. This gives one correlation and the estimate of the first pair of weights or coefficients $\tilde{a}_1^{(1)}$ and $\tilde{a}_2^{(1)}$.

For the k-th canonical correlations when $k > 1$, we still want the projections $U_a^{(k)}$ and $U_b^{(k)}$ that can give maximum correlation, but we also want $U_a^{(k)} \perp U_a^{(k*)}$ and $U_b^{(k)} \perp U_b^{(k*)}$ for any $k* < k$, where $\perp$ means independent with. Thus the number of canonical correlations are equal to the minimum number of possible orthogonal projections from the two random vectors. Sample sizes of the two random vectors should satisfy $n > p_1 + p_2 + 1$, where p_1

and p_2 are the dimensions of $\mathbf{X}_1$ and $\mathbf{X}_2$ respectively.

The reason for choosing canonical correlation analysis in this case is that it gives correlation and coefficients simultaneously. The correlation can be used to represent the relationship between random vectors, and the coefficient vectors can be related to the regression problem in the later chapters.

In multivariate case, solving Eqn (3.2) is equivalent to solving the optimization problem:

$$G = \max_{\tilde{a}_1,\tilde{a}_2} \ \mathrm{cov}(\tilde{a}_1^T\mathbf{X}_1, \tilde{a}_2^T\mathbf{X}_2) - \frac{\eta_1}{2}[Var(\tilde{a}_1^T\mathbf{X}_1) - 1] - \frac{\eta_2}{2}[Var(\tilde{a}_2^T\mathbf{X}_2) - 1]. \tag{3.3}$$

If sample size is n, then we have

$$G = \max_{\tilde{a}_1,\tilde{a}_2} \ \tilde{a}_1^T\mathbf{X}_1^T\mathbf{X}_2\tilde{a}_2 - \frac{\eta_1}{2}[\tilde{a}_1^T\mathbf{X}_1^T\mathbf{X}_1\tilde{a}_1/(n-1) - 1] - \frac{\eta_2}{2}[\tilde{a}_2^T\mathbf{X}_2^T\mathbf{X}_2\tilde{a}_2/(n-1) - 1]. \tag{3.4}$$

Eqn (3.4) can be solved by using Lagrange multiplier:

$$\frac{\partial G}{\partial \tilde{a}_1} = \mathbf{X}_1^T\mathbf{X}_2\tilde{a}_2 - \eta_1\mathbf{X}_1^T\mathbf{X}_1\tilde{a}_1 = 0; \tag{3.5}$$

$$\frac{\partial G}{\partial \tilde{a}_2} = \tilde{a}_1\mathbf{X}_1^T\mathbf{X}_2 - \eta_2\tilde{a}_2^T\mathbf{X}_2^T\mathbf{X}_2 = 0. \tag{3.6}$$

It can be proved that $\eta_1 = \eta_2$. From Eqn (3.5), we can get:

$$\begin{aligned}(\mathbf{X}_1^T\mathbf{X}_1)^{-1}\mathbf{X}_1^T\mathbf{X}_2\tilde{a}_2 &= \rho_1\tilde{a}_1\\ \tilde{a}_1^T(\mathbf{X}_1^T\mathbf{X}_1)(\mathbf{X}_1^T\mathbf{X}_1)^{-1}\mathbf{X}_1^T\mathbf{X}_2\tilde{a}_2 &= \tilde{a}_1^T(\mathbf{X}_1^T\mathbf{X}_1)\rho_1\tilde{a}_1\\ \tilde{a}_1^T\mathbf{X}_1^T\mathbf{X}_2\tilde{a}_2 &= \eta_1.\end{aligned}$$

Similarly from Eqn (3.6), we cat get:

$$\tilde{a}_2^T\mathbf{X}_2^T\mathbf{X}_1\tilde{a}_1 = \eta_2.$$

Thus $\eta_1 = \eta_2$. Set both of η_1 and η_2 to be η. Now we solve the linear system

$$\begin{aligned}\mathbf{X}_1^T\mathbf{X}_2\tilde{a}_2 &= \eta\mathbf{X}_1^T\mathbf{X}_1\tilde{a}_1\\ \mathbf{X}_2^T\mathbf{X}_1\tilde{a}_1 &= \eta\mathbf{X}_2^T\mathbf{X}_2\tilde{a}_2.\end{aligned} \tag{3.7}$$

It is equivalent to solving the generalized eigen decomposition:

$$\begin{pmatrix} 0 & \mathbf{X}_1^T\mathbf{X}_2 \\ \mathbf{X}_2^T\mathbf{X}_1 & 0 \end{pmatrix}\begin{pmatrix} \tilde{a}_1 \\ \tilde{a}_2 \end{pmatrix} = \eta\begin{pmatrix} \mathbf{X}_1^T\mathbf{X}_1 & 0 \\ 0 & \mathbf{X}_2^T\mathbf{X}_2 \end{pmatrix}\begin{pmatrix} \tilde{a}_1 \\ \tilde{a}_2 \end{pmatrix},$$

where η is the eigenvalue and the correlation. Thus $\rho = \eta$. The correlation measure ρ can have more than one values, depending on the dimensions of the random vectors. Alternatively, solving this linear system is equivalent to solving the eigen-decomposition of matrices:

$$\begin{aligned} \text{for } \mathbf{X}_1 \quad & M_1 = (\mathbf{X}_1^T\mathbf{X}_1)^{-1}\mathbf{X}_1^T\mathbf{X}_2(\mathbf{X}_2^T\mathbf{X}_2)^{-1}\mathbf{X}_2^T\mathbf{X}_1, \\ \text{for } \mathbf{X}_2 \quad & M_2 = (\mathbf{X}_2^T\mathbf{X}_2)^{-1}\mathbf{X}_2^T\mathbf{X}_1(\mathbf{X}_1^T\mathbf{X}_1)^{-1}\mathbf{X}_1^T\mathbf{X}_2. \end{aligned}$$

These two matrices can have more a general form. Let $V_1 = Var(\mathbf{X}_1)$, $V_2 = Var(\mathbf{X}_2)$, $V_{1,2} = V_{2,1}^T = Cov(\mathbf{X}_1, \mathbf{X}_2)$, then

$$\text{for } \mathbf{X}_1 \quad M_1 = V_1^{-1}V_{1,2}V_2^{-1}V_{2,1} \tag{3.8}$$

$$\text{for } \mathbf{X}_2 \quad M_2 = V_2^{-1}V_{2,1}V_1^{-1}V_{1,2}. \tag{3.9}$$

Note that MoorePenrose pseudoinverse might be required to solve Eqn (3.8) and Eqn (3.9) to overcome some numerical problems. This is a type of generalized inverse. It is unique for all matrices if the outcome satisfies necessary conditions (Penrose (1955)). Other types of generalized inverse may also be applied here. Vinod (1976) added a ridge penalty into the analysis, which can ease this issue to some extent.

The only correlations of interest are the non-zero ones, or the ones that are fairly distinct from 0. Numerical computation may give results that are non-zero but also very small. Thus practically, a threshold should be set to decide which correlations are too large to be non-zero.

3.3 Canonical correlation analysis between random functional variables

Projections of random vectors are used in the canonical correlation analysis. Similarly, projections of random functional variables can also be used in CCA. The projection of a

functional variable $x(t)$ can be obtained by

$$\int x(t)\beta(t)dt.$$

Without any further modifications, Eqn (3.2) in the functional case can be written as:

$$\rho(x_1(t), x_2(t)) = \max_{\beta_1(t),\beta_2(t)} \frac{cov\left(\int x_1(t)\beta_1(t)dt, \int x_2(t)\beta_2(t)dt\right)}{\sqrt{Var\left(\int x_1(t)\beta_1(t)dt\right) Var\left(\int x_2(t)\beta_2(t)dt\right)}}.$$

However, such naive implementation of Eqn (3.2) disregards any functional aspect of the data. It has been proved that the functional canonical correlation analysis cannot be carried out using Eqn (3.2) by S. E. Leurgans and Silverman (1993) and Ramsay and Silverman (2005). In the later reference, the authors commented that the outcome is 'meaningless'. This is also confirmed in our study. If apply Eqn (3.2) with the generalized inverse, the estimated coefficients would over fit the variations between the two functional variables, and give very large correlations.

The definition of canonical correlation analysis has no requirement of the within group dependence. However, the dependence within functional variables is one of the most important features in functional data analysis. This is because that the data and the coefficient must be smooth functions. From the literature, there are basically two ways to estimate the functional canonical correlation and the component coefficients of the functional variables. One way is to add a roughness penalty in the constraint, for example S. E. Leurgans and Silverman (1993) used a discrete data points with constraints for smoothness on 'curve data', while Ramsay and Silverman (2005) applied a roughness penalty to the basis function expression of the functional variables and coefficients. The other way is a two stage algorithm. The first stage is to use dimension reduction method, such as functional principle component analysis to summarize the data. The second stage is to apply original canonical correlation to the summaries of the data. Guozhong He and Wang (2003) discusses a few different versions of this algorithm. He et al. (2010) combines functional principle component analysis and canonical correlation analysis in functional linear regression with functional response and predictors.

Adding roughness constraints to the functional coefficients is more closely related to the functional regression problem than the other method, since we estimate the coefficient directly with respect to the functional coefficients. By using this method, the information of the functional variables is not changed or discarded before the analysis. Thus we consider

the penalized method here. The alternative way of smoothing with a roughness penalty is to use suitable number of basis functions with suitable order. We use the roughness penalty here in order to reduce the computation cost.

Suppose there are the two functional random variables denoted as $x_1(t)$ and $x_2(t)$, with the corresponding coefficients $\beta_1(t)$ and $\beta_2(t)$ obtained from functional canonical correlation analysis. As shown in the Eqn (3.2), the variances of the projected random vectors are constrained to be 1. This is actually the constraint for the coefficients. Now change the constraint to :

$$Var\left(\int x_i(t)\beta_i(t)dt\right) + \lambda \int [\beta_i^{''}(t)]^2 dt = 1 \quad i = 1, 2,$$

where $\beta^{''}(t)$ is the second order derivative of $\beta(t)$. Therefore the canonical correlation analysis between functional variables is:

$$\rho(x_1(t), x_2(t)) =$$
$$\max_{\beta_1(t),\beta_2(t)} \frac{cov\left(\int x_1(t)\beta_1(t)dt, \int x_2(t)\beta_2(t)dt\right)}{\sqrt{[Var\left(\int x_1(t)\beta_1(t)dt\right) + \lambda_1 \int [\beta_1^{''}(t)]^2 dt][Var\left(\int x_2(t)\beta_2(t)dt\right) + \lambda_2 \int [\beta_2^{''}(t)]^2 dt]}}. \tag{3.10}$$

This is equivalent to

$$\max_{\beta_1(t),\beta_2(t)} \quad cov\left(\int x_1(t)\beta_1(t)dt, \int x_2(t)\beta_2(t)dt\right)$$
$$\text{s.t. } Var\left(\int x_1(t)\beta_1(t)dt\right) + \lambda_1[\beta_1^{''}(t)]^2 = 1, \quad Var\left(\int x_2(t)\beta_2(t)dt\right) + \lambda_2[\beta_2^{''}(t)]^2 = 1.$$

We need to reduce the dimensions of the functional variables before solving Eqn (3.10). S. E. Leurgans and Silverman (1993) used the discrete data points expression, whereas Ramsay and Silverman (2005) used basis functions expression. The details of these methods for performing the canonical correlation analysis will be discussed in the next section.

Eqn (3.10) involves two tuning parameters λ_1 and λ_2. How to choose these values is an important issue. S. E. Leurgans and Silverman (1993), Guozhong He and Wang (2003) and Ramsay and Silverman (2005) mentioned cross validation method in this case. In their articles, the tuning parameters $\lambda_1 = \lambda_2 = \lambda$ from Eqn (3.10). This approximation can simplify the problem and make the computation faster. The cross validation is stated

as follows:

Suppose $(\beta_1^{-i}(t), \beta_2^{-i}(t))_k$ is the k-th pair of coefficients corresponding to the k-th largest canonical correlation between $x_1^{-i}(t)$ and $x_2^{-i}(t)$, where $-i$ means without the i-th sample. Thus for the i-th sample from $x_1(t)$ and $x_2(t)$, there exist projections $\int x_1^i(t)\beta_1^{-i}(t)dt$ and $\int x_2^i(t)\beta_2^{-i}(t)dt$, respectively. For each choice of λ, the correlation between these projections can be expressed as:

$$CV_{\lambda,k} = \mathrm{Cor}\left(\left\{\int x_1^i(t)\beta_1^{-i}(t)dt\right\}_{i=1}^{n}, \left\{\int x_1^i(t)\beta_1^{-i}(t)dt\right\}_{i=1}^{n}\right)_k, \tag{3.11}$$

where $\mathrm{Cor}(.,.)_k$ means the k-th correlation. Depending on the purpose of the analysis, one may want to find the λ such that it can maximise the first canonical correlation as in S. E. Leurgans and Silverman (1993) and Ramsay and Silverman (2005); or one can select a λ that can minimise the k-th canonical correlation as in He et al. (2000). Computationally, this cross validation is very expensive.

3.4 Canonical correlation analysis between a scalar variable and a functional variable

If we replace one of the functional variables by a scalar variable, the canonical correlation can still be calculated. If we denote the scalar variable as y, we can write:

$$\rho(x(t), y) = \max_{\beta(t),\alpha} \frac{\mathrm{Cov}\left(\int x(t)\beta(t)dt, \alpha y\right)}{\sqrt{[\mathrm{Var}\left(\int x(t)\beta(t)dt\right) + \lambda \int [\beta''(t)]^2 dt][\mathrm{Var}(\alpha y)]}}, \tag{3.12}$$

which is equivalent to

$$\max_{\beta(t),\alpha} \quad \mathrm{Cov}\left(\int x(t)\beta(t)dt, \alpha y\right) \text{s.t.} \quad \mathrm{Var}\left(\int x(t)\beta(t)dt\right) + \lambda[\beta''(t)]^2 = 1, \quad \mathrm{Var}(\alpha y) = 1$$

In either CCA between two functional variables or CCA between one functional and one scalar variable, the integration is not the most difficult part. The difficulty lies in defining the second order derivative of the coefficients.

We consider three representative methods of the functional objects here and give the corresponding solutions of the canonical correlation analysis between one functional variable

and one scalar variable. The first one is simply using the representative data points to replace the functional coefficient, and use matrix multiplication to represent the integration. The second one is to introduce numerical integration to our problem. More specifically, one type of Gaussian quadrature, Gauss-Legendre quadrature is used. The third one is to apply basis functions to the functional objects, and reduce the dimension to the number of basis functions. The details of them will be discussed in the following subsections.

3.4.1 Representative data points expression for functional variable and functional coefficients

Suppose the functional variable $x(t)$ has k representative data points. The functional variable can be expressed as:

$$x(t) = \Big(x(t_1), x(t_2), \ldots, x(t_k)\Big) = \mathbf{X}_{n\times k}.$$

As a start, we assume k is large. It is likely to have k larger than the sample size n. The coefficient $\beta(t)$ should have the same number of discrete points as the functional variable. For consistency, the coefficient $\beta(t)$ is approximated by a $1 \times k$ vector $\tilde{\beta}$. We refer to this discrete method as representative data points expression or RDP method.

We now focus on calculation of the second order derivative. Tibshirani et al. (2005) used finite difference to obtain the fusion in fused lasso, which is first order derivative. The same idea can be used in the calculation of second or higher order derivative. We can have:

$$\tilde{\beta}''^{T} \approx L\tilde{\beta}^{T},$$

where L is defined as:

$$L = \begin{pmatrix} 1 & -2 & 1 & 0 & 0 & 0 & \ldots \\ 0 & 1 & -2 & 1 & 0 & 0 & \ldots \\ 0 & 0 & 1 & -2 & 1 & 0 & \ldots \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & \ldots \\ 0 & 0 & 0 & 0 & \ldots & \ldots & \ldots \end{pmatrix} * \frac{1}{\delta t},$$

where δt as the difference between time of the corresponding consecutive data points. This L is from the centred differences formula. The value of δt is arbitrary, thus we set it as 1 for convenience. However, this is only for equally spaced data points. We need to

consider more general case when $\delta t_i \neq \delta t_j$ for $i \neq j$. Suppose $f(t)$ is the original function, which has observations on $t = t_j$, where $j = 1, \ldots, J$. By Taylor expansions,

$$f(t_{j+1}) = f(t_j) + (t_{j+1} - t_j)f'(t_j) + \frac{(t_{j+1} - t_j)^2}{2!}f''(t_j)\ldots$$
$$f(t_{j-1}) = f(t_j) - (t_j - t_{j-1})f'(t_j) + \frac{(t_j - t_{j-1})^2}{2!}f''(t_j)\ldots.$$

Thus the centred differences formula for second order derivative is:

$$f''(t_j) = \frac{f(t_{j+1})(t_j - t_{j-1}) - f(t_j)(t_{j+1} - t_{j-1}) + f(t_{j-1})(t_{j+1} - t_j)}{(t_{j+1} - t_j)(t_j - t_{j-1})\frac{t_{j+1}-t_{j-1}}{2}}.$$

This gives the weight for the entries of the matrix L:

$$L = \begin{pmatrix} \frac{2}{(t_3-t_2)(t_3-t_1)} & -\frac{2}{(t_3-t_2)(t_2-t_1)} & \frac{2}{(t_2-t_1)(t_3-t_1)} & 0 & 0 & 0 & \ldots \\ 0 & \frac{2}{(t_4-t_3)(t_4-t_2)} & -\frac{2}{(t_4-t_3)(t_3-t_2)} & \frac{2}{(t_3-t_2)(t_4-t_2)} & 0 & 0 & \ldots \\ 0 & 0 & \frac{2}{(t_5-t_4)(t_5-t_3)} & -\frac{2}{(t_5-t_4)(t_4-t_3)} & \frac{2}{(t_4-t_3)(t_5-t_3)} & 0 & \ldots \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & \ldots \\ 0 & 0 & 0 & 0 & \ldots & \ldots & \ldots \end{pmatrix}.$$

Such an approximation will create a big bias if the time points are not dense enough to cover the rapid change. Therefore, this method is only useful for the case when records of the data are made densely, such as the data we get from the well known "gait" data in functional data analysis (Ramsay and Silverman (2005)), where 20 records are made per gait cycle.

The integration $\int x(t)\beta(t)dt$ can be approximated by:

$$\int x(t)\beta(t)dt \approx \frac{\mathbf{X}\tilde{\beta}^T}{k}.$$

Now Eqn (3.10) becomes:

$$\rho(x(t), y) \approx \rho(\mathbf{X}, y) \max_{\tilde{\beta},\alpha} \frac{cov(\mathbf{X}\tilde{\beta}^T/k, \alpha y)}{\sqrt{[Var(\mathbf{X}\tilde{\beta}^T/k) + \lambda(L\tilde{\beta})^2][Var(\alpha y)]}},$$

subject to $Var(\mathbf{X}\tilde{\beta}^T/k) + \lambda[L\tilde{\beta}]^2 = 1$ and $Var(\alpha y) = 1$, where $1/k$ in the equation can be written as $\frac{1}{k}I$ and I is the identity matrix with dimension k. Let us denote $K_I = \frac{1}{k}I$. Thus

we have:

$$\begin{aligned}\rho(\mathbf{X}, y) &= \max_{\tilde{\beta},\alpha} \frac{\tilde{\beta} K_I \mathbf{X}^T y\alpha/(n-1)}{(\sqrt{[\tilde{\beta} K_I \mathbf{X}^T \mathbf{X} K_I \tilde{\beta}^T + \lambda \tilde{\beta} L^T K_I L \tilde{\beta}^T][Var(\alpha y)]})/(n-1)} \\ &= \max_{\tilde{\beta},\alpha} \frac{\tilde{\beta}^T K_I \mathbf{X}^T y\alpha}{\sqrt{[\tilde{\beta}(K_I \mathbf{X}^T \mathbf{X} K_I + \lambda L^T K_I L)\tilde{\beta}^T][\alpha^2 y^T y]}}. \end{aligned} \tag{3.13}$$

Eqn (3.8) and Eqn (3.9) can be applied, with $V_1 = K_I \mathbf{X}^T \mathbf{X} K_I + \lambda L^T K_I L$, $V_2 = Var(y)$, $V_{1,2} = K_I \mathbf{X}^T y$.

3.4.2 Gaussian quadrature expression

Gaussian quadrature has been studied for a long time in the literature. It is designed for numerical integrations. Many different versions of Gaussian quadrature have been developed. Gaussian quadrature can produce accurate results if the function can be well approximated by a set of polynomial functions within a certain range. Different Gaussian quadratures have different choices of polynomial functions and different ranges for integration. The corresponding weights and abscissae would also be different. It can be shown that the abscissae of Gaussian quadrature are the roots of certain types of orthogonal polynomials. The basic formula of Gaussian quadrature is:

$$\int_{-1}^{1} f(t)dt \approx \sum_{i=1}^{n} w_i f(t_i),$$

where the upper and lower bound $[-1, 1]$ is for some specific Gaussian Quadrature, such as Gauss-Legendre quadrature or Chebyshev-Gauss quadrature. For commonly used Gaussian quadratures, the weights w_i and the abscissae t_i are all calculated in advance.

In our case, functional variables need to be projected to a one dimensional random variable, which is also done by such integration. The upper and lower bound can be easily

changed. The integration becomes:

$$\begin{aligned}\int_{-1}^{1} x(t)\beta(t)dt &= \int_{-1}^{1} x^*(t)dt \\ &\approx \sum_{i=1}^{n} w_i x^*(t_i) \\ &= \sum_{i=1}^{n} w_i x(t_i)\beta(t_i). \end{aligned} \tag{3.14}$$

In matrix form, the projected random variable is:

$$\int_{-1}^{1} x(t)\beta(t)dt = \mathbf{X}W\tilde{\beta},$$

where W is a diagonal matrix with weights w_i's at the points closest to the abscissas, and zero everywhere else. Here the Gauss-Legendre quadrature is applied. A polynomial of order $(2n-1)$, where n is the number of points, is used to approximate the target function.

The roughness penalty is a little different from the previous two discrete methods. It must be seen as the integration of the the squared second order derivative:

$$\begin{aligned} Pen &= \int [\beta''(t)]^2 dt = \int \beta''^*(t)dt \\ &\approx \sum_{i=1}^{n} w_i \beta''^*(t_i) = \sum_{i=1}^{n} w_i \beta''(t_i)^2 \\ &= \tilde{\beta}'' W \tilde{\beta}''^T, \end{aligned}$$

where the weights and abscissas are the same as those used in the Eqn (3.14). Similar to RDP method, the second order derivative of the functional coefficient $\tilde{\beta}''$ is approximated by multiplying a band matrix L to the original functional coefficient $\tilde{\beta}$. In this case L must be

$$L = \begin{pmatrix} \frac{2}{(t_3-t_2)(t_3-t_1)} & -\frac{2}{(t_3-t_2)(t_2-t_1)} & \frac{2}{(t_2-t_1)(t_3-t_1)} & 0 & 0 & 0 & \dots \\ 0 & \frac{2}{(t_4-t_3)(t_4-t_2)} & -\frac{2}{(t_4-t_3)(t_3-t_2)} & \frac{2}{(t_3-t_2)(t_4-t_2)} & 0 & 0 & \dots \\ 0 & 0 & \frac{2}{(t_5-t_4)(t_5-t_3)} & -\frac{2}{(t_5-t_4)(t_4-t_3)} & \frac{2}{(t_4-t_3)(t_5-t_3)} & 0 & \dots \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & \dots \\ 0 & 0 & 0 & 0 & \dots & \dots & \dots \end{pmatrix},$$

since the abscissas of Gaussian quadrature are not equally spaced. As the gap between abscissas might be large at some part of the curves, this approximation may introduce some bias into the calculation. Therefore, a reasonable number of abscissas is required to reduce the bias.

Note that we can only obtain the estimation of the coefficients $\tilde{\beta}$ at the abscissas. All other points of $\tilde{\beta}$ would be 0. For $\mathbf{X}$, $\tilde{\beta}$, W and L, we use the subscription $_{GQ}$ to denote their values at selected abscissas.

Thus Eqn (3.10) becomes:

$$\begin{aligned}
\rho(x(t), y) &= \max_{\tilde{\beta},\alpha} \frac{cov(\mathbf{X}W\tilde{\beta}^T, \alpha y)}{\sqrt{[Var(\tilde{\beta}W^T\mathbf{X}^T\mathbf{X}W\tilde{\beta}^T) + \lambda\tilde{\beta}\mathbf{L}^T W\mathbf{L}\tilde{\beta}^T][Var(\alpha y)]}} \\
&= \max_{\tilde{\beta}_{GQ},\alpha} \frac{\tilde{\beta}_{GQ}W_{GQ}^T\mathbf{X}_{GQ}^T y\alpha}{\sqrt{[\tilde{\beta}_{GQ}W_{GQ}^T\mathbf{X}_{GQ}^T\mathbf{X}_{GQ}W_{GQ}\tilde{\beta}_{GQ}^T + \lambda\tilde{\beta}_{GQ}\mathbf{L}_{GQ}^T W_{GQ}\mathbf{L}_{GQ}\tilde{\beta}_{GQ}^T][\alpha^2 Var(y)]}} \\
&= \max_{\tilde{\beta}_{GQ},\alpha} \frac{\tilde{\beta}_{GQ}W_{GQ}^T\mathbf{X}_{GQ}^T y\alpha}{\sqrt{[\tilde{\beta}_{GQ}(W_{GQ}^T\mathbf{X}_{GQ}^T\mathbf{X}_{GQ}W_{GQ} + \lambda\mathbf{L}_{GQ}^T W_{GQ}\mathbf{L})\tilde{\beta}_{GQ}^T][\alpha^2 Var(y)]}}.
\end{aligned} \tag{3.15}$$

Similarly Eqn (3.15) can be solved by the same method as Eqn (3.8) and Eqn (3.9). In this case, $V_1 = W_{GQ}^T\mathbf{X}_{GQ}^T\mathbf{X}_{GQ}W_{GQ} + \lambda\mathbf{L}_{GQ}^T W_{GQ}\mathbf{L}$, $V_2 = Var(y)$, and $V_{1,2} = W_{GQ}^T\mathbf{X}_{GQ}^T y$. We refer this discrete method as the GQ method.

3.4.3 Basis function expression for functional coefficients

Suppose $\Phi_k(t)$ are known basis functions. Also, assume that the basis functions are second order differentiable. Thus

$$x(t) = \sum_{k=1}^{\infty} C_k\Phi_k(t) \approx \sum_{k=1}^{K} C_k\Phi_k(t) \quad \beta(t) = \sum_{k=1}^{\infty} \tilde{C}_{\beta,k}\Phi_k(t) \approx \sum_{k=1}^{K} \tilde{C}_{\beta,k}\Phi_k(t),$$

where C_k and $\tilde{C}_{\beta,k}$ are coefficients for the basis functions used for the functional variables and functional coefficients, respectively. We use spline basis in our study, as we have no evidence to support the periodic shape of the curves.

In the actual calculation, the spline basis functions are represented by matrices. The above equations means K basis functions are used. If we reduce the dimension of basis functions to p equally spaced time points, the basis function $\Phi_k(t)$ can be expressed by

a $K \times p$ matrix $\boldsymbol{\Phi}$; while all the coefficient matrices for functional variables are $n \times K$, and coefficient matrices for functional coefficients are $1 \times K$. This would be helpful in the derivation of the solution of correlation analysis later.

We show an example here to explain the process. For the basis function $\phi_k(t)$, we can calculate the values at a series of data points $t_1, t_2, \ldots, t_p$, so that this basis function is expressed as a vector with length p. We have K basis functions in total, so that the set of basis functions is represented by a K by p matrix. For convenience, we can set the the time points to be equally spaced one the interval $[0, 1]$. We can also assume $t_1 = 0$ and $t_p = 1$.

By using the discrete values of the basis functions, the original functional variables can be written as:

$$x(t) \approx \mathbf{X} = \mathbf{C}\boldsymbol{\Phi} \quad \beta(t) \approx \boldsymbol{\beta} = \tilde{C}_\beta \boldsymbol{\Phi}.$$

We refer to this discrete method as the BF method.

Also we denote the the second order derivative of the basis functions $\Phi(t)$ as $\Phi^{(2)} = \Phi''(t)$, so that the second order derivative of $\beta''(t)$ can be written as $\sum_{k=1}^{K} \tilde{C}_{\beta,k} \Phi_k^{(2)}(t)$. Similar to $\boldsymbol{\Phi}$, we denote $\mathbf{\Phi^{(2)}}$ as the discrete values of $\Phi^{(2)}(t)$.

Thus, the integration becomes:

$$\int x(t)\beta(t)dt \approx \mathbf{C}\boldsymbol{\Phi}\boldsymbol{\Phi}^T \tilde{C}_\beta^T / k$$
$$\int \beta''(t)\beta''(t)dt \approx \tilde{C}_\beta \mathbf{\Phi^{(2)}} \mathbf{\Phi^{(2)}}^T \tilde{C}_\beta^T / k.$$

Thus Eqn (3.10) becomes:

$$\begin{aligned}
\rho(x(t), y) &= \max_{\tilde{C}_\beta, \alpha} \frac{cov(\mathbf{C}\boldsymbol{\Phi}\boldsymbol{\Phi}^T \tilde{C}_\beta^T / k, \alpha y)}{\sqrt{[Var(\mathbf{C}\boldsymbol{\Phi}\boldsymbol{\Phi}^T \tilde{C}_\beta^T / k) + \lambda \tilde{C}_\beta \mathbf{L}^T \mathbf{L} \tilde{C}_\beta^T / k][Var(\alpha y)]}} \\
&= \max_{\tilde{C}_\beta, \alpha} \frac{\tilde{C}_\beta \boldsymbol{\Phi}\boldsymbol{\Phi}^T \mathbf{C}^T y\alpha / (k(n-1))}{\sqrt{[\tilde{C}_\beta \boldsymbol{\Phi}\boldsymbol{\Phi}^T \mathbf{C}^T \mathbf{C}\boldsymbol{\Phi}\boldsymbol{\Phi}^T \tilde{C}_\beta^T / k + \lambda \tilde{C}_\beta \mathbf{L}^T \mathbf{L} \tilde{C}_\beta^T / k][\alpha^2 Var(y)]} / (n-1)} \\
&= \max_{\tilde{C}_\beta, \alpha} \frac{\tilde{C}_\beta \boldsymbol{\Phi}\boldsymbol{\Phi}^T \mathbf{C}^T y\alpha / k}{\sqrt{[\tilde{C}_\beta (\boldsymbol{\Phi}\boldsymbol{\Phi}^T \mathbf{C}^T \mathbf{C}\boldsymbol{\Phi}\boldsymbol{\Phi}^T / k + \lambda \mathbf{L}^T \mathbf{L} / k) \tilde{C}_\beta^T / k][\alpha^2 y^T y]}}.
\end{aligned} \tag{3.16}$$

Eqn (3.16) can be solved by the same method as Eqn (3.8) and Eqn (3.9). More specifically, the eigen-decomposition of matrix M_1 and M_2 gives the estimation of the unknown coefficients $\tilde{C}_\beta$ and α respectively. M_1 and M_2 are:

$$\begin{aligned} M_1 &= V_1^{-1} V_{1,2} V_2^{-1} V_{2,1} \\ M_2 &= V_2^{-1} V_{2,1} V_1^{-1} V_{1,2}. \end{aligned}$$

In this case, $V_1 = \boldsymbol{\Phi}\boldsymbol{\Phi}^T \mathbf{C}^T \mathbf{C} \boldsymbol{\Phi}\boldsymbol{\Phi}^T/k + \lambda \mathbf{L}^T \mathbf{L}/k$, $V_2 = y^T y$, and $V_{1,2} = \boldsymbol{\Phi}\boldsymbol{\Phi}^T \mathbf{C}^T y/k$.

3.4.4 General expression and solution

From the above expressions, it is clear that the functional variable $x(t)$ can be replaced by a matrix $\mathbf{X}$, which contains discrete data points. In general, by combining these expressions above, the integration of $x(t)\beta(t)$ and the penalty function $[\beta''(t)]^2$ can be written as:

$$\int x(t)\beta(t)dt = \mathbf{X} W \tilde{C}_\beta^T \tag{3.17}$$

$$\int [\beta''(t)]^2 dt = \tilde{C}_\beta W_2 \tilde{C}_\beta^T, \tag{3.18}$$

where W and W_2 are weight matrices and $\tilde{C}_\beta$ is the unknown coefficient to estimate. If we use RDP method for the functional coefficient, $W = K_I$, which is the diagonal matrix with $1/k$ on the diagonal and $W_2 = L^T K_I L$; $\tilde{C}_\beta$ is the discrete vector of the functional coefficient. If we use the BF method , $W = \boldsymbol{\Phi}/k$; $W_2 = \Phi''\Phi''^T/k$; $\tilde{C}_\beta$ is the coefficient of basis functions for the functional coefficient. If we use Gaussian quadrature, W is the diagonal matrix with weights at the point closest to the abscissae and zero everywhere else. Also $W_2 = L^T W L$ and $\tilde{C}_\beta$ is the functional coefficient, but only the values at or near abscissae can be calculated. The values are $\tilde{C}_{\beta_{GQ}}$. In Eqn (3.17) and 3.18, the only unknown is $\tilde{C}_\beta$. Therefore the data matrix can be seen as $\mathbf{X}W$.

Thus Eqn (3.10) becomes:

$$\begin{aligned} \rho(x(t), y) &= \max_{\tilde{C}_\beta, \alpha} \frac{cov(\mathbf{X}W\tilde{C}_\beta^T, \alpha y)}{\sqrt{[Var(\mathbf{X}W\tilde{C}_\beta^T) + \lambda \tilde{C}_\beta W_2 \tilde{C}_\beta^T][Var(yb)]}} \\ &= \max_{\tilde{C}_\beta, \alpha} \frac{\tilde{C}_\beta W^T \mathbf{X}^T y \alpha}{\sqrt{[\tilde{C}_\beta (W^T \mathbf{X}^T \mathbf{X} W + \lambda W_2) \tilde{C}_\beta^T][\alpha^2 y^T y]}}. \end{aligned} \tag{3.19}$$

Consider the conditions mentioned before in Eqn (3.12), this maximization problem can be rewritten as:

$$G = \tilde{C}_\beta W^T \mathbf{X}^T y\alpha - \frac{1}{2}\rho\tilde{C}_\beta(W^T\mathbf{X}^T\mathbf{X}W + \lambda W_2)\tilde{C}_\beta^T - \frac{1}{2}\rho\alpha^2 y^T y,$$

where ρ is the same as Eqn 3.4. By using the Lagrange multiplier:

$$\frac{\partial G}{\partial \tilde{C}_\beta} = W^T\mathbf{X}^T y\alpha - \rho(W^T\mathbf{X}^T\mathbf{X}W + \lambda W_2)\tilde{C}_\beta^T = 0$$
$$\frac{\partial G}{\partial \alpha} = \tilde{C}_\beta W^T\mathbf{X}^T y - \rho\alpha y^T y = 0.$$

From the constraint $\mathrm{Var}(\alpha y) = 1$, α can be obtained straight-away: $\alpha = 1/\mathrm{SD}(y)$. Thus α is known. The unknown parameters to be estimated are the correlation ρ, the coefficient for functional variables $\tilde{C}_\beta$ and the tuning parameter λ. However, the tuning parameter λ can be assumed as known at the moment. More details are discussed in later sections.

The solutions can be found by solving the following equations:

$$\text{correlation:}\quad \rho^2 = \frac{y^T\mathbf{X}W(W^T\mathbf{X}^T\mathbf{X}W + \lambda W_2)^{-1}W^T\mathbf{X}^T y}{y^T y} \tag{3.20}$$

$$\text{coefficients:}\quad \tilde{C}_\beta = \frac{(W^T\mathbf{X}^T\mathbf{X}W + \lambda W_2)^{-1}W^T\mathbf{X}^T y}{\rho\sqrt{y^T y}} \tag{3.21}$$

The denominator in Eqn (3.21) contains the correlation itself, but from Eqn (3.20), only the squared correlation can be obtained. This means that the sign of $\tilde{C}_\beta$ cannot be found. However, it is difficult to say a functional variable is positively or negatively correlated to a scalar variable. Based on the canonical correlation analysis, we only get the correlation between the projected functional variable and the scalar variable. In this case, the sign is not fixed. Therefore this sign problem has little effect on our case, and we can always assume that the correlation between a functional variable and a scalar variable is positive for convenience.

More generally, the solutions can be written as

$$\text{correlation:}\quad \rho^2 = \frac{V_{X,y}^T P_{X,X}^{-1} V_{X,y}}{V_y} \tag{3.22}$$

$$\text{coefficients:}\quad \tilde{C}_\beta = \frac{P_{X,X}^{-1} V_{X,y}}{\rho\|y\|_2} \tag{3.23}$$

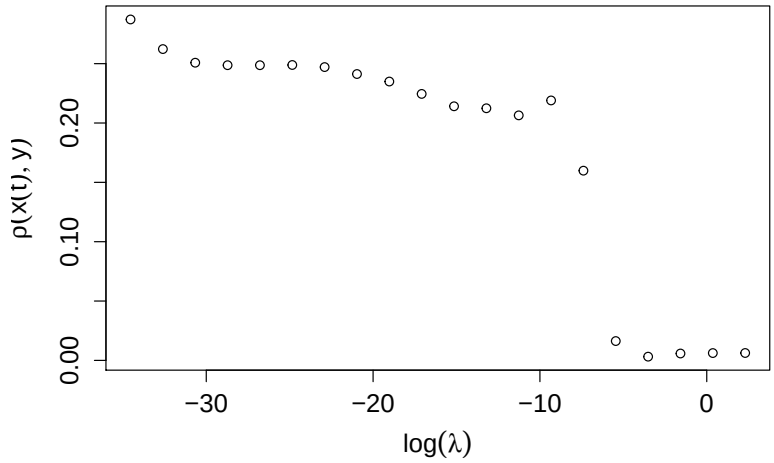


Figure 3.1: The CCA correlation against the log of tuning parameter.

where $P_{X,X}$ is the variance of the functional variable with penalty functions.

3.4.5 Choose the tuning parameters

The value of tuning parameter greatly affects the outcome of the functional canonical correlation. Roughly speaking, the correlation reduces as the value of the tuning parameter increases. The tuning parameter must be non-negative. As we state before, the result is meaningless when no constraint is taken for the smoothness of the coefficient for CCA involving functional variables. On the other hand, when the tuning parameter is too large, the correlation would reduce to almost zero. Figure 3.1 shows the change of the correlation between one functional variable and one scalar variable with the tuning parameters.

Generalized cross validation (GCV)

We can always use leave one out cross validation to find a good tuning parameter. However, the cost of "leave one out" cross validation is extremely high, and makes this method impractical. Golub et al. (1979) proposed generalized cross validation, which is an approximation of the "leave one out" cross validation. The formula is:

$$GCV(\lambda) = \frac{\frac{1}{n}\|[I - H(\lambda)]y\|^2}{[\frac{1}{n}tr(I - H(\lambda)]^2} \tag{3.24}$$

where $H(\lambda)$ is a function of the tuning parameter λ. In a regression problem, the fitted value $\hat{y}$ is obtained by $\hat{y} = Hy$. In this case,

$$H(\lambda) = \mathbf{X}W(W^T\mathbf{X}^T\mathbf{X}W + \lambda W_2)^{-1}W^T\mathbf{X}^T.$$

Since there would be only one canonical correlation, GCV here is close to the cross validation method in S. E. Leurgans and Silverman (1993) for CCA between two functional variables. That is to say, Eqn (3.11) is maximized with $k = 1$.

Apparently, matrix inversion is the most computationally expensive part of GCV calculation. Therefore calculating the inverse matrix for each candidate tuning parameter is impractical. It turns out GCV can be calculated efficiently by the following method.

Suppose we want to have the inverse of matrix M which has the form $M = A + \lambda B$, where A is a positive definite matrix. We solve the generalized eigen decomposition:

$$Bx = \alpha Ax,$$

where x is the eigenvector, and α is the corresponding eigenvalue. If we write it in matrix form:

$$BV = AVD,$$

where columns of the matrix V are the eigenvectors ; D is a diagonal matrix with diagonal to be the eigenvalues. One important feature of generalized eigen decomposition is that the matrix V is 'A-orthogonal', which means $V^TAV = I$. Also V is invertible. Thus,

$$V^TAV = I \quad \text{and} \quad V^TBV = V^TAVD = D$$
$$V^TMV = V^T(A + \lambda D)V = I + \lambda D.$$

And so:

$$M = (V^T)^{-1}(I + \lambda D)V^{-1}$$
$$M^{-1} = [(V^T)^{-1}(I + \lambda D)V^{-1}]^{-1} = V(I + \lambda D)^{-1}V^T. \tag{3.25}$$

By using this method, the matrix inverse for each candidate tuning parameter is no longer necessary. Instead, we only need to calculate the generalized eigen decomposition once.

The only requirement is that the matrix A is positive definite. In our problem, Matrix A

is the 'covariance matrix' $W^T\mathbf{X}^T\mathbf{X}W$. With the smoothing penalty, the singular problem can be avoided when the dimension of the covariance matrix is low. Later when we introduce more than one functional variables in the calculation, the dimension of this matrix would become vary large and it would almost certainly become an ill conditioned matrix. In order to overcome this problem, we introduce the second tuning parameter into the calculation.

The second tuning parameter

Meier et al. (2009) introduces a new type of penalty function, called 'sparsity-smoothness'. It is designed for group variables selection. It uses the combination of inner product of second order derivatives and the inner product of the original function. The penalty function can be defined by the following:

$$Pen = \lambda_1 \int [\beta''(t)]^2 dt + \lambda_2 \int [\beta(t)]^2 dt.$$

The matrix we need to invert in $H(\lambda)$ becomes:

$$W^T\mathbf{X}^T\mathbf{X}W + \lambda_1 W_2 + \lambda_2 W^T W,$$

where λ_1 and λ_2 control the amount of penalization from each penalty functions respectively. Simon and Tibshirani (2012) also mentioned this method when ill conditioning happened with the candidate groups for group lasso selection.

If we keep λ_2 fixed as a small number, such as 0.001, the effect from the second penalty would be small, and the numerical difficulty would be avoided. On the other hand, cross validation can be done with respect to both of the tuning parameters. For each choice of λ_2, a GCV can be done. An optimum can be found for λ_1 and λ_2.

3.4.6 Canonical correlation between more than one functional variables and a scalar variable

When there is more than one functional variable in Eqn (3.12), some minor changes are required. Without loosing any generality, we can assume that there are two functional variables in the following expressions. It can be easily extended to the cases with more

functional variables. The canonical correlation is now:

$$\rho((x_1(t), x_2(t)), y) = \max_{(\beta_1(t),\beta_2(t)),\alpha} \frac{\mathrm{Cov}\left(\left\langle (x_1(t), x_2(t)), (\beta_1(t), \beta_2(t)) \right\rangle, \alpha y\right)}{\sqrt{\left[\mathrm{Var}\left(\left\langle (x_1(t), x_2(t)), (\beta_1(t), \beta_2(t)) \right\rangle\right) + \mathrm{PEN}\right]\left[\mathrm{Var}\left(\alpha y\right)\right]}}, \tag{3.26}$$

$$\mathrm{PEN} = \lambda_1^{(1)} \int [\beta_1''(t)]^2 dt + \lambda_2^{(1)} \int [\beta_1(t)]^2 dt + \lambda_1^{(2)} \int [\beta_2''(t)]^2 dt + \lambda_2^{(2)} \int [\beta_2(t)]^2 dt$$

where $(\cdot, \cdot)$ means a 2-vector, $\langle \cdot, \cdot \rangle$ means inner product of the two elements and $\lambda_i^{(j)}$ means the i-th λ for j-th functional variable. This definition can be easily extended to the case with more than two functional variables.

The inner product in Eqn (3.26) can be rewritten as:

$$< (x_1(t), x_2(t)), (\beta_1(t), \beta_2(t)) >=< x_1(t), \beta_1(t) > + < x_2(t), \beta_2(t) >$$

As in the univariate case, the functional variables in the multivariate case are also expressed as representative data points. The variance of functional variable with penalties is written as block matrices. Suppose there are two functional variables, using the general expression of the functional variables and functional coefficients, the covariance matrix of the input joint functional variables can be written as:

$$\begin{pmatrix} W^T \mathbf{X}_1^T \mathbf{X}_1 W + \lambda_1^{(1)} W_2 + \lambda_2^{(1)} W^T W & W^T \mathbf{X}_1^T \mathbf{X}_2 W \\ W^T \mathbf{X}_2^T \mathbf{X}_1 W & W^T \mathbf{X}_2^T \mathbf{X}_2 W + \lambda_1^{(2)} W_2 + \lambda_2^{(2)} W^T W \end{pmatrix}.$$

The penalty functions are all on the diagonal blocks. This can distinguish between different functional variables, and keep each individual functional variable smooth.

One problem now is that the GCV is not suitable due to the number of tuning parameters selected. According to Ramsay and Silverman (2005), the value of tuning parameter is relevant to the Frobenius norm of the corresponding functional variable. With RDP method, this is $\sqrt{\mathrm{trace}(X^T X)}$. As with the block matrices above, the matrices on the diagonal blocks are identical to each other except for $W^T \mathbf{X}_i^T \mathbf{X}_i W$. In order to keep the advantage of GCV, we normalize the functional variable by their point wise mean and standard deviation. This can at least make the Frobenius norms of $\mathbf{X}_i W$ for all i comparable to each other. By using this approximation, the tuning parameters are shared across all functional variables. Thus, there are at most two parameters that need to be tuned.

3.5 Simulation examples

This section contains a few examples, including canonical correlation analysis between functional variables and canonical correlation analysis between a set of functional variables and scalar variables.

Three functional variables and one scalar variable are generated. The sample size for each of the variables is 80. The data generation process is similar to those used in the simulation section in the later chapter.

The functional variables are shown in Figure 3.2. The three functional variables are denoted as $x_1(t)$, $x_2(t)$ and $x_3(t)$, respectively; the scalar variable is denoted as y. The scalar variable y is generated by:

$$y = \int x_1(t)\beta_1(t)dt + \int x_2(t)\beta_2(t)dt + \int x_3(t)\beta_3(t)dt + \epsilon; \quad \epsilon \sim N(0, 0.02).$$

The coefficients $\beta_1(t)$, $\beta_2(t)$ and $\beta_3(t)$ are adjusted such that the projections of the functional variables have similar measure of spread to each other. Thus the correlations between y and each of the functional variables should be close.

The true values of functional coefficients are shown in Figure 3.3. The canonical correlations between two functional variables do not depended on these true value of functional coefficients.

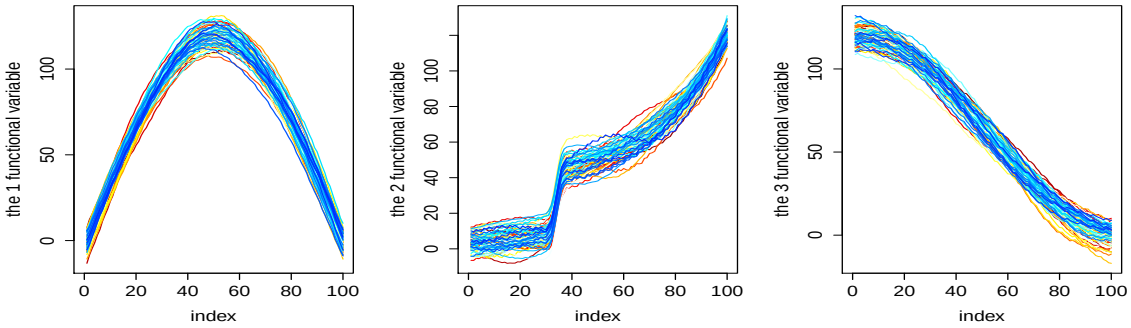


Figure 3.2: Three functional variables generated from three random vectors

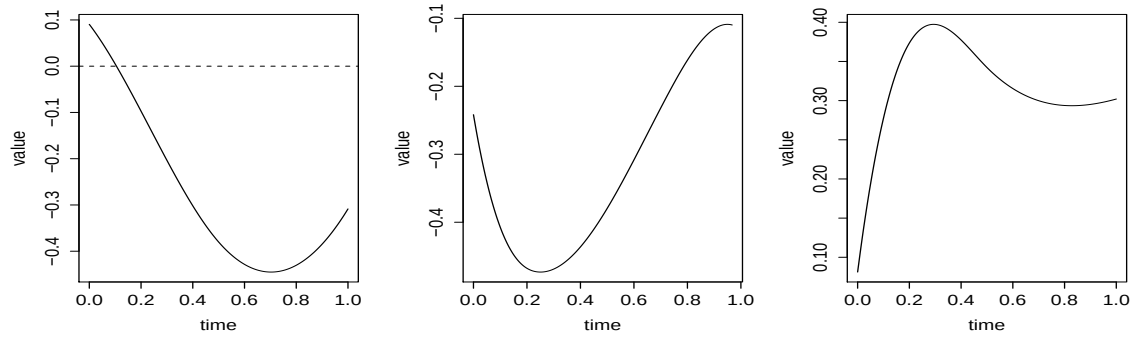


Figure 3.3: Three functional coefficients

3.5.1 Canonical correlation between functional variables

We discussed different discretizing methods for functional variables in Section 3.4. These methods can also be used here. Since the different discretizing methods give similar results, we would just present those that use the RDP method.

With cross validation, the smoothing parameter λ is tested within 15 values from 10^{-4} to 1. Figure 3.4 shows the values of the correlation changes with the smoothing parameters. The x axis shows the candidate smoothing parameters on a log_{10} scale. Within the candidates, the best choice is 0.205. On the other hand, the values of the cross validation seem erratic. This might come from the generalized inverse of the covariance matrices.

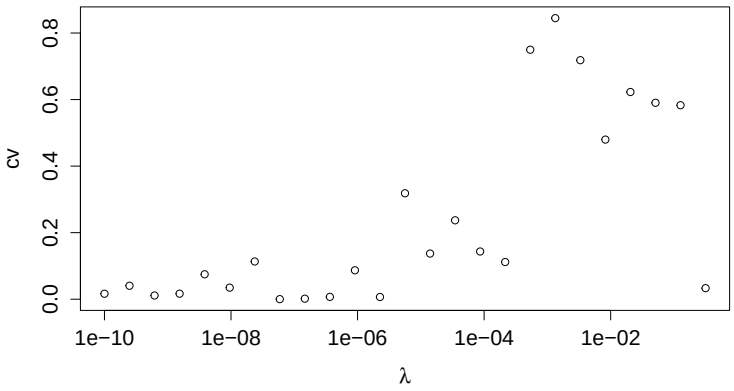


Figure 3.4: The values of cross validation against smoothing parameter in log scale

Table 3.1 shows the first five canonical correlations between $x_1(t)$ and $x_2(t)$; while Figure 3.5 shows the corresponding coefficients from $x_1(t)$ and $x_2(t)$ respectively. The result

is calculated by using the tuning parameter selected by cross validation. Given that the two functional variables are generated from two independently generated random vectors, the values seem reasonable. The correlation values, after the first five, reduce quickly to almost zero.

1	2	3	4	5
0.1106	0.0493	0.0280	0.0101	0.0004

Table 3.1: The five leading canonical correlations between $x_1(t)$ and $x_2(t)$ calculated using the tuning parameter from cross validation

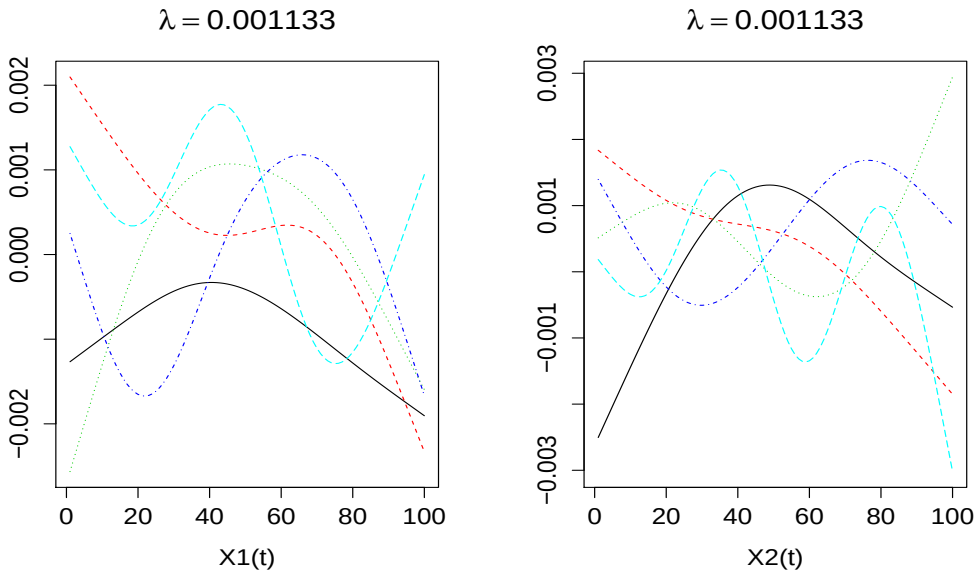


Figure 3.5: The coefficients corresponding to the five leading canonical correlations between $x_1(t)$ and $x_2(t)$. The order of the coefficients are black, red, green, dark blue and light blue.

In order to compare the outcomes corresponding to different values of tuning parameters, the following two examples give results by using extreme values of the tuning parameter. One is $\lambda = 1$ and the other uses $\lambda = 0$. For the later case, the covariance matrix is actually ill-conditioned, thus a result can only be obtained using the generalized inverse. The first five correlations are listed in Table 3.2 and the corresponding coefficients are in Figure 3.6. When $\lambda = 1$, the coefficients corresponding to the leading correlations tend to be a straight line. From the upper two plots, the first two coefficients in black and red from both variables are very smooth. When $\lambda = 0$, the shapes of the coefficients oscillate around zero with high frequency.

	1	2	3	4	5
$\lambda = 1$	3.47×10^{-6}	3.27×10^{-8}	2.75×10^{-10}	0	0
$\lambda = 0$	0.9648	0.7562	0.7002	0.4639	0.3179

Table 3.2: The five leading canonical correlations between $x_1(t)$ and $x_2(t)$

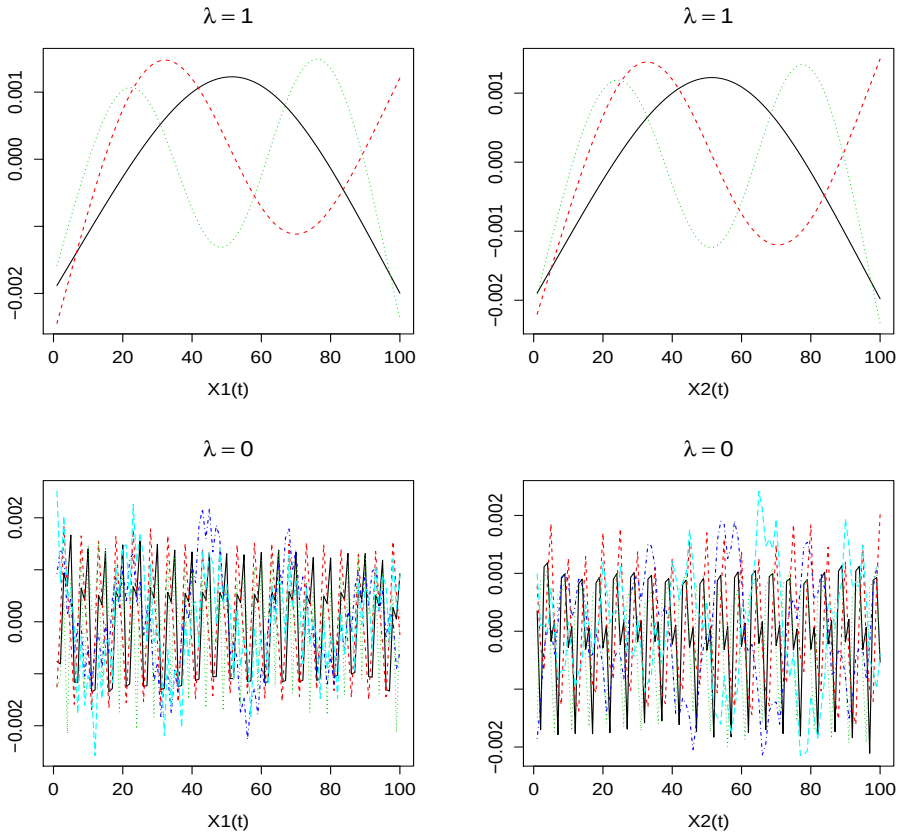


Figure 3.6: The coefficients corresponding to the five leading canonical correlations between $x_1(t)$ and $x_2(t)$ when $\lambda = 1$ or $\lambda = 0$. The order of the coefficients are black, red, green, dark blue and light blue.

3.5.2 Canonical correlation analysis between functional variables and one scalar variable

The correlation between a functional variable and a scalar variable has only one value and therefore, one corresponding functional coefficient. This is because the dimension of the scalar variable is just one. The example contains correlation between $x_1(t)$ and y, and correlation between $(x_1(t), x_2(t))$ and y. Since y is generated by three functional variables together, the correlation between the scalar variable and one or two of these functional variables and should have reasonable big values.

(i). Between one functional variable and one scalar variable

Table 3.3 shows the result using different methods and the smoothing parameters are selected from 41 values between 10^{-20} to 10^5 by generalized cross validation. Clearly all the methods give very similar results.

Discrete data point	basis function	Gaussian quadrature
0.6034	0.6036	0.5966

Table 3.3: Correlation calculated with different discrete methods

We draw the estimated coefficients in Figure 3.7. If we compare with the true coefficient $\beta_1(t)$ in Figure 3.3, the shapes of the estimated coefficients from different discrete methods are all quite close to the true shape. Because of the different discrete methods we use, the scale of the estimated coefficients can be very different from the true coefficient. This does not affect the correlation between the projection and the scalar variable.

For comparison, we show the estimated coefficient using RDP method in Figure 3.8 when the smoothing parameters are very small. The plot on the left shows the outcome when the tuning parameter is zero. The estimated coefficient changes rapidly every where on the curve. The plot on the right shows the outcome when the tuning parameter is large. The shape of the estimated coefficient is very smooth, and close to a straight line.

(ii). Between two functional variables and one scalar variable

Table 3.4 presents the results between y and $\{x_1(t), x_2(t)\}$. As expected, the correlations are larger than the ones between y and $x_1(t)$ from the previous section.

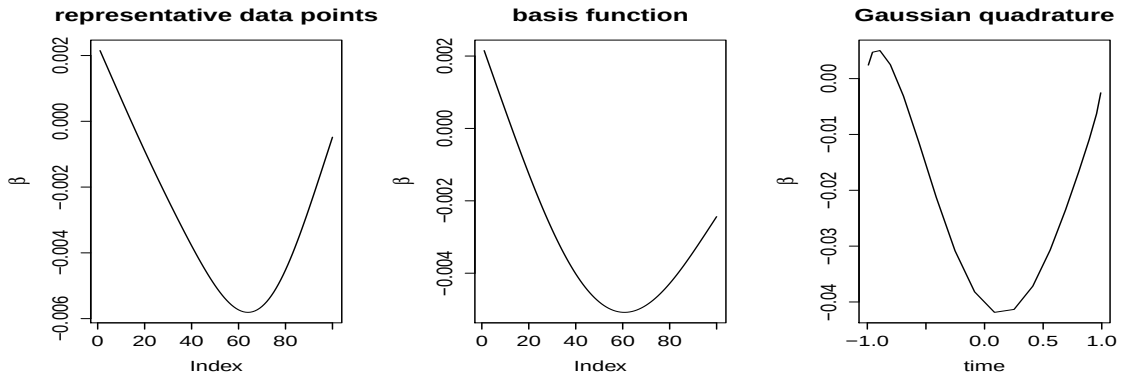


Figure 3.7: Estimated coefficients from the different discrete methods. For RDP expression and BF expression the x axis is the index of data points. For GQ expression, the x axis has the range required by Gaussian quadrature.

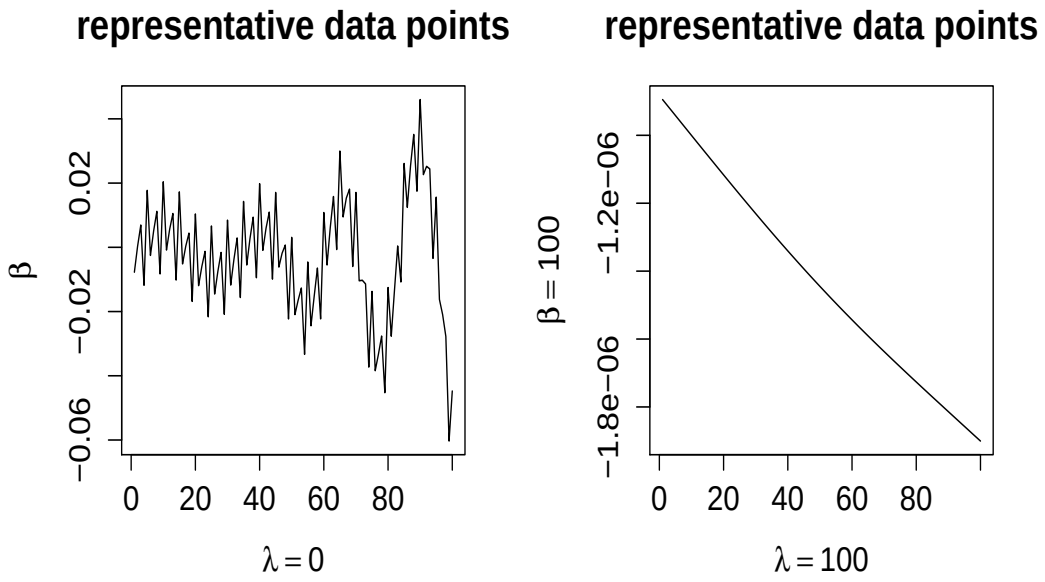


Figure 3.8: Estimations of functional coefficients by using RDP expression.

Discrete data point	basis function	Gaussian quadrature
0.7838	0.7855	0.7648

Table 3.4: Correlation values calculated with different discrete methods between two functional variables and a scalar variable

The coefficients from different discrete methods are shown in Figure 3.9. Recall the generated coefficient $\beta_1(t)$ and $\beta_2(t)$ from Figure 3.3, the shapes of the estimated coefficient in the plots from different discrete methods roughly match the true shape. For the first two discrete methods, the shapes of the coefficients match the true curves very well, while the third way using Gaussian quadrature is slightly worse visually.

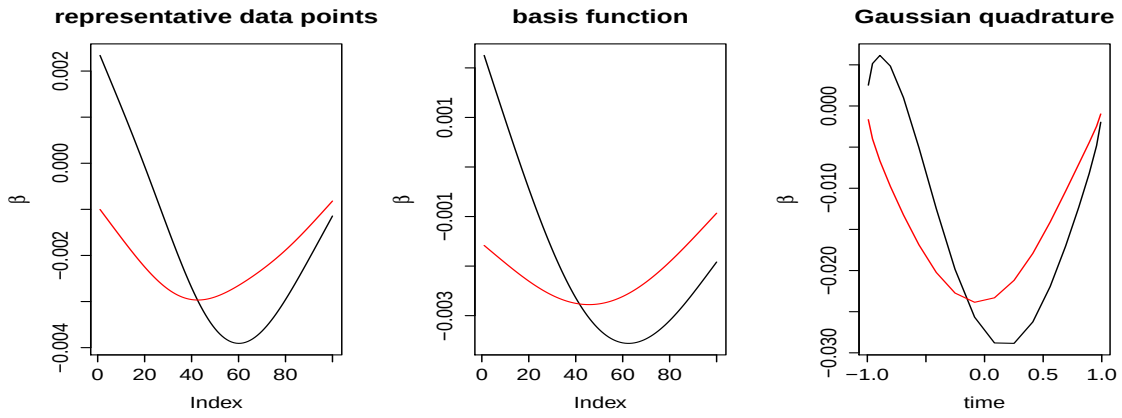


Figure 3.9: The estimated functional coefficient. Black line for the $x_1(t)$ and red line for $x_2(t)$.

3.6 Conclusion and Discussion

We have used the canonical correlations analysis to measure the correlation between functional variables. We noticed that the measures normally have many different non-zero values. If we use a single number to represent the correlation, possible choices are the first canonical correlation, the mean or the mean square of all the non-zero canonical correlations. We used the first canonical correlation (i.e. the maximum value) in this chapter and will do so in the remaining chapters of the thesis. It gives some meaningful results although it usually gives quite a high number. We consider that methods of summarizing these non-zero values is worth a further study.

The other emphasis of this chapter is the relationship between a scalar variable and a group of functional variables. Because the scalar variable has dimension one, we can

only get one correlation from the canonical correlation analysis between a scalar variable and a group of functional variables. From the generalized cross validation formula in Eqn (3.24), we can see that the hat matrix from canonical correlation analysis is related to the hat matrix from a regression problem. In other words, the estimation of the functional coefficient from CCA can be used as the coefficient of the functional regression with functional covariates and scalar response, if we can properly scale the outcome. This is similar to the idea of the partial least square. More precisely, the projection from the canonical correlation between one functional variable and one scalar can explain the same amount of variation as the first component from the functional partial least square (Reiss and Ogden (2007); Aurore Delaigle (2012)).

Chapter 4

Functional Least Angle Regression

In the previous chapter, we discussed how to use canonical correlation analysis to measure the correlation between a group of functional variables and a scalar variable. This enabled us to extend the idea from least angle regression (LARS) and its group selection version to address the functional variable selection problem. This chapter will focus on the regression problem when the response is a scalar variable, and the covariates are functional variables:

$$y = \beta_0 + \sum_{j=1}^{J} \int X_j(t)\beta_j(t)dt + \epsilon \tag{4.1}$$

where y is the response; β_0 is the intercept; $X_j(t)$ is the j-th functional variable with $j = 1, \ldots, J$ and J is the number of candidate functional variables; $\beta_j(t)$ is the j-th functional coefficient; ϵ is the noise term with normal distribution. The dimension J could be very large in this problem. For example, there are more than 500 functional variables in the motion data discussed in Chapter 7. We propose functional least angle regression (functional LARS) for functional variable selection.

The original LARS algorithm and group LARS algorithm will be introduced briefly in Section 4.1. Functional LARS will be defined in Section 4.2 and the technical details will also be provided. The modifications of functional LARS algorithm will be given in Section 4.3, followed by the definition of our stopping rules in Section 4.4. A comprehensive simulation study is conducted and the results are presented in Section 4.5.

4.1 LARS and Group LARS Algorithm

4.1.1 LARS algorithm

In multivariate data analysis, LARS, short for least angle regression, is a variable selection and parameter estimation method. It is a iterative, piecewise linear algorithm with great computational efficiency. This algorithm is designed for the linear multivariate regression model, i.e.:

$$y = \beta_0 + \sum_{j=1}^{J} x_j \beta_j + \epsilon$$

where both response variable y and candidate variables x_j, $(j = 1, \ldots, J)$ are scalar variables; ϵ is the noise term with a normal distribution. This algorithm was proposed by Efron et al. (2003) in order to find the solution for the lasso method proposed by Tibshirani (1996).

The key idea is to do the selection by using the correlation between variables and residuals, such that the redundant, relevant and irrelevant variables are clearly separated. More specifically, the algorithm first finds the variable whose correlation with the current residual is the highest and then defines the direction of the parameter vector such that it moves accordingly. It then moves the parameter vector in the ordinary least squares direction until any of the remaining candidate variables has as much correlation with the current residual as the current direction. The algorithm adds the new predictor to the selected variables. This procedure is then repeated.

The algorithm to be described here is based on a statistical understanding, which is different from the original algebraic understanding. This was pointed out by the discussions of the least angle regression from Turlach (2004) and Efron et al. (2004). In order to give a better description of the algorithm, we put Figure 4.1 and a more detailed descriptions after the algorithm. Figure 4.1 is based on Figure 2 in the original LARS paper with a little modification.

Let the full set of candidate variables $\mathbf{X} = (x_1, x_2, \ldots, x_J)$, where J is the number of variables in total. Let $\mathbf{X}_A$ be the set containing all the selected variables and $\mathbf{X}_{A^c}$ be the rest of the candidates. Suppose that the residuals from the previous iteration are $r^{(k)}$, where k is the index for the current iteration. Note that for the first iteration, $r^{(1)} = y$, where y is the response variable and $\boldsymbol{\beta}$ is the coefficient vector. The variables are centred and scaled

before commencing the regression, so the intercept can be ignored in the algorithm.

The algorithm starts with $k = 1$, $r^{(1)} = y$, $\boldsymbol{\beta} = \mathbf{0}$. The first selection is based on the correlation between $r^{(1)}$ and $\mathbf{X}$. The variable corresponding to the largest absolute value of the correlation is selected. After the first variable is selected, we perform the following steps:

1. Build a unit vector that has equal correlation to each of the selected covariates, and call it $u^{(k)}$ in the k-th iteration. This is done by:

$$u^{(k)} = \frac{\mathbf{X}_{A_k}(\mathbf{X}_{A_k}^T\mathbf{X}_{A_k})^{-1}\mathbf{X}_{A_k}^T r^{(k)}}{||\mathbf{X}_{A_k}(\mathbf{X}_{A_k}^T\mathbf{X}_{A_k})^{-1}\mathbf{X}_{A_k}^T r^{(k)}||_2}$$

   where $||\mathbf{a}||_2$ is the l_2 norm of vector $\mathbf{a}$: $||\mathbf{a}||_2 = \sqrt{\mathbf{a}^T\mathbf{a}}$.

2. Find the new variable and find how far the current $u^{(k)}$ can go in its direction by solving the equation

$$cor(u^{(k)}, r^{(k)} - \alpha_{m^c}^{(k)}u^{(k)})^2 = cor(X_{m^c}, r^{(k)} - \alpha_{m^c}^{(k)}u^{(k)})^2 \quad \text{for } m^c \in A^c. \tag{4.2}$$

   This equation can be rewritten as a quadratic equation with respect to $\alpha_{m^c}^{(k)}$, which is the distance to go in the direction of $u^{(k)}$ for each $m^c \in A^c$. In all $\alpha_{m^c}^{(k)}$, the algorithm finds the one with a minimum positive value and the corresponding covariate. We denote the index of that covariate as m^c*. It then add m^c* into set A and get the distance to go: $\alpha^{(k)} = \alpha_{m^c*}^{(k)}$.

3. The new residual for next iteration is:

$$r^{(k+1)} = r^{(k)} - \alpha^{(k)}u^{(k)}. \tag{4.3}$$

   The estimated coefficient at the k-th iteration $\hat{\beta}^{(k)}$ is made from the non-zero coefficients $\hat{\beta}_m^{(k)}$ and the zero coefficients $\hat{\beta}_{m^c}^{(k)}$, where

$$\hat{\beta}_m^{(k)} = \frac{\alpha^{(k)}(\mathbf{X}_{A_k}^T\mathbf{X}_{A_k})^{-1}\mathbf{X}_{A_k}^T r^{(k)}}{||\mathbf{X}_{A_k}(\mathbf{X}_{A_k}^T\mathbf{X}_{A_k})^{-1}\mathbf{X}_{A_k}^T r^{(k)}||_2}$$

   and $\hat{\beta}_{m^c}^{(k)} = 0$. Note that at this point the newly selected variable still has a zero coefficient.

We repeat steps 1-3 until the algorithm meets the stopping rule or runs out of candidate variables. Since LARS is computational efficient, practically the complete solution path is calculated, and users can find where to stop with different rules and requirements. The estimated coefficient $\hat{\beta}^{[K]}$ at K-th iteration is the sum of the coefficients from all the iterations before and including the K-th one:

$$\hat{\beta}^{[K]} = \sum_{k=1}^{K} \hat{\beta}^{(k)},$$

where $\hat{\beta}^{(k)}$ is made from $\hat{\beta}_m^{(k)}$ and $\hat{\beta}_{m^c}^{(k)}$.

The original algebraic explanation covers the situation when the response variable is negatively correlated with the selected variable, while here we ignore that point. This is because the negative correlation can always be transferred to positive correlation at the beginning of each iteration, and transferred back after the iteration. The description later are all based on the situation where all variables are positively correlated with the current residual in all iterations.

The key step in the algorithm is certainly the second step. There are two important points in this step. Firstly, with respect to each of the candidate variables, the distance that the unit direction vector moves is calculated via an equal squared-correlation equation. Secondly, the variable corresponds to the smallest positive distance that the unit direction vector moves would is selected.

The first point is represented in Figure 4.1. It shows how to find the distance to move for direction vector u with respect to a candidate variable x. The iteration number is omitted for convenience. In the plot, y_1 represents the projection of the current residual on u; y_2 is the projection of the space spanned by x and u. u' is the new direction vector. Suppose that x is selected as the new variable. If x is the last candidate variable, the solution of the regression will reach to the solution of least square estimation, i.e. y_2. If there are still candidate variables available, the distance to move for u' will depend on the next iteration.

The plot shows that the new variable x joins in the regression equation at the point $\alpha \times u$. Unlike forward stepwise selection, the LARS algorithm does not take a full least square step for the direction vector. More specifically, the algorithm takes the next variable x in when x is equally important as the current direction vector u with respect to the current residual $r - \alpha u$, or $y_2 - \alpha u$ in Figure 4.1. The new direction vector u' represents the direction of the current residual $r - \alpha u$ projects on the space spanned by x and u. Because

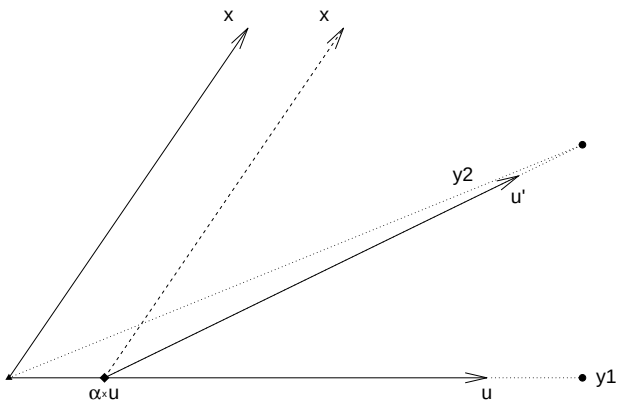


Figure 4.1: Plot relating to finding the distance to move for direction vector u with respect to the variable x. x, u and u' are all unit vectors. The projections from y to u, u', (x, u) are drawn in dotted lines. This plot represents the Eqn (4.2) in the algorithm.

x and u are equally important to the current residual, the new direction vector, u', must have equal angle or equal correlation with x and u. This leads to Eqn (4.2).

For the second point, using the smallest α as the criterion to select the next variable can be understood from the correlation point of view: in each iteration, the variable which has the highest correlation with the current residual is selected. In the first iteration, this statement is obvious. For the later iterations, it is less so.

The LARS algorithm does not take full OLS step for the direction vector, therefore the squared correlation between the residual and the direction vector, i.e.,

$$cor(r - \alpha u, u)^2$$

always reduces when α increases from 0. The minimum value of this squared correlation appears when the full OLS step is taken. In other words $0 \leq \alpha \leq (u^T u)^{-1} u^T r$. If α keeps increasing after it reaches the OLS solution, αu would make the squared correlation between the current residual and u larger. Therefore, if one candidate joins the direction vector earlier than others, the squared correlation of the variable is certainly larger than others. This is also true for the first iteration, where the direction vector is $\mathbf{0}$. It is important to understand that the algorithm always picks the most correlated variable as the next

variable to join the direction vector. Later, our new stopping rule will depend on this behaviour of the algorithm.

For more detailed explanations and discussions see the original paper by Efron et al. (2003).

4.1.2 Group LARS algorithm

Yuan and Lin (2006) proposed a series of group variable selection algorithms, including the group LARS algorithm. The group version of the algorithm solves the variable selection problem for the following regression model:

$$y = \beta_0 + \sum_{j=1}^{J} \tilde{x}_j \tilde{\beta}_j + \epsilon$$

where $\tilde{x}_j$ is the j-th group of variables, with dimension p_j, and ϵ is the usual noise term which follows a normal distribution.

In all the algorithms proposed in that paper, each of the groups of variables are transformed to the orthonormal matrices via QR decomposition before doing any selection (Simon and Tibshirani (2012)). The basic idea of group LARS is using the correlation (or cosine of the angle) between the current residual and the groups. The correlation between $\tilde{x}_j$ and $r^{(k)}$ now is more complex than that of the scalar case, but the correlation between the orthonormal basis of $\tilde{x}_j$ and $r^{(k)}$ is not hard to calculate. Assume that $\tilde{x}_j$ are orthonormal matrices for $j \in 1, \ldots, J$. The squared correlation between $\tilde{x}$ and a scalar response y is defined as:

$$cor^2(\tilde{x}, y) = ||\tilde{x}^T y||^2 / ||y||^2 \tag{4.4}$$

Note that this formula gives only the squared correlation.

Let $\tilde{x}$ be a single group variable, and $\mathbf{X}$ be the collection of all the candidate group variables. Let $\mathbf{X}_A$ be the set that contains all the selected groups and $\mathbf{X}_{A^c}$ be the rest of the candidates. Suppose that the residual from the previous iteration is $r^{(k)}$, where k is the index for the current iteration. Also define X_{A_k} as the matrix comprising the columns of all the variables selected in the k-th iteration. Note that for the first iteration, $r^{(1)} = y$, where y is the response variable. As before, $\boldsymbol{\beta}$ is the coefficient vector and $\boldsymbol{\beta}_j$ is the coefficient corresponding to the j-th covariate. The intercept is ignored by centring and scaling the

variables.

The algorithm starts with $k = 1$, $r^{(1)} = y$, $\boldsymbol{\beta} = \mathbf{0}$. The first selection is based on the correlation between $r^{(1)}$ and $\mathbf{X}$. The group of variables that has the largest absolute correlation with $r^{(1)}$ is selected. After the first variable is selected, we carry out the following steps

1. Define the direction to go by projecting the column combined selected groups of variables $\mathbf{X}_A$ to the current residual $r^{(k)}$:

$$u^{(k)} = X_{A_k}(X_{A_k}^T X_{A_k})^{-1} X_{A_k}^T r^{(k)} \tag{4.5}$$

   Group LARS does not normalize the direction vector. so the distance vector $u^{(k)}$ is not a unit vector.

2. Find the new group and the distance to move in the direction of $u^{(k)}$ by solving the following equation:

$$||\tilde{x}_m^T(r^{(k)} - \alpha_{m^c}^{(k)} u^{(k)})||^2 / p_i = ||\tilde{x}_{m^c}^T(r^{(k)} - \alpha_{m^c}^{(k)} u^{(k)})||^2 / p_{m^c} \quad \text{for } m^c \in A^c, m \in A \tag{4.6}$$

   where p_m and p_{m^c} are the dimensions of the corresponding group variables. Here m can be chosen arbitrarily. This equation can be rewritten as a quadratic equation with respect to $\alpha_{m^c}^{(k)}$, which is the distance to move in the direction of $u^{(k)}$ for each $m^c \in A^c$. In all $\alpha_{m^c}^{(k)}$, the algorithm finds the one with minimum positive value and its corresponding covariate. Denote the index of that covariate as m^c*. It then adds m^c* into the set A and gets the distance to go as $\alpha^{(k)} = \alpha_{m^c*}^{(k)}$.

3. The new residual for next iteration is:

$$r^{(k+1)} = r^{(k)} - \alpha^{(k)} u^{(k)}.$$

   The coefficient for the k-th iteration $\boldsymbol{\beta}^{(k)}$ is made from the non-zero coefficients $\boldsymbol{\beta}_m^{(k)}$ and the zero coefficients $\boldsymbol{\beta}_{m^c}^{(k)}$, where

$$\boldsymbol{\beta}_m^{(k)} = \alpha^{(k)} X_{A_k}(X_{A_k}^T X_{A_k})^{-1} X_{A_k}^T r^{(k)}$$

   and $\boldsymbol{\beta}_{m^c}^{(k)} = 0$.

Similarly with the scalar case, we repeat steps 1-3 until the algorithm meets the stopping rule or runs out of candidates.

For non-orthonormal matrices $\tilde{x}_j$, QR decomposition is applied such that:

$$\tilde{x}_j = Q_j R_j \text{ for } j \in 1, \ldots, J.$$

We define matrix Q_{A_k} as the matrix comprised the columns of Q_m for $m \in A$ in the k-th iteration. The estimated coefficient $\hat{\beta}^*$ using Q_{A_k} at iteration k made from the non-zero coefficients $\hat{\boldsymbol{\beta}}_m^{(k)}$ and the zero coefficients $\hat{\boldsymbol{\beta}}_{m^c}^{(k)}$, where

$$\hat{\boldsymbol{\beta}}_m^{*k} = \alpha^{(k)} Q_{A_k} (Q_{A_k}^T Q_{A_k})^{-1} Q_{A_k}^T r^{(k)}$$

and $\boldsymbol{\beta}_{m^c}^{(k)} = 0$.

For the j-th groups, the estimated coefficient $\hat{\boldsymbol{\beta}}_j^{[K]}$ until the K-th iteration would be:

$$\hat{\boldsymbol{\beta}}_j^{[K]} = \sum_{k=1}^{K} R_j^{-1} \hat{\boldsymbol{\beta}}_j^{*k}, \tag{4.7}$$

where $\hat{\boldsymbol{\beta}}_j^{*k}$ corresponds to the j-th group of variables from iteration k.

4.2 Functional LARS Algorithm

We extend the idea of the LARS to a functional data analysis framework and propose functional LARS algorithm in this section. Recall the multivariate functional regression problem in Eqn (4.1):

$$y^r = \beta_0 + \sum_{j=1}^{J} \int x_j^r(t) \beta_j(t) dt + \epsilon$$

where y^r is the response variable; $x_j^r(t)$'s are functional variables; ϵ is the noise term that follows a normal distribution with mean zero and unknown variance σ^2. The superscript r means they are raw data, which are not centred or scaled.

All of the discrete methods we use to represent a functional variable would give us a group of variables. More specifically, it gives a group of very high dimensional variables by using the RDP method; it gives a group of variables with dimension corresponding to the p-point-rule by using the GQ method; and it gives a group of variables with dimension equals to the number of basis functions we choose by using the BF method. Therefore, functional variables naturally have the group property. Thus group LARS can be extended

to the functional case, with some necessary modifications.

However group LARS cannot be applied directly to functional case. The most serious problem is that the dimension of the discrete functional variable is often high, for all of the discrete methods we use. The dimension could be even higher than the sample size when we use the RDP method. Therefore, the discrete functional variables are usually short rank matrices, and it would be difficult to carry out the QR decomposition for them. Moreover, recall Eqn (4.7), R_j^{-1} is required to get the estimated functional coefficient for $x_j(t)$. Thus inverting R_j would create numerical error for the estimation of the functional coefficients.

If the group variables cannot be represented as orthonormal matrices, the squared correlation can no longer be calculated by Eqn (4.4). Thus other correlation measures must be applied in the algorithm. Many correlation measures have been proposed in the functional case. We first propose an algorithm without specifying the choice of correlation measures.

We centre and scale the response variable and the functional variables by their mean and point-wise standard deviation: μ_y, σ_y, $\mu_j(t)$ and $\sigma_j(t)$. Thus the intercept can be ignored during the selection. We will recover the intercept and the functional coefficients for the original variables after selection.

After centring and scaling, we get y and $x_j(t)$. Let $x_j(t)$ be a single functional variable, and $\mathbf{x}(t)$ be the set that contains all the candidates. Let $\mathbf{x}_A(t)$ be the set that contains all the selected functional variables and $\mathbf{x}_{A^c}(t)$ contains the rest of the candidates. Suppose that the residuals obtained from the previous iteration is $r^{(k)}$, where k is the index of the current iteration. Note that for the first iteration, $r^{(1)} = y$, where y is the response variable. $\boldsymbol{\beta}(t)$ is the functional coefficient vector. Denote $\rho^2(x(t), y)$ as the squared correlation between a functional variable $x(t)$ and a scalar variable y.

The algorithm starts with $k = 1$, $r^{(1)} = y$, $\boldsymbol{\beta}(t) = \mathbf{0}$. The first selection is based on the correlation between $r^{(1)}$ and $\mathbf{x}(t)$. The covariate which has the largest absolute correlation with $r^{(1)}$ is selected. After the first variable is selected, we carry out the following steps:

1. Define the direction $u^{(k)}$ to move in by projecting the scalar variable to the selected functional variables:

$$u^{(k)} = \frac{\sum_m \int x_m(t)\beta_m^{(k)}(t)dt}{\mathrm{sd}(\sum_m \int x_m(t)\beta_m^{(k)}(t)dt)} \quad m \in A.$$

   $u^{(k)}$ must have positive correlation with the residual $r^{(k)}$. Note that $\beta_m^{(k)}(t)$ is un-

known, and needs to be estimated in this step with respect to $r^{(k)}$. Unlike the group LARS case, we normalize the direction vector $u^{(k)}$ using its standard deviation. This normalization will be useful in defining our stopping rule later.

2. For each $m^c \in A^c$, compute $\alpha_{m^c}^{(k)}$ using

$$cor(u^{(k)}, r^{(k)} - \alpha_{m^c} u^{(k)})^2 = \rho^2(x_{m^c}(t), r^{(k)} - \alpha_{m^c} u^{(k)})^2 \text{ for } m^c \in A^c \tag{4.8}$$

For all $\alpha_{m^c}^{(k)}$, the algorithm finds the one with the minimum positive value and its corresponding covariate. Denote the index of that covariate as m^c*. We then add m^c* into the set A and get $\alpha_{m^c*}^{(k)}$ as the distance to move in the direction vector.

3. The new residual for next iteration is:

$$r^{(k+1)} = r^{(k)} - \alpha_{m^c*}^{(k)} u^{(k)}.$$

As before, the functional coefficient up to the K-th iteration is the sum of all the coefficients calculated up to and including the current iteration.

The model we get from the algorithm is up to the K-th iteration is as following:

$$y = \sum_{j=1}^{J} \int x_j(t) \hat{\beta}_j^*(t) dt,$$

where we omit the iteration index for the coefficients.

After transforming with the means and standard deviations from the data, the model becomes:

$$y^r = \mu_y - \sum_{j=1}^{J} \int \frac{\mu_j(t) \hat{\beta}_j^*(t) \sigma_y}{\sigma_j(t)} dt + \sum_{j=1}^{J} \int x_j^r(t) \frac{\hat{\beta}_j^*(t) \sigma_y}{\sigma_j(t)} dt.$$

So

$$\hat{\beta}_0 = \mu_y - \sum_{j=1}^{J} \int \frac{\mu_j(t) \hat{\beta}_j^*(t) \sigma_y}{\sigma_j(t)} dt,$$

$$\hat{\beta}_j(t) = \frac{\hat{\beta}_j^*(t) \sigma_y}{\sigma_j(t)}.$$

One remaining problem is how to find the solution to Eqn (4.8). We provide the details of

one way to do it below.

4.2.1 The solutions of Eqn (4.8)

The functional canonical correlation analysis discussed in the previous chapter is a good choice of correlation measure in the functional LARS algorithm. It provides a projection of a group of functional variables on a scalar variable and gives a squared correlation between this groups of functional variables and the scalar variable. By using this correlation, we can utilise the piece-wise linear property that LARS has.

All LARS, group LARS and functional LARS algorithms use direction vectors to represent the information in the selected variables with respect to the response variable. By using the direction vectors, all the coefficients of selected variables change linearly with the distance moved at each iteration. This piece-wise linear property gives LARS algorithm significant computational advantages.

If we disregarded the piece-wise linear property in functional LARS, we would not have a direction vector u in each iteration. Instead, we would need to calculate

$$\rho^2\left(\mathbf{x}_A(t), r-\alpha_{m^c}\int\sum_{m\in A}x_m(t)\beta_m(t)dt\right)=\rho^2\left(x_{m^c}(t), r-\alpha_{m^c}\int\sum_{m\in A}x_m(t)\beta_m(t)dt\right)$$

in the k-th iteration to find the distance to move for each candidate functional variables. Both correlation and the functional coefficients $\beta_m(t)$ are unknown. They are estimated with respect to $\mathbf{x}_A(t)$ and the residual $r-\alpha_{m^c}\int\sum_{m\in A}x_m(t)\beta_m(t)dt$. In addition, $\beta_m(t)$ would changes with the value of α_{m^c}. Thus the search for next variable to select would be computationally impractical if we want very accurate results by using a fine grid for the value of α.

Therefore, in the functional LARS algorithm, piece-wise linearity also means that for each iteration, the projection of a group of functional variables with respect to the current residual $r-\alpha u$ does not change with the distance moved, and neither does the corresponding functional coefficients. More specifically, in Eqn (4.8), the projection of the selected functional variables stays the same when the residual $r-\alpha u$ changes with α.

Recall the general solution of the functional canonical correlation in Eqn (3.20):

$$\begin{aligned}\rho^2 &= \frac{(r-\alpha u)^T \mathbf{X} W (W^T \mathbf{X}^T \mathbf{X} W + \lambda_1 W_2 + \lambda_2 W^T W)^{-1} W^T \mathbf{X}^T (r-\alpha u)}{(r-\alpha u)^T (r-\alpha u)} \\ &= \frac{(r-\alpha u)^T S (r-\alpha u)}{(r-\alpha u)^T (r-\alpha u)} \end{aligned} \tag{4.9}$$

where $S = \mathbf{X} W (W^T \mathbf{X}^T \mathbf{X} W + \lambda_1 W_2 + \lambda_2 W^T W)^{-1} W^T \mathbf{X}^T$; $\mathbf{X}$ is the discrete data matrix; W is the matrix depending on the discrete method we choose for $\beta(t)$; W_2 is the matrix for smoothness penalty; λ_1 and λ_2 are two tuning parameters in 'sparsity-smoothness' penalty we mentioned in Chapter 3.4.5. The correlation ρ_s between the residual $r - \alpha$ and the direction vector u is:

$$\begin{aligned}\rho_s = \mathrm{Cor}(r-\alpha u, u) &= \frac{\mathrm{Cov}(r-\alpha u, u)}{\sqrt{\mathrm{Var}(r-\alpha u)\mathrm{Var}(u)}} \\ &= \frac{(r-\alpha u)^T u/(n-1)}{\sqrt{(r-\alpha u)^T (r-\alpha u)/(n-1)}\sqrt{u^T u/(n-1)}} \\ &= \frac{(r-\alpha u)^T u}{\sqrt{(r-\alpha u)^T (r-\alpha u) u^T u}}.\end{aligned}$$

Thus if the squared canonical correlation between functional variable $x(t)$ and residual $r - \alpha u$ equals to the squared correlation between two scalar variables u and $r - \alpha u$, i.e. $\rho^2 = \rho_s^2$, Eqn (4.8) can be written as:

$$\begin{aligned}\frac{(r-\alpha u) S (r-\alpha u)}{(r-\alpha u)^T (r-\alpha u)} &= \left(\frac{(r-\alpha u)^T u}{\sqrt{(r-\alpha u)^T (r-\alpha u) u^T u}} \right)^2 \\ &= \frac{(r^T u - \alpha u^T u)(r^T u - \alpha u^T u)}{(r-\alpha u)^T (r-\alpha u) u^T u}.\end{aligned}$$

The only unknown item in the equation is α, and thus the equation can be written as a quadratic function with respect to α:

$$\alpha^2 (u^T S u u^T u - u^T u u^T u) - 2\alpha (r^T S u u^T u - u^T u r^T u) + (r^T S r u^T u - r^T u r^T u) = 0 \tag{4.10}$$

This equation can be solved analytically.

There are two tuning parameters in the matrix S arising from the 'sparsity-smoothness' penalty. The smoothing parameter λ_1 is tuned by generalized cross validation, or GCV and λ_2 is tuned by cross validation. Because GCV is computationally efficient, we tune

λ_1 from a large interval with fine grid. On the other hand, we tune λ_2 by cross validation, so we tune it on a fairly small interval with coarse grid. We discussed details about the tuning of the two parameters in Chapter 3.

4.3 Modifications

The algorithm works well in general. However, in some special cases, the algorithm may not work as we expected. Here we propose two modifications to make the algorithm more reliable and efficient. The first one can be used when Eqn (4.10) has no real solution for all candidate variables. The second modification is to remove the redundant variables from the selected variables.

4.3.1 Modification I

In the algorithm, we must calculate α's for all candidate variables in order to perform the selection. However, in some cases, there may be no real solutions for α with respect to all candidate functional variables from their quadratic equations. This happens when the candidate variables contain very little information about the current residual. More precisely, the correlation between any candidate variable and the current residual is smaller than the correlation between the current direction and the current residual for all possible values of α, even when αu reaches the OLS solution. In addition, if all candidates are selected, it is impossible to find the next selection and the distance to move.

In both cases, the selection and the calculation of α fail. Therefore, the only choice is to carry out the OLS for the current direction vector. Thus the algorithm is modified by taking the full OLS using the projection of the selected variables in the regression model when α has no real solutions. If there are candidate variables left, the algorithm may still be able to carry on after this step, but the contribution from the new variables would be little. In other words, it is safe to stop the selection in this case. This is a special case of finding the stopping point. More details about the stopping rules will be given in the next section.

4.3.2 Modification II

The LARS algorithm itself would not reduce the number of selected variables without modification. But, with the right modification, it can give lasso solution, which will involve dropping some variables selected. A variable is removed from the regression equation when the sign of its coefficient changes. Changing of the coefficient sign indicates that there is a point at which the coefficient is exactly zero. This implies that the corresponding variable contributes nothing to explain the variation of the response variable at that point. So we can remove that variable when its coefficient becomes zero.

For the functional LARS algorithm, the sign change is no longer feasible, since the sign of $\beta(t)$ cannot be defined directly. An alternative way is to measure the contribution of a variable by calculating the variance of its projection on the response variable, i.e., $\mathrm{Var}\left(\int x(t)\beta(t)dt\right)$. When the contribution of a variable reduces to zero, the variance of its projection in the regression equation would also be zero.

However, it is impossible to find the point when the variance of the projection of a functional variable reaches zero exactly. An alternative option is to remove the variable when the variance of its projection reduces to a small value. To implement this idea, a threshold is required. The variable is removed from the model when the following two conditions are met. Firstly, the variance of the projection of the variable is smaller than the maximum variance from the same variable:

$$\mathrm{Var}\left(\int x_j(t)\beta_j^{(k)}(t)\right) < \mathrm{Var}\left(\int x_j(t)\beta_j^{(k*)}(t)\right) \text{ for } k* \in 1,\ldots,k-1.$$

Secondly, the variance of the projection of the variable is less than a certain percentage of the total variance of the response variable:

$$\mathrm{Var}\left(\int x_j(t)\beta_j^{(k)}(t)\right) < \kappa \mathrm{Var}(y),$$

where κ is the threshold.

The first condition is to make sure that the variable to remove becomes less and less useful when new variables join the regression equation. The second condition is to make sure that the variable is removed when its contribution is small. The second condition requires a threshold. In our simulation, a figure of 10% is used.

4.4 Stopping Rules

Similar to the LARS and group LARS algorithms, functional LARS also calculates the full solution path or solution surface more precisely. Practically, the final model should be one of the estimates on the solution path. We can always use leave-one-out cross validation to find the optimal stopping point, but it is very time consuming to do so. Therefore we need to use other stopping rules which are less expensive computationally. Mallow's C_p-type criterion (Mallows (1973)) has been used in the LARS and group LARS as a stopping rule. In addition, other traditional methods, including Akaike information criterion (AIC) (Akaike (1998)), Bayesian information criterion (BIC) (Schwarz et al. (1978)), R^2 coefficient and adjusted R^2 coefficient can also be used. These criteria can be used in functional LARS but only when the number of variables is not too large. There are other issues in functional LARS of using these criteria. In order to overcome this shortcoming, we propose a new stopping rule.

We explain Mallow's C_p in details here because it is preferred in many other related algorithms such as LARS, group LARS and group lasso. It is also very closely related to other criteria, such as AIC and BIC. It would be very easy to calculate other information based criteria if we can calculate Mallow's C_p. However, in functional data analysis model selection problems, such as smoothing, leave-one-out cross validation or generalized cross validation are preferred (Ramsay and Silverman (2005), Gertheiss:2013). One important reason is that when we discretize the functional variables, the calculation of the degrees of freedom would be inaccurate.

4.4.1 C_p-type criterion

For a linear regression problem:

$$y = \beta_0 + \sum_{j=1}^{J} z_j \beta_j + \epsilon$$

where $\epsilon \sim N(0, \sigma^2)$. We can assume that all the variables are centred, so that $\beta_0 = 0$. Thus the response variable y has the distribution $N(\mu, \sigma^2)$, where $\mu = \sum_{j=1}^{J} z_j \beta_j$.

Assume that we have n independent sets of observations, where the i-th set is:

$$\mathcal{D}^i = (y_i, z_{i,1}, \ldots, z_{i,J}) \text{ for } i = 1, \ldots, n.$$

Thus we have:

$$y_i \sim N(\mu_i, \sigma^2)$$

where $\mu_i = \sum_{j=1}^{J} z_{i,j}\beta_j$; $i = 1, \ldots, n$.

The value $\hat{\mu}_i$ is obtained by leave one out cross-validation to estimate the mean value of $\mathbf{y}_i$. In other words, they are obtained from a prediction point of view (Efron (1986); Efron and Tibshirani (1997a)). Denote $g(\cdot)$ as the function to calculate this value. In linear regression, $g(y_i) = \hat{\mu}_i = H^{-i}y_i$, where H^{-i} is the "hat" matrix estimated from all the samples excluding the i-th sample. The squared error is:

$$(\hat{\boldsymbol{\mu}} - \boldsymbol{\mu})^2 = (\mathbf{y} - \hat{\boldsymbol{\mu}})^2 - (\mathbf{y} - \boldsymbol{\mu})^2 + 2(\hat{\boldsymbol{\mu}} - \boldsymbol{\mu})(\mathbf{y} - \boldsymbol{\mu}).$$

The expectation of the squared error is:

$$\begin{aligned}\mathbb{E}[(\hat{\boldsymbol{\mu}} - \boldsymbol{\mu})^2] &= \mathbb{E}[(\mathbf{y} - \hat{\boldsymbol{\mu}})^2] - \mathbb{E}[(\mathbf{y} - \boldsymbol{\mu})^2] + 2\mathbb{E}[(\hat{\boldsymbol{\mu}} - \boldsymbol{\mu})(\mathbf{y} - \boldsymbol{\mu})] \\ &= \mathbb{E}[(\mathbf{y} - \hat{\boldsymbol{\mu}})^2] - \mathbb{E}[\mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{\mu}] + 2\mathbb{E}[(\hat{\boldsymbol{\mu}} - \boldsymbol{\mu})(\mathbf{y} - \boldsymbol{\mu})].\end{aligned}$$

For each i, $\mathbb{E}[y_i^2 - 2y_i\mu_i + \mu_i^2] = \mathrm{Var}(\epsilon) = \sigma^2$, and $\mathbb{E}[(\hat{\mu}_i - \mu_i)(y_i - \mu_i)] = \mathrm{Cov}(\hat{\mu}_i, y_i)$. Because $\hat{\mu}_i$ is obtained from all the samples but the i-th one, the covariance of $\mathrm{Cov}(\hat{\mu}_i, y_j)_{i \neq j}$ is equal to 0. Therefore:

$$\mathbb{E}[(\hat{\boldsymbol{\mu}} - \boldsymbol{\mu})^2] = \mathbb{E}[(\mathbf{y} - \hat{\boldsymbol{\mu}})^2] - n\sigma^2 + 2\mathrm{tr}(\mathrm{Cov}(\hat{\boldsymbol{\mu}}^T, \mathbf{y}^T)),$$

where $\mathrm{tr}(\cdot)$ means the trace of a matrix. The normalized squared error is:

$$\mathbb{E}\left[\frac{(\hat{\boldsymbol{\mu}} - \boldsymbol{\mu})^2}{\sigma^2}\right] = \mathbb{E}\left[\frac{(\mathbf{y} - \hat{\boldsymbol{\mu}})^2}{\sigma^2}\right] - n + 2\mathrm{tr}\left(\frac{\mathrm{Cov}(\hat{\boldsymbol{\mu}}^T, \mathbf{y}^T)}{\sigma^2}\right).$$

This equation leads to the estimation of C_p-type criterion:

$$C_p = \mathbb{E}\left[\frac{(\mathbf{y} - \hat{\boldsymbol{\mu}})^2}{\sigma^2}\right] - n + 2\mathrm{df}, \tag{4.11}$$

where df is the degrees of freedom, defined as:

$$\mathrm{df} = \mathrm{tr}\left(\frac{\mathrm{Cov}(\hat{\boldsymbol{\mu}}^T, \mathbf{y}^T)}{\sigma^2}\right). \tag{4.12}$$

This definition of degrees of freedom is discussed in Ye (1998) as 'generalized degrees of freedom'. Efron and Tibshirani (1997b) also mentioned this type of degrees of freedom. Both the LARS and group LARS algorithms use Eqn (4.11) and Eqn (4.12) in their stopping rules. However, group LARS uses the variance of the response variable rather than the variance of the error as σ^2, and thus they only obtain approximated degrees of freedom. Zou et al. (2007) discussed the calculation of the degrees of freedom for the lasso estimation obtained by LARS.

The estimation of C_p and the degrees of freedom involve unknown parameters: $\boldsymbol{\mu}$ and σ^2. Usually, these parameters are estimated from the ordinary least square estimation using all the variables. Therefore, if the residual from the ordinary least square has zero variance, we cannot calculate C_p, the degrees of freedom and other information criteria which requires σ^2. At the moment, we assume the residual from OLS using all variables has non-zero variance σ^{*2} and mean μ^*. We use μ^* as the expectation of y.

Practically, in order to calculate $\text{Cov}(\hat{\boldsymbol{\mu}}^T, \mathbf{y}^T)$, the Bootstrap method or the Monte Carlo method are used. For each of set of the samples $\mathcal{D}_i$, we generate B new realizations of the response y_i from $N(\mu_i^*, \sigma^{*2})$. Denote the new generated values as y_b^*, $b = 1, \ldots, B$. New fitted values $\hat{\boldsymbol{\mu}}_b^*$ with respect to these new realizations can be calculated by using the same function $g(.)$ as before. Both Ye (1998) and Efron et al. (2003) used this method to find degrees of freedom. The draw back of bootstrap method is that the computational cost would be high.

Here we propose our definition of the degrees of freedom. Recall Eqn (4.3), in the functional LARS algorithm, the residual after iteration k can be written as

$$r^{(k+1)} = r^{(k)} - \alpha_k u^{(k)}$$

where u is the direction vector, calculated by:

$$u^{(k)} = \frac{H_k r^{(k)}}{\text{SD}(H_k r^{(k)})}.$$

Therefore, the true "hat" matrix at iteration k is :

$$H_k^* = \frac{H_k \alpha_k}{\text{SD}(H_k r^{(k)})},$$

where $H_k = \mathbf{X}W(W^T\mathbf{X}^T\mathbf{X}W + \lambda_1 W_2 + \lambda_2 W^T W)^{-1}W^T\mathbf{X}^T$ from Eqn (4.9).

The residual after iteration k becomes:

$$r^{(k+1)} = (I - H_k^*)r^{(k)}.$$

Recursively, the fitted value after iteration K with respect to the response variable is:

$$\begin{aligned}\hat{y} =& y - r^{(K+1)} \\ =& y - [\prod_{k=1}^{K}(I - H_k^*)]y \\ =& [I - \prod_{k=1}^{K}(I - H_k^*)]y,\end{aligned}$$

hence the "hat" matrix $\bar{H}_K$ after iteration K is

$$\bar{H}_K = I - \prod_{k=1}^{K}(I - H_k^*). \tag{4.13}$$

We then define the degrees of freedom for functional LARS as follows:

$$\begin{aligned}\mathrm{df}^* &= \mathrm{tr}\left(\frac{\mathrm{Cov}(\hat{\boldsymbol{\mu}}^{*T}, \mathbf{y}^{*T})}{\sigma^2}\right) \\ &= \mathrm{tr}\left(\frac{\mathrm{Cov}(\mathbf{y}^{*T}\bar{H}_k^T, \mathbf{y}^{*T})}{\sigma^2}\right) \\ &= \mathrm{tr}\left(\frac{\bar{H}_k\mathbf{y}^*\mathbf{y}^{*T}/(n-1)}{\sigma^2}\right)\end{aligned}$$

where $\mathbf{y}^*\mathbf{y}^{*T}/(n-1)$ is an $n \times n$ matrix. The i, j-th element is $\mathrm{Cov}(y_i, y_j)$. Its value is σ^2 if $i = j$ and 0 otherwise. Hence:

$$\begin{aligned}\mathrm{df}^* &= \mathrm{tr}\left(\frac{\bar{H}_k\mathbf{y}^*\mathbf{y}^{*T}/(n-1)}{\sigma^2}\right) \\ &= \mathrm{tr}\left(\frac{\bar{H}_k\sigma^2 I}{\sigma^2}\right) \\ &= \mathrm{tr}(\bar{H}_k).\end{aligned} \tag{4.14}$$

Combining Eqn (4.11), Eqn (4.14) and the estimated value of σ^{*2}, we have

$$C_p = \frac{\mathbb{E}[(y - \hat{y})^2]}{\sigma^{*2}} - n + 2\mathrm{df}^*. \tag{4.15}$$

Other criteria such as AIC and BIC also require the error variance and the degrees of freedom. We omit the equations of these traditional criteria.

Remarks

1. The degrees of freedom defined above uses the "hat" matrix from each iteration. The "hat" matrix at the k-th iteration is:

   $$H_k = \mathbf{X}W(W^T\mathbf{X}^T\mathbf{X}W + \lambda_1 W_2 + \lambda_2 W^T W)^{-1}W^T\mathbf{X}^T$$

   where λ_1 and λ_2 are tuning parameters. Recall from Section 3.4.5, that the tuning parameters are calculated with respect to both scalar and functional variables. Therefore the values of tuning parameters can be thought of as calculated from a function of the scalar and functional variables. The values of the tuning parameters involves the information about the relationship between the two variables. So the value of the degrees of freedom does not purely dependent on the functional covariates in the regression model. More specifically, the value of the degrees of freedom depends more on the relationship between the functional variables and the response variable. If the correlation between the functional variable and a scalar variable is small, the corresponding tuning parameters would be large. Large tuning parameters leads to small degrees of freedom. Thus the degrees of freedom reduces when the correlation between the scalar response and the functional variable in the regression model reduces and vice versa.

   We expect that when the number of variables increases in the model, the value of degrees of freedom increases and the expected normalized squared residual decreases, so we can have an optimal point to stop the algorithm. However, in functional LARS, the degrees of freedom does not necessarily increase when then number of functional variables increases. Usually, the stopping rules using degrees of freedom would stop at an optimal point, but non in the functional LARS. Thus it would be very difficult to use those stopping rules which depend on the degrees of freedom. More discussions will be given in the first scenario of the simulation study in Section 4.5.

2. Not only tuning parameters have disadvantages, but they also have some advantages. Firstly, the tuning parameters make sure that no singularity problem occurs in the inverse in the calculation. This means that we will always have a result even with a large number of functional variables in the regression equation. Secondly, when the correlation between the functional variables and the response variable is small, the tuning parameter would be large, and this leads to a very small estimated coefficient. This property guarantees us an estimation of the error variance using OLS with all candidate variables. In addition, these estimated coefficient with small scale gives the prediction a certain amount of robustness.

4.4.2 New general stopping rules

If the residual from OLS using all candidate variables has zero mean, we cannot use the criteria discussed above. We will propose a new stopping rule based on the underlying idea of algorithm itself.

The idea is to compare the values of the distance that the direction unit vector goes. This is the quantity α calculated from Eqn (4.8) in each iteration when we looking for the next variable:

$$\mathrm{Cor}(u^{(k)}, r^{(k)} - \alpha_j u^{(k)})^2 = \rho(Z_j, r^{(k)} - \alpha_j u^{(k)})^2 \text{ for } j \in A^c, i \in A,$$

where $\rho(\cdot)$ could be the functional canonical correlation. Recall that $u^{(k)}$ is the direction vector in the k-th iteration; $r^{(k)}$ is the residual from previous iteration; Z_j is one of the candidate functional variables. We will omit the iteration index in the later description.

Since the direction vector u is unit vector, the distance α is equivalent to:

$$\alpha = \alpha||u||_2 = ||\alpha u||_2. \tag{4.16}$$

$r - \alpha u$ is the residual left after the current iteration. Thus αu represents the variation explained in the current iteration by the current direction vector u. From Eqn (4.16), we know that α can be used to replace αu to represent the amount of variation explained in the current iteration. A very small α means that the current direction vector u contributes very little in the current iteration. Such a phenomenon can happen in two situations: firstly, the current direction vector u is informative and it is almost equally important as the newly selected variable Z with respect to the residual r. Secondly, u is unable to explain much variation in the residual r.

In the latter situation, u would have small correlation with r. Therefore it only needs a small distance for u to move to reach the OLS solution with respect to r. This indicates that we can stop the algorithm when the distance is very small, or before the distance becomes very small.

However, we need to exclude the first situation mentioned above. When u is actually an informative direction vector with respect to $r^{(k)}$, and it is as important as the next selected variable, this simple criterion would give an incorrect result. We can check the correlation or angle between the newly selected variable $Z^{(k+1)}$ and the current residual $r^{(k)} - \alpha^{(k)}p^{(k)}$ to find out how informative u is. If the correlation between $Z^{(k+1)}$ and the current residual has very small correlation, or equivalently, very large angle (close to orthogonality), $Z^{(k+1)}$ would not be very useful to explain the variation in the current residual. As we discussed at the end of Section 4.1.1, the algorithm selects the variable that has the highest correlation with the current residual at each iteration. Therefore, if $Z^{(k+1)}$ has a small correlation with the current residual, the rest of the candidates can only do worse than $Z^{(k+1)}$. Thus the algorithm can stop when the new selected variable has a very small correlation with the current residual. Also, because

$$\mathrm{Cor}(p^{(k)}, r^{(k)} - \alpha^{(k)}p^{(k)})^2 = \rho(Z^{(k+1)}, r^{(k)} - \alpha^{(k)}p^{(k)})^2,$$

we can use $\mathrm{Cor}(p^{(k)}, r^{(k)} - \alpha^{(k)}p^{(k)})^2$ instead of $\rho(Z^{(k+1)}, r^{(k)} - \alpha^{(k)}p^{(k)})^2$ to measure the correlation between new variable and the current residual to avoid the heavy calculation of functional canonical correlation. Denote *entering correlation* as the correlation between the current direction $u^{(k)}$ and the current residual $r^{(k)} - \alpha^{(k)}u^{(k)}$ when the new variable $Z^{(k+1)}$ is selected. From this correlation, we can also get the corresponding *entering angle* by

$$\text{entering angle} \;=\; \arccos(\text{entering correlation}).$$

Thus we have two ways to find the stopping point: one is to stop when the distance to go is very small; the other one is to stop when the *entering correlation* is too small or the *entering angle* is too close to orthogonality. At the k-th iteration, if we observe that α is smaller than the threshold, or that the *entering correlation* is smaller than the threshold, or that the *entering angle* is larger than the threshold, we should stop at iteration k.

However, we need to define the terms 'too small' and 'too big' in a numerical sense. It is difficult to define thresholds even for simulated data set. On the other hand, our experience shows that the values of α would have an abrupt drop right at the stopping point. In the

simulated data, the value of α at the stopping point could be dropped to less than 1% of the maximum value of α. Similar phenomenon happens to the *entering correlation* and *entering angle*, but to a less significant degree than α. This is due to the value of the correlation and the angle being bounded between [-1,1], and [0,90], respectively.

Both of the ways we defined above have no optimal points, and the thresholds are difficult to determine. To overcome this problem, we combine both the entering correlation or angle and the corresponding distance $\alpha^{(k)}$ to get a new stopping rule. When the algorithm reaches the stopping point, all the relevant variable should be selected, thus the next variable should have small correlation with the current residual. At that point, the new direction vector still needs a large distance to move. So α in that iteration is still a large number. Thus we can have a large angle and a large distance at the stopping point.

The new stopping rule is defined as:

$$AD^{(k)} = entering\ angle^{(k)} \times \alpha^{(k-1)}$$

where AD is short for 'angle times distance'. The algorithm should stop at iteration k if AD reaches the maximum point at iteration k.

To sum up, we propose two stopping rules here. Firstly, the algorithm should stop when the distance to move α is smaller than a threshold. Secondly, the algorithm should stop when AD reaches the maximum. These stopping rules can be used without any information about the error variance. Thus it can be used when the number of variables is vary large.

4.5 Simulation study

4.5.1 The true model and data generation

The true model uses three functional variables:

$$y = \mu + \int x_1(t)\beta_1(t)dt + \int x_2(t)\beta_2(t)dt + \int x_3(t)\beta_3(t)dt + \epsilon \tag{4.17}$$

where μ is the intercept; $\beta_j(t)$ are functional coefficients; $x_j(t)$ are functional variables and ϵ is the noise follows a normal distribution with mean 0 and standard deviation 0.05. The variance of the noise is adjusted such that the ratio between it and variance of the response

variable is around 1:30.

We represent the functional variables by discrete data points generated on 100 equally spaced time points within [0, 1]. The i-th sample of the functional variable $x_j(t)$ is generated from:

$$x_{i,j}(t) = \mu_j(t) + e_{i,j}(t) + \varepsilon_{i,j}(t)$$

where $\mu_j(t)$ is an overall shape, and it is the same for all $i \in 1, \ldots, n$. The term $e_{i,j}(t)$ is a functional object obtained from the i-th sample of a 10 dimensional random vector $\nu_{i,j}$ by smoothing using basis functions, where $\nu_{i,j}$ is the i-th realization from a Gaussian process:

$$\nu_{i,j} \sim GP\,(0, \kappa(.,.;\boldsymbol{\theta}))\,.$$

Our kernel function κ is the sum of a squared exponential kernel and a linear kernel:

$$\kappa(\nu_i, \nu_{i*}; \boldsymbol{\theta}) = v \exp\left(w(\nu_i - \nu_{i*})^2\right) + a\nu_i\nu_{i*} + \sigma^2 I_{i,i*},$$

where $\boldsymbol{\theta} = (v, w, a, \sigma) = (20, 50, 2, 0.5)$, and $I_{i,i*} = 1$ if $i = i*$, and 0 otherwise. Squared exponential kernel gives smooth curves while linear kernel adds a small amount of roughness to the curves. The term $\varepsilon_{i,j}(t)$ is the functional error. In the literature, most of the errors are independent, but here we consider more difficult situation where the error term is also functional. The errors are also generated from normal distributions.

To help us understand the result of simulation, we want to control the correlation between any two functional variables. This is done by controlling the canonical correlation between random vectors ν_j^0, such that largest canonical correlation between any two random vectors is almost zero. In our simulation, we generate eight random vector ν_j^0, $j = 1, \ldots, 8$. Set $\nu_1 = \nu_1^0$. For $j > 1$, we perform the linear regression:

$$\nu_j^0 = \tau_0 + \sum_{j^*<j} \boldsymbol{\tau}_{j^*}\nu_{j^*}^0 + \epsilon_j$$

and set $\nu_j = \epsilon_j$ to remove the correlations. We use this method to generate linearly uncorrelated functional variables.

As an illustrative example, one set of functional variables and the corresponding functional coefficients are shown in Figure 4.2 and Figure 4.3, respectively. Figure 4.3 contains three functional coefficients for $x_1(t)$, $x_2(t)$ and $x_3(t)$, respectively. The other functional coefficients are all zero.

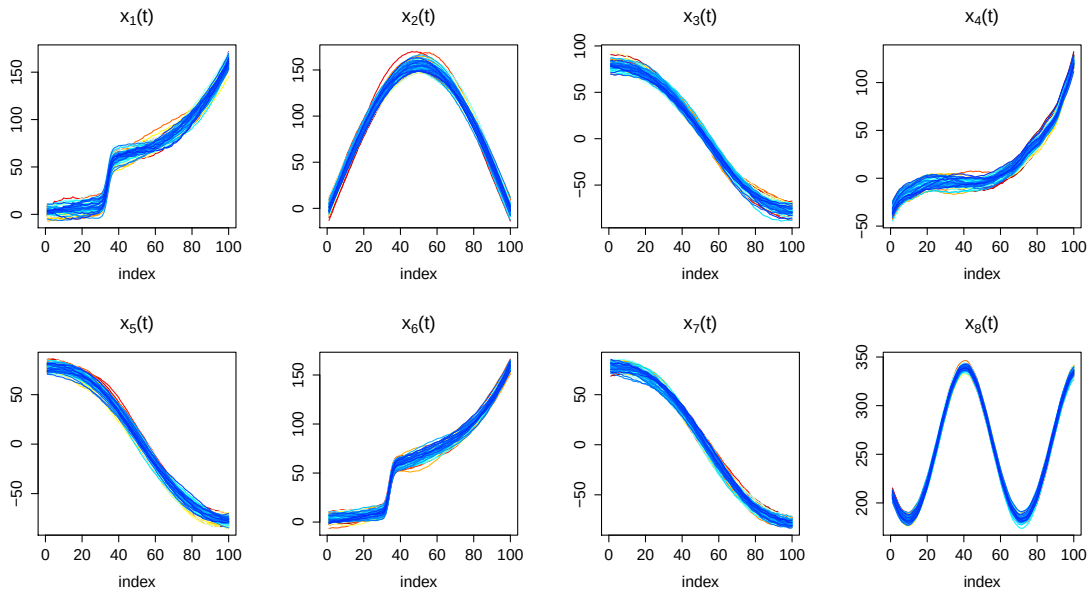


Figure 4.2: Eight linearly uncorrelated functional variables

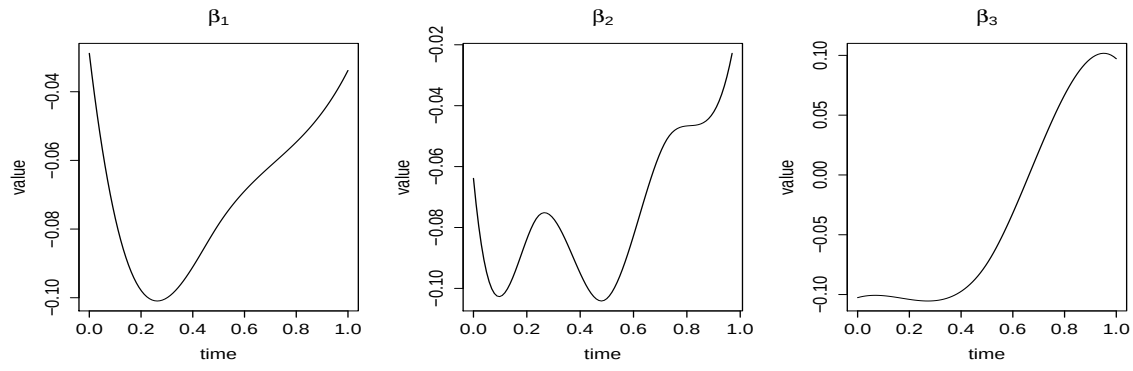


Figure 4.3: The true values of the functional coefficients

We will consider five scenarios in the simulation study. Each of the scenarios are repeated 1000 times. In each of the replications, we generate 120 samples. The 80 of them are used as training data, and the remaining 40 are used as test data.

Different scenarios have different correlation structures. The details are listed below.

1. Eight functional variables are generated, namely $x_1(t)$, $x_2(t)$, ..., $x_8(t)$. The maximum functional canonical correlation between any two of the functional variables are almost zero.

2. Similar to Scenario 1, eight uncorrelated functional variables are generated, namely $x_1(t)$, $x_2(t)$, ..., $x_8(t)$. We replace $x_4(t)$ by combining $x_1(t)$ and $x_4(t)$ such that the correlation between the first and fourth functional variables is quite large, measured by the maximum functional canonical correlation.

3. Similar to Scenario 2, we first generate eight uncorrelated functional variables $x_1(t)$, $x_2(t)$, ..., $x_8(t)$. Then we replace $x_4(t)$, $x_5(t)$ and $x_6(t)$, such that they are correlated with $x_1(t)$, $x_2(t)$ and $x_3(t)$ respectively.

4. Similar to Scenario 2, we generate eight uncorrelated functional variables $x_1(t)$, $x_2(t)$, ..., $x_8(t)$. Then we replace $x_4(t)$ and $x_5(t)$, such that they are both correlated with $x_1(t)$.

5. 100 functional variables are generated without any constraints on the correlation between any pairs of the variables. The first three variables are still the true variables, but the correlation between them are no longer as small as previous scenarios.

In each scenario, we will report the order of variables entering the regression equation, RMSE of prediction, and find the model chosen by using different stopping rules.

4.5.2 Scenario 1

In this scenario, we have eight linearly uncorrelated functional variables $x_1(t)$, $x_2(t)$, ..., $x_8(t)$. Figure 4.4 shows the correlation map between functional variables and the response from the example data set shown in Figure 4.2 and Figure 4.3. The last column shows the correlation between functional covariates and the response. The correlation between the functional variables are almost zero.

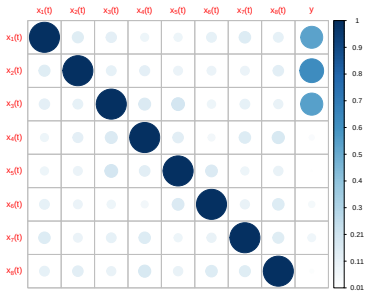


Figure 4.4: Correlation map of the example data set from Scenario 1. The dark blue dots on the diagonal represent the correlations between the functional variables and themselves, which are always 1.

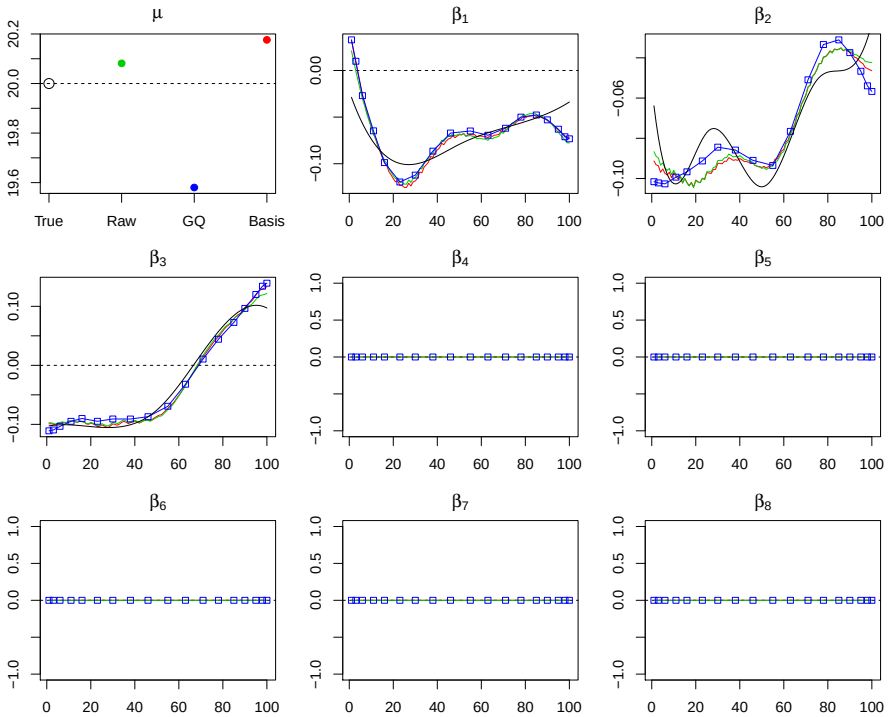


Figure 4.5: Estimated parameters at the third iteration for Scenario 1. Black lines are the true functional coefficients; green lines are the estimates of the functional coefficients using the RDP method; blue squares are the point estimates of those using the GQ method; red lines are the estimates from the BF method. The colour of the lines and points remains the same meaning through out all the simulations.

We use three different discrete methods for the functional coefficient discussed in Chapter 3, i.e. RDP, GQ and BF methods. The estimates of the intercept and the functional coefficients at the third and eighth iterations are drawn in Figure 4.5 and Figure 4.6 for the example data set, together with the true value of the intercept and the true values of the functional coefficients.

Note that the we use 18-point rule for Gaussian quadrature here. Thus only 18 points on the estimated functional coefficients are returned. Also these 18 points are not equally spaced.

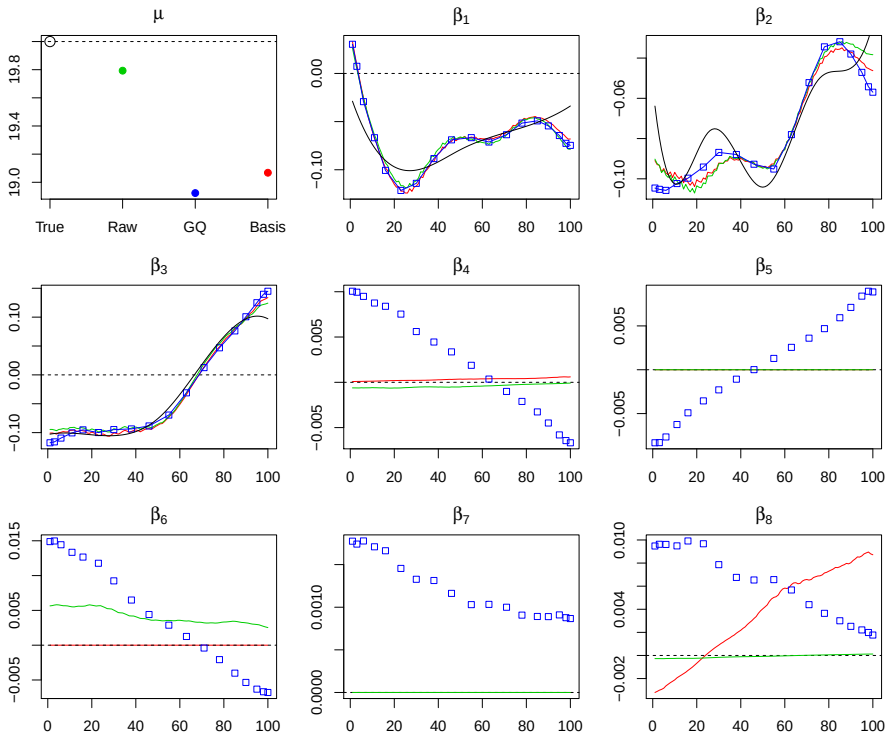


Figure 4.6: Estimated parameters at the eighth iteration for Scenario 1. Black lines are the true functional coefficients; green lines are the estimates of the functional coefficients using RDP method; blue squares are the point estimates of the ones using GQ method; red lines are the estimates from BF method.

Figure 4.5 shows the estimated parameters at iteration three. It also shows that only the parameters of the relevant variables are non-zero, which means that the relevant variables are successfully selected before any other irrelevant variables. Figure 4.6 shows that at the end of the algorithm, the irrelevant variables might be selected, but with very small coefficients.

The estimates of the intercept from different discrete methods are all close to the true

value. The estimates of functional coefficients are also accurate in general. To make sure the selected irrelevant variables have little effects to the regression, we compare the contribution of each functional variables, by using the R^2 coefficient. More specifically, we calculate $p_j = \int x_j(t)\beta_j(t)dt$, and calculate the R^2 coefficients using each p_j with respect to y. We list the results of both training and testing from iteration three and eight in Table 4.1. Clearly, the contributions of the irrelevant variables are little. This is further confirmed by the results in Table 4.3.

	RDP		GQ		BF	
Iteration	3	8	3	8	3	8
Training	0.2892	0.2892	0.2920	0.2920	0.2871	0.2873
	0.3869	0.3862	0.3909	0.3914	0.3859	0.3859
	0.3015	0.3009	0.3001	0.2995	0.3000	0.2994
	0	0.0005	0	0.0022	0	0
	0	0	0	-0.0008	0	0
	0	0.0073	0	0.0047	0	0
	0	-0.0049	0	-0.0012	0	0
	0	0.0007	0	-0.0028	0	-0.0014
Testing	0.3944	0.3969	0.3920	0.3932	0.4015	0.4007
	0.3077	0.3103	0.3142	0.3132	0.3111	0.3114
	0.2657	0.2662	0.2740	0.2755	0.2684	0.2689
	0	-0.0051	0	-0.0058	0	0.0001
	0	0.0026	0	0.0042	0	0
	0	-0.0156	0	-0.0091	0	0
	0	0.0069	0	0.0017	0	0
	0	-0.0007	0	0.0089	0	0.0045

Table 4.1: Contribution of each variable in iteration three and eight.

Table 4.3 shows more numerical results. We list the explanation of the notation used in Table 4.3 in Table 4.2. Here we clarify that when a stopping criterion suggests that the algorithm should stop at iteration m, or after the m-th variable enters the regression equitation, it actually means that there are $(m-1)$ variables in the regression equation. This is because the last variable selected is only used for finding the distance to move for the $(m-1)$-th selection.

The first part of Table 4.3 shows the result by using RDP method for functional coefficients. The relevant functional variables are selected before the others. Recall that the true standard deviation of the error is 0.05. Thus the prediction RMSE in the table represents accurate prediction after the fourth functional variable enters the regression equation. After the third iteration, some irrelevant variables are selected into the model. However

selection	The index of the variables selected in each iteration or '-' if no variable is selected.
RMSE	The RMSE of the prediction after each iteration.
entering correlation	The correlation between the current direction and current residual when the new variable joins the selected set of variables. Because the current direction in the first iteration is $\mathbf{0}$, the first value of this summary is invalid.
AD	The correlation between the current direction and current residual when the new variable joins the selected set of variables over the distance that the current direction goes. The first value of this measurement is also not valid.
C_p	Mallow's C_p using the estimated variance from error. The estimated variance is calculated from OLS using all variables.
AIC	AIC using the estimated variance from error.
BIC	BIC using the estimated variance from error.
R^2	R^2 coefficient at each iteration.
Adj R^2	Adjusted R^2 coefficient at each iteration.

Table 4.2: Meanings of the notations used in Table 4.3

the contributions they made to the prediction are negligible, since the prediction RMSE remains almost unchanged. This confirms the finding from Table 4.1. The entering correlation reduces across all iterations, and the entering angle increases across all iterations. The parameter α reduces to a very small value after the fourth variable joins the regression equation. Although we do not have a criterion to find the threshold, the rapid decreasing in α after the third iteration seems a good indicator that the algorithm should stop at the fourth iteration. At iteration four, AD reaches a maximum, which also indicates that the algorithm should stop at the fourth iteration. Both α and AD suggest that the number of variables in the regression is three. C_p, AIC and BIC using estimated error variance reach their optimal value at the end of the iterations. As we know, the first three selections cover the true variables, but the actual distance to move for theses three functional variables is decided by the fourth selection. So we should stop at the fourth iteration. Therefore α and AD give an accurate stopping point in this case.

The second part of this table shows the result from the algorithm using the GQ method for functional coefficients. Similar to the previous part of the table, the relevant variables

	Iteration	1	2	3	4	5	6	7	8
RDP	selection	2	3	1	6	8	4	5	7
	RMSE	0.320	0.273	0.062	0.062	0.061	0.064	0.067	0.073
	entering cor	-	0.565	0.682	0.255	0.237	0.162	0.157	0.074
	entering angle(°)	-	55.583	46.961	75.218	76.292	80.673	80.969	85.771
	α	0.067	0.227	0.746	0.017	0.028	0.017	0.021	0.017
	AD	-	3.735	10.670	56.075	1.316	2.253	1.370	1.844
	C_p	336.001	264.279	6.630	7.085	7.260	7.237	7.223	7.206
	AIC	198.798	162.936	34.112	34.340	34.427	34.415	34.409	34.400
	BIC	733.399	664.091	432.129	432.599	432.808	432.787	432.776	432.755
	R^2	0.115	0.401	0.980	0.980	0.980	0.980	0.980	0.980
	Adj R^2	0.112	0.392	0.976	0.976	0.976	0.976	0.976	0.976
GQ	selection	2	1	3	6	4	5	7	8
	RMSE	0.320	0.265	0.062	0.062	0.061	0.065	0.080	0.066
	entering cor	-	0.561	0.679	0.260	0.223	0.076	0.045	-0.388
	entering angle(°)	-	55.904	47.211	74.927	77.135	85.659	87.448	67.146
	α	0.059	0.218	0.759	0.011	0.031	0.002	0.071	0.062
	AD	-	3.277	10.300	56.861	0.828	2.613	0.194	4.753
	C_p	336.562	266.793	7.625	6.595	4.476	3.662	13.239	5.449
	AIC	199.078	164.194	34.609	34.095	33.035	32.628	37.416	33.522
	BIC	733.941	666.296	433.871	432.913	431.094	430.600	441.941	435.948
	R^2	0.113	0.392	0.980	0.980	0.982	0.983	0.977	0.983
	Adj R^2	0.110	0.383	0.976	0.977	0.978	0.979	0.973	0.980
BF	selection	2	3	1	8	4	-	-	-
	RMSE	0.320	0.273	0.062	0.062	0.062	0.062	0.062	0.062
	entering cor	-	0.570	0.676	0.139	0.001	0.047	0.000	0.379
	entering angle(°)	-	55.253	47.499	81.998	89.958	87.326	89.994	67.753
	α	0.051	0.239	0.759	0.001	0.000	0.031	-0.060	-0.026
	AD	-	2.807	11.332	62.274	0.091	0.012	2.825	4.052
	C_p	336.057	270.086	7.412	8.469	8.338	8.316	8.309	8.289
	AIC	198.826	165.840	34.503	35.031	34.966	34.955	34.952	34.942
	BIC	733.463	670.173	434.215	435.326	435.223	435.211	435.232	435.222
	True BIC	2790.232	2380.919	749.950	755.893	758.090	739.464	803.165	757.531
	R^2	0.115	0.380	0.980	0.980	0.980	0.980	0.980	0.980
	Adj R^2	0.082	0.349	0.974	0.974	0.974	0.976	0.968	0.975

Table 4.3: Summaries of the functional LARS for the first scenario using the example data set. The functional coefficients are represented by using BF, GQ and RDP. In each type of discrete methods, the selected variable, prediction RMSE, and different measures of stopping rules in each iteration are presented.

No. of variables	RDP			GQ			BF		
in the regression	$\log(\lambda_1)$	d.f.	cor	$\log(\lambda_1)$	d.f.	cor	$\log(\lambda_1)$	d.f.	cor
1	-9.7476	0.2683	0.6294	-2.1798	0.2606	0.6337	-16.7935	0.2720	0.6298
2	-8.4428	1.2822	0.7960	0.1689	1.1523	0.7975	-17.0545	1.3977	0.7878
3	-19.9250	12.0659	0.9833	-11.8353	12.3793	0.9840	-26.1881	12.6134	0.9838
4	-1.1359	12.0719	0.0546	4.6052	12.4092	0.2968	0.6908	12.6360	0.0288
5	4.6052	12.0864	0.1088	4.6052	12.5356	0.3015	2.5175	12.6475	0.0697
6	-1.9188	12.0873	0.0282	4.6052	12.6699	0.1973	2.2565	12.6519	0.0312
7	-2.1798	12.0883	0.0223	4.6052	13.4106	0.0821	2.2565	12.6635	0.0307
8	-2.1798	12.0867	0.0211	3.8223	14.1648	0.5103	2.2565	12.6676	0.0301

Table 4.4: The values of tuning parameter on log scale, degrees of freedom, and the corresponding correlation in the regression model with different number of variables. λ_1 controls the smoothness; $d.f.$ is the degrees of freedom up to the corresponding iteration; *cor* is the correlation between the selected variables and the current response variable.

are selected first. The RMSE also indicate accurate prediction. AD reaches its maximum value at iteration four, and α has a sudden drop at iteration four. Both of them indicate that the algorithm should stop there. All other criteria fail to find their optimal points before that last selection.

The result in the third part of the table is obtained by using the BF method for the functional coefficients. The true variables are selected in the first three iterations. The algorithm fails to select variables after iteration five. Only AD and α successfully found the correct stopping point. All other criteria failed to find the stopping point.

As discussed in Chapter 4.4, here we present the details of the change of tuning parameters in log scale, degrees of freedom, correlation between the selected variables and the current response variable in Table 4.4. Since the first three selections cover the true variables, the correlation is expected to be large. When the number of variables increasing, the correlation between the selected variables and the current response variable drops and stays low. The corresponding tuning parameter, however, keeps increasing. The corresponding degrees of freedom barely change when the tuning parameters are large. The stopping rules using this definition of degrees of freedom, such as C_p from Eqn (4.15), certainly cannot get enough penalty for the normalized squared error. Thus it would be difficult to find optimal stopping points using the traditional stopping rules.

Summary of the repeated runs

Table 4.5 shows the summary of the prediction RMSE's from 1000 replications. The average and the standard deviation of prediction RMSE's after the m-th covariates enters the

regression equation are given. The prediction RMSE reaches an optimal value when the fourth variable joins the regression equation (i.e. three functional variables are selected) using any of the three discrete methods. This matches the fact that the true model includes three functional variables.

Table 4.6 summarises the percentage of successfully selected stopping points using different criteria. The correct stopping point is after the fourth covariate enters the regression equation according to the prediction RMSE in Table 4.5. The traditional stopping rules seem unlikely to find the best stopping point, while AD is much more promising. By using AD, the probabilities of successfully selecting the stopping points are almost 1 for different discrete methods applied on functional coefficients.

No. of variables	RDP		GQ		BF	
in the regression	mean	sd	mean	sd	mean	sd
1	0.3321	0.0356	0.3333	0.0355	0.3331	0.0356
2	0.2667	0.0321	0.2680	0.0323	0.2673	0.0325
3	0.0577	0.0072	0.0610	0.0125	0.0565	0.0107
4	0.0570	0.0070	0.0605	0.0117	0.0569	0.0085
5	0.0573	0.0071	0.0610	0.0120	0.0575	0.0073
6	0.0586	0.0073	0.0625	0.0145	0.0582	0.0075
7	0.0611	0.0082	0.0656	0.0150	0.0583	0.0076
8	0.0636	0.0083	0.0634	0.0093	0.0583	0.0075

Table 4.5: Summary of prediction RMSE's after the m-th covariates enters the regression equation (the first column).

	RDP	GQ	BF
α	99.46	99.00	98.91
AD	100.00	99.64	99.64
C_p	0.00	1.36	15.04
AIC	0.00	1.36	15.31
BIC	0.00	2.90	15.04
R^2	0.00	0.18	14.67
Adj R^2	0.00	0.27	15.22

Table 4.6: The percentage of different stopping criteria successfully finding the stopping point.

4.5.3 Scenario 2

The difference between Scenario 2 and Scenario 1 is that we replace one of the irrelevant variable $x_4(t)$ such that the correlation between $x_4(t)$ and $x_1(t)$ is around 0.9. This also indicates that the correlation between $x_4(t)$ and y is large. The aim of this scenario is to see if the algorithm can still do the selection and estimation when one of the irrelevant variable is correlated with the response as well as one of the relevant variables. The data set we use here is generated from the one we showed in Scenario 1. Therefore the response variable, true intercept and functional coefficients remain the same.

Figure 4.7 shows the correlation map of the data set. The correlation between $x_1(t)$ and $x_4(t)$ is very high. Also the correlation between the response variable y and $x_4(t)$ is quite high. Figure 4.8 and Figure 4.9 show the estimated parameters at iterations three and eight. The plot from iteration three gives very good estimations to the true parameters. It also indicates that the first three selections cover the three true variables. Figure 4.9 shows that at the end of the algorithm, the irrelevant variables might be selected, but with very small coefficients.

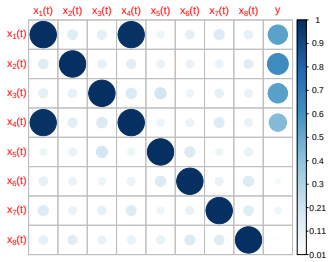


Figure 4.7: Correlation map of example data set from Scenario 2.

Summary of the repeated runs

We report the summary of the results from 1000 replications. Table 4.7 shows the prediction RMSE's after the m-th covariate joins the regression. The prediction RMSE's using different discrete methods reach their optimal values when the fourth variable joins the regression equation. This suggests that the algorithm should stop after the fourth variable is selected. Therefore the algorithm can give accurate prediction in the case when the variables are correlated. In the 1000 replications, 97.1% of the runs successfully selected the

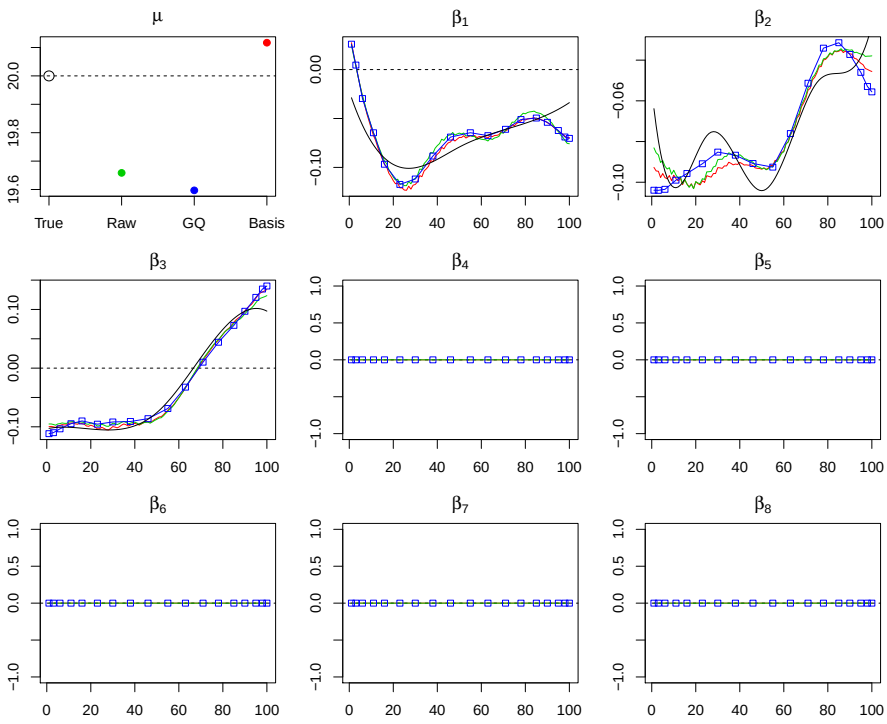


Figure 4.8: Estimated parameters at the third iteration for Scenario 2. Black lines are the true functional coefficients; green lines are the estimations of the functional coefficients using the RDP method; blue squares are the point estimations of the ones using the GQ method; red lines are the estimation from the BF method.

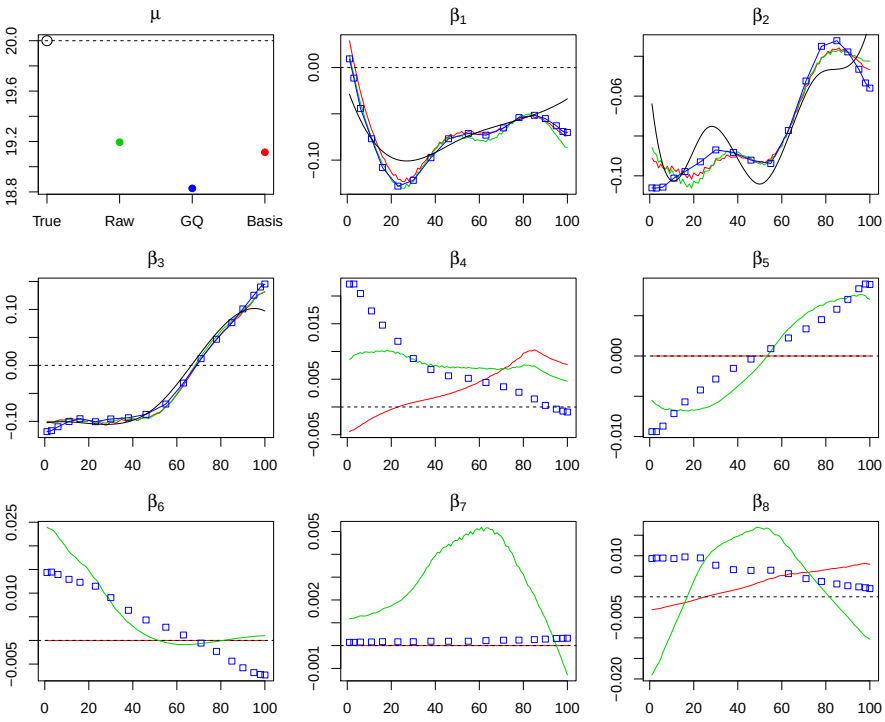


Figure 4.9: Estimated parameters at the eighth iteration for Scenario 2. Black lines are the true functional coefficients; green lines are the estimations of the functional coefficients using the RDP method; blue squares are the point estimations of the ones using the GQ method; red lines are the estimation from the BF method.

true variables in the first three iterations using the RDP method to represent the functional coefficients. The percentage for using the GQ method and the BF method are 94.6% and 95.4%, respectively.

Number of	RDP		GQ		BF	
selected covariates	mean	sd	mean	sd	mean	sd
1	0.3337	0.0363	0.3351	0.0364	0.3346	0.0363
2	0.2681	0.0332	0.2700	0.0335	0.2691	0.0340
3	0.0585	0.0082	0.0629	0.0172	0.0582	0.0148
4	0.0576	0.0073	0.0607	0.0083	0.0575	0.0072
5	0.0579	0.0072	0.0610	0.0083	0.0580	0.0073
6	0.0591	0.0074	0.0623	0.0086	0.0586	0.0073
7	0.0617	0.0083	0.0652	0.0093	0.0588	0.0073
8	0.0637	0.0082	0.0635	0.0087	0.0587	0.0073

Table 4.7: Summary of prediction for RMSE's after the m-th covariates enters the regression equation (the first column).

Table 4.8 shows the percentage of successfully selected stopping points using different criteria from 1000 replications. The correct stopping point is after the fourth covariate enters the regression equation. The conventional stopping rules seem unlikely to find the best stopping point, and the results are similar to those from scenario 1. By using AD and α, the successful selection rates are very high for all three methods. The difference to Table 4.6 in Scenario 1 is ignorable, meaning that the new stopping rules perform very well even when some redundant variables are correlated to the relevant variables.

	RDP	GQ	BF
α	98.82	97.18	96.27
AD	100.00	99.27	99.54
C_p	0.00	0.27	12.57
AIC	0.00	0.27	12.75
BIC	0.09	2.00	12.57
R^2	0.00	0.00	12.39
Adj R^2	0.00	0.00	12.20

Table 4.8: The percentage of different stopping criteria successfully finds the stopping point.

4.5.4 Scenario 3

We further introduce correlations to the data set in Scenario 3. We replace the irrelevant variables $x_4(t)$, $x_5(t)$ and $x_6(t)$ such that $x_1(t)$, $x_2(t)$ and $x_3(t)$ are correlated with $x_4(t)$,

$x_5(t)$ and $x_6(t)$, respectively. This also means that the correlations between $x_4(t)$ $x_5(t)$ and $x_6(t)$ and the response variable are all around 0.9. The aim of this scenario is to see if the algorithm can do the selection and estimation when all the relevant variables are correlated with other variables. We show the estimation with respect to the data set which is generated from that used in Scenario 1. Thus the response variable, true intercept and functional coefficients remain the same as before.

Figure 4.10 shows the correlation map of the data set. Clearly the response variable is correlated with the first six variables. Figure 4.A.1 and Figure 4.A.2 in Appendix 4.A show the results for one replication, which are very similar to those in Scenario 1 and 2.

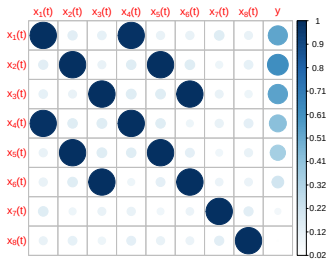


Figure 4.10: Correlation map of example data set from Scenario 3.

We now report the summary of the results from 1000 replications. Table 4.9 shows the prediction RMSE after the m-th covariate joins the regression. The prediction RMSE's using different discrete methods reach their optimal values when the fourth variable joins the regression equation. This suggests that the algorithm should stop after the fourth variable is selected. Therefore the algorithm can give accurate prediction in the case when the variables are correlated. In the 1000 replications, the percentages of successfully selecting the true variables in the first three selection using the RDP, GQ and BF methods are 93.6%, 90.1% and 92.0%, respectively. Even though these percentages drop from previous two scenarios, the corresponding prediction RMSE's are still good.

Table 4.10 shows the percentage of successfully selecting the right stopping points using different criteria from 1000 replications. The percentage of successful selection from different methods and stopping rules are shown in Table 4.9. It shows a similar result to Table 4.5 and Table 4.7 from Scenario 1 and 2, respectively. The new stopping rules are still performing well in this scenario when there are more correlated irrelevant variables.

Number of	RDP		GQ		BF	
selected covariates	mean	sd	mean	sd	mean	sd
1	0.3321	0.0357	0.3337	0.0357	0.3330	0.0358
2	0.2667	0.0335	0.2690	0.0345	0.2677	0.0336
3	0.0588	0.0099	0.0661	0.0279	0.0597	0.0146
4	0.0575	0.0080	0.0615	0.0098	0.0581	0.0085
5	0.0578	0.0077	0.0616	0.0096	0.0583	0.0079
6	0.0592	0.0080	0.0626	0.0098	0.0586	0.0079
7	0.0613	0.0083	0.0647	0.0104	0.0587	0.0080
8	0.0626	0.0082	0.0638	0.0098	0.0588	0.0079

Table 4.9: Summary of prediction RMSE's after the m-th covariates enters the regression equation (the first column).

	RDP	GQ	BF
α	97.90	94.03	93.94
AD	99.92	98.57	99.66
C_p	0.00	0.25	10.77
AIC	0.00	0.25	11.19
BIC	0.17	0.93	10.77
R^2	0.00	0.00	10.68
Adj R^2	0.00	0.00	10.68

Table 4.10: The percentage of successes in finding the stopping point for different criteria.

4.5.5 Scenario 4

Scenario 4 has two irrelevant variables correlated with one relevant variable. We replace the irrelevant variables $x_4(t)$ and $x_5(t)$ such that both of them are correlated with $x_1(t)$. This also means that the correlations between $x_4(t)$ or $x_5(t)$ and the response variable could be large. We want to see if the algorithm can do the selection and estimation when one relevant variable is correlated with more than one irrelevant variables. Similar to previous scenarios, we show the estimation from the data set generated from that used in Scenario 1, such that the response variable, true intercept and functional coefficients remain the same.

Figure 4.11 shows the correlation map of the data set. Now the response variable is correlated with the first five variables, and the first functional variable is highly correlated with the fourth and fifth variables. The estimates at iterations three and eight for one replication are shown in Figure 4.B.1 and Figure 4.B.2 in Appendix 4.B. Both plots show good results similar to those from previous scenarios.

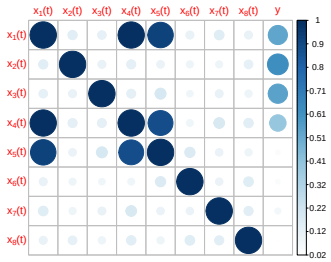


Figure 4.11: Correlation map of example data set from Scenario 4.

Hence, we report the summary of the results from 1000 replications. Table 4.11 shows the prediction RMSE after the m-th covariate joins the regression. The prediction RMSE's using different discrete methods reach their optimal values when the fourth variable joins the regression equation. This suggests that the algorithm should stop after the fourth variable is selected. The algorithm gives accurate prediction in the case when the variables are correlated. In the 1000 replications, the percentages of successfully selecting the true variables in the first three selection using the RDP, GQ and BF methods are 100%, 99.81% and 99.9%, respectively.

Number of	RDP		GQ		BF	
selected covariates	mean	sd	mean	sd	mean	sd
1	0.3319	0.0360	0.3333	0.0361	0.3328	0.0361
2	0.2660	0.0326	0.2679	0.0333	0.2666	0.0329
3	0.0579	0.0072	0.0633	0.0211	0.0577	0.0132
4	0.0571	0.0069	0.0608	0.0106	0.0574	0.0092
5	0.0575	0.0069	0.0610	0.0108	0.0578	0.0085
6	0.0588	0.0072	0.0624	0.0112	0.0582	0.0087
7	0.0612	0.0079	0.0652	0.0134	0.0586	0.0110
8	0.0635	0.0081	0.0635	0.0092	0.0585	0.0074

Table 4.11: Summary of prediction RMSE's after the m-th covariates enters the regression equation (the first column).

Table 4.12 shows the percentage of successfully selected stopping points using different criteria from 1000 replications. The correct stopping point is after the fourth covariate enters the regression equation according to the prediction RMSE in Table 4.11. As expected, the conventional stopping rules are not good at finding the best stopping point in this case. On the other hand, *AD* will almost certainly find the right stopping point.

	RDP	GQ	BF
α	99.74	98.01	97.58
AD	100.00	99.05	99.57
C_p	0.00	0.69	12.12
AIC	0.00	0.69	12.47
BIC	0.00	1.65	12.12
R^2	0.00	0.09	12.03
Adj R^2	0.00	0.17	11.95

Table 4.12: The percentage of successes in finding the stopping point for different criteria.

4.5.6 Scenario 5

In this scenario we generate 100 functional variables without any constraints on the correlation between functional variables. The true model stays the same, i.e. the response variable depends on the first three variables $x_1(t)$, $x_2(t)$ and $x_3(t)$. It is not feasible to present the correlation map for all the 100 functional variables. Thus we only present the correlation map for the first eight functional variables here in Figure 4.12. There are no large correlations between any pairs of functional variables, but the correlations are generally larger than before. Moreover, the irrelevant functional variables may be correlated with the response variables.

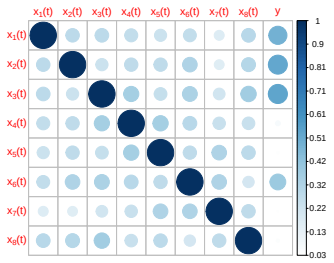


Figure 4.12: Correlation map of example data set from Scenario 5.

Figure 4.13 shows the estimated parameters at iteration three for one replication. The plot from iteration four gives reasonable estimations of the true parameters. The overall shapes of the functional coefficients are well represented, but they are not as accurate as the those estimated from the previous scenarios. This is as expected, since the correlations between variables are larger than before.

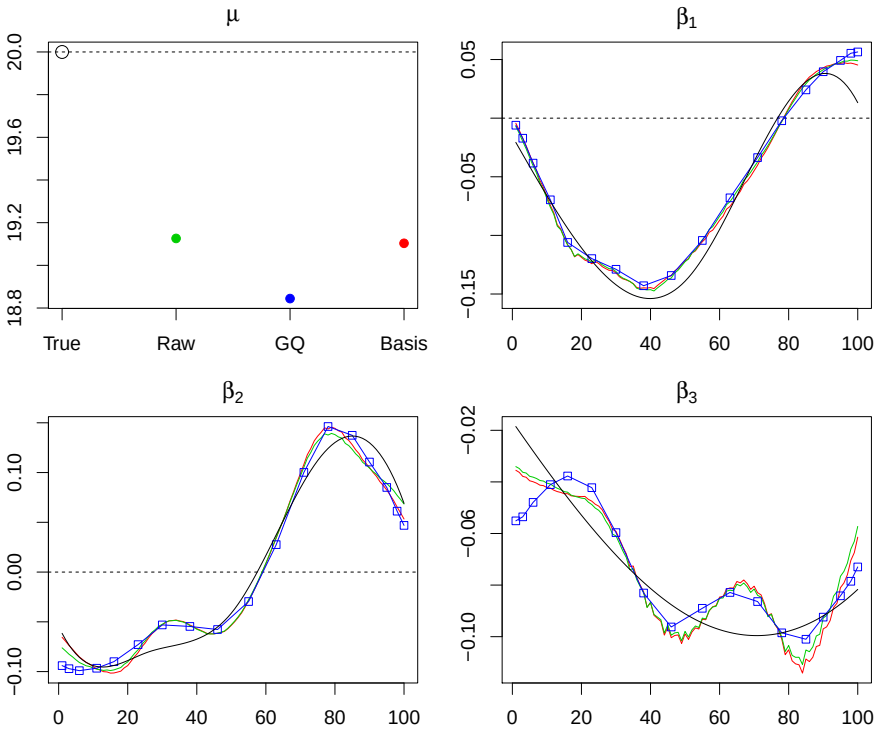


Figure 4.13: Estimated parameters at the eighth iteration for Scenario 5. Black lines are the true functional coefficients; green lines are the estimations of the functional coefficients using the RDP method; blue squares are the point estimations of the ones using the GQ method; red lines are the estimation from the BF method.

Summary of the repeated runs

Similar to the scenarios before, we run 1000 replicates in this scenario. In the 1000 replications, the percentages of successfully selecting the true variables in the first three selection using the RDP, GQ and BF methods are 99.9%, 96.9% and 98.1%, respectively. Table 4.13 shows the prediction RMSE after the m-th covariate joins the regression. Recall that the error generated in the data has standard deviation 0.05. The prediction RMSE's listed in Table 4.13 indicates good predictions, even though the estimation of the functional coefficients are not extremely accurate.

The prediction RMSE's using different discrete methods reach their optimal values at different points when using different discrete methods for functional coefficients. When using RDP method, the prediction RMSE suggests that the algorithm should stop when the fourth variable enters the regression. The other two discrete methods both lead the algorithm to stop at the sixth iteration. The prediction RMSE's near the optimal points are almost the same with each other, for example, when using RDP, the prediction RMSE at iteration 4 is the same as the value at iteration 5, but the latter one has a slightly smaller standard deviation. Even though the optimal points from prediction RMSE are not clear, we still believe the algorithm should stop after the fourth variable enters the regression equation, since the true variables are almost certainly covered in the first three selections.

Number of	RDP		GQ		BF	
selected covariates	mean	sd	mean	sd	mean	sd
1	0.3243	0.0394	0.3260	0.0395	0.3257	0.0397
2	0.2566	0.0336	0.2605	0.0359	0.2578	0.0339
3	0.0605	0.0105	0.0737	0.0423	0.0618	0.0203
4	0.0585	0.0072	0.0662	0.0194	0.0594	0.0115
5	0.0586	0.0072	0.0654	0.0170	0.0591	0.0101
6	0.0600	0.0076	0.0653	0.0168	0.0590	0.0101
7	0.0619	0.0081	0.0655	0.0162	0.0591	0.0101
8	0.0642	0.0086	0.0660	0.0168	0.0590	0.0101

Table 4.13: Summary of prediction RMSE's after the m-th covariates enters the regression equation (the first column).

Table 4.14 shows the percentage of successfully selected stopping points using different criteria from 1000 replications. The correct stopping point is after the fourth covariate enters the regression equation according to the prediction RMSE in Table 4.11. Similar to the previous scenarios, AD can almost certainly find the right stopping point, while other methods have even worse performance than those in previous scenarios. When we use

the estimated error variance in the calculation, the normalized squared error in Eqn (4.11) tends to infinity, because the estimated error variance is almost zero.

	RDP	GQ	BF
α	98.44	94.54	97.56
AD	99.90	95.22	98.15
C_p	0.00	0.00	0.00
AIC	0.00	0.00	0.00
BIC	0.00	0.00	0.00
R^2	0.00	0.00	0.00
Adj R^2	0.00	0.00	0.00

Table 4.14: The percentage of different stopping criteria successfully finds the stopping point.

4.6 Conclusions and Discussions

We propose the functional least angle regression algorithm in this chapter. It performs very well. The relevant variables are usually selected prior to the irrelevant variables. In general, the functional coefficients are estimated accurately. Accurate variable selection and parameter estimation lead to accurate prediction. The proposed algorithm is consistently good under different circumstances. This is supported by our comprehensive simulation studies, including scenarios involving many correlated or large number of irrelevant variables.

However, when there are a large number of candidate variables, the computation cost would increase. As the number of variables in the regression equation increases, the computation cost for each iteration increases exponentially. This is due to the calculation of the canonical correlation analysis in the algorithm to find the correlation between the selected variables and the current response variable. The number of selected variables increases or stays the same after each iteration. Thus the dimensionality of the calculation increases faster and faster. Therefore, even though we can calculate the full solution path, it would be better to have a good stopping rule, especially when the number of candidate variables is large, so we can stop before we have the full solution path. The conventional stopping rules as discussed in Section 4.4 require the error variance and the degrees of freedom. The estimates of these terms in functional LARS make these methods difficult to use. Because of the existence of the tuning parameters, the degrees of freedom do not necessarily increase with the number of variables in the regression equation. Thus

when the normalized squared error decreases, the penalty from the degrees of freedom might not be big enough to define an optimal point. We proposed two new stopping rules. They do not rely on either variance of the error or the degrees of freedom. One of them could have an optimal point, the other one requires a threshold. Although an accurate threshold is difficult to define, an empirical method based on the rapid decrease of the value performs extremely well in all the scenarios in the simulation study.

The tuning parameters in functional the LARS algorithm bring robustness into the estimation. If irrelevant variables are selected, the prediction would not be much worse. More specifically, even if we select irrelevant variables into the regression equation, as long as we have all the relevant variables, the prediction would not be much worse than the perfect model without any irrelevant variables. This is also supported by the simulation study.

Appendix

4.A Plots of estimated parameters for Scenario 3

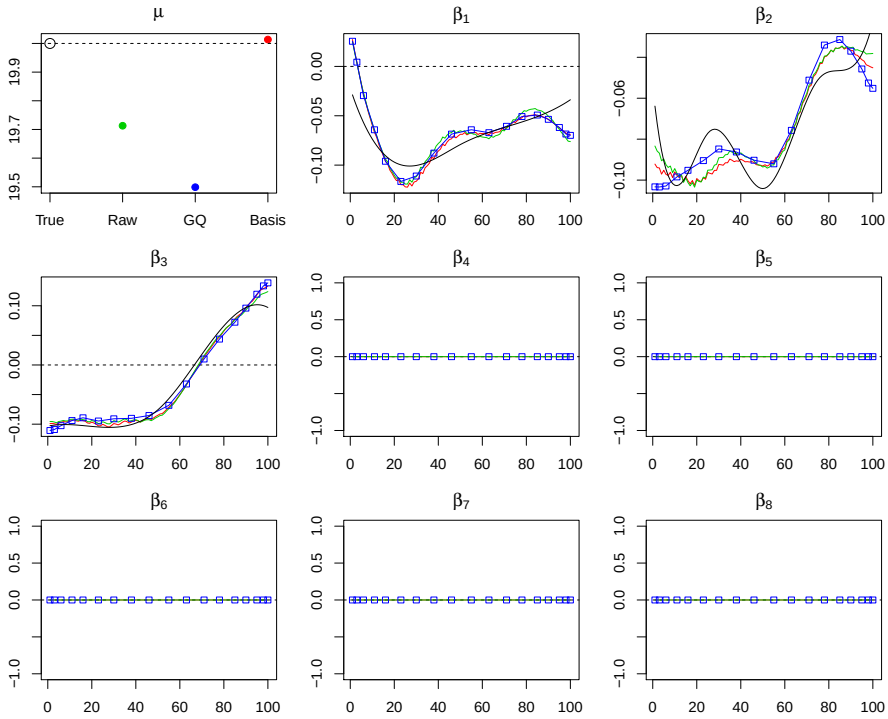


Figure 4.A.1: Estimated parameters at the third iteration for Scenario 3. Black lines are the true functional coefficients; green lines are the estimations of the functional coefficients using RDP method; blue squares are the point estimations of the ones using GQ method; red lines are the estimation from BF method.

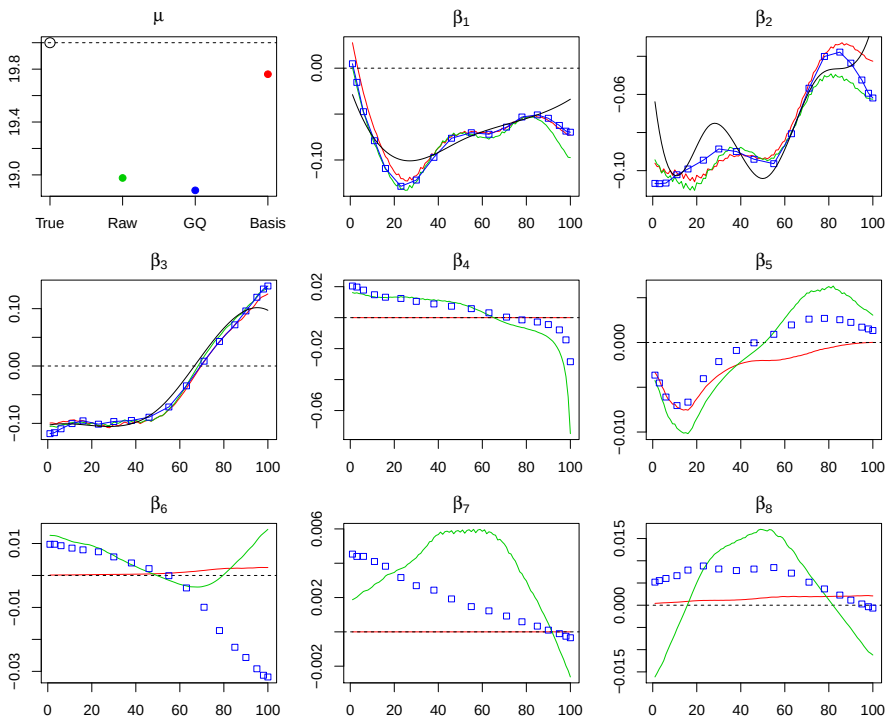


Figure 4.A.2: Estimated parameters at the eighth iteration for Scenario 3. Black lines are the true functional coefficients; green lines are the estimations of the functional coefficients using RDP method; blue squares are the point estimations of the ones using GQ method; red lines are the estimation from BF method.

4.B Plots of estimated parameters for Scenario 4

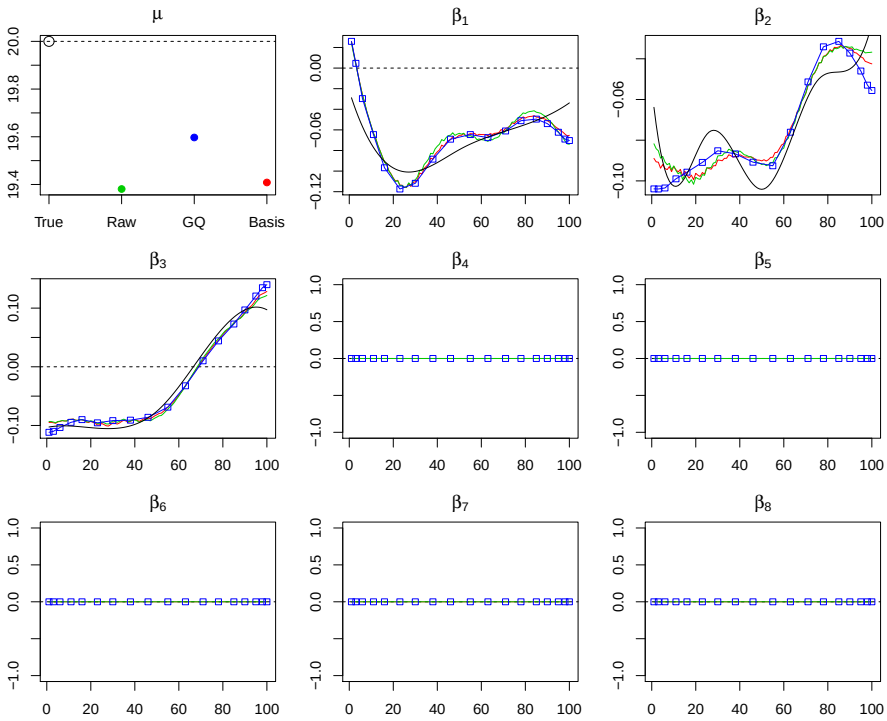


Figure 4.B.1: Estimated parameters at the third iteration for Scenario 4. Black lines are the true functional coefficients; green lines are the estimations of the functional coefficients using RDP method; blue squares are the point estimations of the ones using GQ method; red lines are the estimation from BF method.

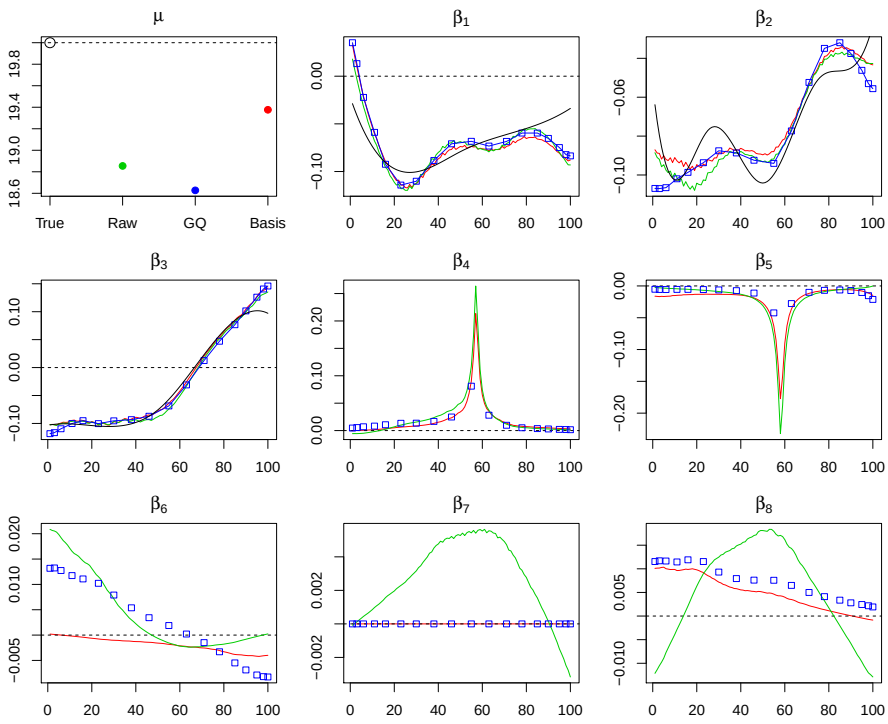


Figure 4.B.2: Estimated parameters at the eighth iteration for Scenario 4. Black lines are the true functional coefficients; green lines are the estimations of the functional coefficients using RDP method; blue squares are the point estimations of the ones using GQ method; red lines are the estimation from BF method.

Chapter 5

Selection of Mixed Scalar and Functional Variables Using Functional LARS

In the previous chapter, we discussed functional least angle regression. The model we considered there only involves functional variables in the candidates. In a more realistic situation, there would also be scalar variables in the candidates. So we want our variable selection method to be able to select both functional and scalar variables. In this chapter, we focus on the model with scalar response and a mixture of functional and scalar covariates:

$$y = \beta_0 + \sum_{j=1}^{J} \int X_j(t)\beta_j(t)dt + \sum_{m=1}^{M} z_m\gamma_m + \epsilon \tag{5.1}$$

where y is the response; β_0 is the intercept; $X_j(t)$ is the j-th functional variable with $j = 1, \ldots, J$ and J is the number of candidate functional variables; $\beta_j(t)$ is the j-th functional coefficient; z_m is the m-th scalar variable with $m = 1, \ldots, M$ and M is the number of candidate scalar variables; γ_m is the m-th scalar coefficient; ϵ is the noise follows normal distribution with mean 0 and variance σ^2. Both J and M can be large. The motion data we are going to discuss in Chapter 7 include more than 500 functional variables and more than 200 scalar variables.

We first extend the group lasso algorithm in functional variable selection to include mixed scalar and functional variable selection. We also extend the functional LARS method discussed in the previous chapter to deal with this more general problem by introducing

additional normalizations.

The extensions of the existing algorithms are introduced in section 5.1, followed by additional normalizations to the functional LARS when selecting mixed scalar and functional variables in section 5.2. A comprehensive simulation study using different algorithms for data with complex correlation structures is in section 5.3. We specifically check the performance of Modification II from Section 4.3.2 in the scenarios with the very complex correlation structures.

5.1 Selection of Mixed Scalar and Functional Variables Using Extened Group Variable Selection Methods

In the previous chapter, we treated each of the functional variables as one group of variables using any of the discrete methods. If we have both functional variables and scalar variables in the candidate, we can also treat each of the scalar variables as one group of variable with dimension one. It is, therefore, natural to use group variable selection when there are mixed functional and scalar variables in the candidate.

There are a few algorithms in the literature working with functional variable selection problems, such as Matsui and Konishib (2011); Mingotti et al. (2013); J.Gertheiss (2013). These algorithms can be thought of as the combinations of group variable selection methods and the univariate functional regression. The same idea can be used when we select from both scalar variables and functional variables in the regression, because group variable selection methods are generally flexible with regard to the dimensions of the candidate groups.

Thus there are two aspects in this type of method: the first one is the group variable selection method and the second one is the univariate functional or scalar regression. The choice of group variable selection method is not necessarily crucial. Here we use group lasso to illustrate the algorithms. We will also briefly discuss the solution of univariate functional regression in order to check the accuracy of the result from using this idea.

5.1.1 Group variable selection

The first aspect of the problem we address is the group variable selection method. We start by briefly introducing group lasso and its solution found by the shooting algorithm

(Fu (1998)). More detailed derivation, which gives us the solution of the lasso method using the shooting algorithm, is in Appendix 5.A.

Group lasso was initially proposed by Bakin et al. (1999) with a general but computationally very expensive algorithm. Later Yuan and Lin (2006) introduces some special settings into the algorithm which greatly reduced computation time. Group lasso targets on the following model:

$$y = \beta_0 + \sum_{j=1}^{J} \mathbf{x}_j \boldsymbol{\beta}_j + \epsilon, \tag{5.2}$$

where $\mathbf{x}_j$ is the j-th group variables, $\boldsymbol{\beta}_j$ is the corresponding coefficient. Without loss of generality, we assume that the response variable and the group covariates are centred, such that the mean values of the response and the group variables are zero, and thus the intercept $\beta_0 = 0$. The penalty is added to the least square equation:

$$G = (y - \sum_{j=1}^{J} \mathbf{x}_j \boldsymbol{\beta}_j)^2 + \lambda \sum_{j=1}^{J} (\boldsymbol{\beta}_j^T K_j \boldsymbol{\beta}_j)^{1/2},$$

where λ is the tuning parameter and K_j is the kernel matrix for variable j.

The shooting algorithm solves this equation iteratively. At each iteration, it solves a one dimensional regression problem with penalized least square, conditional on the values of all the other coefficients. The computation burden at each iteration leads to expensive computation cost for solving the early version of group lasso. The solution is given in Appendix 5.B.

Yuan and Lin (2006) proposed an improved version of group lasso. They assume that $\mathbf{x}_j$ is orthonormal matrix for all j, and the kernel matrices $K_j = p_j I_{p_j}$, where p_j is the dimension of the j-th group variable. Thus $\mathbf{x}_j^T \mathbf{x}_j = I_{p_j}$. These two assumptions can greatly reduce computation time. The penalized least square is now:

$$G = (y - \sum_{j=1}^{J} \mathbf{x}_j \boldsymbol{\beta}_j)^2 + \lambda \sum_{j=1}^{J} \|\boldsymbol{\beta}_j\|_2,$$

where $\|\cdot\|_2$ means the l_2 norm. The Lagrange multiplier, under this setting for the j-th

variable, is:

$$\begin{aligned}\frac{\partial G}{\partial \boldsymbol{\beta}_j} &= -\mathbf{x}_j^T r_j + \mathbf{x}_j^T \mathbf{x}_j \boldsymbol{\beta}_j + \lambda \mathbf{s}_j \\ &= -\mathbf{x}_j^T r_j + \boldsymbol{\beta}_j + \lambda \mathbf{s}_j = \mathbf{0}, \end{aligned} \tag{5.3}$$

where the response r_j is:

$$r_j = y - \sum_{j^*=1}^{J} \mathbf{x}_{j^*} \boldsymbol{\beta}_{j^*}, \quad j^* \neq j,$$

and $\mathbf{s}_j$ is:

$$\begin{cases} \|\mathbf{s}_j\|_2 \leq \sqrt{p_j} & \boldsymbol{\beta}_j = \mathbf{0}; \\ \mathbf{s}_j = \frac{\sqrt{p_j}\boldsymbol{\beta}_j}{(\boldsymbol{\beta}_j^T \boldsymbol{\beta}_j)^{1/2}} & \text{otherwise.} \end{cases}$$

When

$$\mathbf{x}_j^T r_j - \boldsymbol{\beta}_j < \lambda \sqrt{p_j},$$

$\hat{\boldsymbol{\beta}}_j = \mathbf{0}$; otherwise, the solution of this version of group lasso is obtained by:

$$\begin{aligned}\boldsymbol{\beta}_j &= \left(1 + \frac{\lambda \sqrt{p_j}}{\|\boldsymbol{\beta}_j\|_2}\right)^{-1} \mathbf{x}_j^T r_j \\ &= \left(1 - \frac{\lambda \sqrt{p_j}}{\lambda \sqrt{p_j} + \|\boldsymbol{\beta}_j\|_2}\right) \mathbf{x}_j^T r_j. \end{aligned} \tag{5.4}$$

From Eqn (5.3), we can get:

$$\mathbf{x}_j^T r_j = \frac{\boldsymbol{\beta}_j \lambda (\sqrt{p_j} + \|\boldsymbol{\beta}_j\|_2)}{\|\boldsymbol{\beta}_j\|_2}.$$

By taking l_2 norm of both sides of the equation, we have:

$$\begin{aligned}\|\mathbf{x}_j^T r_j\|_2 &= \left\| \frac{\boldsymbol{\beta}_j (\lambda \sqrt{p_j} + \|\boldsymbol{\beta}_j\|_2)}{\|\boldsymbol{\beta}_j\|_2} \right\|_2 \\ &= \frac{\|\boldsymbol{\beta}_j\|_2 (\lambda \sqrt{p_j} + \|\boldsymbol{\beta}_j\|_2)}{\|\boldsymbol{\beta}_j\|_2} \\ &= (\lambda \sqrt{p_j} + \|\boldsymbol{\beta}_j\|_2)\end{aligned}$$

Thus Eqn (5.4) can be simplified as:

$$\hat{\boldsymbol{\beta}}_j = \left(1 - \frac{\lambda\sqrt{p_j}}{\|\mathbf{x}_j^T r_j\|_2}\right)^+ \mathbf{x}_j^T r_j.$$

The above solutions are obtained under the condition that the candidate groups of variables are orthonormal. In general, the groups of variables can be transferred into orthonormal matrices by QR decomposition. More specifically, we have:

$$\mathbf{x}_j = Q_j R_j \tag{5.5}$$

where Q_j is orthonormal with dimension $n \times p_j$ and R_j is invertible with dimension $p_j \times p_j$. Q_j is orthonormal, which means each column of Q_j is a basis for $\mathbf{x}_j$. In other words, each column of $\mathbf{x}_j$ can be obtained by linear combination of the columns of Q_j. Thus Q_j can be used to represent $\mathbf{x}_j$. So for a general $\mathbf{x}_j$, the solution can be written as:

$$\hat{\boldsymbol{\beta}}_j = \left(1 - \frac{\lambda\sqrt{p_j}}{\|\mathbf{Q}_j^T r_j\|_2}\right)^+ R_j^{-1}\mathbf{Q}_j^T r_j.$$

5.1.2 Univariate regression

As we mentioned before, variable selection methods, with both scalar and functional candidates based on group variable selection method, are normally the combination of group variable selection methods and univariate regressions. We discussed the solution of group variable selection in the previous subsection. We now briefly review the solution of the univariate regression. The solution of univariate regression with scalar variables is simple. Thus we omit the formula here and focus on the univariate functional regression. The model in this section is:

$$y = \int x(t)\beta(t)dt + \epsilon \tag{5.6}$$

The conventional discrete method in functional data analysis is the basis function method. As we discussed in Chapter 3, functional variables can be expressed by a set of known basis functions. Here we use the same set up as in Chapter 3.

Suppose we have a functional variable $x(t)$. Let $\Phi_k(t)$ be the known basis functions. Also

assume that the basis functions are second order differentiable. Thus,

$$x(t) = \sum_{k=1}^{\infty} C_k \Phi_k(t) \approx \sum_{k=1}^{K} C_k \Phi_k(t) \quad \beta(t) = \sum_{k=1}^{\infty} \tilde{C}_{\beta,k} \Phi_k(t) \approx \sum_{k=1}^{K} \tilde{C}_{\beta,k} \Phi_k(t),$$

where C_k and $\tilde{C}_{\beta,k}$ are coefficients for the basis functions used for the functional variable and functional coefficient, respectively. We can use the same basis functions for all functional items. It is impossible to have an infinite number of basis functions, so we use K basis functions instead. If we calculate the basis functions on p equally spaced time points, the basis function $\Phi_k(t)$ can be expressed by a $K \times p$ matrix $\boldsymbol{\Phi}$; while all the coefficient matrices for functional variables are $n \times K$, and coefficient matrices for functional coefficients are $1 \times K$.

By using discrete values of the basis functions, the original functional variables can be written as:

$$x(t) \approx \mathbf{x} = \mathbf{C}\boldsymbol{\Phi} \quad \beta(t) \approx \boldsymbol{\beta} = \tilde{b}\boldsymbol{\Phi}$$

Let us denote the second order derivative of the basis functions $\Phi(t)$ as $L(t)$. The second order derivative of the functional coefficient $\beta''(t)$ can be written as $\sum_{k=1}^{K} \tilde{C}_{\beta,k} L_k(t)$. Similarly, we denote $\mathbf{L}$ as the discrete values of $L(t)$.

Thus the integration in the model Eqn (5.6) becomes:

$$\int x(t)\beta(t)dt \approx \mathbf{C}\boldsymbol{\Phi}\boldsymbol{\Phi}^T\tilde{b}^T/k \tag{5.7}$$

$$\int \beta''(t)\beta''(t)dt \approx \tilde{b}\mathbf{L}\mathbf{L}^T\tilde{b}^T/k. \tag{5.8}$$

In our analysis, we treat the functional variable $x(t)$ as a matrix of discrete data points $\mathbf{x}$. The univariate functional regression can be written as:

$$\begin{aligned} y &= \int x(t)\beta(t)dt + \epsilon \\ &\approx \mathbf{x}\boldsymbol{\Phi}^T\tilde{b}^T/k + \epsilon \end{aligned}$$

Since only $\tilde{b}_j$ are unknown, we can treat $\mathbf{x}\boldsymbol{\Phi}^T/k$ as the new input variable.

The smoothness of the functional coefficient is one of the most important aspect of functional regression. It can be controlled by the number and order of basis functions, or

by the use of a roughness penalty. Different methods lead to different solutions for the estimated parameter.

1. Control smoothness by basis functions
   If we use basis functions to control the smoothness, the parameter $\tilde{b}$ is estimated by the following least square function:

$$G = (y - (\mathbf{x}\boldsymbol{\Phi}^T/k)\tilde{b}^T)^2.$$

   Thus the estimated functional coefficient is:

$$\begin{aligned}\hat{\beta}(t) &\approx \hat{\tilde{b}}\boldsymbol{\Phi} \\ &= y^T(\mathbf{x}\boldsymbol{\Phi}^T/k)^T[\boldsymbol{\Phi}^T\mathbf{x}^T\mathbf{x}\boldsymbol{\Phi}^T/k^2]^{-1}\boldsymbol{\Phi}.\end{aligned}$$

   To avoid singularity, we use a small number of basis functions in this case. The small number and low order of the basis functions used would lead to a low computation cost, but would also lead to poor estimation at the locations with rapid change on the functional coefficient.

2. Control smoothness by basis functions and roughness penalty
   Another way to control the smoothness of functional coefficients is to use a large number, and high order of basis functions together with the roughness penalty. The penalized least square becomes:

$$\begin{aligned}G =&(y - \int x(t)\beta(t)dt)^2 + \varphi\int[\beta''(t)]^2dt \\ \approx&(y - (\mathbf{x}\boldsymbol{\Phi}^T/k)\tilde{b}^T)^2 + \varphi(\tilde{b}(\boldsymbol{\Phi}_2\boldsymbol{\Phi}_2^T/k)\tilde{b}^T)\end{aligned}$$

   where Φ_2 is the matrix representing the second order derivative of the basis functions.

   The estimation for $\beta(t)$ from penalized functional regression is:

$$\begin{aligned}\hat{\beta}(t) &\approx \hat{\tilde{b}}\boldsymbol{\Phi} \\ &= y^T\mathbf{x}\boldsymbol{\Phi}^T(\boldsymbol{\Phi}\mathbf{x}^T\mathbf{x}\boldsymbol{\Phi}^T + \varphi\boldsymbol{\Phi}_2\boldsymbol{\Phi}_2^T)^{-1}\boldsymbol{\Phi}\end{aligned} \tag{5.9}$$

   The roughness penalty allows us to use relatively large number and high order basis functions to fit the functional coefficient. The smoothness of the estimation is

then adjusted by the roughness penalty. This penalty term also helps avoiding the singular problem.

5.1.3 Mixed scalar and functional variable selection based on the group lasso method

With centred and scaled response and candidate variables, the equation for the regression model is

$$y = \sum_{j=1}^{J} \int x_j(t)\beta_j(t)dt + \sum_{m=1}^{M} z_m\gamma_m + \epsilon. \tag{5.10}$$

If we approximate $x_j(t)$ by $\mathbf{x}_j$, approximate $\beta_j(t)$ by $\tilde{b}\boldsymbol{\Phi}$, and also consider Eqn (5.7), Eqn (5.8), the above equation can be written as:

$$y = \sum_{j=1}^{J} \mathbf{x}_j\boldsymbol{\Phi}^T\tilde{b}_j^T/k + \sum_{m=1}^{M} z_m\gamma_m + \epsilon. \tag{5.11}$$

Thus the variable selection problem Eqn (5.10) becomes Eqn (5.11), which is similar to the group variable selection problem. The group lasso penalty is added to the least square equation:

$$G = (y - \sum_{j=1}^{J} \mathbf{x}_j\boldsymbol{\Phi}^T\tilde{b}_j^T/k - \sum_{m=1}^{M} z_m\gamma_m)^2 + \lambda\ PEN, \tag{5.12}$$

where PEN is the penalty function. Each of the univariate functional regression methods corresponds to one type of PEN. Now we show the details of the difference in the following subsections.

Control smoothness by basis functions

If we use basis functions to control the smoothness of the functional coefficient, the penalty function would target on the functional coefficients. And thus the penalty function is given by:

$$PEN = \sum_{j} ||\tilde{b}_j||_2 + \sum_{m} ||\gamma_m||_2.$$

The functional coefficients are also the target of the penalty functions used in some functional variable selection methods. For example, Matsui and Konishib (2011) uses Gaussian basis functions to represent the functional variables, and then use group SCAD proposed by Wang et al. (2007) to do the selection; Mingotti et al. (2013) uses B-spline basis functions and the lasso method to do the variable selection.

The least square equation Eqn (5.12) becomes:

$$G = (y - \sum_{j=1}^{J} \mathbf{x}_j \boldsymbol{\Phi}^T \tilde{b}_j^T / k - \sum_{m=1}^{M} z_m \gamma_m)^2 + \lambda (\sum_{j=1}^{J} \sqrt{p_j} \|\tilde{b}_j\|_2 + \sum_{m=1}^{M} \|\gamma_m\|_2).$$

The estimation of the j-th functional coefficient is:

$$\hat{\beta}_j(t) \approx \left(1 - \frac{\lambda \sqrt{p_j}}{\|\boldsymbol{\Phi} \mathbf{x}_j^T r_j\|_2}\right)^+ \boldsymbol{\Phi} \mathbf{x}_j^T r_j,$$

and the estimation of the m-th scalar coefficient is:

$$\hat{\gamma}_m = \left(1 - \frac{\lambda}{\|z_m^T r_j\|_2}\right)^+ z_m^T r_j.$$

Instead of transferring the new group variables z_m to orthonormal matrices, they are assumed to be orthonormal. This approximation would bring some errors to the estimation, but the error would be small if the variables are centred and scaled.

The advantage of using basis functions to control the smoothness of the functional coefficients is the low computational cost, since the dimension of each group would be restricted to a small number to avoid the singularity. The disadvantage, however, of this restriction is the inaccurate estimation of the functional coefficients, when the shape of the curve is complex. We refer to this method as FGL_B.

Control smoothness by basis functions and roughness penalty

J.Gertheiss (2013) proposed the functional variable selection method based on group lasso with roughness penalty for the functional coefficients. They use B-spline basis functions to represent the functional variable and the functional coefficients.

The roughness penalty is added to the kernel matrix such that the penalized least square

becomes:

$$G = (y - \sum_{j=1}^{J} \mathbf{x}_j \boldsymbol{\Phi}^T \tilde{b}_j^T / k - \sum_{m=1}^{M} z_m \gamma_m)^2 + \lambda (\sum_{j=1}^{J} \sqrt{p_j} (\tilde{b}_j^T (\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \varphi \boldsymbol{\Phi}_2 \boldsymbol{\Phi}_2^T) \tilde{b}_j)^{1/2} + \sum_{m=1}^{M} \|\gamma_m\|_2).$$

Apparently $K = \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \varphi \boldsymbol{\Phi}_2 \boldsymbol{\Phi}_2^T$ is not a constant times an identity matrix in this equation. The authors decomposite this kernel matrix by Choleskey decomposition and get $K = L^T L$. Thus the penalized least square becomes:

$$\begin{aligned}
G =& (y - \sum_{j=1}^{J} \mathbf{x}_j \boldsymbol{\Phi}^T \tilde{b}^T / k - \sum_{m=1}^{M} z_m \gamma_m)^2 + \lambda (\sum_{j=1}^{J} \sqrt{p_j} (\tilde{b}_j^T L^T L \tilde{b}_j)^{1/2} + \sum_{m=1}^{M} \|\gamma_m\|_2) \\
=& (y - \sum_{j=1}^{J} \mathbf{x}_j \boldsymbol{\Phi}^T L^{-1} L \tilde{b}^T / k - \sum_{m=1}^{M} z_m \gamma_m)^2 + \lambda (\sum_{j=1}^{J} \sqrt{p_j} (\tilde{b}_j^T L^T L \tilde{b}_j)^{1/2} + \sum_{m=1}^{M} \|\gamma_m\|_2) \\
=& (y - \sum_{j=1}^{J} \mathbf{x}_j^* \tilde{b}^{*T} / k - \sum_{m=1}^{M} z_m \gamma_m)^2 + \lambda (\sum_{j=1}^{J} \sqrt{p_j} (\tilde{b}_j^{*T} \tilde{b}_j^*)^{1/2} + \sum_{m=1}^{M} \|\gamma_m\|_2),
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{x}_j^* &= \mathbf{x}_j \boldsymbol{\Phi}^T L^{-1} \\
\mathbf{b}_j^* &= L \tilde{b}_j.
\end{aligned}$$

Additionally, the input variables are assumed to be orthonormal in the calculation rather than transformed into orthonormal basis. More specifically, $\mathbf{z}_j^{*T} \mathbf{z}_j^*$ is assumed to be as identity matrix times the dimension K.

One benefit of this is that the design matrices normally have high dimensions, which might lead to short rank design matrices. The QR decomposition of these matrices brings numerical error to the regression. Inverting the upper triangular matrix R_j from the QR decomposition would bring even more error to the regression. The final estimation of the functional coefficient would be extremely inaccurate.

Thus the solution of the functional coefficient $\beta_j(t)$ in theory should be:

$$\hat{\beta}_j(t) = \begin{cases} \mathbf{0} & \|\mathbf{z}_j^{*T} r_j - \mathbf{z}^{*T} \mathbf{z}^* \boldsymbol{\beta}_j^*\|_2 \leq \lambda \sqrt{K}; \\ \boldsymbol{\Phi} (\mathbf{z}_j^{*T} \mathbf{z}_j^* + \frac{\lambda}{\|\boldsymbol{\beta}_j^*\|_2})^{-1} L^{-1} \mathbf{z}_j^{*T} r_j & \text{otherwise.} \end{cases}$$

However, because they assume that $\mathbf{z}_j^{*T}\mathbf{z}_j^* = I_{p_j}$, the actual solution is :

$$\hat{\beta}_j(t) = \left(1 - \frac{\lambda\sqrt{K}}{\|\mathbf{z}_j^{*T} r_j\|_2}\right)^+ \boldsymbol{\Phi} L_j^{-1} \mathbf{z}_j^{*T} r_j$$

where $1 - \frac{\lambda\sqrt{K}}{\|\mathbf{z}_j^{*T} r_j\|_2}$ is a constant, related to the shrinkage of the coefficients. The directions of the functional coefficients are controlled by $\boldsymbol{\Phi} L_j^{-1} \mathbf{z}_j^{*T} r_j$.

We can substitute the basis functions into the solution, such that the direction can be written as:

$$\begin{aligned}
\hat{\beta}_j(t) &\approx \boldsymbol{\Phi} L^{-1} \mathbf{z}_j^{*T} r_j \\
&= \boldsymbol{\Phi} L^{-1} L^{-1T} \mathbf{x}_j^{T*} r_j \\
&= \boldsymbol{\Phi} K^{-1} \mathbf{x}_j^{T*} r_j \\
&= \boldsymbol{\Phi} (\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \varphi \boldsymbol{\Phi}_2^T \boldsymbol{\Phi}_2)^{-1} \boldsymbol{\Phi}^T \boldsymbol{\Phi} \mathbf{c}_j^T r_j.
\end{aligned}$$

Recall that the coefficient estimated using penalized functional regression from Eqn (5.9) is

$$\boldsymbol{\Phi} \left(\boldsymbol{\Phi}^T \mathbf{x}^T \mathbf{x} \boldsymbol{\Phi} + \varphi \boldsymbol{\Phi}_2^T \boldsymbol{\Phi}_2\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{\Phi} \mathbf{c}^T y.$$

The solutions are fairly similar to each other. If the functional variables are centred and scaled to have mean 0 and variance $n - 1$ column-wise before entering the regression model, the error made by assuming $\mathbf{z}_j^{*T}\mathbf{z}_j^* = K\, I_K$ would be small.

The solution for the scalar variable is same as before. The estimation for the m-th scalar variable is:

$$\hat{\gamma}_m = \left(1 - \frac{\lambda}{\|z_m^T r_j\|_2}\right)^+ z_m^T r_j.$$

We refer to this method as FGL_P.

The advantage of this algorithm is that it can give fairly accurate estimation of the the functional coefficients even if they have complex shapes.

The disadvantages of this algorithm are the follows. Firstly, the estimation involves approximation, which would bring errors to the results. Secondly, there are two tuning parameters in the algorithm and GCV cannot be used. So it would be computationally expensive to find a good set of tuning parameters.

5.2 Functional LARS with Scalar Variables in the Candidates

In the previous chapter, we introduced functional LARS for the variable selection problem involving only functional variables. The algorithm works well when all the functional variables are represented by using matrices with same dimension. In this chapter, we introduce scalar candidates to the variable selection problem. Thus the dimension of the variables would be different from each other. We need to modify the algorithm and deal with certain issues in the algorithm that could be effected by the different dimensions.

There are two aspects in the algorithm that might be effected: firstly the calculation of the correlation between the selected variables and the response variable; and secondly, the calculation of the solution of equal correlation equation.

5.2.1 Correlation between one scalar variable and a group of scalar and functional variables

The linear correlation between two scalar variables in the LARS is the Pearson's correlation coefficient.

$$\rho = \frac{Cov(Z, Y)}{\sqrt{Var(Z)Var(Y)}}$$

Recall that the general formula to calculate the functional canonical correlation in Eqn (3.22) and the corresponding coefficient in Eqn (3.23):

$$\text{correlation: } \rho^2 = \frac{V_{X,y}^T P_{X,X}^{-1} V_{X,y}}{V_y}$$
$$\text{coefficients: } \tilde{b} = \frac{P_{X,X}^{-1} V_{X,y}}{\rho \|y\|_2}$$

The matrix $P_{X,X}$ is a block matrix. The (i, j)-th block is:

$$P_{X_i,X_j} = W^T \mathbf{x}_i^T \mathbf{x}_j W + \delta_{i,j} \text{PEN},$$

where $\delta_{i,j} = \mathbf{I}$ if $i = j$, and $\mathbf{0}$ otherwise; PEN is the penalty function. This is the general expression we used in Chapter 3, and the matrix W depends on the choice of the discrete method for the functional coefficient.

Similarly, $V_{X,y}$ is also a block matrix. The i-th block is:

$$V_{X_i,y} = W\mathbf{x}_i^T y.$$

Both scalar and functional variables are centred and scaled to have mean 0 and variance 1. For scalar variables, centring is obtained by subtracting the sample mean, and scaling is obtained by dividing the sample standard deviation. Functional variables are centred by the column mean and scaled by the column standard deviation.

The blocks corresponding to the scalar variables certainly have no penalties to add on. As an example, suppose that there are two variables, $x(t)$ and z. The covariance matrix $P_{X,X}$ can be written as:

$$P_{X,X} = \begin{pmatrix} W^T\mathbf{x}_i^T\mathbf{x}_j W + \delta_{i,j}\text{PEN} & W^T\mathbf{x}_i^T z \\ \mathbf{x}_j W z & z^T z \end{pmatrix}.$$

Also, $V_{X,y}$ stays the same. In order to unify the formula, we can have $W = 1$ and $PEN = \mathbf{0}$ for the blocks corresponding to scalar variables. The cases with more functional variables and scalar variables can be extended from this expression easily.

5.2.2 Equal squared correlation for scalar and functional variables.

Recall that the LARS algorithm uses the following equation to find the distance to move on the direction unit vector with respect to the scalar candidate variable z:

$$\text{Cor}(r - \alpha u, z)^2 = \text{Cor}(r - \alpha u, u)^2$$

where u is the unit vector representing the direction of the current iteration. The iteration number is omitted.

The group LARS from Yuan and Lin (2006) has a similar equation:

$$\|(r - \alpha u)^T \mathbf{z}_j\|_2^2 / p_j = \|(r - \alpha u)^T \mathbf{z}_i\|_2^2 / p_i$$

where $\|\cdot\|_2$ means the Euclidean norm of the vector; the group of variables $\mathbf{z}_i$ is one of the

variables selected; $\mathbf{b}_i$ is the corresponding coefficient; $\mathbf{z}_j$ is one of the candidate variables; p_i and p_j are the dimensions of $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively. The dimensions p_i and p_j are used to remove the effect of the difference in dimensions. Note that the direction vector u here is not unit vector.

This equation is equivalent to:

$$\begin{aligned}\frac{(r-\alpha u)^T(\mathbf{z}_j\mathbf{z}_j^T)(r-\alpha u)}{p_j} =& \frac{(r-\alpha u)^T(\mathbf{z}_i\mathbf{z}_i^T)(r-\alpha u)}{p_i}\\ (r-\alpha u)^T\frac{\mathbf{z}_j\mathbf{z}_j^T}{p_j}(r-\alpha u) =& (r-\alpha u)^T\frac{\mathbf{z}_i\mathbf{z}_i^T}{p_i}(r-\alpha u).\end{aligned}$$

Clearly the normalization is on the matrices $\mathbf{z}_j\mathbf{z}_j^T$ and $\mathbf{z}_i\mathbf{z}_i^T$. These two matrices are the only different elements between left and right hand sides of the equation. The dimension of the matrix x could be the rank, trace and Frobenius norm of the matrix $\mathbf{z}\mathbf{z}^T$ if $\mathbf{z}$ is orthonormal as required in the group LARS algorithm.

In functional cases, we write this equation in two versions for functional variable and scalar variable, respectively. For functional candidates:

$$\mathrm{Cor}\left(r-\alpha u, \int x(t)\beta(t)dt\right)^2 /N_f = \mathrm{Cor}(r-\alpha u, u)^2/N_u, \tag{5.13}$$

where the correlation between a scalar variable and the projection of a group of functional variables is calculated by functional canonical correlation analysis.

For scalar candidates, we have:

$$\mathrm{Cor}\,(r-\alpha u, z)^2\,/N_z = \mathrm{Cor}(r-\alpha p, p)^2/N_u, \tag{5.14}$$

where N_f, N_z and N_u are all constants for normalization.

For functional candidate variables

We omit the subscript of the variables in the following derivation. If we substitute Eqn (3.20) into left hand side of Eqn (5.13) and expand the right hand side of Eqn (5.13), we can get:

$$\begin{aligned}\frac{(r-\alpha u)^T \mathbf{x} W K^{-1} W^T \mathbf{x}^T (r-\alpha u)}{(r-\alpha u)^T (r-\alpha u) N_f} &= \frac{[(r-\alpha u)^T u/(n-1)]^2}{(r-\alpha u)^T (r-\alpha u) u^T u/(n-1)^2 N_u} \\ \frac{(r-\alpha u)^T \mathbf{x} W K^{-1} W^T \mathbf{x}^T (r-\alpha u)}{(r-\alpha u)^T (r-\alpha u) N_f} &= \frac{(r-\alpha u)^T u u^T (r-\alpha u)}{(r-\alpha u)^T (r-\alpha u) u^T p N_u} \\ (r-\alpha u)^T \bar{S} (r-\alpha u) &= (r-\alpha u)^T \bar{U} (r-\alpha u).\end{aligned}$$

where $\bar{S} = \frac{\mathbf{x} W K^{-1} W^T \mathbf{x}^T}{N_f}, \bar{U} = \frac{u(u^T u)^{-1} u^T}{N_u}$; $\mathbf{x}$ is the discrete data points of the functional variable $x(t)$. The estimated α is the solution of the quadratic function:

$$\alpha^2 [u^T (\bar{S} - \bar{U}) u] - 2\alpha [r^T (\bar{S} - \bar{U}) u] + [r^T (\bar{S} - \bar{U}) r] = 0 \tag{5.15}$$

For scalar candidate variables

If we expand both side of Eqn (5.14):

$$\begin{aligned}\frac{[(r-\alpha u)^T z/(n-1)]^2}{(r-\alpha u)^T (r-\alpha u) z^T z/(n-1)^2 N_z} &= \frac{[(r-\alpha u)^T u/(n-1)]^2}{(r-\alpha u)^T (r-\alpha u) u^T u/(n-1)^2 N_u} \\ \frac{(r-\alpha u)^T z z^T (r-\alpha u)}{z^T z N_z} &= \frac{(r-\alpha u)^T u u^T (r-\alpha u)}{u^T u N_u} \\ (r-\alpha u)^T \bar{Z} (r-\alpha u) &= (r-\alpha u)^T \bar{U} (r-\alpha u).\end{aligned}$$

where $\bar{Z} = \frac{z(z^T z)^{-1} z^T}{N_z}$. The solution of α is the solution of the quadratic function:

$$\alpha^2 [u^T (\bar{Z} - \bar{U}) u] - 2\alpha [r^T (\bar{X} - \bar{U}) u] + [r^T (\bar{X} - \bar{U}) r] = 0 \tag{5.16}$$

The intuitions from Eqn(5.15) and Eqn(5.16) are identical. Thus we can unify the formulas for scalar and functional cases. From the group LARS algorithm, the choice of normalization method is not clear. As stated before, the choices we could try are the rank, trace and Frobenius norm of the matrix. The findings will be listed in the simulation section.

5.3 Simulation Study

We have seen good results from the previous chapter for functional LARS. As the formulas in this chapter do not change the estimation of the functional coefficients, we believe that the estimation accuracy would not change when we use different types of normalization methods. The speed of the calculation largely depends on the dimension of each of the functional variables. Since we use generalized cross validation to find a good tuning parameter, the computation cost will be low by using the functional LARS.

In this simulation study, we have seven scenarios with increasingly complex correlation structures. We focus on the performance of the three functional variable selection methods in the first four scenarios and we check the performance of the functional LARS with Modification II in Section 4.3.2 in the Scenarios 5 and 6. We also include the scenario when a large number of variables are in the candidate. In addition, we compare the performance of functional LARS with different normalization methods. The stopping rule for the two variable selection methods based on group lasso is 5-fold cross validation, suggested by J.Gertheiss (2013). The stopping rule for functional LARS is *AD* proposed in the previous chapter. We refer to functional LARS algorithm as $flars$.

5.3.1 True model and data

The true model uses three functional variables and three scalar variables:

$$y = \int x_1(t)\beta_1(t)dt + \int x_2(t)\beta_2(t)dt + \int x_3(t)\beta_3(t)dt + z_1\gamma_1 + z_2\gamma_2 + z_3\gamma_3 + \epsilon. \quad (5.17)$$

In the regression equation, $x_j(t)$ are functional variables; z_m are scalar variables. $j = 1, \ldots, 7$, and $m = 1, \ldots, 5$. Thus the variables are $x_1(t)$ to $x_7(t)$ and z_1 to z_5. The variables are generated using the same idea as that used in the last chapter. The variables generated are linearly uncorrelated with each other. Canonical correlation is used to measure the relationship involving functional variables, and Pearson's correlation is used to measure the relationship between scalar variables.

We take one data set as an illustrative example. Figure 5.1 shows the seven uncorrelated functional variables. Figure 5.2 shows the functional coefficients $\beta_1(t)$, $\beta_2(t)$ and $\beta_3(t)$. The coefficients for scalar variables are: $\gamma_1 = 0.2201$, $\gamma_2 = 0.2087$ and $\gamma_3 = 0.1931$. The intercept is 10.

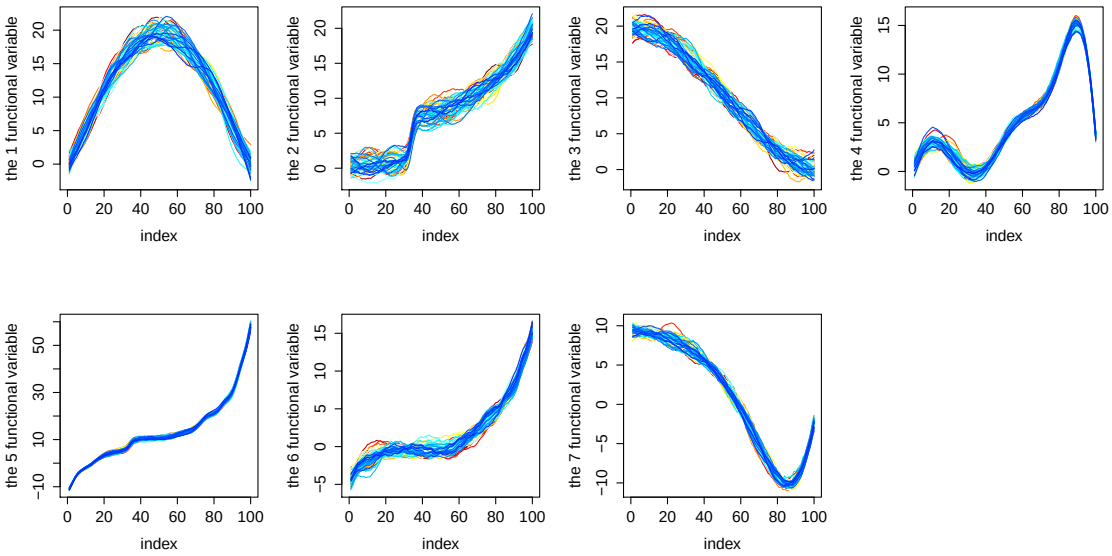


Figure 5.1: Seven linear uncorrelated functional variables

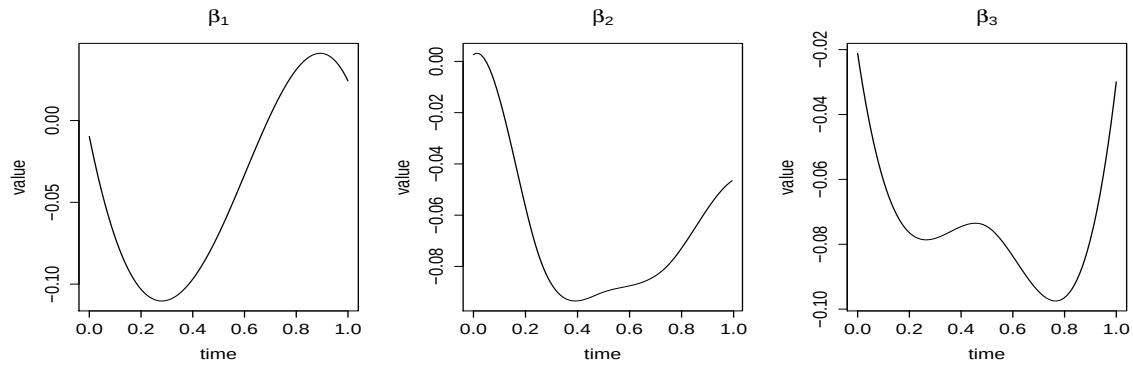


Figure 5.2: The true values of the functional coefficients

5.3.2 Different Scenarios

We consider six scenarios in this simulation study. Each of the scenarios is repeated 1000 times. In each replication, 120 samples are generated, where 80 of them are used for training, and the remaining 40 samples are used for testing.

The correlations between the variables are more complex in the later scenarios. Here we list the details:

1. Seven functional variables and five scalar variables are generated. They are named as $x_1(t)$, $x_2(t)$, ..., $x_7(t)$, z_1, ..., z_5. The variables are uncorrelated with each other.

2. Seven functional variables and five scalar variables are generated in Scenario 2, namely $x_1(t)$, $x_2(t)$, ..., $x_7(t)$, z_1, ..., z_5. We replace functional variables $x_4(t)$ and $x_5(t)$, such that they are both correlated with $x_1(t)$. All the other variables remain uncorrelated with each other.

3. Similar to Scenario 2, we first generate 12 uncorrelated variables $x_1(t)$, $x_2(t)$, ..., $x_7(t)$, z_1, ..., z_5 in Scenario 3. We replace scalar variables z_4 and z_5 such that they are correlated with z_1. All the other variables remain uncorrelated with each other.

4. We combines both correlation structures from Scenario 2 and 3 in Scenario 4. More specifically, $x_4(t)$ and $x_5(t)$ are correlated with $x_1(t)$, and z_4 and z_5 are correlated with z_1. The scalar variables and functional variables are uncorrelated.

5. We introduce correlation between one functional variable and one scalar variable in Scenario 5. We replace $x_6(t)$ such that it is correlated with z_1. All the other variables remains the same as the ones from Scenario 4.

6. Scenario 6 has the same correlation structure as Scenario 5. The special feature for this scenario is that some irrelevant variables are more correlated with the response variable than all of the true variables. We want to test the performance of Modification II discussed in the previous chapter.

7. Scenario 7 has 50 scalar variables and 50 functional variables. We still use the first three scalar variables and first three functional variables in the true model. We generate these random variables independently from normal distributions, but we do not try to control the correlation structure of them.

Scenario 1

We set all candidate variables to be linearly uncorrelated with each other in Scenario 1. Figure 5.3 shows the correlation map between all variables and the response from the example data set. The last column shows the correlation between the response variable and all of the candidate variables.

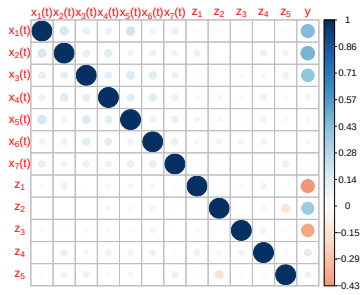


Figure 5.3: Correlation map of one of the simulated data set from Scenario 1

Comparison between algorithms We first compare the performance of three different functional variable selection algorithms. We include the estimations from the algorithms based on group lasso and functional LARS with the the *RDP* method and identity normalization in this comparison. The comparison between other results from functional LARS with different representative methods for the coefficients and different normalization is in a later paragraph.

Figure 5.4 shows the estimations of the parameters for each of the variables. The top left plot shows the true value and the estimations of the intercept; the right bottom plot shows the true values and the estimations of the parameter for the scalar variables; the rest of the plots show the true values and the estimations of the functional coefficients.

The intercept estimated from different algorithms are all close to the true value. The shapes of the functional coefficients are also close to those of the true coefficients. However, the estimates from the algorithms using group lasso would diverge from the true coefficient, if the true functional coefficient had sharp turns. Estimation from functional LARS shows better performance in this situation.

As we mentioned above, the result in Figure 5.4 corresponding to $flars$ is suggested by our stopping rule *AD*; the results corresponding to the other two algorithms are suggested

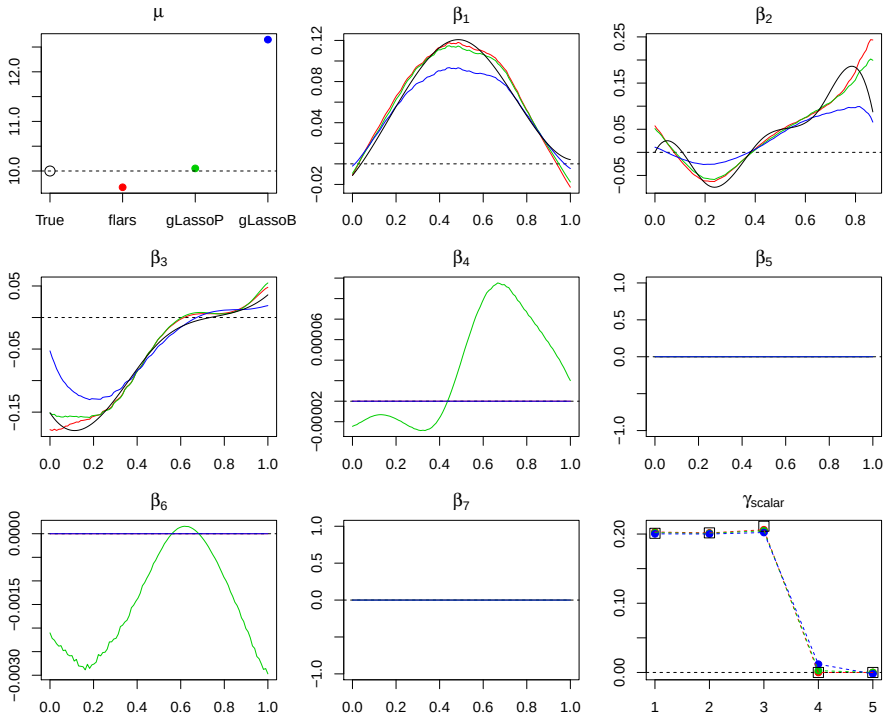


Figure 5.4: Estimated parameters from $flars$, FGL$_B$ and FGL$_P$. Black lines are the true functional coefficients; red lines and points are the estimations of the coefficients from $flars$; green lines and points are the estimations of the coefficients from FGL$_P$; blue lines and points are the estimations of the coefficients from FGL$_B$. The colours of the lines and points remain the same meaning through out all the simulations.

by 5-fold cross validation. The stopping point chosen by AD gives excellent selection of the variables, i.e. only $\beta_1(t)$, $\beta_2(t)$, $\beta_3(t)$, γ_1, γ_2 and γ_3 are non-zero. However, the variables chosen by FGL_P and FGL_B with 5-fold cross validation include irrelevant scalar variables. We found that this is a typical problem when using FGL_P and FGL_B with 5-fold cross validation. More specifically, 5-fold cross validation tends to reach the optimal when some irrelevant variables are included. It is also worth noting that FGL_P and FGL_B are more likely to select scalar variables rather than functional variables.

We present the estimation accuracy and selection accuracy of the three algorithms for this data set in Table 5.1. The estimation accuracy is measured by the RMSE; the selection accuracy is measured by the number of true variables selected and the number of irrelevant variables selected. We have 6 true variables and 6 irrelevant variables in the candidates. The prediction from functional LARS is the most accurate among the three algorithms; the prediction from FGL_P is slightly worse, while the prediction from FGL_B is poor. This confirms the estimation of the parameters in Figure 5.4.

	RMSE	expected	unexpected
$flars$	0.0640	6/6	0/6
FGL_P	0.0660	6/6	3/6
FGL_B	0.1177	6/6	2/6

Table 5.1: The estimation accuracy and the selection accuracy from three algorithms.

Comparison between different discrete methods and normalizations We present the results for Scenario 1 from functional LARS using three different discrete methods for the functional coefficients with four different normalization methods in Figure 5.5. Recall that each of the RDP, GQ and BF discrete methods has four normalization choices: Frobenius norm, rank, trace and the identity matrix. Thus we have 12 sets of outcomes from the combinations between the discrete methods and the normalization methods. The results using the RDP method for functional coefficients are in the top row; the results from the GQ method are in the middle row; the results from the BF method are in the bottom row. The same colour in different rows indicates the same normalization method. In each of the plots, results corresponding to Frobenius norm, rank, trace and the identity matrix are in red, green, blue and purple, respectively. Similar to previous plots, the true values are drawn in black. The estimates of the functional coefficients corresponding to the GQ method are point estimates from the 18-points rule.

From the plots, we can see that the outcomes from different discrete and normalization

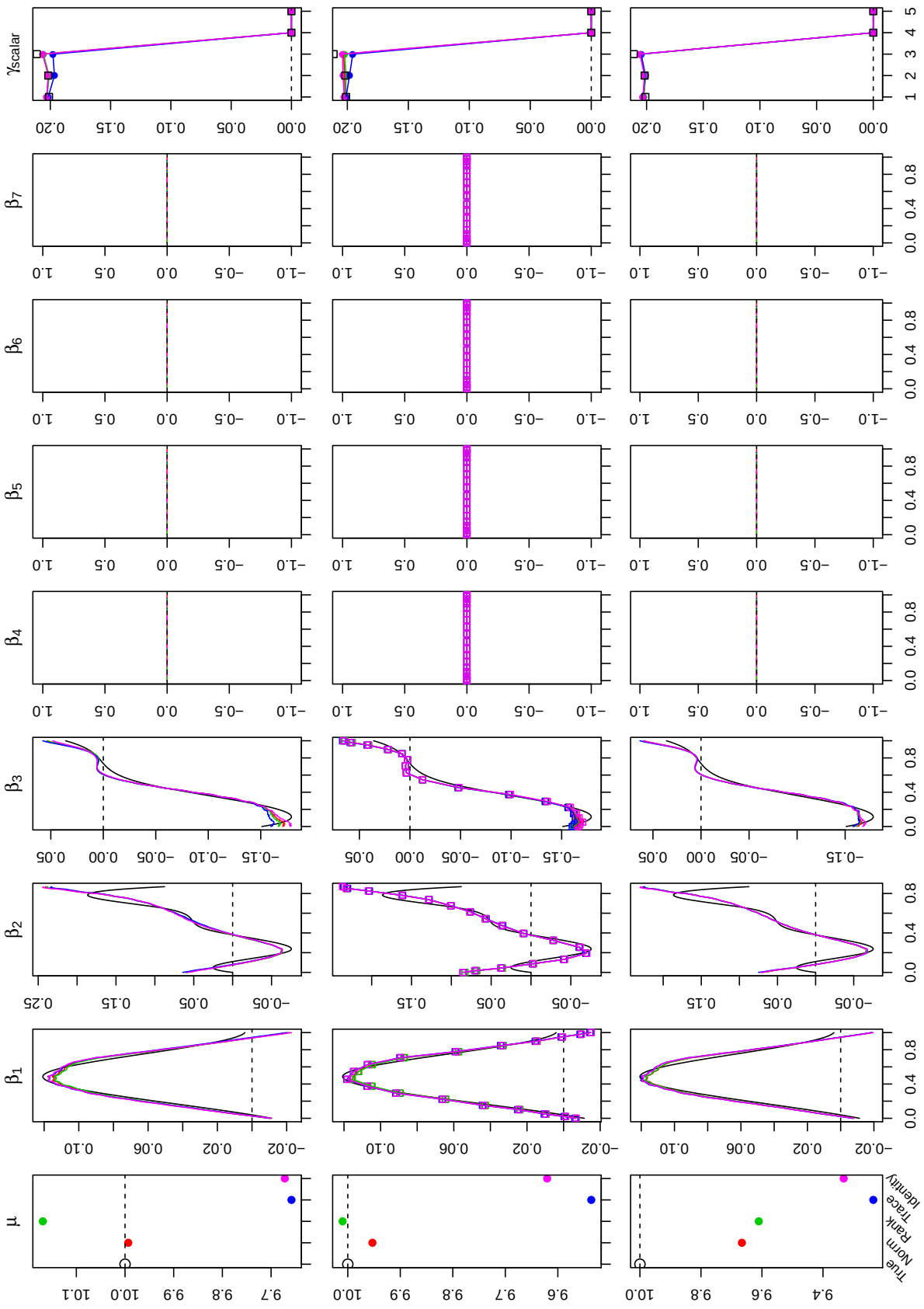


Figure 5.5: Comparison between different discrete methods and normalization methods.

methods are identical to each other. The results are also fairly accurate with respect to the true values of the parameters. In addition, the plots show that the selection made by the algorithm using different discrete and normalization methods and the stopping rule *AD* are accurate for this data set. The calculation from the combination of the *RDP* and identity normalization stops with seven variables in the regression equation suggested by *AD*. This could happen with very small probability. We will show more results from the 1000 replications later.

The prediction RMSE's from functional LARS using different combinations of discrete methods and normalization methods are shown in Table 5.2. Recall that the true error has standard deviation 0.05. Thus we obtain accurate predictions. None of the combinations has a large impact on the outcome for this data set.

	Norm	Rank	Trace	Identity
RDP	0.0631	0.0627	0.0643	0.0640
GQ	0.0639	0.0643	0.0653	0.0638
BF	0.0612	0.0614	0.0612	0.0605

Table 5.2: RMSE from different discrete methods for functional coefficients and different normalization methods.

Simulation study results based on 1000 replications We focus on the following aspects of the algorithms: selection accuracy, estimation accuracy and computation time. For each of the algorithms, we use the corresponding stopping rules to find the estimation of the parameters for prediction. More specifically, we use *AD* for functional LARS and we use 5-fold cross validation with prediction RMSE for FGL_P and FGL_B. The comparison of selection accuracy has two parts, one is the percentage of true variables selected, the other one is the the percentage of irrelevant variables selected. The comparison of the estimation accuracy is only dependent on the prediction RMSE. The computation time from all algorithms are calculated in the following ways. For the functional LARS, it is the computational time up to the stopping point. For others the computational time is the time up to the point at which we find the best tuning parameters.

Table 5.3 contains the average of the outcomes from 1000 replications using different algorithms. We first compare functional LARS using different discrete methods and different normalization methods. The first column of the table is the average prediction RMSE. We can see that the functional LARS normalized by rank shows poor performance for all three discrete methods. The Second column of the table shows the percentage of expected

			RMSE	Expected (%)	Unexpected (%)	Time (sec)
flars	*RDP*	Norm	0.0591	99.89	0.00	3.4058
		Trace	0.0615	99.38	0.00	3.2836
		Rank	0.1723	76.62	0.42	2.2518
		Identity	0.0591	100	0.00	3.8382
	GQ	Norm	0.0619	100	0.07	0.6083
		Trace	0.0655	99.53	0.94	0.5999
		Rank	0.1560	80.29	0.35	0.6203
		Identity	0.0623	99.94	0.08	0.5799
	BF	Norm	0.0585	99.94	0.21	0.7154
		Trace	0.0709	98.44	3.75	0.7478
		Rank	0.1434	82.32	0.44	0.7421
		Identity	0.0586	99.94	0.06	0.7322
FGL_P			0.0616	100	51.95	260.3309
FGL_B			0.1166	100	23.86	1.0205

Table 5.3: Summaries of functional LARS with different discretizing and normalization methods, FGL_P and FGL_B.

selected variables. It confirms that *AD* sometimes stops the algorithm before all the true variables are selected when normalized by rank. Thus rank might not be a good choice for normalization in functional LARS, if we use *AD* as the stopping rule. From the third column, we can see that *AD* may select irrelevant variables with small probabilities. The normalization using trace and rank are slightly worse than the others. In addition, the prediction performance from trace is also slightly worse than the prediction from norm and identity. The computation time from the *RDP* methods are certainly the longest.

By considering all four columns, we can say that: first of all, the best normalization methods are Frobenius norm and identity; secondly, for best selection or fastest computation, we should choose thet *RDP* and the *GQ*, respectively. The difference in prediction accuracy from three methods are identical to each other.

The prediction RMSE from FGL_P is comparable with the good ones from functional LARS algorithm, but FGL_B is not good enough. Both FGL_P and FGL_B successfully selected all the true variables, but they also tend to select large proportion of irrelevant variables. The computation time for FGL_P is the longest due to the tuning for two tuning parameters. The computation time for FGL_B can only compare with the ones for functional LARS using the *RDP* method.

Scenario 2

The difference between Scenario 2 and Scenario 1 is that two irrelevant functional variables $x_4(t)$ and $x_5(t)$ are altered so that they are correlated with one of the relevant functional variable $x_1(t)$. Thus the correlations between the scalar response and these two functional variables are large. The correlation map is in Figure 5.6.

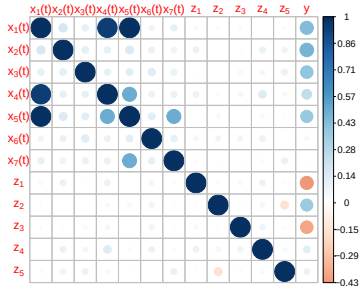


Figure 5.6: Correlation map for one of the simulated data set from Scenario 2

The comparison between the estimations from three different algorithms and the comparison between the estimations between different normalizations are in Figure 5.C.1 and Figure 5.C.2 respectively in Appendix 5.C. The figures are similar to those from Scenario 1.

The models estimated by algorithms using group lasso and 5-fold cross validation tend to include irrelevant variables in the covariates. For this data set, FGL_P selects one irrelevant functional variable and one irrelevant scalar variable, and FGL_B selects two irrelevant scalar variables. Compared to the true values of the parameters, estimates from functional LARS and FGL_P are more accurate than the ones from FGL_B. Details of the results are listed in Table 5.4. The table shows that the prediction RMSE from functional LARS using the combination of *RDP* method and identity normalization is the best of the three algorithms. All three algorithms successfully select relevant variables. The functional LARS algorithm select no irrelevant variables, while the other two algorithms select two irrelevant variables.

The estimations from functional LARS using different discrete methods and different normalization methods are similar to each other and close to the true value of the parameters. Table 5.5 shows the results from functional LARS using different discrete methods and different normalization methods. All prediction RMSE's are close to each other and indi-

	RMSE	expected	unexpected
flars	0.0646	6/6	0/6
FGL_P	0.0660	6/6	2/6
FGL_B	0.1177	6/6	2/6

Table 5.4: The estimation accuracy and the selection accuracy for three algorithms.

cate accurate predictions.

	Norm	Rank	Trace	Identity
RDP	0.0634	0.0630	0.0638	0.0646
GQ	0.0642	0.0643	0.0653	0.0642
BF	0.0612	0.0613	0.0612	0.0601

Table 5.5: RMSE from different discrete methods for functional coefficients and different normalization methods.

The results from 1000 replications are more credible. We put the summary of the replications in the Table 5.6. The functional LARS, normalized by rank using different discrete methods, has inaccurate results in terms of the prediction RMSE. Similar to the previous scenario, such poor results are caused by the inaccurate stopping points selected by *AD*. From the second column we can see that the algorithm normalized by rank may stop before all the relevant variables are selected. Thus if we want to use *AD* as the stopping rule, we should use other normalization methods. The third column shows the percentage of irrelevant variables selected. It appears that the probability of selecting irrelevant variables is small when using any combination of discrete methods and normalization methods.

The prediction RMSE's from FGL_P are slightly worse than those from functional LARS. The RMSE from FGL_B is much worse than the others. Both methods successfully select all the relevant variables, but they also select many irrelevant variables, especially the FGL_P method.

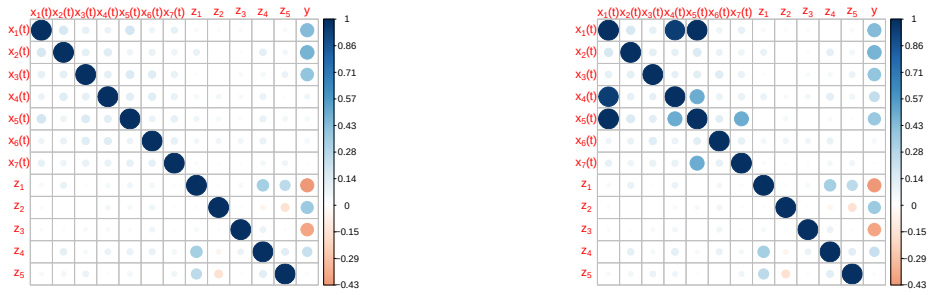
Functional LARS using the *GQ* and *BF* as discrete methods are the fastest algorithms. However, their selection accuracy is not as good as that using the *RDP* method. FGL_P takes the longest time to find a good pair of tuning parameters, while FGL_B takes a more reasonable amount of time to find a good tuning parameter.

Scenario 3 and 4 Scenario 3 has correlated scalar variables and no correlated functional variables, while Scenario 4 combines both Scenario 2 and 3. The data sets and the results from these three scenarios are all similar. Thus we present the findings of Scenario 3 and

			RMSE	Expected (%)	Unexpected (%)	Time (sec)
flars	*RDP*	Norm	0.0586	99.74	0.26	3.1231
		Trace	0.0613	99.21	0.20	3.3727
		Rank	0.1707	76.66	0.50	2.0830
		Identity	0.0601	99.33	0.56	3.8276
	GQ	Norm	0.0621	99.35	0.68	0.9492
		Trace	0.0661	98.61	1.70	0.5288
		Rank	0.1577	79.72	0.59	0.5517
		Identity	0.0630	99.13	0.97	0.5123
	BF	Norm	0.0596	99.14	0.95	0.6556
		Trace	0.0682	97.19	4.21	1.0814
		Rank	0.1468	81.45	0.85	0.6619
		Identity	0.0594	99.09	1.10	0.9101
FGL_P			0.0617	1.00	50.63	250.7358
FGL_B			0.1181	1.00	24.55	1.0692

Table 5.6: Summaries of functional LARS with different discrete and normalization methods, FGL_P and FGL_B.

4 together with Scenario 2. The correlation maps of these two scenarios are in Figure 5.7. In these scenarios, correlations between any functional variables and any scalar variables are almost zero.



(a) Scenario 3 (b) Scenario 4

Figure 5.7: Correlation map of one of the simulated data set from Scenario 3 and 4

The comparison between the estimations from three different algorithms and the comparison between the estimations between different normalizations in Scenario 3 and 4 are in the Figure 5.C.3, Figure 5.C.5, Figure 5.C.4 and Figure 5.C.6 in Appendix 5.C. Also the summary tables Table 5.C.1, Table 5.C.4, Table 5.C.2 and Table 5.C.5 for the two comparisons are in the Appendix 5.C. For this data set, the new correlation structures have

little effect on the estimates from the three algorithms and the estimates from functional LARS with different discrete methods and normalization methods. The prediction RMSE from functional LARS and FGL_P are close to each other, but the one from FGL_B is not good.

The summary for the results from 1000 replications in Scenario 3 and 4 are in Table 5.C.3 and Table 5.C.6 in Appendix 5.C. We can get similar conclusions from these two tables. By using *AD* as the stopping criteria, functional LARS normalized by rank may stop before finishing the selection of all the relevant variables. Thus the corresponding prediction RMSE is larger than the others. The prediction RMSE from FGL_P is slightly worse than those from functional LARS. Even though FGL_B selects all of the relevant variables, the prediction RMSE is still poor. This is certainly caused by the inaccurate estimation of the parameters.

Scenario 5

Previously we focused on the comparison between functional LARS with different discretization and normalization methods, also with the other two variable selection methods extended from functional variable selection. In this scenario and the next one, we focus on the selection and estimation when using Modification II from Section 4.3.2. Scenario 5 and 6 have the most complex correlation structure in our simulation study. Similar to previous scenarios, we have seven functional variables $x_1(t), \ldots, x_7(t)$, and five scalar variables $z_1, \ldots, z_5$. The true variables are still the first three functional variables and first three scalar variables. Irrelevant functional variables $x_4(t)$ and $x_5(t)$ are correlated with the true variable $x_1(t)$, with correlation of about 0.9; the irrelevant functional variable $x_6(t)$ is correlated with relevant scalar variable z_1, with correlation 0.5. The correlation between the response variable and some irrelevant variables could be large. The correlation map is shown in Figure 5.8.

Recall that Modification II is designed to remove variables from the regression equation. The variables should be removed when they are no longer useful. There are two aspects of the performance we want to test: before the algorithm reaches the stopping point, whether or not the modification remove relevant variables and whether or not it remove irrelevant variables.

Modification II removes variables based on their contribution to the model. A functional variable is removed from the model when the following two conditions are met. First, the

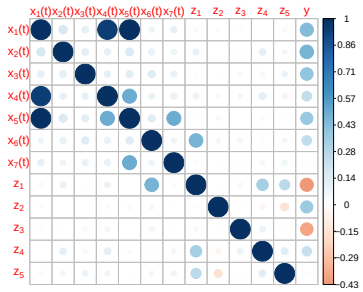


Figure 5.8: Correlation map of one of the simulated data set from Scenario 5

variance of the projection of the variable is smaller than the maximum variance from the same variable:

$$\text{Var}\left(\int x_j(t)\beta_j^{(k)}(t)\right) < \text{Var}\left(\int x_j(t)\beta_j^{(k*)}(t)\right) \quad \text{for} \quad k* \in 1,\ldots,k-1.$$

Secondly, the variance of the variable is less than a certain percentage of the total variance of the response variable:

$$\text{Var}\left(\int x_j(t)\beta_j^{(k)}(t)\right) < \kappa \text{Var}(y),$$

where κ is the threshold. In our simulation, $\kappa = 0.1$. This criteria also works for scalar variables. We omitted the equations for scalar variables here.

For this data set, unmodified and modified functional LARS give the same results before the algorithms reach the stopping point suggested by *AD*. The estimation of the parameters from unmodified functional LARS FGL_P, FGL_B are shown in Figure 5.D.1 and Figure 5.D.2 in Appendix 5.D. The result from modified functional LARS is omit in the plots. The conclusion made from these figures are similar to those from previous scenarios.

Table 5.7 shows the comparison of the prediction RMSE and the variable selected in the corresponding iteration up to Iteration 9 from unmodified and modified functional LARS. We only show the result using the *RDP* method and identity normalization here. The results from the unmodified and modified algorithms are identical. The first six iterations select these six variables for both unmodified and modified algorithms. This means that the modification successfully avoid removing important variables from the model for this

data set.

Iteration	RMSE		Selection	
	Unmodified	Modified	Unmodified	Modified
1	0.4132	0.4132	$x_1(t)$	$x_1(t)$
2	0.3850	0.3850	z_2	z_2
3	0.3614	0.3614	z_1	z_1
4	0.3142	0.3142	z_3	z_3
5	0.2258	0.2258	$x_3(t)$	$x_3(t)$
6	0.0591	0.0591	$x_2(t)$	$x_2(t)$
7	0.0636	0.0636	z_5	z_5
8	0.0649	0.0649	z_4	z_4
9	0.0661	0.0661	$x_7(t)$	$x_7(t)$

Table 5.7: Comparison of the prediction RMSE and the selection between unmodified functional LARS and modified functional LARS from Scenario 5.

Each of the data set in the 1000 replications are checked such that the largest correlation between the response variable and all 12 candidates falls onto one of relevant variables. The results from 1000 replicates are omitted here, because the outcome from the unmodified and modified functional LARS algorithms are identical to each other before the corresponding stopping points. The difference made by the modifications happens after all the relevant variables are selected. However, those differences have little influence on the results. Thus it further confirms that the modification is extremely unlikely to be triggered for any combinations of discrete and normalization methods before reaching the stopping point if no irrelevant variables are selected before the relevant variables. This also indicates that the threshold $\kappa = 0.1$ is a good choice for not removing the relevant variables.

Scenario 6

Scenario 6 has the same correlation structure as that in Scenario 5. The only difference is that one of the irrelevant variable $x_4(t)$ and $x_5(t)$ has the biggest correlation with the response variable among all the 12 variables. In other words, Scenario 6 is the extreme and rare case of Scenario 5. In order to obtain the required correlation structure,the data generated here are checked such that the largest correlations between the response and the 12 candidate variables falls on to $x_4(t)$ or $x_5(t)$. The correlation map is shown in Figure 5.9.

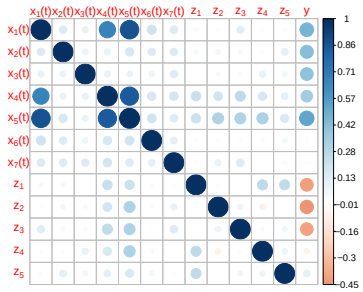


Figure 5.9: Correlation map of a simulated data set from Scenario 6

For this data set, unmodified and modified functional LARS give the different results before the algorithms reach the corresponding stopping points suggested by *AD*. The estimates of the parameters from unmodified functional LARS with the *RDP* and identity normalization, FGL_P, FGL_B are shown in Figure 5.10. All the other estimated parameters from unmodified and modified algorithms are shown in Figure 5.E.1 and Figure 5.E.2 respectively in Appendix 5.E.

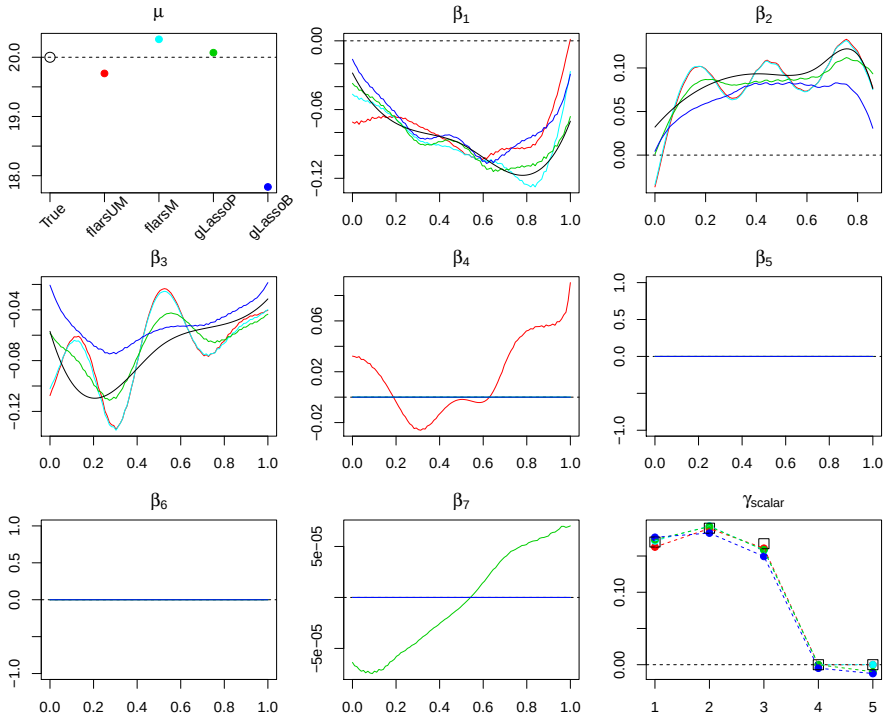


Figure 5.10: Estimated parameters from unmodified and modified $flars$, FGL_B and FGL_P.

Table 5.8 shows the results from unmodified and modified algorithm. The irrelevant func-

Iteration	RMSE		Selection	
	Unmodified	Modified	Unmodified	Modified
iter1	0.4761	0.4761	$x_4(t)$	$x_4(t)$
iter2	0.4605	0.4605	$x_1(t)$	$x_1(t)$
iter3	0.4134	0.4134	$x_2(t)$	$x_2(t)$
iter4	0.3900	0.4040	z_1	z_1(drop $x_4(t)$)
iter5	0.3576	0.3478	z_2	z_2
iter6	0.2868	0.2752	$x_3(t)$	$x_3(t)$
iter7	0.0641	0.0621	z_3	z_3
iter8	0.0630	0.0609	$x_6(t)$	$x_6(t)$
iter9	0.0636	0.0608	$x_7(t)$	$x_7(t)$

Table 5.8: Comparison of the prediction RMSE and the selection between unmodified functional LARS and modified functional LARS from Scenario 6.

tional variable $x_4(t)$ is selected before all the other variables. The modification removes one irrelevant variable $x_4(t)$ from the regression equation at iteration 4. As expected, the prediction RMSE from modified functional LARS is smaller than that the from unmodified version after all the true variables are selected for inclusion in the regression equation.

It indicates that the threshold $\kappa = 0.1$ is a reasonable value for this data set. From our experience, we tend to choose a small value of κ. When κ is large, it is more likely to trigger the modification and remove variables. Thus when κ is large, it is possible that some of the true variables could be removed and are never selected again.

Scenario 6 is also replicated 1000 times. Since Scenario 6 is actually the extreme and rare case of Scenario 5, we have approximately a 6% chance to get one data set that satisfy the requirements. Table 5.9 shows the summary of the results from these replicates. As usual, functional LARS algorithm using rank as the normalization has most unsatisfying performance among all the algorithms in the table. By using the modification, we can successfully obtain correct variables in about 60% of cases. However, the prediction RMSE is not improved much immediately after all the variables are selected. On the other hand, the selection made by FGL_P and FGL_B are not as accurate as expected. Normally these two algorithms select irrelevant variables for inclusion in the regression equation. The prediction RMSE from FGL_P is better in this case.

		RMSE		Correct selection (%)	
		Unmodified	Modified	Unmodified	Modified
RDP	Norm	0.0784	0.0834	18.90	54.33
	Trace	0.0886	0.0935	18.90	51.81
	Rank	0.1781	0.1500	9.29	33.86
	Identity	0.0816	0.0910	15.59	59.69
GQ	Norm	0.0799	0.0862	14.17	65.98
	Trace	0.0854	0.0888	14.17	61.73
	Rank	0.1651	0.1382	8.19	39.21
	Identity	0.0828	0.0874	10.39	65.51
BF	Norm	0.0777	0.0850	0.00	61.73
	Trace	0.0869	0.0937	0.00	56.06
	Rank	0.1637	0.1375	0.00	35.91
	Identity	0.0778	0.0889	0.00	63.15
FGL_P		0.0615		0.31	
FGL_B		0.1142		10.55	

Table 5.9: Comparison of the prediction RMSE's and the selection between unmodified functional LARS and modified functional LARS from Scenario 6.

Scenario 7

In this scenario, we generate 50 functional variables and 50 scalar variables without any constraints on the correlations between the variables. The true model remains the same, i.e., the first three functional variables and the first three scalar variables are the true variables. As the number of variables is large, we omit the correlation map in this case.

We show the results from the 1000 replications in Table 5.10. The best outcome is from the functional LARS with the *RDP* method and normalised by the norm or the identity matrix. The outcomes from the *BF* with the same normalizations are also very good. The difference between the unmodified and modified functional LARS is small. The functional LARS normalized by rank still gives unsatisfied results. The computational time for FGL_P is very long. Practically, when the number of variables is large, FGL_P is not a good choice. In addition, both FGL_P and FGL_B selected a large number of irrelevant variables. Note that the percentage of the unexpected variables is calculated based on 94 irrelevant variables in the candidates.

		RMSE—		Expected (%)		Unexpected (%)		Time (sec)	
		Unmodified	Modified	Unmodified	Modified	Unmodified	Modified	Unmodified	Modified
RDP	Norm	0.0665	0.0645	99.23	98.85	0.18	0.23	14.9329	14.8827
	Trace	0.0902	0.0884	94.02	94.36	0.28	0.31	13.7672	13.8482
	Rank	0.3235	0.3240	58.94	57.71	0.65	0.67	10.8376	10.9249
	Identity	0.0662	0.0645	99.54	99.35	0.22	0.23	15.6785	15.6053
GQ	Norm	0.0777	0.0690	98.53	98.27	0.55	0.62	3.6820	3.9109
	Trace	0.0911	0.0820	95.54	95.84	0.67	0.72	3.4859	3.7339
	Rank	0.3158	0.3111	58.78	62.09	1.84	1.54	3.9560	4.1507
	Identity	0.1166	0.1054	92.42	94.98	1.34	1.22	3.6994	3.8699
BF	Norm	0.0674	0.0730	99.17	96.43	0.28	0.49	5.1893	5.7727
	Trace	0.1120	0.1078	90.87	91.59	0.82	0.89	5.0884	5.5577
	Rank	0.3120	0.3183	60.11	57.19	1.30	1.27	5.0217	5.2892
	Identity	0.0755	0.0813	98.21	95.13	0.52	0.73	5.2248	5.5784
FGL_P		0.1101		99.99		43.57		1718.6690	
FGL_B		0.3683		54.66		13.81		8.7383	

Table 5.10: Comparison of the prediction RMSE's and the selection between all algorithms from Scenario 7.

5.4 Conclusion and discussion

We extend two existing functional variable selection methods to mixed scalar and functional variable selection methods in this chapter. But our focus is on the functional LARS with additional normalizations. The functional LARS algorithm works better than the other two extended algorithms when there are both scalar and functional variables in the candidates.

The algorithms are compared with the model selected using the corresponding stopping rules. For functional LARS normalised by rank, the outcome is always worse than others. This is because the stopping rule *AD* tends to stop the algorithm before the relevant variables are all selected. Thus, even though functional LARS normalized by rank is able to select relevant variables priorly, it is not a good choice to use while using *AD* to find the stopping point. Among four normalization methods, Frobenius norm and identity matrix are the best ones. The selection made by using functional LARS with *RDP* method is generally more accurate than other discrete methods. For FGL_P and FGL_B, 5-fold cross validation is applied. The combination of this criteria and the variable selection algorithms normally select with all the relevant variables and a few more irrelevant variables. The extra irrelevant variables disturb the prediction outcome. In addition, by

only using the basis functions to control the smoothness, FGL_B would not give accurate estimations of the functional coefficients, neither would the prediction using the estimated coefficients.

The other aspect we considered in the simulation study is the computation cost. The most expensive algorithm is FGL_P, due to the tuning of two tuning parameters with cross validation. The functional LARS algorithms are generally fast, except for the ones using the *RDP* method. The computation cost for using *RDP* method as the discrete method is certainly expensive due to the very high dimension of the functional variables.

In the first five scenarios, the performance of functional LARS is generally much better than the ones from FGL_P and FGL_B. In the last scenario of the simulation, we look at one extreme case with complex correlation structure. Functional LARS algorithm is able to avoid selecting correlated variables for inclusion in the model in most of the cases. Additionally, because the robustness of the algorithm, we can stop the algorithm one of two iterations later in the worst cases to help facilitate convergence.

Appendix

5.A Lasso and shooting algorithm

Lasso targets on the variable selection and parameter estimation in the linear model with scalar variables:

$$y = \beta_0 + \sum_{j=1}^{J} x_j \beta_j + \epsilon, \tag{5.18}$$

where x_j is the j-th variables, β_j is the corresponding coefficient. We can always assume that the the response variable and the covariates are centred and normalized, so that the intercept $\beta_0 = 0$ and $\|x_j\|_2 = 1$ for all j. The variable selection and the parameter estimation is done by the following penalized least square:

$$G = (y - \sum_{j=1}^{J} x_j \beta_j)^2 + \lambda \sum_{j=1}^{J} |\beta_j|, \tag{5.19}$$

where λ is the tuning parameter.

Shooting algorithm is proposed by Fu (1998), designed to solve lasso by iterative method. It is replaced by LARS later for faster computation. The key idea of shooting algorithm is to find the solution of β_j in Eqn (5.19) conditional on the values of all the other coefficients. The process is repeated for $j \in 1, \ldots, J$ until all the β_j converge.

Suppose $J = 1$, Eqn (5.19) can be rewritten as:

$$G = (y - x\beta)^2 + \lambda|\beta|, \tag{5.20}$$

Define $s = \partial|\beta|/\partial\beta$. $|\beta|$ is not differentiable when $\beta = 0$. Thus by subgradient we have:

$$\begin{cases} \|s\|_2 \leq 1 & \beta = 0; \\ s = \frac{\beta}{\|\beta\|_2} = \text{sign}(\beta) & \text{otherwise.} \end{cases}$$

Then we use Lagrange multiplier, and get:

$$\frac{\partial G}{\partial \beta} = -x^T y + x^T x\beta + \lambda s = -x^T y + \beta + \lambda s, \tag{5.21}$$

where $x^T x\beta = \beta$ because x is normalized to have $\|x\|_2 = 1$.

Set $\frac{\partial G}{\partial \beta} = 0$, we have:

$$x^T y - \beta = \lambda s. \tag{5.22}$$

s should have the same sign as β, thus the estimation of β is:

$$\hat{\beta} = x^T y - \lambda \text{sign}(\beta)$$

When β is small, and we have:

$$\|x^T y - \beta\|_2 < \lambda,$$

the estimation of β would be shrunk to zero. Thus $\hat{\beta}$ can be written as:

$$\hat{\beta} = \text{sign}(\beta)(|x^T y| - \lambda)^+,$$

where $(a)^+$ is equivalent to $\max(a, 0)$.

If $J > 1$, the above process is repeated through all variables. The response y is replaced by

$$r_j = y - \sum_{j^* \in \{1,\ldots,J\}, j^* \neq j} x_{j^*} \beta_{j^*},$$

and the solution becomes:

$$\hat{\beta}_j = \text{sign}(\beta)_j(|x^T r_j| - \lambda)^+$$

For a value of λ, the above solution is calculated from each $j \in 1, \ldots, J$, until the values of β_j converge.

5.B Group lasso and the solution from shooting algorithm

The variable selection and the parameter estimation is done by the following penalized least square:

$$G = (y - \sum_{j=1}^{J} \mathbf{x}_j \boldsymbol{\beta}_j)^2 + \lambda \sum_{j=1}^{J} (\boldsymbol{\beta}_j^T K_j \boldsymbol{\beta}_j)^{1/2}, \tag{5.23}$$

where λ is the tuning parameter, K_j is the kernel matrix for variable j. Initially the design matrices $\mathbf{x}_j$ and the kernel matrices K_j have no special form, so the same idea from the shooting algorithm can be used here. Suppose $J = 1$, we have:

$$G = (y - \mathbf{x}\boldsymbol{\beta})^2 + \lambda(\boldsymbol{\beta}^T K \boldsymbol{\beta})^{1/2}. \tag{5.24}$$

Define $\mathbf{s} = \partial(\boldsymbol{\beta}^T K \boldsymbol{\beta})^{1/2} / \partial \boldsymbol{\beta}$. By subgradient, we have:

$$\begin{cases} \|s\|_2 \leq \frac{(\boldsymbol{\beta}^T K K \boldsymbol{\beta})^{1/2}}{(\boldsymbol{\beta}^T K \boldsymbol{\beta})^{1/2}} & \boldsymbol{\beta} = \mathbf{0}; \\ s = \frac{K\boldsymbol{\beta}}{(\boldsymbol{\beta}^T K \boldsymbol{\beta})^{1/2}} & \text{otherwise.} \end{cases}$$

By Lagrange multiplier:

$$\frac{\partial G}{\partial \boldsymbol{\beta}} = -\mathbf{x}^T y + \mathbf{x}^T \mathbf{x}\boldsymbol{\beta} + \lambda \mathbf{s} = \mathbf{0}. \tag{5.25}$$

This equation is difficult to solve, since we cannot separate the parameter β out easily. It can be solved by optimization methods. In the multi-group-variable situation, the solution of j-th coefficient $\boldsymbol{\beta}_j$ is also from this equation, with respect to each of the groups of variables. Similar to the scalar case, the response becomes:

$$r_j = y - \sum_{j^*=1, j^* \neq j}^{J} \mathbf{x}_{j^*} \boldsymbol{\beta}_{j^*}.$$

A sequential optimization algorithm is used to find $\boldsymbol{\beta}_j$ for all j in Bakin et al. (1999). It is certainly impractical when the number of variables is large.

5.C Scenario 2, 3 and 4

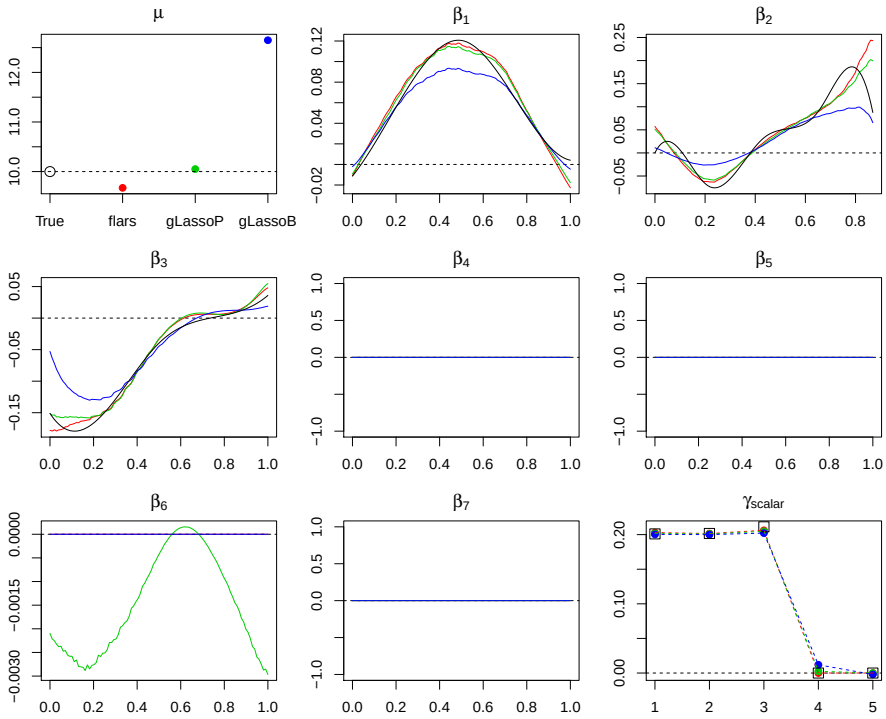


Figure 5.C.1: Estimated parameters from $flars$, FGL$_B$ and FGL$_P$. Black lines are the true functional coefficients; red lines and points are the estimations of the coefficients from $flars$; green lines and points are the estimations of the coefficients from FGL$_P$; blue lines and points are the estimations of the coefficients from FGL$_B$. The colours of the lines and points remains the same meaning through out all the simulations.

	RMSE	expected	unexpected
$flars$	0.0640	6/6	0/6
FGL$_P$	0.0661	6/6	3/6
FGL$_B$	0.1180	6/6	2/6

Table 5.C.1: The estimation accuracy and the selection accuracy from three algorithms.

5.D Scenario 5

5.E Scenario 6

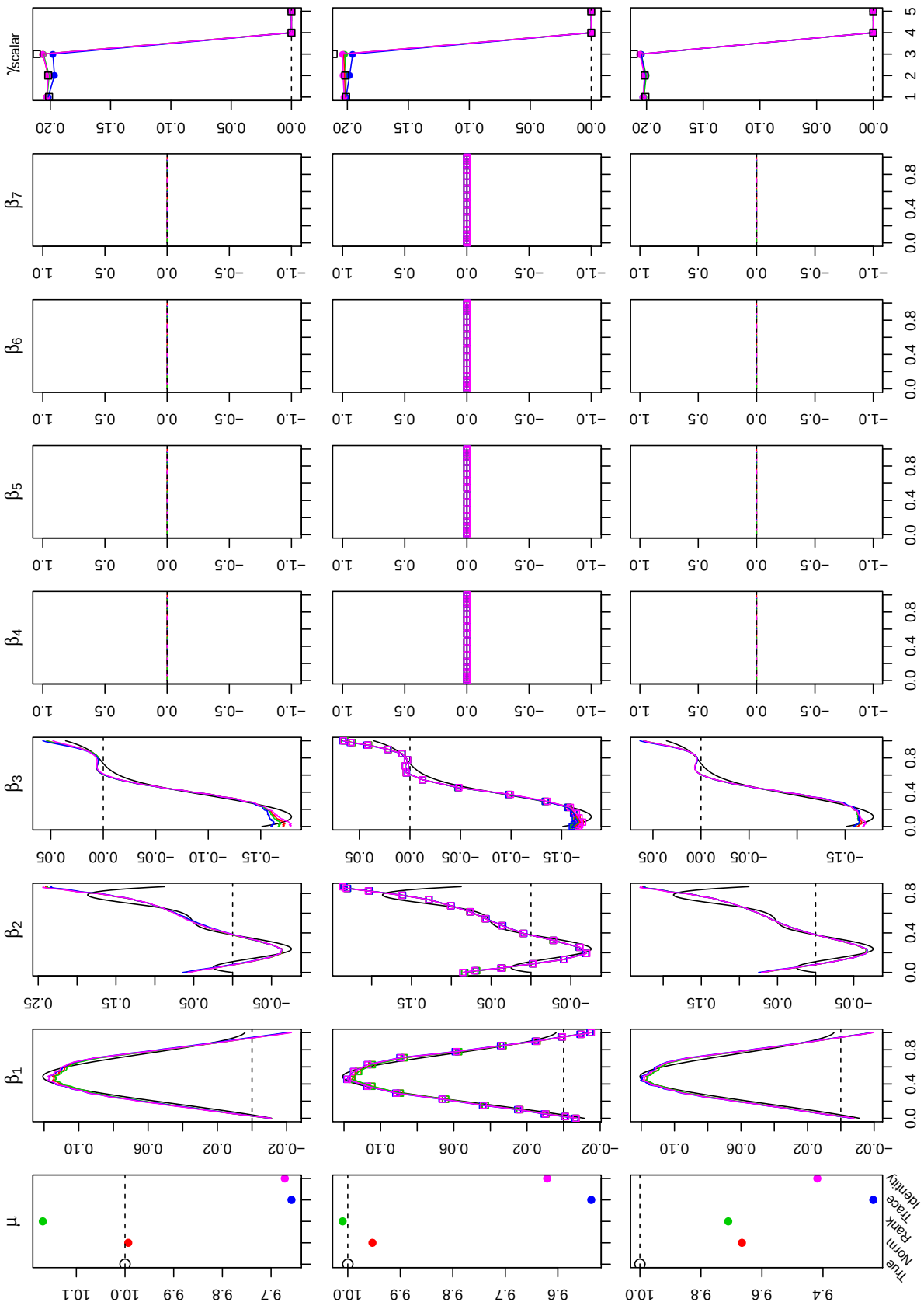


Figure 5.C.2: Comparison between different discrete methods and normalization methods.

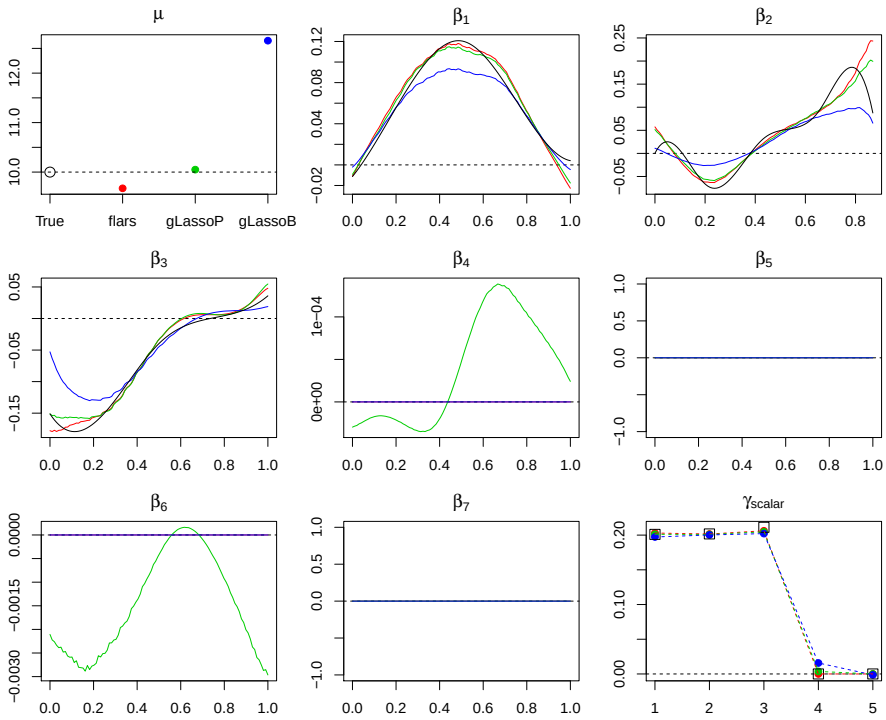


Figure 5.C.3: Estimated parameters from $flars$, FGL_B and FGL_P.

	Norm	Rank	Trace	Identity
RDP	0.0631	0.0627	0.0643	0.0640
GQ	0.0639	0.0643	0.0653	0.0638
BF	0.0612	0.0614	0.0611	0.0606

Table 5.C.2: RMSE from different discrete methods for functional coefficients and different normalization methods.

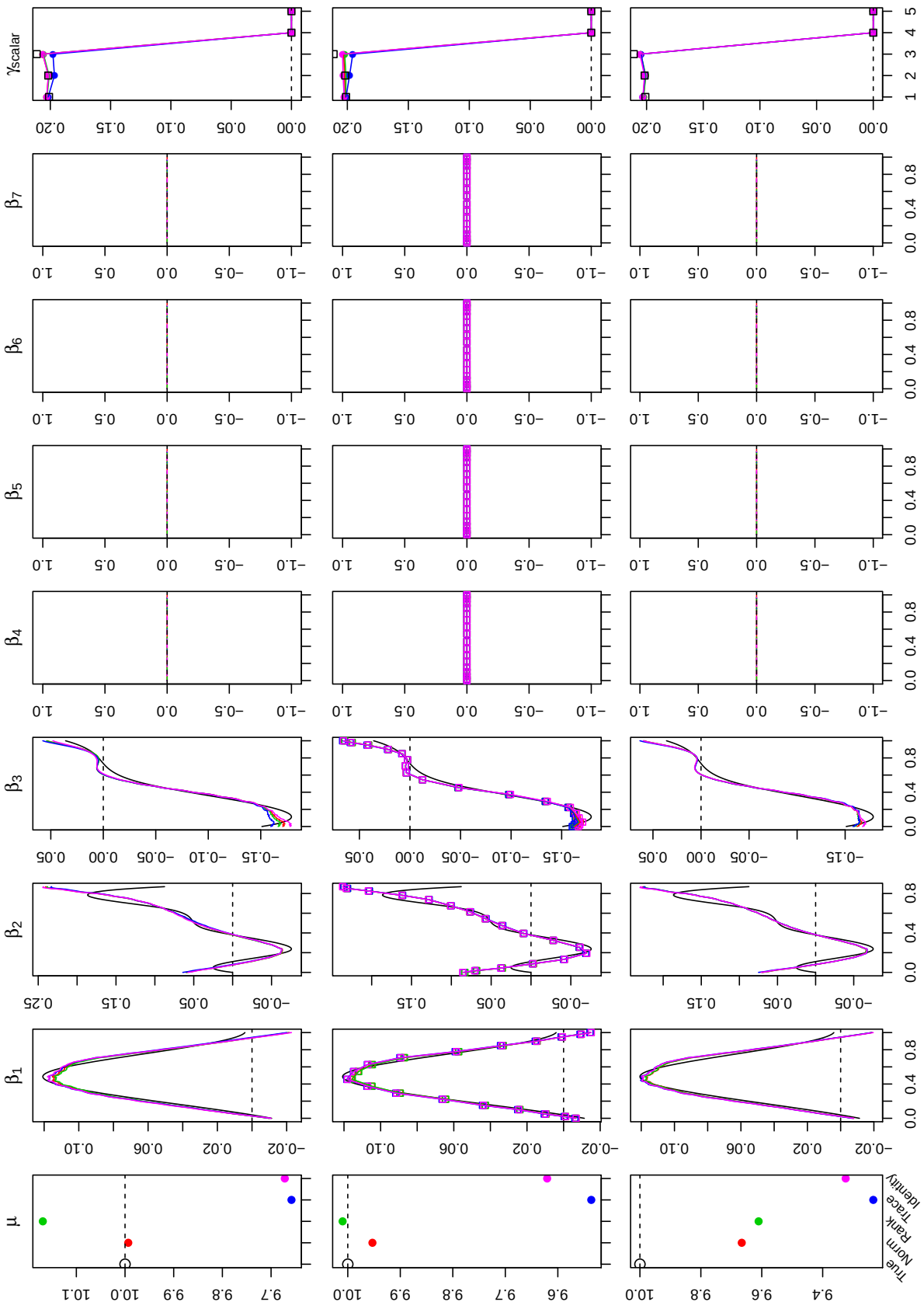


Figure 5.C.4: Comparison between different discrete methods and normalization methods.

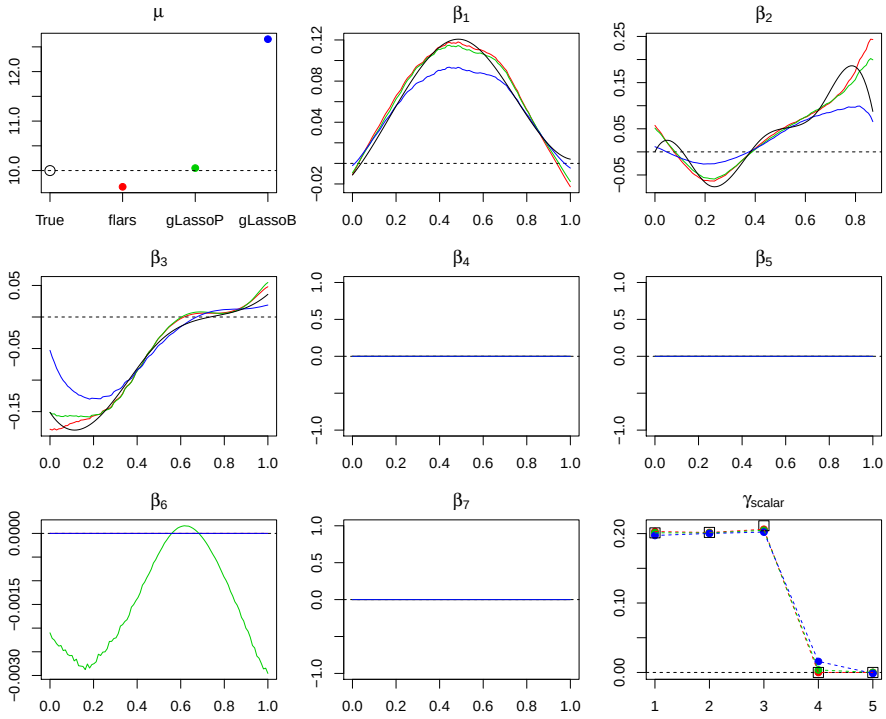


Figure 5.C.5: Estimated parameters from $flars$, FGL$_B$ and FGL$_P$.

			RMSE	Expected (%)	Unexpected (%)	Time (sec)
flars	*RDP*	Norm	0.0585	100	0.00	3.1276
		Trace	0.0613	99.44	0.00	2.9690
		Rank	0.1696	77.25	0.53	2.0865
		Identity	0.0591	100	0.00	3.8419
	GQ	Norm	0.0612	100	0.02	0.5040
		Trace	0.0652	99.48	1.08	0.4884
		Rank	0.1519	81.11	0.40	0.5145
		Identity	0.0615	100	0.03	0.4744
	BF	Norm	0.0585	99.90	0.12	0.6509
		Trace	0.0686	98.59	3.79	0.6459
		Rank	0.1388	83.15	0.56	0.6454
		Identity	0.0583	100	0.08	0.6252
FGL$_P$			0.0615	100	51.03	271.5067
FGL$_B$			0.1165	100	23.61	1.2410

Table 5.C.3: Summaries of functional LARS with different discrete and normalization methods, FGL$_P$ and FGL$_B$.

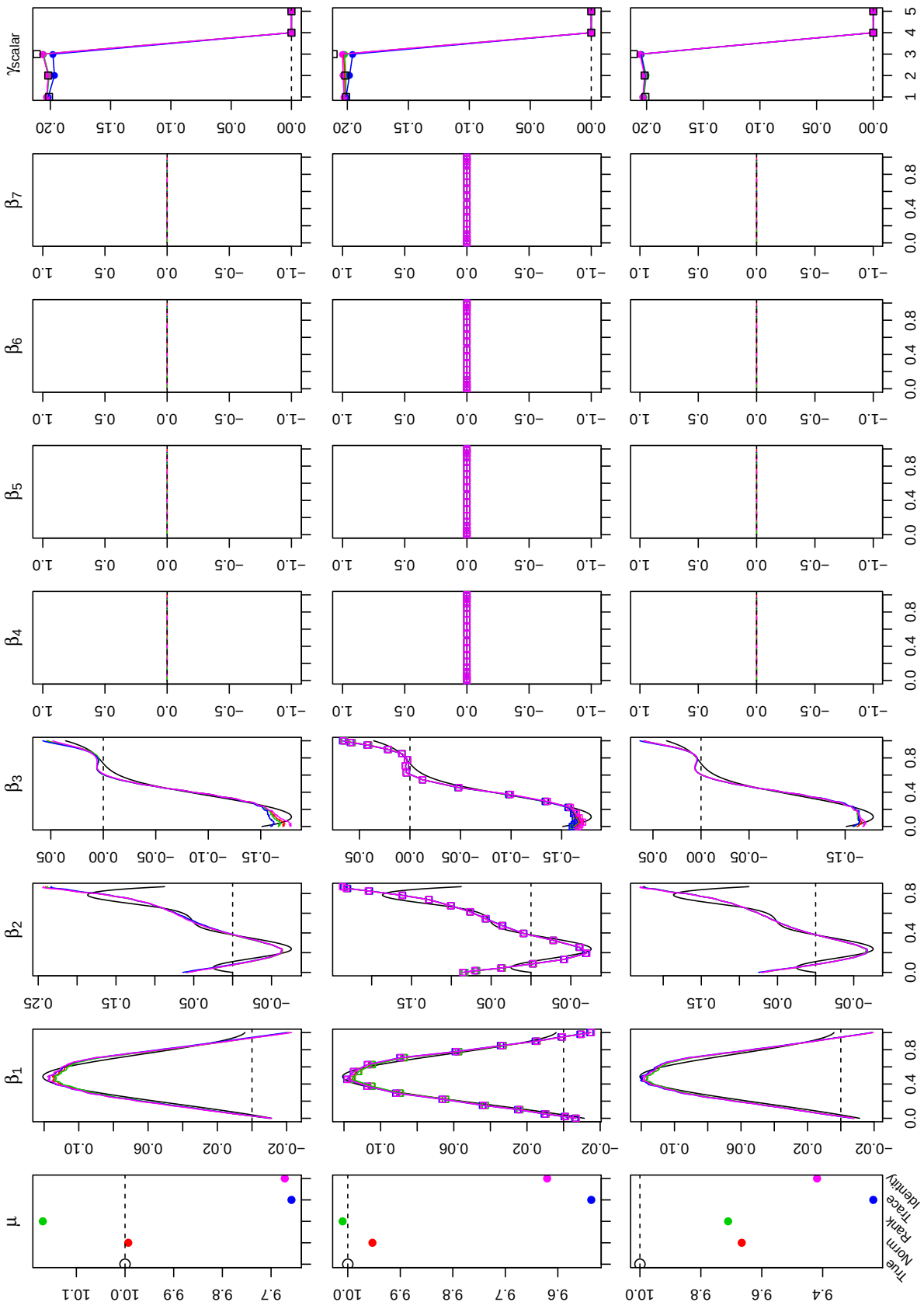


Figure 5.C.6: Comparison between different discrete methods and normalization methods.

	RMSE	expected	unexpected
$flars$	0.0646	6/6	0/6
FGL_P	0.0661	6/6	2/6
FGL_B	0.1180	6/6	2/6

Table 5.C.4: The estimation accuracy and the selection accuracy from three algorithms.

	Norm	Rank	Trace	Identity
RDP	0.0634	0.0630	0.0638	0.0646
GQ	0.0642	0.0643	0.0653	0.0643
BF	0.0612	0.0612	0.0611	0.0601

Table 5.C.5: RMSE from different discrete methods for functional coefficients and different normalization methods.

			RMSE	Expected (%)	Unexpected (%)	Time (sec)
$flars$	*RDP*	Norm	0.0593	99.59	0.27	2.9560
		Trace	0.0637	98.67	0.30	3.1152
		Rank	0.1687	77.26	0.59	2.0014
		Identity	0.0600	99.33	0.59	3.4102
	GQ	Norm	0.0631	99.22	0.63	0.4995
		Trace	0.0674	98.37	1.52	0.4905
		Rank	0.1556	80.51	0.44	0.5132
		Identity	0.0641	99.11	0.79	0.4772
	BF	Norm	0.0597	99.10	0.81	0.6467
		Trace	0.0709	96.89	4.03	0.6390
		Rank	0.1449	82.10	0.54	0.6400
		Identity	0.0602	99.00	0.98	0.6243
FGL_P			0.0616	100	50.97	265.8056
FGL_B			0.1171	100	24.36	1.0849

Table 5.C.6: Summaries of functional LARS with different discrete and normalization methods, FGL_P and FGL_B.

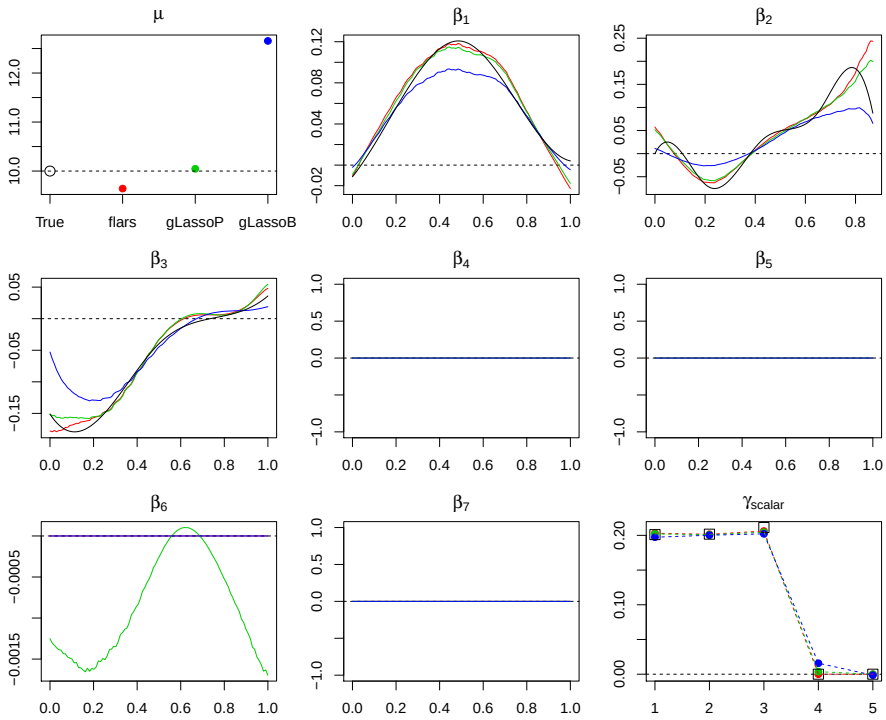


Figure 5.D.1: Estimated parameters from $flars$, FGL_B and FGL_P.

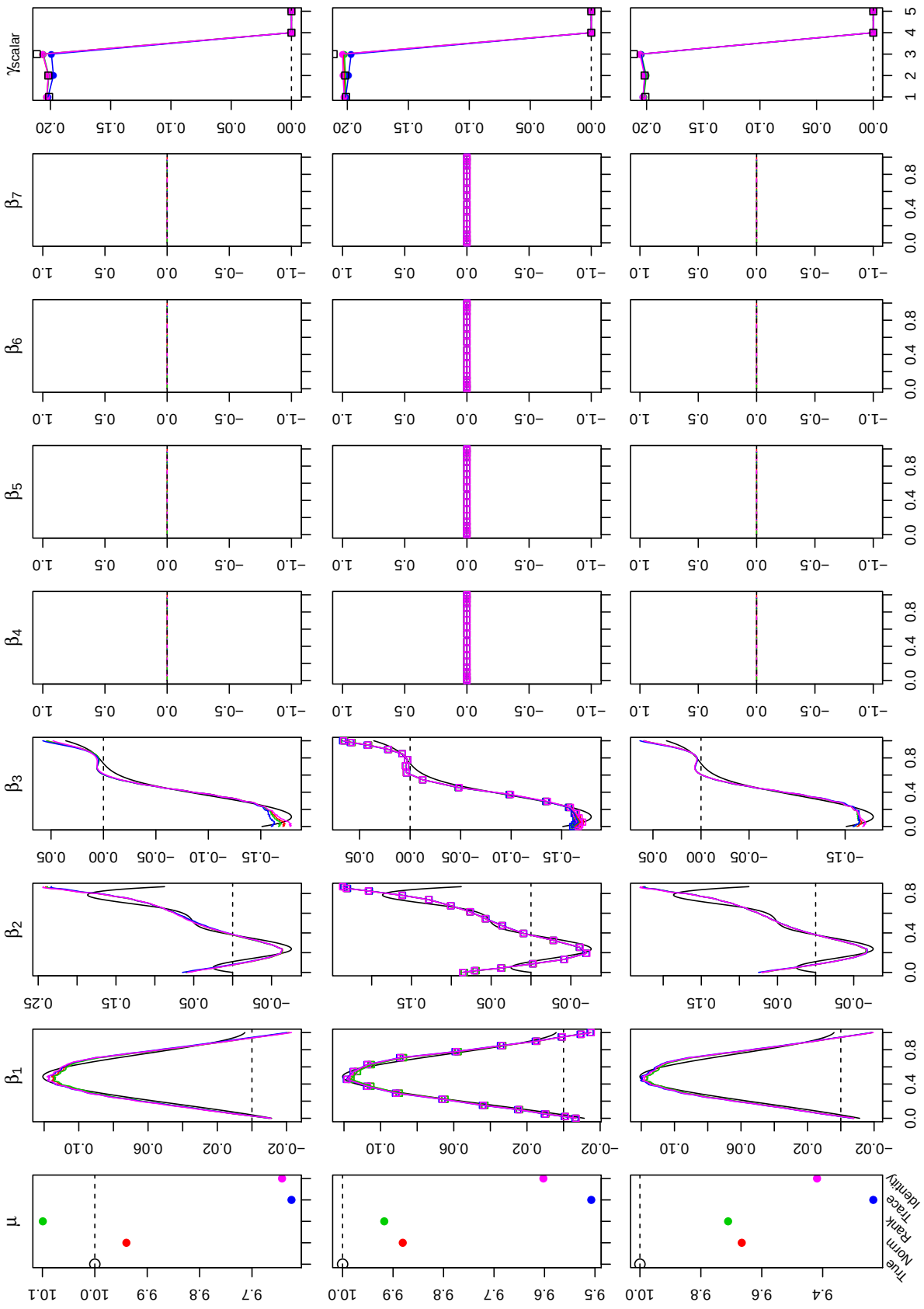


Figure 5.D.2: Comparison between different discrete methods and normalization methods.

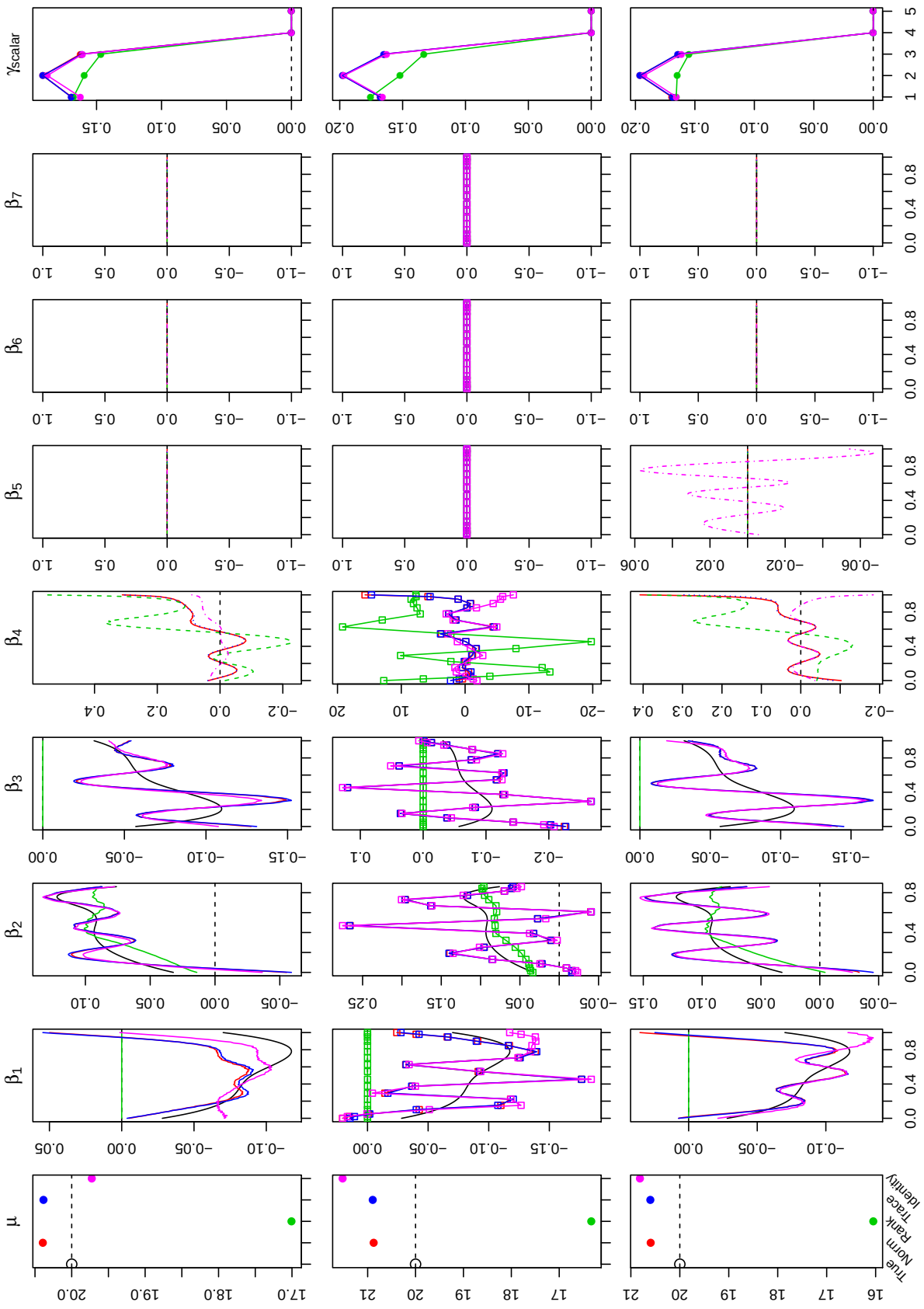


Figure 5.E.1: Comparison between different discrete methods and normalization methods.

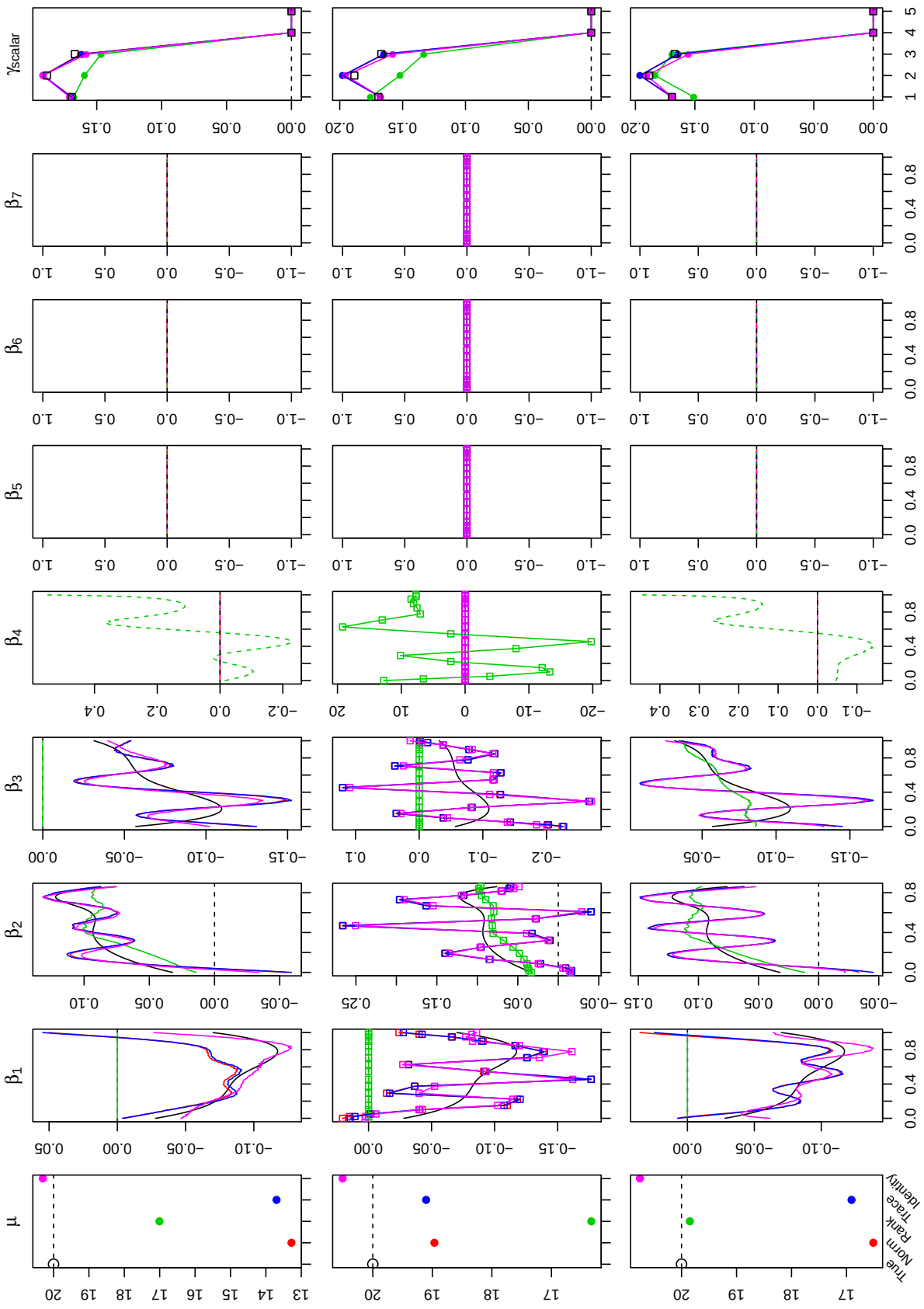


Figure 5.E.2: Comparison between different discrete methods and normalization methods.

Part II

Applications of Functional Regression Analysis in Motion Data

Chapter 6

Background and Preprocessing of the Motion Data

Stroke is a major health problem. Globally, it is one of the most common causes for both death and severe disability (Ian L. Dryden and Zhou (2009)). Advanced age can increase the risk of stroke (Lip et al. (2010)). As the world population is significantly ageing, the number of people suffer a first-time stroke is expected to increase from 16 million in 2005 to approximately 23 million by 2030 (Strong et al. (2007)). One of the common disabilities after surviving stroke is hemiparesis. This means the left or right side of the body is weak or loses its functions because of the brain injury caused by stroke. This is a highly common syndrome for stroke survivors. Among the acute patients, who had their first stroke within three month, 80% suffer hemiparesis (Party (2012)). This number reduces to 50-70% for chronic patients, who had their first stroke after six months (Centre for Disease Control (2012), Kelly-Hayes et al. (2003)).

Studies have confirmed that significant improvements can be achieved in upper limb function even for the patients who had stroke a few years ago (Laver et al. (2012)). But this requires intensive, repetitive and challenging practice and exercises (P. Langhorne (2009)). In addition, such treatment requires involvement of therapists. However, therapist time is a limited and expensive resourced compare to the large population of stroke patients (Party (2012)).

Similar rehabilitation programs can be carried at home by patients themselves. However, improvements without therapists are questionable (Touillet et al. (2010)). On the other hand, video games have been used for therapeutic purposes for a long time (Shi et al.

(2013)). It can bring fun and focus to the treatment carried by the patients themselves at home. The value of video games in such treatments has been proven by increasingly strong evidence. Primack et al. (2012) determines more than one thousand publications giving positive results up to February 2010.

In this chapter we introduce the detailed background of the motion data we are going to analyse, including the design of the trial in Section 6.1. Signal data and the preprocessing of signal data are in Section 6.2. After preprocessing the signal data, we obtain functional variables and scalar variables for the modelling in the next chapter.

6.1 Backgroup Information about the Motion Data

Subjects and the trial There were 70 stroke survivors without significant cognitive or visual impairment in the study. The inclusion criteria was that the patient should be able to understand the information on the screen, grasp with their paretic hand, move their paretic limb against gravity and do the most basic movements to calibrate the game. Patients have a wide range of levels of abilities in their upper limb functions. None of these patients have previously played video games. They were asked to participate in a home-based rehabilitation programme using the Circus Challenge video games over a three month period. There were no requirements for the lower limb ability of the participants. Thus, the patients can either stand or sit to play the game.

Data were collected using a longitudinal design. According to the plan, each patient should have eight assessments in three months time. The first four assessments were arranged weekly, and the following four were arranged fortnightly. For each patient, the first assessment gave the baseline dependence level only. Patients had the video game to play during their own time after the baseline assessment. In the following seven assessments, patients were visited by the therapists and both assessment game and assessment of the dependence level were carried out.

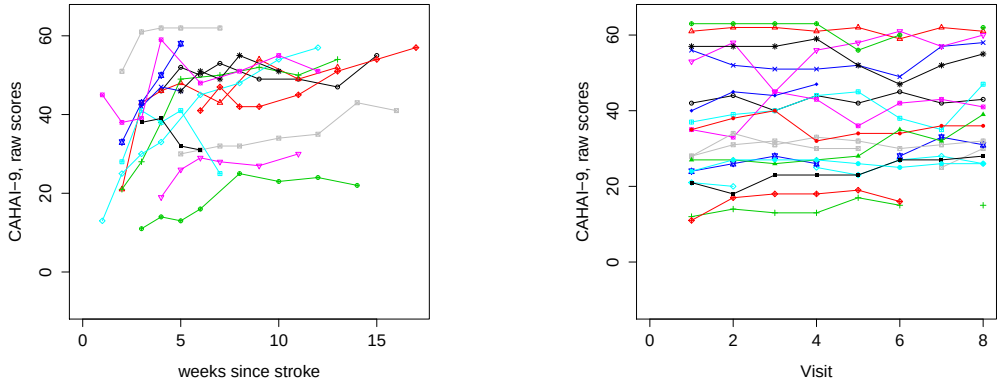
The video game Circus Challenge is a commercial game used to help patients recover the upper limbs functions. It was produced for the company Limbs Alive by the professional video game studio Pitbull (http://www.pitbullstudio.co.uk) and computer scientists from the project team. The video game contained a series of gaming scenarios within which players participate in a number of circus oriented activities by completing the re-

quired movements. The movements were designed and ordered by the therapists, in order to help patients recover during an efficient and smooth process. Patients must move as required by the game to score enough points to ensure successful completion of circus activities in the game.

The corresponding assessment game is called the Circus Challenge Assessment game. It is a simpler version of the full game. All the important movements in the commercial game are included in the assessment game and ordered roughly according to the difficulty levels. The data used in the model are from the assessment game rather than the full game. Further details about the data is in the following chapter.

The dependence level The dependence level means the level of dependency of independency of the patients in their daily life. The measurement used to quantify the patient's dependence level is called Chedoke Arm and Hand Activity Inventory, or CAHAI for short (`http://www.cahai.ca/`). The medical team in the project decided to use CAHAI-9 as the reference clinical standard. CAHAI-9 has nine different tasks from daily activities, such as dial 999 and open a jar. This assessment is a fully validated measure of upper limb functional ability (S. Barreca (2005)). Each task uses seven points to quantify the patients in general. A score of 7 means the patient is fully independent with regards to that task, while a score 1 means that the patients is fully dependent. This gives us a measure from 9 to 63 to represent the dependency level of the patients.

According to the time from stroke to first first assessment, patients included in the study are split into two phases: chronic and acute. Chronic patients had their first stroke six months time ago. Some patients had stroke years ago. Acute patients had their stroke within the last six months. Such separation is because of the different behaviour of recovery from acute patients and chronic patients. It is well known in the literature (P. Langhorne (2009)) that generally, stroke patients recover with a relatively high speed immediately after the stroke, and slow down over time. After half a year to one year, normally the recovery speed will reduce to a very low level. This is confirmed by this set of motion data. Figure 6.1 show the clinical CAHAI-9 values against time for acute patients and chronic patients respectively. Note that in the plots, the x axis for acute patients is the time since stroke, and for chronic patients is the visit time. This is because the time since stroke for chronic patients varies from a few months to a few years, which would be difficult to visualize if using time since stroke. It is clear to see that the acute patients have an increasing trend, while the trend for chronic patients is not obvious.



(a) Acute patients (b) Chronic patients

Figure 6.1: Dependence level (CAHAI) against assessment time

The movements The therapists designed many movements in the commercial game. The 40 most important movements were selected in the assessment game according to therapists experience. These movements can be categorized as four types:

Mirrored Two hands are required to do the same movement, at the same time, mirror each other with no phase lag.

In-phase Two hands are required to do the same movement, at the same time, but with a phase lag.

Sequential Two hands are required to do the same movement alternately. To be more specific, when one hand moves, the other hand is required to stay still with no requirement about the posture. The order of which hand moves first does not effect the result.

Coordinated Non-paretic hand is required to do a relatively complicated movement, while the paretic hand is required to stay still with a certain posture. This group has the hardest movements.

The most important movements are in the category “mirrored” and “in-phase”. This is because the last two types may be too difficult for the patients to carry out. Thus the data we have from the last two types of movements have less information than the ones from the first two types.

6.2 Device and the signal data

Our target is to model and predict the dependency level obtained from the clinical assessments. The prediction will based on the motion data from the assessment game played by the patients at home. These motion data are collected by 3-D tracking devices.

3-D tracking devices have been used in the research for a long time. The most accurate devices are the camera based devices, such as VICON system (`http://www.vicon.com/`). However, these devices are usually expensive, and have to be used in a laboratory environment but not at home. These accurate devices might require complex set-ups or calibrations before use every time. The Circus Challenge and the Circus Challenge Assessment game are both rely on the home-based portable device from the company Sixense (`http://sixense.com/`). This is one of the most state-of-the-art devices with wireless controllers in the video game market. It is very easy to use. Each set of devices contains one base unit connected to the computer to gain power and collect signals from the controllers. This base unit provides a coordinate system in 3-dimensional space. The origin is the base unit itself, with x axis going from left to right, y axis going from down to up and z axis going from front to back. In our case, each set of devices also contains three controllers: one for left hand, one for right hand and one on the belt round the waist. The three controllers provide signals of the relative position in the 3-D space with respect to the base unit during the game. The controllers use power from AAA batteries. The controller has been designed for commercial video games, and provides greater accuracy than any other game input device on the market. The technology powering the controller is based on magnetic motion tracking (Hansen (1987)). The signals are not as accurate as camera based devices, such as VICON. Because of the nature of the magnetic field, the readings are not linearly correlated to the distance from the base unit, especially when the target is far away from the base unit. However, we can get different readings when the positions with respect to the base unit are different. According to the validation done in Shi et al. (2013), this system is fairly accurate and approximately linear in the range that covers most patients can reach. The other disadvantage using this system is that the signals might be influenced by big metal objects such as radiators, if the device is too close to them. According to therapists' experience, the devices work well in a normal home environment.

The data from the controllers are 60 Hz high frequency data. Therefore the signals can be visualized as curves. For example, we show ten sample trajectories from different patients

doing the movement forward roll in Figure 6.1. This movement asks the player to raise the hands up to the chest hight, and draw a circle with diameter about half arm length vertical to the torso.

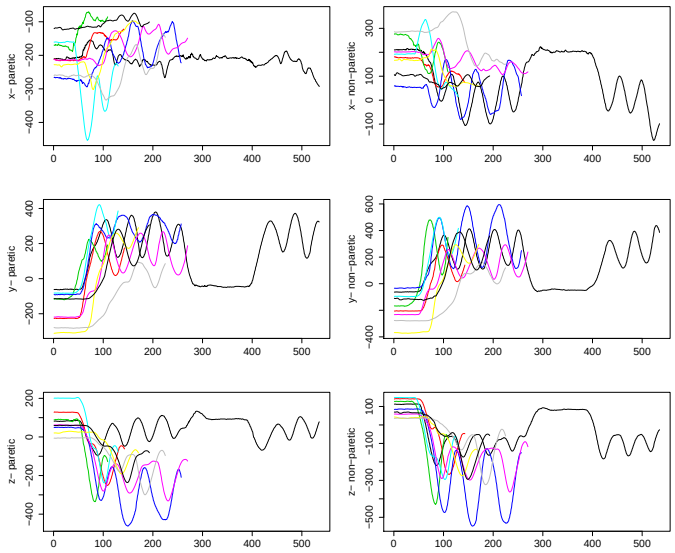


Figure 6.1: A few samples of the raw data on each of the axes and hands of the movement "forward roll" (LA05)

The signals contain information about both position and orientation of the controllers. For some movements, position data are more important than the orientation data, and vice verse for some others. The movement we show in Figure 6.1 is more about the position rather than the orientation.

6.2.1 Position data and calibration

It is clear that the position data are represented by the 3-dimensional signals from x, y and z axes with respect to the base unit. However, from both gaming and modelling points of view, the absolute distances from the base unit are not convenient to use, since they are effected by not only the movements of the hands, but also many other factors. Therefore, calibration and normalization are necessary before any game or data pretreatment for modelling.

Calibration is done by a set of the most basic postures. The postures include the closest and furthest reach when patients put the hands up, front and down. Figure 6.2 shows the ideal standing of the patients with respect to the base unit. Ideally, the player should stand

on the z axis one arm length away from the base unit, with the torso (segment between two controllers) parallel to the x axis.

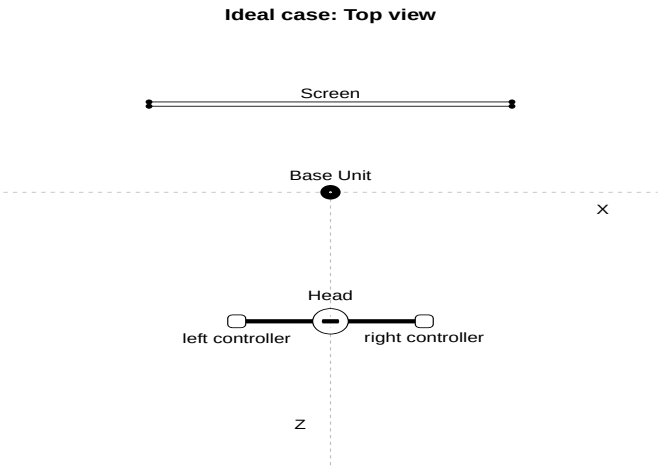


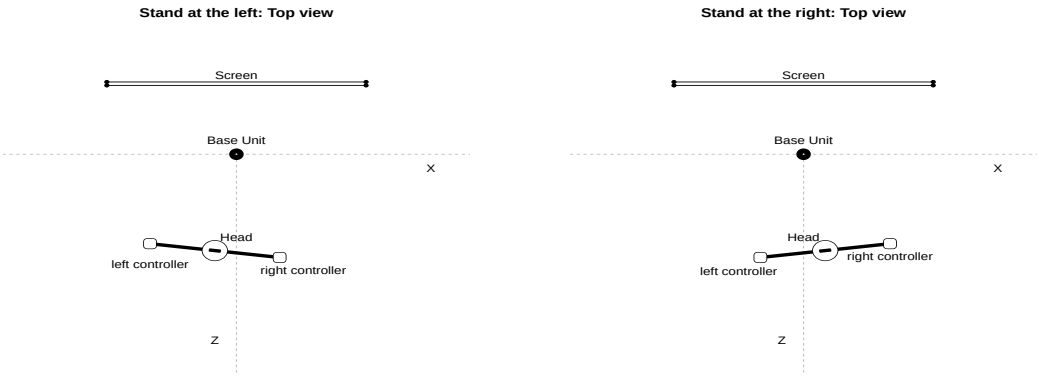
Figure 6.2: Top view of the ideal standing position. The coordinate system is centred at base unit. Dashed line is the x axis and z axis.

However, it is not possible to guarantee such standing practically. Patients may stand with a different angle and position, as shown in Figure 6.3a and Figure 6.3b. People may also move during the game. In addition, patients have different arm lengths. Thus, any model based on such data directly will lead to unreliable result. To solve these problems, we need to carry out the following steps to normalize the data: centring, standardizing and rotating.

Centring

By using the base unit as the reference point, the trajectories of the hands would be very complicated. One patient may also change their standing position during the one assessment game. However, if the reference becomes the shoulders, the movement will be constrained in a sphere of radius one arm length. Therefore, the first step is to change the reference point from base unit to the shoulders. This step will not change the shape of the trajectories, however, because there are no sensors on the shoulder, the position of the shoulders must be estimated.

We first transfer the reference point to the middle controller on the belt around patients waist to remove the effect from changing standing positions. This can be done by sub-



(a) One possible standing position (b) Another possible standing position

Figure 6.3: Possible standings of the players.

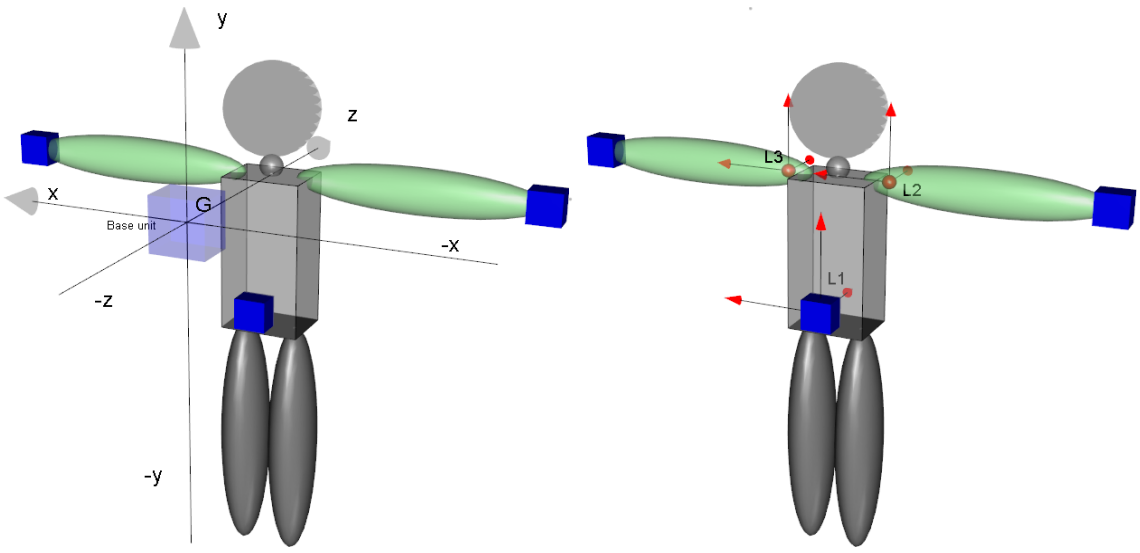
tracting the positions of the third controller from the positions of other two controllers:

$$(x_1^c(t), y_1^c(t), z_1^c(t)) = (x_1(t) - x_3(t), y_1(t) - y_3(t), z_1(t) - z_3(t))$$
$$(x_2^c(t), y_2^c(t), z_2^c(t)) = (x_2(t) - x_3(t), y_2(t) - y_3(t), z_2(t) - z_3(t))$$

where x, y, z are readings of the 3-D positions of the controllers. Subscripts are the indices of the controllers and the superscript ^{c} means the centred signals.

The calibration data give us the information to estimate the shoulder positions. When patients put their arms down to the lowest positions, we can have the x and z coordinates for the shoulders. The y coordinate can be obtained by the arm length, which is given in the clinical information of the patients. This posture is chosen because it is the easiest posture, and thus the most reliable posture for us to calculate the shoulder position. By some simple calculations, the shoulder positions can be obtained from this information. The hand positions can therefore be transferred to the sphere, with radius one arm length for further analysis.

These two transformations only shift the signals. Figure 6.4a and Figure 6.4b show the coordinate system before and after two transformations respectively. The arrows are the positive direction of each axis, the blue boxes are the controllers, the purple box is the base unit.



(a) Before transformation (b) After transformation

Figure 6.4: Three dashed lines and three gray arrows represent the coordinating system before any transformation. The purple box represents the base unit and the three blue boxes are the positions of the controllers. There are three smaller coordinating systems on two shoulders and the waist.

Standardizing

To make the position data more comparable, the values of the signals are standardized by dividing the arm length of the corresponding patient. The unit of the values are standardized to the number of arm lengths, and all the position data should be in the range from -1 to 1.

Rotating

As shown in Figure 6.3a and Figure 6.3b, the simple shifting in the 3-D space is not enough to make movements comparable across different samples. Rotation is also required. As an illustrative example, Figure 6.5 shows how to rotate the red coordinate to the green coordinate. The rotation happens on the x-z plane, so we look at the two dimensional plot here. For example, if the coordinate is rotated 30 degrees anti-clockwise, as shown in red, it should be rotated back to the green coordinate. In other words, all of the position data need to be rotated 30 degrees clockwise. We define this angle by using the furthest position that the patient can reach on their left and right sides. These positions can be found in the calibration movement and in some assessment movements.

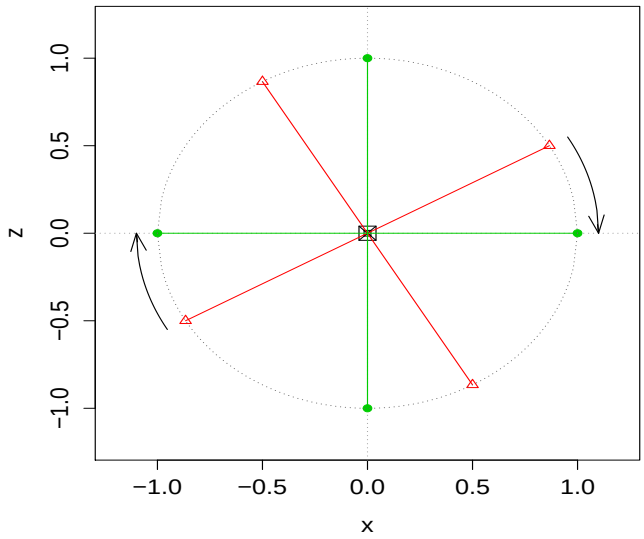


Figure 6.5: Rotation from the red triangle to the green triangle

6.2.2 Orientation data

The signals also include the orientation information of the controllers. Many different methods are used to describe the orientation data and the commonly used ones are Euler angles, rotation matrices, quaternions. The *Euler angles* are the easiest way to understand the rotation. The idea is that any orientation can be obtained by three rotations around each of the x, y, and z axes (Diebel (2006)). However, each orientation can be represented by more than one set of such rotations. On the other hand, it is less obvious how the rotation information is represented by a rotation matrix and a quaternion. Rotation matrices and quaternions are both used in the motion data.

Rotation matrix

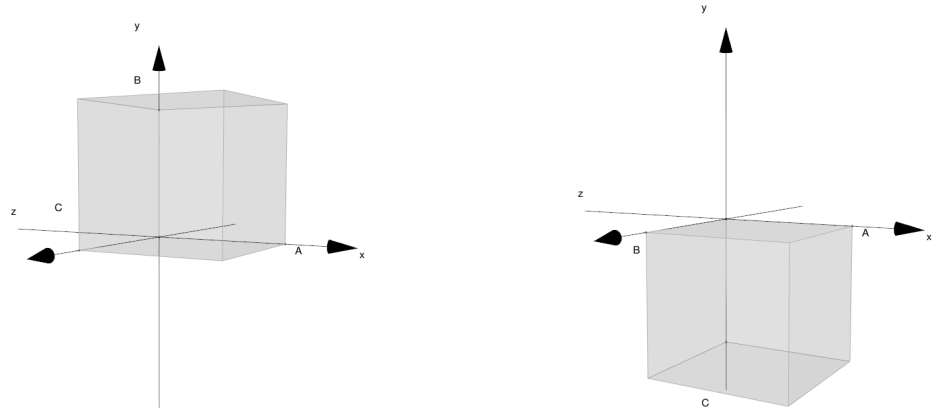
The rotation matrices representing the state of the rotation in this motion data are the simplified version with dimension 3 by 3. The idea is to rotate a cube with each edge of length 1 unit. As shown in Figure 6.6a, the cube is in the position such that vertices A, B and C have coordinates (1, 0, 0), (0, 1, 0), (0, 0 ,1),respectively. The vertex at the origin does not move in any case. This is the default state of the orientation. A rotation matrix is built with the coordinates of these three particular vertices, where each column is the coordinate of A, B and C in order corresponding to x, y, z axes, respectively. Thus the

rotation matrix for the default orientation is

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

If any rotation happens to this default cube, the coordinates of the vertices A, B and C will change, and give a new rotation matrix to represent the state of the new orientation. For example, Figure 6.6b shows a rotation of 180 degrees around the x axis from the default state of orientation. In this case, the rotation matrix is

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$$



(a) Default state of orientation

(b) Rotate around x axis for 180 degrees from default state of orientation

Figure 6.6: Illustrative example of a rotation matrix

Quaternion

It is clear that the rotation matrix requires nine numbers to store the information of rotation, which adds a large burden to the storage since the recording frequency is 60 Hz. There is another frequently used method to represent the orientation data, called quaternion.

A *quaternion* is a four dimensional vector, contains a vector and an angle related to the

rotation. Originally, a quaternion is a three dimensional complex number, containing one real number and three complex number on three different directions. It was first described by Sir William Rowan Hamilton in 1843. A quaternion can be written as:

$$q = (q_x, q_y, q_z, q_w) = (\mathbf{v}, q_w)$$

where $\mathbf{v}$ is a vector containing q_x, q_y, q_z. The vector $\mathbf{v}$ is the axis of the rotation, and $q_w = cos(\frac{r}{2})$, where r is the angle of rotation. In our case q is a unit vector. Since the quaternions represent the rotation of the controllers in this case, it is meaningless to consider the complex part of the quaternion. Figure 6.7 shows a right circular cone in three dimensional space, with vertex at the origin, vectors $(0, 0, 1)$ and $(-1, 0, 0)$ on the surface of the cone. Vectors $(0, 0, 1)$ and $(-1, 0, 0)$ are the direction of positive z and negative x axis. For illustration, we use an example quaternion to represent the rotation from positive z axis to negative x axis through the surface of the right circular cone. The rotation should be around the unit vector $(-\frac{\sqrt{2}}{2}, 0, \frac{\sqrt{2}}{2})$, and the angle is π. Therefore, the quaternion from $(0, 0, 1)$ to $(-1, 0, 0)$ is $(q_x = -\frac{\sqrt{2}}{2}, q_y = 0, q_z = \frac{\sqrt{2}}{2}, q_w = 0)$. It seems that the quaternion is the representation of the rotation rather than the orientation, but if we fix the starting vector, the orientation state can be represented by the rotation.

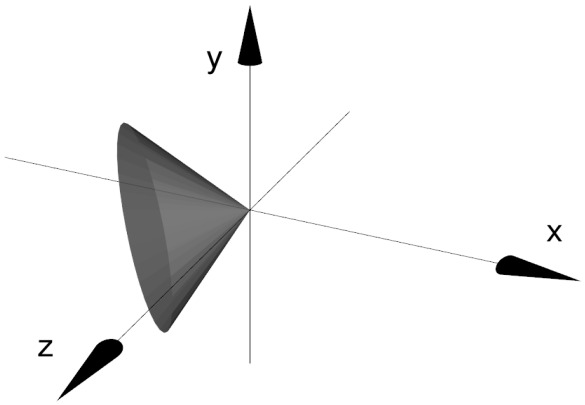


Figure 6.7: Illustrative example of quaternion.

A quaternion has its own rules of elementary arithmetic, but in our case, only quaternion multiplication has physical meanings. Suppose the unit quaternion is defined as $q = (q_x, q_y, q_z, q_w) = (v, q_w)$, with $q_x^2 + q_y^2 + q_z^2 + q_w^2 = 1$. Then multiplication between two unit

quaternion q_1 and q_2 can be written as:

$$\begin{aligned} q_1 \times q_2 &= (q_{1,x}, q_{1,y}, q_{1,z}, q_{1,w})(q_{2,x}, q_{2,y}, q_{2,z}, q_{2,w}) \\ &= (\mathbf{v}_1, q_{1,w})(\mathbf{v}_2, q_{2,w}) \\ &= (q_{1,w}\mathbf{v}_2 + q_{2,w}\mathbf{v}_1 + \mathbf{v}_1\mathbf{v}_2, q_{1,w}q_{2,w} - \mathbf{v}_1\mathbf{v}_2), \end{aligned}$$

where $\mathbf{v}_1\mathbf{v}_2$ is a number from the vector multiplication.

Compared to Euler angles and rotation matrix, quaternions are much simpler to use. It is often preferred for practical reason.

Usage of the orientation data

A rotation matrix has nine dimensions, since it represents the rotation by using the three dimensional position of three vertices. A quaternion has four dimensions, since it represents the rotation by using the three dimensional position of a reference vector and an angle. We can treat these signals equivalent to the signals from the 3D position data. However, directly applying the signals introduces a large number of variables to the analysis. In addition to the extra space for data storage, it is also difficult to interpret the meanings of each dimension of the signals. Most of the assessment orientation movements requires simple rotations on a two dimensional plane around the third axis. For example, the movement similar to rotating a door handle requires a hand to rotation around z axis and move on the x-y plane. We use the following two transformations to highlight the orientation information.

- **Direction angle**. The idea is from the direction cosine (Kuipers (2002)). Direction cosines are the cosines of the angles between the vector and the three coordinated axes, or the projection to each axis, if the vector is a unit vector. The direction angle is calculated by using one of the three direction cosines. We often choose the one from the plane on which rotation happens. It is easy to interpret the outcome. However, the cosine value is same for an angle with or without sign, therefore, the angles are limited in the range $[-\pi/2, \pi/2]$.
- **Projection angle**. This is an extension of the direction angle. Rather than using one direction cosine, the projection angle combines two of them from the plane on which rotation happens. In other words, the rotation is projected to the plane on

which rotation happens and calculate the angle in the range $[-\pi, \pi]$. It avoids the limit of range in direction angle, since two cosines can define a angle properly.

6.3 Functional Variables

As we can see in Figure 6.1, the shapes of the curves are very smooth. Because of the high sampling frequency, the data can be seen as densely sampled functions with small amount of observation error. However, there are a few defects in the data set. First of all, all the samples are recorded within a fairly long period, much longer than the necessary time to complete one replication of the movement. Thus the samples would include preparation movements and failed replications. Since we are only interest in one replication of the target movement, segmentation of the curves becomes necessary. Secondly, the time periods of completing one replication for one movement from different patients and different visits would be different, and thus different samples are not comparable. We need to transfer the curves to functional objects to do the analysis. Finally, we align the important features of the segmented and smoothed functional objects by registration.

Segmentation

The important features in the movements can often be represented by quadratic shapes on some of the signals. However, automatic segmentation is extremely unreliable here due to the complexity of the shapes from different samples. We therefore use automatic segmentation to remove the preparation part and segment the desired part of the curves manually. Take the data in Figure 6.1 as an example, according to the movement and the scale of the three axes, the majority of the movement is on y-z plane. We can use either the y or z axis as the reference to segment the curve. We show the segmented curves in Figure 6.1 using y from non-paretic hand as the reference for the whole data set.

Smoothing

We use smoothing to transfer the data into functional objects. As we mentioned above, different samples have different length. Because the segments we use are from very short time periods, and the sampling frequency is very high, we therefore assume that all the

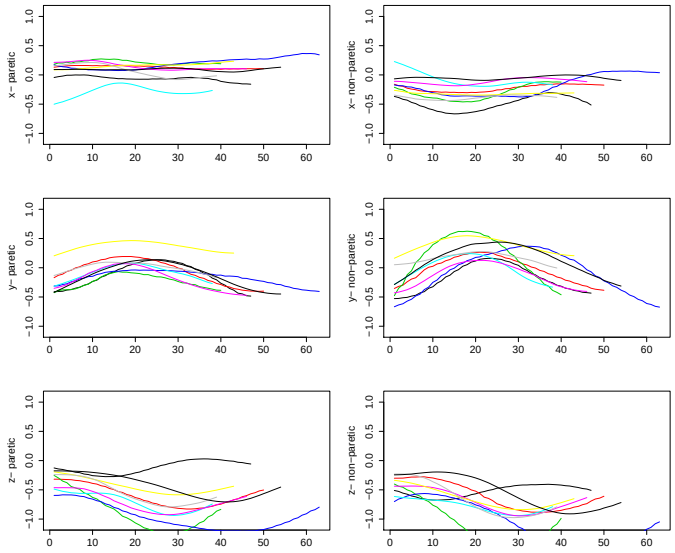


Figure 6.1: Segmented standardized data of a few samples from "forward roll" (LA05)

segmented curves are recorded using the same time period. More specifically, we assume $t \in [0, 1]$ for convenience.

The noise level of the signals are relatively low. Thus the level of smoothness we introduce to the functional objects should be low. We use penalized B-spline basis to calculate the basis coefficients. If the number of data points of a curve is k, then the number of basis function is $\max(k/3, 9)$, where $\max(a, b)$ means the maximum value of a and b. The order of the basis function is set to be 6 and the smoothing parameter corresponding to the roughness penalty is 10^{-7}. We show the smoothed segmented curve as 100 discrete points in Figure 6.2.

Registration

Because the shape of the functional variables are very simple, advanced registration methods seem to be over complicated here. We use the basic landmark registration (Ramsay and Silverman (2005)) to align the important features. We decide the important feature to the peaks or troughs on the quadratic shape. The registration is only done on the reference axis. The same change is then applied to all the other axes.

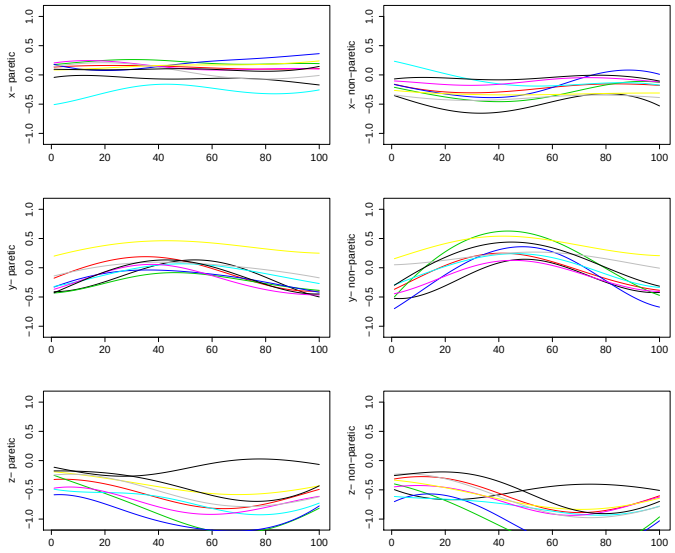


Figure 6.2: Smoothed segmented standardized data of a few samples from "forward roll" (LA05)

6.4 Scalar Variables

From the last section, we can see that it takes many steps including manual segmentation to prepare the functional variables. We therefore have summary statistics of the signals to represent part of the information in the data set. We still use automatic segmentation to remove as much useless information as possible. Unlike the functional variables, the summary statistics can be generated automatically.

6.4.1 Kinematic variables

According to the therapists, there are a few features that provide reliable and essential information related to the upper limbs functions: speed, synchrony and accuracy. The summary statistics of the curves or kinematic variables chosen for each of the movement are as follows:

1. Overall speed: let $p_t = (x_t, y_t, z_t)$, $t = 0, \ldots, t, \ldots, T$ be the vector of normalized 3D positions at time t; then T is the total time taken to perform the movement and $p_0 = (x_0, y_0, z_0)$ the starting position vector. The *displacement* distance at time t is given by the Euclidean distance between the current position and the starting position:

$$d_t = \|p_t - p_0\|_2 = \sqrt{(x_t x_0)^2 + (y_t y_0)^2 + (z_t z_0)^2}. \tag{6.1}$$

where $\|.\|_2$ means the L_2 norm or Euclidean distance. The vector formed with all these displacement distances follows as $d = (d_0, \ldots, d_T)$. Then, the *cumulative* distance travelled by the upper limb can be calculated as $D = \sum_{t=1}^{t=T} \|p_t - p_{t-1}\|_2$. The speed is more likely to be composed of summaries from the cumulative distance, which is $v = D/T$. Because this method uses the ratio between cumulative distance and the overall time as speed, multiple replicates appearing in the curves would have little effect to the outcome.

2. Synchrony between two hands: due to the different requirements in the movements, two summary statistics were used to account for synchrony:

   - For *mirrored* and $in-phase$ movements that requires activity from both hands at same time without (mirrored) or with (in-phase) a phase lag, the maximum or minimum cross-correlation between lags [5, 5] is used. The reason for using cross-correlation rather than basic correlation is because of the high sampling frequency of the signal. This statistic is robust with respect to the number of replicates in the curves. For the other two types of movements, cross-correlation is too sensitive. Therefore, it is more suitable to apply this statistics in mirrored and in-phase movements.
   - For *sequential* and *coordinated* movements, because these types of movements requires one hand stay still when the other hand moves, the amplitudes from two hands are more important. The standard deviation ratio is designed for this problem. It is defined as:

   $$\mathrm{SD}_{ratio} = \mathrm{SD}(d^P)/\mathrm{SD}(d^{NP}) \tag{6.2}$$

   where $\mathrm{SD}(d^P)$ means the standard deviation of the displacement distance from paratic hand. It is expected to give a small number from this statistics for these two types of movements. This statistics also works for mirrored and in-phase movements, as two hands are expected to have similar standard deviation of the displacement distance. It is expected to be close to 1 for mirrored and in-phase movements.

3. Accuracy - the Range-of-Movement (ROM) is used as a proxy. This is defined as

   $$ROM = range(d) = \max(d) - \min(d) \tag{6.3}$$

where max and min are the maximum and minimum respectively. Similarly, this simple statistic is also irrelevant to the multiple replicates in the curves.

6.5 Conclusion

We have shown the process of preparing functional and scalar variables from signal data in the chapter. First of all, we scale and transfer the signal data in 3D space without changing the shape of the signals. For functional variables, we use segmentation, smoothing and registration to transfer the signal data into functional objects. For scalar variables, we use kinematic variables obtained by summarizing the signals with mild segmentation only. We will use these variables in the regression model in Chapter 7.

Chapter 7

Modelling of the Motion Data

From Chapter 6, we have all the scalar variables and functional variables prepared. We now focus on the modelling of motion data using those variables. Among a large number of models we studied, we present a few typical models here, including models with only scalar variables and models with mixed scalar and functional variables. The latter ones would rely on functional LARS from the previous part of this thesis.

From the last chapter, we can see that the variation in the acute patients is much more significant and complex than the ones in chronic patients. Therefore, we focus on the modelling of acute patients here. Even though the models are built with respect to the acute patients, they can still be used to model the variation in the chronic samples.

We organize this chapter as follows. First we introduce the models we are going to compare in Section 7.1. Then we show the details about the model learning and model prediction together with a comparison of the outcome of the models for both acute and chronic patients in Section 7.2. The numerical comparison is in the Section 7.3.

7.1 Models for Recovery after Stroke

In the last chapter, we obtain a large number of candidate scalar and functional variables. Now our aim is to build a model that will explain the variation of the dependence level of stroke patients measured by CAHAI-9, and use this model to do the same prediction for

new patients. This model can be written as:

$$y_{i,k} = f(x_{i,k;1}(t), \ldots, x_{i,k;j}(t), \ldots, z_{i,k;1}, \ldots, z_{i,k;m}, \ldots) + \epsilon_{i,k}; \quad (7.1)$$
$$i = 1, \ldots, N_{\text{patient}};\ k = 1, \ldots, n_i;\ j = 1, \ldots, J;\ m = 1, \ldots, M$$

where the response variable y represents CAHAI-9; $x_{i,k;j}(t)$ is the j-th functional variable for patient i and visit k; z_m is the m-th scalar variable; n_i is the number of visits for the i-th patient.

7.1.1 Linear regression with scalar variable

If we assume that the samples are independent from each other, we can use the following linear regression model with scalar variables only:

$$y_{i,k} = \beta_0 + \sum_{m=1}^{M} z_{i,k;m}\gamma_m + \epsilon_{i,k}, \quad (7.2)$$

where β_0 is the intercept. We refer to this model as *lm*.

We select variables by lasso carried out by least angle regression. As we stated in the previous part of the thesis, LARS is an efficient, accurate and valid algorithm for variable selection and parameter estimation in linear regression problem. We are aware that there does not exist a unique set of variables that can consistently give us the best prediction for any subset of the samples and any kind of models. The variables selected in this linear regression model are also used later in the models with scalar variables.

7.1.2 Auto-regressive time series model with scalar variables

It is reasonable to assume that the samples from different patients are independent, but the samples recorded at different visits from the same patient should be dependent. We observe that the dependence level changes with time for acute patients. Thus we consider an auto-regressive time series model with lag 1 or AR(1) in addition to the linear regression model (7.2):

$$y_{i,k} = \beta_0 + \sum_{m=1}^{M} z_{i,k;m}\gamma_m + \alpha y_{i,k-1} + \epsilon_{i,k}, \quad (7.3)$$

where α is the coefficient for the AR model.

Recall that for each patient, the data are collected weekly in the first month and fortnightly in the following two months. Thus the time between observations is not always the same. We can use an example to illustrate the relationship between time since stroke, visit and weeks from Table 7.1. The unit of the numbers in the table is weeks. In addition to the planned time gaps, some patients might not play as planned. The missing visits give more variability to the time between observations for these patients.

t	5.4	6.4	7.4	8.4	9.4	10.4	11.4	12.4	13.4	14.4	15.4	16.4
k	1(baseline)	2	3	4	-	5	-	6	-	7	-	8
l	1(baseline)	2	3	4	5	6	7	8	9	10	11	12

Table 7.1: An illustrative example of the time for one acute patient. t is the time since stroke; k is the index of visit; l is the index of the actual week.

Most of the studies using time series model focus on evenly spaced time series, due to various reasons such as limited computation technologies (Eckner (2012)). The most commonly used way to deal with this is to transfer the unevenly spaced time series to evenly spaced time series by some types of interpolation (Adorf (1995)). Such interpolation would bring bias to the analysis when the observations of time series change rapidly. However, our response variable is the dependence levels. It would be impossible for the stroke patients to have large amount of change in dependence levels in a short period. Thus the interpolation would not bring too much noise to the analysis. In addition, instead of linear interpolation, we interpolate the missing observation using Gaussian process. This is because the kinematic variables should changes smoothly over time. Also GP is known to be good at interpolation estimation. More specifically, for each m, we have:

$$\begin{aligned} z_m =& g_m + \epsilon_m \\ g_m \sim& GPR(0, \kappa(l, l'; \boldsymbol{\theta}_m)); \qquad \epsilon_m \sim N(0, \sigma_m^2). \end{aligned} \tag{7.4}$$

We use the training data to learn this Gaussian process and interpolate the missing observation in both training data and testing data by predictions.

With all the missing observations interpolated, the model becomes:

$$y_{i,l} = \beta_0 + \sum_{m=1}^{M} z_{i,l;m}\gamma_m + \alpha y_{i,l-1} + \epsilon_{i,l}, \tag{7.5}$$

where the index l represents the number of weeks since stroke for acute patients and number of weeks since the baseline for chronic patients. The value of l should lie in $1, \ldots, 12$, but it could be larger for a small number of patients. We refer this model as tsN.

From Table 7.1, we can see that there may be no observations for some of l. In this case, we can recursively substitute $y_{i,l}$ by $y_{i,l-1}$ on the right hand side of Eqn (7.5) until we have the value of $y_{i,l-1}$:

$$\begin{aligned} y_{i,l} &= \beta_0 + \sum_{m=1}^{M} z_{i,l;m}\gamma_m + \alpha y_{i,l-1} + \epsilon_{i,l} \\ &= \beta_0 + \sum_{m=1}^{M} z_{i,l;m}\gamma_m + \alpha\left(\beta_0 + \sum_{m=1}^{M} z_{i,l-1;m}\gamma_m + \alpha y_{i,l-2} + \epsilon_{i,l-1}\right) + \epsilon_{i,l} \\ &= \ldots \\ &= \sum_{\delta=0}^{t_{i,l}-t_{i,k-1}-1} \alpha^{\delta}\left(\beta_0 + \sum_{m=1}^{M} z_{i,l-\delta;m}\gamma_m\right) + \alpha^{t_{i,l}-t_{i,k-1}} y_{i,t_{i,k-1}} + \sum_{\delta=0}^{t_{i,l}-t_{i,k-1}-1} \alpha^{\delta}\epsilon_{i,l-\delta}. \end{aligned} \tag{7.6}$$

Note that the subscripts on the right hand side in Eqn (7.6) are changed. This is under the assumption, with respect to l, that the previous available observation have index $k-1$. It means that we can predict the response at any time, even when we have no observations at that time, using the previous available observation and the interpolated values.

7.1.3 Mixed effects model with scalar fixed effect and Gaussian process random effects

The lag 1 auto-regressive model includes a linear regression part and a linear auto-correlation term. The dependency of the response observations from the same patients is captured by the linear auto-regressive parameter. It is shared across all the patients. We can generalize this model by a mixed effects model using a Gaussian process, where a Gaussian process can be used to capture the dependency between the observations from the same patient non -parametrically. More specifically, we assume the dependence level of any of the patients follows a Gaussian process, since the dependence level must change smoothly over time. The Gaussian process for all the patients shares the same set of hyper-parameters. This model is modified from the Gaussian process functional regression model proposed by Shi and Wang (2008).

We use some kinematic variables, time since stroke at each observations and patient's baseline dependence level in the Gaussian process random effect model. From many possible choices, we will report the results from three typical ones listed in Table 7.2.

ME_1	model	$y = \beta_0 + y^{(0)}\gamma^{(0)} + \mathbf{z}\boldsymbol{\gamma} + g(\boldsymbol{\phi}) + \epsilon$	
	parameters	$\boldsymbol{\phi}_{\text{fix}} = (1, y^{(0)}, \mathbf{z})$	$\boldsymbol{\phi} = t$
ME_2	model	$y = \beta_0 + y^{(0)}\gamma^{(0)} + g(\boldsymbol{\phi}) + \epsilon$	
	parameters	$\boldsymbol{\phi}_{\text{fix}} = (1, y^{(0)})$	$\boldsymbol{\phi} = (t, \mathbf{z})$
ME_3	model	$y = \beta_0 + y^{(0)}\gamma^{(0)} + g(\boldsymbol{\phi}) + \epsilon$	
	parameters	$\boldsymbol{\phi}_{\text{fix}} = (1, y^{(0)})$	$\boldsymbol{\phi} = (t, \mathbf{z}, \mathbf{z}^{(1)})$

Table 7.2: Three different mixed effects models. $g(\cdot)$ is the random effect part using Gaussian process.

In the table, β_0 is the intercept; $y^{(0)}$ is the baseline measure of each patients; $\gamma^{(0)}$ is the linear coefficient corresponding to the baseline measure; $\epsilon \sim N(0, \sigma^2)$. $\mathbf{z}^{(1)}$ is the movement measure from previous week. All three models in Table 7.2 contain a fixed effect and a random effect $g(\boldsymbol{\phi})$. We assume that

$$g \sim GP(0, \kappa(\boldsymbol{\phi}, \boldsymbol{\phi}'; \boldsymbol{\theta})),$$

where $\kappa(\cdot, \cdot)$ is the kernel function providing the covariance matrix for the Gaussian process and $\boldsymbol{\theta}$ is the hyper-parameter of the kernel function. We use the linear kernel and the squared exponential kernel to build the covariance matrix suggested by Shi and Choi (2011):

$$\text{Cov}_{l,l'} = \sum_{m=1}^{M} \alpha_m \phi_l^{(m)} \phi_{l'}^{(m)} + \nu \exp\left\{ -\frac{1}{2} \sum_{m=1}^{M} \omega_m (\phi_l^{(m)} - \phi_{l'}^{(m)})^2 \right\}.$$

Model ME_1 only uses time t in the random effect. It is a straight forward extension from the linear regression model Eqn (7.2). Model ME_2 moves the kinematic variables to the random effect part. Thus the fixed effect model has only the baseline measurement. Model ME_3 adds the kinematic variables from the previous week to the random effect model. Similar to Eqn (7.4) in the time series model, the data from the previous week may not exist in the observation, and are augmented by interpolation using the Gaussian process approach discussed in Section 7.1.2.

7.1.4 Mixed effects model using both scalar and functional variables

The models we have discussed so far use scalar variables only. The motion data also contain many functional variables. In this subsection, we consider mixed effect models using both scalar and functional variables. We do not use the time series model for functional variables, because it requires us to interpolate missing functional samples, which we are not able to do it at the moment.

As in the case of the previous mixed effects model, we have many choices about the variables in the fixed effect and random effect respectively. We still use scalar variables in the random effect part, but add functional variables to be fixed effect part.

fFE	model	$y = \beta_0 + y^{(0)}\gamma^{(0)} + t\gamma^{(t)} + \mathbf{z}\boldsymbol{\gamma} + \sum_{j=1}^{J} \int x_j(t)\beta_j(t)dt + \epsilon$	
	parameters	$\boldsymbol{\phi}_{\text{fix}} = (1, y^{(0)}, t, \mathbf{z}, x_j(t))$	-
fME_1	model	$y = \beta_0 + y^{(0)}\gamma^{(0)} + \mathbf{z}\boldsymbol{\gamma} + \sum_{j=1}^{J} \int x_j(t)\beta_j(t)dt + g(\boldsymbol{\phi}) + \epsilon$	
	parameters	$\boldsymbol{\phi}_{\text{fix}} = (1, y^{(0)}, \mathbf{z}, x_j(t))$	$\boldsymbol{\phi} = (t)$
fME_2	model	$y = \beta_0 + y^{(0)}\gamma^{(0)} + \sum_{j=1}^{J} \int x_j(t)\beta_j(t)dt + g(\boldsymbol{\phi}) + \epsilon$	
	parameters	$\boldsymbol{\phi}_{\text{fix}} = (1, y^{(0)})$	$\boldsymbol{\phi} = (t, \mathbf{z})$
fME_3	model	$y = \beta_0 + y^{(0)}\gamma^{(0)} + \sum_{j=1}^{J} \int x_j(t)\beta_j(t)dt + g(\boldsymbol{\phi}) + \epsilon$	
	parameters	$\boldsymbol{\phi}_{\text{fix}} = (1, y^{(0)})$	$\boldsymbol{\phi} = (t, \mathbf{z}, \mathbf{z}^{(1)})$

Table 7.3: One fixed effect functional regression model and three mixed effects models. $g(\cdot)$ is the random effect part using a Gaussian process.

Four models are listed in Table 7.3. The first model fFE is a fixed effect model while others are mixed effects model. Similar to the models with scalar variables only, the variables are selected from the fixed effect model.

The functional variables and the kinematic variables used in the models are selected in the linear fixed effect model by functional LARS. We have a large number of functional variables and scalar variables in the data, but it is impossible to put all the functional variables in the selection at the moment. This is because that we cannot interpolate missing values for any functional variable. We can only use complete samples to do the analysis. Thus if we put all of the variables together, we may end up with only a few samples with complete observations. The variables used in the scalar fixed effect model will not necessarily get

selected again. Detailed selection outcomes will be shown in Section 7.3.

The model fME_1 uses time t in the random effect, and all the others in the fixed effect. The model fME_2 uses time t and the scalar movement variable $\mathbf{z}$ in the random effect and the others in the fixed effect. The model fME_3 added the previous scalar movement variable to the random effect in Model fME_2.

7.2 Model Learning and Predicting

We showed the models and the corresponding reasons for using them in the last section. We will discuss the details about training and predicting procedures for each of the models with respect to both acute patients and chronic patients in this section.

Due to the missing data problem with functional variables, we reduce the number of candidate variables and remove a few samples. For acute patients, there are 173 samples from 34 patients with 72 functional variables from 10 movements and 68 kinematic scalar variables from 17 movements; for chronic patients, there are 196 samples from 36 patients with 60 functional variables from 8 movements and 68 kinematic scalar variables from 19 movements. In addition to the functional variables and the kinematic variables, we also include the time since stroke and the baseline measurement in the candidate variables.

7.2.1 Linear regression with scalar variables

The scalar variables used in the linear regression are selected by lasso calculated by LARS. The stopping points are selected based on Mallow's C_p. For acute patients, we select seven kinematic variables from six movements and for the chronic patients we select eight kinematic variables from eight movements. Both of the models include baseline dependence level and the acute model also selects time since stroke.

After we have selected the variables, we learn the models using only the training data and get the estimated parameters by ordinary least square. These parameters are then used to provide the predictions with respect to the rest of the data.

7.2.2 Auto-regressive time series model

The AR(1) model is a non-evenly spaced time series model. We only fit non-evenly spaced model for acute patients, as the time since stroke for chronic patients may range from a few months to a few years. Thus assuming they have the same auto-regressive structure is not sensible.

The parameters in non-evenly spaced time series model are learned by using a maximum likelihood estimator. Recall the model in Eqn (7.6), we have:

$$y_{i,l} = \sum_{\delta=0}^{t_{i,l}-t_{i,k-1}-1} \alpha^{\delta} \left(\beta_0 + \sum_{m=1}^{M} z_{i,l-\delta;m} \gamma_m \right) + \alpha^{t_{i,l}-t_{i,k-1}} y_{i,t_{i,k-1}} + \sum_{\delta=0}^{t_{i,l}-t_{i,k-1}-1} \alpha^{\delta} \epsilon_{i,l-\delta}. \tag{7.7}$$

In order to learn the model, we replace all the l's by k's in the subscript, as values related to l are unknown.

To simplify the formula, we define $\Delta_{i,k} = t_{i,k} - t_{i,k-1}$, $E_{i,k} = \alpha^{\Delta_{i,k}}$, $u_{i,k} = y_{i,k-1}$, $\mathbf{A}_{i,k} = \sum_{\delta=0}^{\Delta_{i,k}-1} \alpha^{\delta} \left(1 + \sum_{m=1}^{M} z_{i,k-\delta;m}\right)$. Thus Eqn (7.6) can be rewritten as:

$$y_{i,k} = \mathbf{A}_{i,k}\boldsymbol{\gamma} + E_{i,k} u_{i,k} + \epsilon^*_{i,k}$$

where $\epsilon^*$is the error. We assume that $\epsilon \sim N(0, \sigma^2)$ and $\epsilon_{i,k}$ is independent with $\epsilon_{i',k'}$. Thus

$$\epsilon^*_{i,k} = \sum_{\delta=0}^{\Delta-1} \alpha^{\delta} \epsilon_{k-\delta} \sim N\left(0, \frac{\alpha^{2\Delta_{i,k}} - 1}{\alpha^2 - 1}\sigma^2\right),$$

and

$$y_{i,k} \sim N\left(\mathbf{A}_{i,k}\boldsymbol{\gamma} + E_{i,k} u_{i,k}, \frac{\alpha^{2\Delta_{i,k}} - 1}{\alpha^2 - 1}\sigma^2\right).$$

We also define $D_{i,k} = \frac{\alpha^{2\Delta_{i,k}-1}}{\alpha^2-1}$.

The log-likelihood is

$$\ell = \sum_{i,k} \left[\log\left(\frac{1}{\sqrt{2\pi D_{i,k}\sigma^2}} \right) - \frac{(y_{i,k} - \mathbf{A}_{i,k}\boldsymbol{\gamma})^2}{2D_{i,k}\sigma^2} \right].$$

It can be rewritten into matrix form:

$$\ell = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log\sigma^2 - \frac{1}{2}\log|D| - \frac{(y - \mathbf{A}\boldsymbol{\gamma} - Eu)^T D^{-1} (y - \mathbf{A}\boldsymbol{\gamma} - Eu)}{2\sigma^2}, \tag{7.8}$$

where n is the sample size in total.

Using a Lagrange multiplier, we have

$$\frac{\partial \ell}{\partial \boldsymbol{\gamma}} = 0; \quad \frac{\partial \ell}{\partial \alpha} = 0; \quad \frac{\partial \ell}{\partial \sigma} = 0,$$

but we are only interested in $\boldsymbol{\gamma}$ and α. To simplify the problem, we replace σ^2 by $\exp v$, and then calculate $\partial \ell / \partial v$ instead of $\partial \ell / \partial \sigma$.

For $\boldsymbol{\gamma}$ we have:

$$\frac{\partial \ell}{\partial \boldsymbol{\gamma}} \propto \mathbf{A}^T D^{-1}(y - \mathbf{A}\boldsymbol{\gamma} - Eu).$$

Thus

$$\hat{\boldsymbol{\gamma}} = (\mathbf{A}^\mathbf{T}\mathbf{D}^{-\mathbf{1}}\mathbf{A})^{-1}(\mathbf{A}^T D^{-1}(y - Eu)). \tag{7.9}$$

We transfer σ to a function of v to simplify the calculation. Since σ is not the parameter of interest, this change does not affect the outcome. So we have:

$$\frac{\partial \ell}{\partial \boldsymbol{\gamma}} = -\frac{n}{2} + (y - \mathbf{A}\boldsymbol{\gamma} - Eu)^T D^{-1}(y - \mathbf{A}\boldsymbol{\gamma} - Eu) \exp(-v).$$

Thus

$$\hat{v} = \log \frac{2(y - \mathbf{A}\boldsymbol{\gamma} - Eu)^T D^{-1}(y - \mathbf{A}\boldsymbol{\gamma} - Eu)}{n}. \tag{7.10}$$

The partial differential for α is difficult to calculate and write. Fortunately we can write $\hat{v}$ and $\hat{\boldsymbol{\gamma}}$ by α only, so we can use profile likelihood to find the estimations of the parameters.

The learning of this model can include the following two steps:

1. interpolate the covariates in the missing weeks using the observed values by a Gaussian process;
2. use profile likelihood to calculate the estimates of the parameters.

The predicting is also done with two steps:

1. interpolate the covariates in the missing weeks using the observed values by a Gaussian process learned from the training data;

2. calculate predictions using the interpolated data and the estimated parameters.

Note that for the i-th patient, $\hat{y}_{i,k}$ depends on $\hat{y}_{i,k-1}$. Thus the prediction procedure must be done iteratively.

7.2.3 Mixed effects model with scalar variables

We have three mixed effects models with scalar variables. Recall that ME_1 has baseline dependence level and the scalar movements variables in the fixed effect, with only time in the random effect. We move the kinematic variables into the random effect for ME_2. In addition to ME_2, we also have the kinematic variables from previous week in the random effect.

Mixed effects model ME_1

Recall that the usage of a Gaussian process as random effect in the model is based on the assumption that patients dependence level changes smoothly with time. For the i-th patient, the model can be written as:

$$\begin{aligned} y_i &= \mathbf{z}_i\boldsymbol{\gamma} + g(\mathbf{t}_i) + \boldsymbol{\epsilon}_i \\ g(\mathbf{t}_i) \sim GP(0, \kappa(\mathbf{t}_i, \mathbf{t}_i'); \boldsymbol{\theta}) &\quad \epsilon_{i,k} \sim N(0, \sigma^2) \end{aligned} \tag{7.11}$$

The covariance of the Gaussian process is built by the kernel κ. We use the linear kernel and the squared exponential kernel to build the covariance matrix as before. The log likelihood for the i-th patient is:

$$\ell_i = -\frac{n_i}{2}\log(2\pi) - \frac{1}{2}|\Sigma_i| - \frac{1}{2}(y_i - \mathbf{z}_i\boldsymbol{\gamma})^T\Sigma_i^{-1}(y_i - \mathbf{z}_i\boldsymbol{\gamma}).$$

To simplify the problem, we can put the error, ϵ, in the Gaussian process, and treat the variance σ as one additional hyper-parameter. The posterior distribution is the log likelihood with respect to all the patients times the hyper-prior distributions. If we assume all the hyper-parameters follow uniform distribution, the posterior distribution is:

$$p(\boldsymbol{\theta}, \boldsymbol{\gamma}|\mathcal{D}) \propto \sum_i \ell_i. \tag{7.12}$$

We use maximum a posterior (MAP) to estimate the hyper-parameters.

Since the Gaussian process is the random effect with mean zero, we can learn this model iteratively. More specifically, we use ordinary least squares to estimate the coefficients in the fixed effect, then use the residual to estimate the hyper-parameters using MAP. We repeat these two steps until convergence. However, since the estimations from the later iterations barely changes the result, we only use the estimates from the first iteration as the outcome.

Thus the learning of this model has two steps:

1. learn the fixed effect with ordinary least squares and obtain $\hat{\boldsymbol{\gamma}}$,
2. use the residual from the fixed effect to learn the Gaussian process, and obtain $\hat{\boldsymbol{\theta}}$.

Similarly the prediction also has two steps:

1. use $\hat{\boldsymbol{\gamma}}$ to give prediction for $\hat{y}_{fix}$,
2. use $\hat{\boldsymbol{\theta}}$ to predict the $\hat{y}_{random}$, and sum up the two parts to have $\hat{y}$.

Mixed effects models ME_2 and ME_3

The other two mixed effects models ME_2 and ME_3 only differ slightly to the model ME_1 from learning and predicting points of view. Model ME_2 moves all the kinematic variables to the random effect part while model ME_3 also has the samples from previous week of the kinematic variables in the random effect part in addition to the model ME_2. Thus the training and predicting for model ME_3 also requires interpolation of the kinematic variables from the missing weeks.

7.2.4 Mixed effects model with mixed scalar and functional variables

We consider four models with mixed scalar and functional variables. The variables in the model are selected via functional LARS from model fFE. All the functional objects are calculated using the *RDP* method.

Fixed effect model fFE

Similar to the scalar case, we select variables in the following functional linear fixed effect model:

$$y = \beta_0 + y^{(0)}\gamma^{(0)} + t\gamma^{(t)} + \mathbf{z}\boldsymbol{\gamma} + \sum_{j=1}^{J} \int x_j(t)\beta_j(t)dt + \epsilon.$$

Functional LARS is used here to select both scalar variables and functional variables. The stopping points for both acute patients and chronic patients are based on the suggestions from the two stopping rules we proposed in the Chapter 4, i.e., AD and the difference in α. Table 7.1 shows the values from the two stopping rule corresponding to the iteration number.

Acute	Iter	2	3	4	5	6	7	8	9	10	11
Unmodified	AD	8.38	13.77	12.87	6.61	7.17	13.30	2.78	11.56	0.08	0.03
	$\alpha_{\times 1000}$	141.36	213.35	179.92	88.31	94.76	174.22	36.37	148.83	0.87	0.36
Modified	AD	8.38	13.77	12.87	6.15	10.05	5.38	9.56	13.25	0.29	0.22
	$\alpha_{\times 1000}$	141.36	213.35	179.92	88.31	135.65	71.06	133.26	176.10	3.48	2.89
Chronic	Iter	2	3	4	5	6	7	8	9	10	11
Unmodified	AD	55.44	3.39	3.60	1.91	4.36	2.35	3.06	0.00	0.04	0.00
	$\alpha_{\times 1000}$	830.40	48.27	47.60	24.90	55.23	29.73	38.50	0.03	0.43	0.02
Modified	AD	55.44	3.39	3.60	1.91	4.36	2.29	3.48	0.02	4.53	0.25
	$\alpha_{\times 1000}$	830.40	48.27	47.60	24.90	55.23	29.73	44.03	0.21	57.31	3.23

Table 7.1: The changes of the stopping criteria with iterations for both unmodified and modified functional LARS in both acute and chronic models.

For both types of patients, the first few variables selected by functional LARS are much more informative than others. This is reflected by the distance α corresponding to these variables is vary large compared to the later ones. In this case, the expected peak from AD is no longer valid. However, consider both stopping rules, it is reasonable for both acute and chronic models to stop the algorithm at iteration 9 and 8 respectively using either the unmodified and modified algorithms.

For acute patients, modified functional LARS removes three variables from the regression equation, including two kinematic variables and one functional variable. For chronic patients, modified functional LARS removes three variables from the regression equation, including one kinematic variable and two functional variables. In the following analysis, results from the variables selected from both unmodified and modified functional LARS will be used for comparison.

Functional LARS selects the variables and gives the estimates of the parameters simultaneously, so we can use the estimated variables to perform the predictions.

Mixed effects models fME_1, fME_2 and fME_3

Recall that the model fME_1 for the i-th patient is:

$$y_i = \beta_0 + y_i^{(0)}\gamma^{(0)} + \mathbf{z}_i\boldsymbol{\gamma} + \sum_{j=1}^{J}\int x_{i,j}(t)\beta_j(t)dt + g(\boldsymbol{\phi}_i) + \epsilon \tag{7.13}$$
$$g(\mathbf{t}_i) \sim GP(0, \kappa(\mathbf{t}_i, \mathbf{t}_i'); \boldsymbol{\theta}) \quad \epsilon_{i,k} \sim N(0, \sigma^2)$$

Similar to the previous mixed effects model, we use a two step procedure to learn this model:

1. Use functional LARS to estimate the coefficients for the fixed effect part and obtain the residual r_{fix};
2. use MAP to find the estimated hyper-parameters in the Gaussian process for random effect with respect to r_{fix} and obtain a new residual r_{ran};
3. use functional LARS to update the coefficients for the fixed effect part with respect to r_{ran} and update r_{fix};

repeat Step 2 & 3 until convergence.

In the step of learning for fixed effect, the coefficients related to the fixed effect part are recalculated using only the training data. Because the variables are all pre-selected, the stopping point will not be a problem any more. Thus, we estimate the coefficients by functional LARS and stop after all the variables are in the regression equation. The model fME_2 and model fME_3 are learned in a similar way.

The prediction procedure using these models is also similar to the mixed effects model with only scalar variables. More specifically, we predict the fixed effect and random effect separately, and sum them up to give the final prediction.

Model prediction for a new patient using Gaussian process as random effect

The dependence level of a new patient should follow a Gaussian process with the same hyper-parameters from the training patients. For a Gaussian process, we need four ele-

ments to carry out the prediction: the hyper-parameters, the input variables for prediction, observed samples of the response variable and observed samples of the input variables. The prediction can be thought as the weighted average of the observed samples of the response variables. The weight is decide by the covariance kernel.

In the prediction procedure using random effect, we may have two types of predictions after we finish training the model depending on the data structure. Type I is that we have observed a few samples for the new patient including response variables and the input variables. Type II is that we have no samples observed. For both types we have hyper-parameters and input variables for prediction. We briefly describe one way to do prediction for Type II data from Shi and Choi (2011). We assume that we observe from each of the training patients the same set of input variables for prediction, and thus we can have a set of outcomes from each of the training patients. The final prediction is the average from all these outcomes. However, the accuracy from Type II prediction is questionable.

In our problem, we have no observed samples of response variable for a completely new patient. We have the following two choices. Recall that the first week only gives baseline measurement; we can use the second week's observation from the new patient as the second baseline and use Type I prediction for the subsequent samples. Or we can use Type II prediction to predict the first week's observation and use Type I prediction for the following samples. We use the first method. The main reason is the poor performance of the second one. In addition, practically speaking, two baseline measurements are not difficult to obtain. Also, we use one step ahead prediction from Shi et al. (2012) to predict the response variable from later weeks.

7.3 Numerical Comparison

We split 34 acute patients into five folds with seven patients per fold for the first five folds, and six patients in the last fold. Similarly we split 36 chronic patients into five folds and assign eight patients into the last fold. Five-fold cross validation is then applied. More specifically, we do the predictions for one of the fold by using models learned with the other four folds. The splitting for both types of patients are randomly selected. In order to remove the odds from the random splitting, we repeat this process 500 times for both acute and chronic patients. We compare the models by using the prediction root-mean-squared-error, as it is the aim of the study for this set of motion data.

	Scalar	lm	tsN	ME_1	ME_2	ME_3
Acute	RMSE	6.985	7.226	6.614	7.565	7.610
	SD	0.229	0.287	0.179	0.141	0.147
Chronic	RMSE	3.904	-	4.047	4.270	4.319
	SD	0.106	-	0.105	0.080	0.091
	Unmodified	fFE	fME_1	fME_2	fME_3	
Acute	RMSE	6.212	6.212	6.507	6.587	
	SD	0.224	0.220	0.237	0.249	
Chronic	RMSE	3.649	3.749	4.229	4.207	
	SD	0.094	0.094	0.120	0.123	
	modified	fFE	fME_1	fME_2	fME_3	
Acute	RMSE	6.200	6.061	6.513	6.568	
	SD	0.186	0.176	0.198	0.215	
Chronic	RMSE	3.698	3.854	4.197	4.148	
	SD	0.090	0.095	0.108	0.116	

Table 7.1: Model comparison using prediction RMSE

The results are listed in the Table 7.1. Consider the range of the response is from 9 to 63, the prediction RMSEs from all the models above indicate fairly accurate predictions.

For acute patients, the best results come from the functional mixed effects model with only time t in the random effect fME_1 using the variables from the modified functional LARS. The predictions from this model have a fairly small standard deviation. The same model using the variables from unmodified LARS has a slightly worse prediction RMSE. It represents that the variables selected from modified functional LARS actually improve the prediction performance for acute patients. If we use only the scalar variables, the best model is the mixed effects model with only time t in the random effect ME_1.

The variables selected in the model with the best performance for acute patients are 'LA05.lx', 'LA09.ly', 'LA28.rqx' and 'sp.P.LA05'. The first three variables are functional variables and the last one is the scalar variable. We omit the variables about the personal information here. The name of the functional variable contains the following information: the movement number and the name of the coefficient. The names of the scalar variables contains the following information: the kinematic variable, the side of the limb and the movement number.

'LA05.lx' is the data from x axis on paretic limb for forward circle movement we used in the example. The movement is mainly on y-z plane. But the variation on the x axis shows the capability to keep stable when moving. 'LA09.ly' is the data from y axis on paretic

limb for a sawing movement. This movement is mainly on z axis, but also requires the arms to move at the elbow positions. Patients, who cannot control their arms very well, might not be able to bend their arms to around 90 degree to do this movement. So the variation on y axis is also informative. 'LA28.rqx' is the data from the x axis of the quaternion on non-paratic limb for a orientation movement. The movement requires the patient to rotate their hands around z axis. 'sp.P.LA05' is the speed of the paretic limb for movement forward circle.

For chronic patients, the best model is the functional linear regression model fFE using variables selected from unmodified functional LARS. The standard deviation of the prediction RMSE's is very small. The same model using variables from the modified functional LARS is slightly worse than the best one with a 0.05 difference in the prediction RMSE. Thus the variables removed by the modification in functional LARS is not making much differences to the outcome.

The variables selected in the model with the best performance for chronic patients are: 'LA07.rx', 'LA28.lqz', 'rom.P.LA21' and 'rom.P.LA35'. The first two variables are functional and the last two are scalar.

'LA07.rx' is the data from x axis on non-paretic limb for the movement which requires patients to move their hands from bottom to neck. Similar to 'LA05', the movement is on the y-z plane, but more capable patients should be more likely to keep their limbs stable on the y-z plane. 'LA28.lqz' is the data from the z axis of the quaternion on paratic limb for movement 'LA28'. 'rom.P.LA21' is the range-of-movement of the paretic limb for a movement which requires patients to move their hands from bottom to about chest hight, and put one hand on the other. This movement is difficult for the paretic limb since it requires a fairly precise hight. Patients who have poor capability might not be able to move to that precise position. 'rom.P.LA35' is the range-of-movement of the paretic limb for a movement which requires patients to move their hands like they are chopping with a knife. Patients who have poor capability might move their whole arms rather than hands and give very large values of range-of-movements.

The best model without the functional variables is still the linear regression. The results from time series model and the mixed effects models confirm that taking count of the within patient correlation may not be beneficial to the prediction for chronic patients.

7.4 Conclusion

We have discussed six models with scalar variables only and four models with mixed scalar and functional variables for acute and chronic patients in this chapter. The changes in the dependence level for acute patients have more variation than those from chronic patients. This is reflected by the prediction RMSE's from all the models. More specifically, the prediction RMSE's are generally larger than those from the chronic patients.

The models with scalar variables only can do well in predicting for new patients. By involving the functional variables, the models can always be improved. For acute patients, the best model with scalar variables only is improved about 10% and 5% by including functional variables selected by the modified and unmodified functional LARS, respectively. For chronic patients, this improvement is about 10% for both sets of variables. It confirmed the quality of variables selected by the functional LARS and the improvement by using modifications in the selection. In addition, the performance of the functional linear regression models trained by functional LARS is satisfactory for both acute and chronic patients.

Chapter 8

Conclusion and Future Work

This thesis is made of two parts. The first part is about a new variable selection algorithm for functional linear regression. The second part is concerned with the modelling of a motion data set using different models, including the functional linear regression model which uses the new algorithm to select variables and estimate coefficients.

In the first part, we proposed a new functional variable selection algorithm called functional least angle regression. It aimed ad doing variable selection for functional linear regression model with scalar response and mixed scalar and functional predictors. The algorithm is an extension of the least angle regression for multivariate linear regression model with scalar variables only.

The key to the new algorithm is the calculation of the correlation between a scalar variable and a group of scalar and functional variables. We use the modified functional canonical correlation here. Functional variables are represented by discrete data points and the functional coefficients are represented by different methods, including discrete data points, basis function and Gaussian quadrature. The last method is new and the performances are satisfactory. The smoothness of the functional coefficients is controlled by the roughness penalty functions while the smoothing parameters are estimated via generalized cross validation for fast computation. With this correlation analysis, we can achieve selection of the variables and estimation of the parameters simultaneously. Two modifications are used to overcome the difficulties in the extreme cases. As the conventional stopping rules are not useful for the new algorithm due to the failure of the calculation of the degrees of freedom, we proposed two new stopping rules for practical use and further reduce the computational cost as we can stop before the whole solution path is drawn.

There are a few limitations in this algorithm. First of all, the computational cost for creating the solution path would increase exponentially as the number of candidate predictors increases. Secondly the algorithm is incapable of capturing the non-linear relationship between the response and the predictors.

The first limitation is due to the nature of the algorithm. With the help from the new stopping rules, practical use is not affected. On the other hand, more research can be done to overcome the second limitation. The linearity of this model is captured by the canonical correlation analysis, which only measures the linear relations. In future research, we will replace this correlation analysis with others to achieve non-linearity. As long as the correlation analysis can give an estimated correlation coefficient and an estimated coefficient to project a group of scalar and functional variables into a one dimension variable, the idea of least angle regression can still be used.

In the second part, we analyse a motion data set. The motion data here contain rich information. However, many pretreatments must be carried out to allow modelling of the data set. With different types of predictors, different models are tested and compared. The best model in general is the random effects model with functional regression in one part and a Gaussian process in the other.

The 'best' model can still be improved. One way is to do variable selection properly. More specifically, the variable selection should be carried out with respect to functional regression and a Gaussian process together as a whole. Due to the practical reason, variable selection for the Gaussian process is omitted in our model. Also, the model only considers non-linearity in the scalar variables. The prediction accuracy would be further improved if we can also include functional variables in the non-linear model.

Bibliography

Abdi, H. (2007). Distance. *Encyclopedia of measurement and statistics*, pages 280–284.

Abdi, H. (2010). Congruence: congruence RV coefficient, and Mantel coefficient. *Encyclopedia of Research Design*.

Adorf, H.-M. (1995). Interpolation of Irregularly Sampled Data Series—A Survey. In *Astronomical Data Analysis Software and Systems IV*, volume 77, page 460.

Akaike, H. (1998). Information theory and an extension of the maximum likelihood principle. In *Selected Papers of Hirotugu Akaike*, pages 199–213. Springer.

Aurore Delaigle, P. H. (2012). Methodology and theory for partial least squares applied to functional data. *The Annals of Statistics*, 40:322352.

Bakin, S. et al. (1999). Adaptive regression and model selection in data mining problems.

Cardot, H., Ferraty, F., and Sarda, P. (1999). Functional linear model. *Statistics & Probability Letters*, 45(1):11–22.

C.Crainiceanu, A. Staicu, C. (2009). Generalized Multilevel Functional Regression. *Journal of American Statistical Association*, 104(488):1550–1561.

Di, C.-Z., Crainiceanu, C. M., Caffo, B. S., and Punjabi, N. M. (2009). Multilevel functional principal component analysis. *The annals of applied statistics*, 3(1):458.

Diebel, J. (2006). Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors.

Eckner, A. (2012). A framework for the analysis of unevenly-spaced time series data. *Preprint*.

Efron, B. (1986). How biased is the apparent error rate of a prediction rule? *Journal of the American Statistical Association*, 81(394):461–470.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Rejoinder to "Least angle regression" by Efron et al.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. (2003). Least angle regression. *The Annals of statistics*, 32(2):407–499.

Efron, B. and Tibshirani, R. (1997a). Improvements on cross-validation: the 632+ bootstrap method. *Journal of the American Statistical Association*, 92(438):548–560.

Efron, B. and Tibshirani, R. (1997b). Improvements on cross-validation: the 632+ bootstrap method. *Journal of the American Statistical Association*, 92(438):548–560.

Fan, Y., Foutz, N., James, G. M., Jank, W., et al. (2014). Functional response additive model estimation with online virtual stock markets. *The Annals of Applied Statistics*, 8(4):2435–2460.

Fan, Y. and James, G. (2013). Functional additive regression. *Preprint*.

Fu, W. J. (1998). Penalized regressions: the bridge versus the lasso. *Journal of computational and graphical statistics*, 7:397–416.

Golub, G. H., Heath, M., and Wahba, G. (1979). Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223.

Guozhong He, H.-G. M. and Wang, J.-L. (2003). Methods of Canonical Analysis for Functional Data. *Journal of multivariate analysis*, 85:54–77.

Hall, P., Müller, H.-G., and Wang, J.-L. (2006). Properties of principal component methods for functional and longitudinal data analysis. *The annals of statistics*, pages 1493–1517.

Hansen, P. (1987). Method and apparatus for position and orientation measurement using a magnetic field and retransmission.

He, G., Müller, H., and Wang, J. (2000). Extending correlation and regression from multivariate to functional data. *Asymptotics in statistics and probability*, pages 197–210.

He, G., Mller, H.-G., Wang, J.-L., and Yang, W. (2010). Functional linear regression via canonical analysis. *Bernoulli*, 16:605–908.

Ian L. Dryden, A. K. and Zhou, D. (2009). Stroke. *The Annals of Applied Statistics*, 3(3):881–1231.

J. Berrendero, A. Cuevas, J. T. (2013). A Distance-Covariance Approach to Variable Selection in Functional Discrimination. *Processing*.

James, G. M., Wang, J., and Zhu, J. (2009). Functional linear regression that's interpretable. *The Annals of Statistics*, pages 2083–2108.

J.Gertheiss, A.Maity, A. (2013). Variable Selection in Generalized Functional Linear Models. *Stat*.

J.Goldsmith, Bobb, J., C.Crainiceanu, B.Caffo, and D.Reich (2011). Penalized Functional Regression. *Journal of Computational and Graphical Statistics*, 20(4):830–851.

Josse, J. (1971). Statistical methods of comparing different multivariate analyses of the same data. *Mathematics in the archaeological and historical sciences*, pages 138–149.

Josse, J. and Holmes, S. (2013). Measures of dependence between random vectors and tests of independence. literature review. *arXiv preprint arXiv:1307.7383*.

Kelly-Hayes, M., Beiser, A., Kase, C. S., Scaramucci, A., DAgostino, R. B., and Wolf, P. A. (2003). The influence of gender and age on disability following ischemic stroke: the Framingham study. *Journal of Stroke and Cerebrovascular Diseases*, 12(3):119–126.

Kuipers, J. B. (2002). *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*. Princeton University Press, 1 edition.

Laver, K., George, S., Thomas, S., Deutsch, J. E., and Crotty, M. (2012). Virtual reality for stroke rehabilitation. *Stroke*, 43(2):e20–e21.

Lip, G. Y., Nieuwlaat, R., Pisters, R., Lane, D. A., and Crijns, H. J. (2010). Refining clinical risk stratification for predicting stroke and thromboembolism in atrial fibrillation using a novel risk factor-based approach: the euro heart survey on atrial fibrillation. *Chest Journal*, 137(2):263–272.

Lorenzo-Seva, U. and Ten Berge, J. M. (2006). Tucker's congruence coefficient as a meaningful index of factor similarity. *Methodology*, 2(2):57–64.

Mallows, C. L. (1973). Some comments on c_p. *Technometrics*, 15(4):661–675.

Mantel, N. (1967). The detection of disease clustering and a generalized regression approach. *Cancer research*, 27(2 Part 1):209–220.

Matsui, H. and Konishib, S. (2011). Variable selection for functional regression models via the L1 regularization. *Computational Statistics and Data Analysis*, 55:33043310.

Meier, L., de, G. S. V., and Peter, B. (2009). High-dimensional additive modeling. *The Annals of Statistics*, 37:3779–3821.

Minas, C., Curry, E., and Montana, G. (2013). A distance-based test of association between paired heterogeneous genomic data. *Bioinformatics*, page btt450.

Mingotti, N., Lillo, R. E., and Romo, J. (2013). Lasso variable selection in functional regression.

Müller, H.-G. and Stadtmüller, U. (2005). Generalized functional linear models. *Annals of Statistics*, pages 774–805.

Müller, H.-G. and Yao, F. (2012). Functional additive models. *Journal of the American Statistical Association*.

Olshen, R. A., Biden, E. N., Wyatt, M. P., and Sutherland, D. H. (1989). Gait analysis and the bootstrap. *The annals of statistics*, pages 1419–1440.

P. Langhorne, F. Coupar, A. P. (2009). Motor recovery after stroke: a systematic review. *Lancet Neurol*, 8:741–754.

Party, I. S. W. (2012). National clinical guideline for stroke.

Penrose, R. (1955). A generalized inverse for matrices. In *Mathematical proceedings of the Cambridge philosophical society*, volume 51, pages 406–413. Cambridge Univ Press.

Primack, B. A., Carroll, M. V., McNamara, M., Klem, M. L., King, B., Rich, M., Chan, C. W., and Nayak, S. (2012). Role of video games in improving health-related outcomes: a systematic review. *American journal of preventive medicine*, 42(6):630–638.

Ramsay, J. O. and Silverman, B. (2005). *Functional Data Analysis*. Springer, 2 edition.

Ramsay, J. O. and Silverman, B. W. (2002). *Applied functional data analysis: methods and case studies*, volume 77. Springer New York.

Reiss, P. T. and Ogden, R. T. (2007). Functional Principal Component Regression and Functional Partial Least Squares. *Journal of the American Statistical Association*, 102:984–996.

Robert, P. and Escoufier, Y. (1976). A unifying tool for linear multivariate statistical methods: the rv-coefficient. *Applied statistics*, pages 257–265.

Rosen, O. and Thompson, W. K. (2009). A bayesian regression model for multivariate functional data. *Computational Statistics & Data Analysis*, 53(11):3773–3786.

S. Barreca, P. Stratford, C. L. L. M. D. S. (2005). Test-retest reliability, validity, and sensitivity of the Chedoke arm and hand activity inventory: a new measure of upper-limb function for survivors of stroke. *Arch Phys Med Rehabil*, 86:1616–1622.

S. E. Leurgans, R. A. M. and Silverman, B. W. (1993). Canonical Correlation Analysis when the Data are Curves. *Journal of the Royal Statistical Society. Series B*, 55:725–740.

Schwarz, G. et al. (1978). Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464.

Shi, J., Cheng, Y., Serradilla, J., Morgan, G., Lambden, C., Ford4, G., Price, C., Rodgers, H., Cassidy, T., Rochester, L., and Eyre, J. (2013). Evaluating Functional Ability of Upper Limbs after Stroke Using Video Game Data. pages 181–192.

Shi, J. and Choi, T. (2011). *Gaussian Process Regression Analysis for Functional Data*. Chapman and Hall/CRC, 1 edition.

Shi, J. and Wang, B. (2008). Curve prediction and clustering with mixtures of gaussian process functional regression models. *Statistics and Computing*, 18(3):267–283.

Shi, J., Wang, B., Will, E., and West, R. (2012). Mixed-effects gaussian process functional regression models with application to dose–response curve prediction. *Statistics in medicine*, 31(26):3165–3177.

Simon, N. and Tibshirani, R. (2012). Standardization and the group lasso penalty. *Statistica Sinica*, 22(3):983.

Strong, K., Mathers, C., and Bonita, R. (2007). Preventing stroke: saving lives around the world. *The Lancet Neurology*, 6(2):182–187.

Székely, G. J., Rizzo, M. L., Bakirov, N. K., et al. (2007). Measuring and testing dependence by correlation of distances. *The Annals of Statistics*, 35(6):2769–2794.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B*, 58.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., and Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B*, 67:91–108.

Touillet, A., Guesdon, H., Bosser, G., Beis, J.-M., and Paysant, J. (2010). Assessment of compliance with prescribed activity by hemiplegic stroke patients after an exercise programme and physical activity education. *Annals of physical and rehabilitation medicine*, 53(4):250–265.

Turlach, B. A. (2004). A note on the group lasso and a sparse group lasso. *arXiv:math/0406472*.

Vinod, H. D. (1976). Canonical ridge and econometrics of joint production. *Journal of Econometrics*, 73:147–166.

Wang, L., Chen, G., and Li, H. (2007). Group scad regression analysis for microarray time course gene expression data. *Bioinformatics*, 23(12):1486–1494.

Yao, F., Müller, H.-G., and Wang, J.-L. (2005a). Functional data analysis for sparse longitudinal data. *Journal of the American Statistical Association*, 100(470):577–590.

Yao, F., Müller, H.-G., Wang, J.-L., et al. (2005b). Functional linear regression analysis for longitudinal data. *The Annals of Statistics*, 33(6):2873–2903.

Ye, J. (1998). On measuring and correcting the effects of data mining and model selection. *Journal of the American Statistical Association*, 93(441):120–131.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

Zou, H., Hastie, T., Tibshirani, R., et al. (2007). On the degrees of freedom of the lasso. *The Annals of Statistics*, 35(5):2173–2192.