CONSERVATIVE CONFIDENCE REGIONS FOR KERNEL ESTIMATES OF THE VARYING COEFFICIENT MODEL

By

JAMES EDWARD BLUM

Bachelor of Arts
University of Michigan
Ann Arbor, Michigan
1992

Master of Science
Oklahoma State University
Stillwater, Oklahoma
1998

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
December, 2000

CONSERVATIVE CONFIDENCE REGIONS FOR

KERNEL ESTIMATES OF THE VARYING

COEFFICIENT MODEL

Thesis Approved

[illegible]

Thesis Advisor

[illegible]

[illegible]

Jeanne S Hill

[illegible]

Dean of the Graduate College

ACKNOWLEDGMENTS

First, I would like to extend my appreciation to my advisor, Dr. Moser. It would be difficult to overestimate his influence on my academic development. Both as an advisor and as an instructor, Dr. Moser's positive attitude, exceptional technical skill and great ability for communication make him one of the best teachers and mentors that a student could hope for.

I would also like to thank the members of my committee for their review of and helpful suggestions on my work. In particular, I would like to thank Dr. McCann for introducing me to the paper by Daniel Q. Naiman, Dr. Hill for her constructive comments on presentation style, and Dr. Fuqua for our many interesting conversations about the state of statistics education and statistics in practice.

I must also thank those who were my fellow students at one time or another, and it is a rather long list: Joy, Scott, James, Chris, Diane, Gerald, Kelly, Jennifer, Bryan, Sierra, Joel, Marlo, Tracey, Claudia, Luke, and Desiree. All of you have helped make what might have been a dreary and boring five years into a fun, interesting, and always unpredictable trip; thanks to each of you for your friendship and kindness. I should especially thank former student Dr. Laura Coombs, for without her influence, I might never have sought a doctoral degree. Both she and her husband, Tom, helped make Stillwater a home away from home for me. Thanks to them for all of the helpful conversations and counsel.

Finally, I need to thank my family. My parents and my sister have always been patient and willing to help in any way as I have travelled this seemingly endless road. That I have reached this point is testament to their love and support.

TABLE OF CONTENTS

LIST OF TABLES

4.5 Estimated coverage probabilities on $t \in [1, 29]$ for case 5 61

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

CHAPTER ONE

INTRODUCTION AND REVIEW OF LITERATURE

The defining characteristic of longitudinal studies is the repeated observation, usually over time, of both an outcome and a set of covariates for a group of randomly selected subjects. For a group of n subjects, let m_i denote the number of observations on the i^{th} subject, and let t_{ij} denote the measurement time for the j^{th} observation on the i^{th} subject, $i = 1, ..., n$ and $j = 1, ..., m_i$. The observed outcome for the i^{th} subject at time t_{ij} is denoted by Y_{ij} and the corresponding set of $(k + 1)$ covariates by $\boldsymbol{X}_{ij} = (X_{ij0}, X_{ij1}, ..., X_{ijk})^{\mathrm{T}}$. The longitudinal observations; $(Y_{ij}, \boldsymbol{X}_{ij}, t_{ij})$, $i = 1, ..., n$ and $j = 1, ..., m_i$, are assumed independent across subjects, while observations on the same subject are likely to be correlated in some way.

Many parametric approaches have been put forth in an effort to model the relationship between the outcomes and covariates over the course of a longitudinal study; however, the focus here centers on recent developments in non-parametric approaches. Many authors; Hart and Wehrly (1986), Altman (1990), and Hart (1991), have considered non-parametric estimates of Y strictly as a function of t, without consideration of the presence of any covariate effects. In an effort to include covariate effects, Moyeed and Diggle (1994) and Zeger and Diggle (1994) studied the following semi-parametric model:

$$Y_{ij} = \mu(t_{ij}) + \boldsymbol{X}_{ij}^{\mathrm{T}}\beta + \epsilon_i(t_{ij}). \tag{1.1}$$

Here $X_{ij0} = 1$ for all i, j; $\mu(t)$ is an arbitrary smooth function of t, $\beta = (\beta_0, ..., \beta_k)^{\mathrm{T}}$ is a vector of unknown constants, and $\epsilon_i(t)$ is a mean zero stochastic process. This model allows for covariate effects to be included together with a time varying intercept term, yet this may be too restrictive for some settings.

A useful generalization of (1.1) is the following:

$$Y_{ij} = \boldsymbol{X}_{ij}^{\mathrm{T}}\boldsymbol{\beta}(t_{ij}) + \epsilon_{ij}(t_{ij}) \tag{1.2}$$

where $\epsilon_i(t)$ is a mean zero stochastic process independent of $\boldsymbol{X}_i(t)$, $\boldsymbol{\beta}(t) = (\beta_0(t), \beta_1(t), ..., \beta_k(t))^{\mathrm{T}}$, and $\beta_l(t)$ is a real-valued, smooth function of t for $l = 0, ..., k$. Fixing $X_{ij0} = 1, \forall\, i, j$, allows for the time varying intercept term that appears in (1.1); however, (1.2) also allows for the possibility of any of the model coefficients varying over time. It should be noted that (1.2) is probably still best described as semi-parametric, for even though $\boldsymbol{\beta}(t)$ is defined non-parametrically, there is a specific structure imposed on the relationship between Y and $\boldsymbol{X}$. As Wu, Chiang, and Hoover (1998) point out, a purely non-parametric approach, *i.e.* modelling Y as a smooth function of $(\boldsymbol{X}(t),\ t)$, suffers from two major problems: the dimensionality of the problem becomes quite high as the number of covariates or the number of observation points increases, and it is potentially difficult to interpret.

Model (1.2) is actually a special case of both a broader class of varying coefficient models studied by Hastie and Tibshirani (1993), and a class of models investigated by Ramsay and Dalzell (1991) known as functional linear models. In its own right, (1.2) has been investigated by Fan and Zhang (1998), Brumback and Rice (1998), and Hoover, Rice, Wu, and Yang (1998). The investigation by Hoover *et al.* includes estimates of $\boldsymbol{\beta}(t)$ based on smoothing splines, locally weighted polynomials, and kernel estimates.

The kernel estimation technique put forth in Hoover *et al.* is based on a locally-weighted least squares criterion. The estimator of $\boldsymbol{\beta}(t)$, $\widehat{\boldsymbol{\beta}}(t) = (b_0(t), b_1(t), ..., b_k(t))^{\mathrm{T}}$, is chosen to minimize:

$$l_N(t) = \sum_{i=1}^{n}\sum_{j=1}^{m_i}\left[Y_{ij} - \left(\sum_{l=0}^{k} X_{ijl} b_l(t)\right)\right]^2 K\left(\frac{t - t_{ij}}{h}\right) \tag{1.3}$$

where $N = \sum_{i=1}^{n} m_i$ is the total number of observations, $K(\cdot)$ is a Borel measurable kernel function, and h is a positive bandwidth. Defining for each subject a vector of outcomes $\boldsymbol{Y}_i = (Y_{i1}, ..., T_{im_i})^{\mathrm{T}}$, a design matrix

$$\boldsymbol{X}_i = \begin{pmatrix} X_{i10} & X_{i11} & ... & X_{i1k} \\ ... & & & ... \\ X_{im_i0} & X_{im_i1} & ... & X_{im_ik} \end{pmatrix},$$

and a kernel matrix

$$\boldsymbol{K}_i(t,h) = \begin{pmatrix} K\left(\frac{t-t_{i1}}{h}\right) & 0 & ... & ... & 0 \\ 0 & K\left(\frac{t-t_{i2}}{h}\right) & 0 & ... & 0 \\ ... & & ... & & ... \\ & & & ... & 0 \\ 0 & ... & ... & 0 & K\left(\frac{t-t_{im_i}}{h}\right) \end{pmatrix},$$

(1.3) can be rewritten as:

$$l_N(t,h) = \sum_{i=1}^{n}\Big(\boldsymbol{Y}_i - \boldsymbol{X}_i\widehat{\boldsymbol{\beta}}(t)\Big)^{\mathrm{T}} \boldsymbol{K}_i(t,h)\Big(\boldsymbol{Y}_i - \boldsymbol{X}_i\widehat{\boldsymbol{\beta}}(t)\Big). \tag{1.4}$$

It is then straightforward to show that the unique minimizer of (1.4) is given by

$$\widehat{\boldsymbol{\beta}}(t,h) = \left(\sum_{i=1}^{n} \boldsymbol{X}_i^{\mathrm{T}} \boldsymbol{K}_i(t,h) \boldsymbol{X}_i\right)^{-1} \left(\sum_{i=1}^{n} \boldsymbol{X}_i^{\mathrm{T}} \boldsymbol{K}_i(t,h) \boldsymbol{Y}_i\right) \tag{1.5}$$

provided $\sum_{i=1}^{n} \boldsymbol{X}_i^{\mathrm{T}} \boldsymbol{K}_i(t,h) \boldsymbol{X}_i$ is invertible. It should be noted that, in order to produce an estimate of $\boldsymbol{\beta}(t)$ over some interval *I*, (1.5) is computed for several $t \in I$.

Of particular importance in obtaining estimates given by (1.5) is the choice of a suitable bandwidth. It is widely known in general kernel smoothing that increasing bandwidth increases the bias of the estimator, while decreasing the bandwidth increases the variance of the estimator. A difficulty even more basic than compromising between bias and variance is the need for a method to select a bandwidth in a manner that is feasible and practical. Intuitively, bandwidth selection should be most dependent on the behavior of the function to be estimated and the magnitude of the variability of the random error term. Since neither of these is known in advance, data driven bandwidth selection procedures are generally required. The method most frequently used, and that is chosen by Hoover, Rice, Wu, and Yang (1998) and Wu, Chiang, and Hoover (1998), attempts to minimize average predictive squared error (APSE) via a pseudo cross-validation technique.

If Y_{ij}^* is a new observation at $(\boldsymbol{X}_{ij}^*, t_{ij}^*)$, then

$$\mathrm{APSE}(\widehat{\boldsymbol{\beta}}) = \frac{1}{N} \sum_{i=1}^{n} \sum_{j=1}^{m_i} \mathrm{E}\left[\left(Y_{ij}^* - \boldsymbol{X}_{ij}^{*\mathrm{T}} \widehat{\boldsymbol{\beta}}(t_{ij}^*)\right)^2\right].$$

To generate a cross-validation sample for which APSE can be measured, subjects in the study are left out one at a time. Let $\widehat{\boldsymbol{\beta}}^{(-i)}$ be the estimate given by (1.5) leaving out all observations on the i^{th} subject. The cross-validation APSE criterion for bandwidth selection is given by

$$\mathrm{CV}(h) = \frac{1}{N}\sum_{i=1}^{n}\sum_{j=1}^{m_i}\left(Y_{ij} - \boldsymbol{X}_{ij}^{\mathrm{T}}\widehat{\boldsymbol{\beta}}^{(\text{-}\boldsymbol{i})}(t_{ij})\right)^2.$$

The cross-validation bandwidth, h_{CV}, is defined to be the value of h that minimizes $\mathrm{CV}(h)$. It is of interest to note that, in practice, h_{CV} generally provides under-smoothed estimates. It is also worth noting that (1.5) depends on a single smoothing parameter irrespective of the number of coefficient functions to be estimated, which is a potentially significant problem if different coefficient functions require different degrees of smoothing.

The ability to make inferences based on estimates of $\boldsymbol{\beta}(t)$ is highly desirable. Essentially, we would like to be able to discern whether fluctuations in a particular coefficient function appear to be beyond simple random error and indicative of a significant time effect. Wu, Chiang, and Hoover (1998) derived the asymptotic distribution of estimators given by (1.5) and developed a method for generating conservative confidence regions for an arbitrary linear combination of the coefficient functions. The method of construction for these confidence regions is based on an adaptation of a Bonferroni style method developed by Knafl, Sacks, and Ylvisaker (1985).

The method of Wu *et al.* is as follows: for a known vector of $k+1$ real constants, $\boldsymbol{A} = (a_0, a_1, ..., a_k)^{\mathrm{T}}$, a confidence region for $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\beta}(t)$ on $t \in [a,b]$ is constructed by choosing a grid of g points $\{\xi_1, \xi_2, ..., \xi_g\}$ such that $a = \xi_1, b = \xi_g$, and $\xi_{j+1} - \xi_j = \gamma$ for $j = 1, ..., g-1$. For each point ξ_j in the grid, $l(\xi_j)$ and $u(\xi_j)$ are chosen to satisfy

$$\lim_{n\to\infty} P\left[l(\xi_j) \leq \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\beta}(\xi_j) \leq u(\xi_j)\right] \geq 1 - \frac{\alpha}{g}, \tag{1.6}$$

thus satisfying

$$\lim_{n\to\infty} P\left[l(\xi_j) \leq \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\beta}(\xi_j) \leq u(\xi_j), \forall\, j = 1, ..., g\right] \geq 1 - \alpha. \tag{1.7}$$

Typically, the choices for $l(\xi_j)$ and $u(\xi_j)$ are given by

$$\boldsymbol{A}^{\mathrm{T}}\widehat{\boldsymbol{\beta}}(\xi_j) \pm z_{\frac{\alpha}{2g}}(Nh)^{-\frac{1}{2}}\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{D}(\xi_j)\boldsymbol{A}\right)^{\frac{1}{2}}$$

where $\boldsymbol{D}(t)$ is the asymptotic covariance matrix for $\widehat{\boldsymbol{\beta}}(t)$.

To bridge the gaps between grid points, for $t \in [\xi_j, \xi_{j+1}]$ define $l^{(\mathrm{I})}(t)$ to be the linear interpolation of $l(\xi_j)$ and $l(\xi_{j+1})$:

$$l^{(\mathrm{I})}(t) = [(\xi_{j+1} - t)l(\xi_j) + (t - \xi_j)l(\xi_{j+1})]/\gamma$$

Defining $u^{(\mathrm{I})}(t)$ and $(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\beta})^{(\mathrm{I})}(t)$ similarly, it follows directly from (1.7) that

$$\lim_{n\to\infty} P\left[l^{(\mathrm{I})}(\xi_j) \leq \left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\beta}\right)^{(\mathrm{I})}(\xi_j) \leq u^{(\mathrm{I})}(\xi_j), \forall\, t \in [a, b]\right] \geq 1 - \alpha.$$

Shifting the interval from $(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\beta})^{(\mathrm{I})}(t)$ to $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\beta}(t)$ is done based on one of the following smoothness conditions:

$$(i)\ \sup_{t\in[a,b]}|\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\beta}'(t)| \leq c_1$$

$$(ii)\ \sup_{t\in[a,b]}|\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\beta}''(t)| \leq c_2$$

where c_1 and c_2 are known constants.

If condition (i) is satisfied, it follows that

$$\left|\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\beta}(t) - \left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\beta}\right)^{(\mathrm{l})}(t)\right| \leq \frac{c_1}{\gamma}(\xi_{j+1}-t)(t-\xi_j), \forall\, t \in [\xi_j, \xi_{j+1}]$$

and

$$\lim_{n\to\infty} P[l^{(\mathrm{l})}(t) - \frac{c_1}{\gamma}(\xi_{j+1}-t)(t-\xi_j) \leq \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\beta}(t) \leq u^{(\mathrm{l})}(t) - \frac{c_1}{\gamma}(\xi_{j+1}-t)(t-\xi_j),$$
$$\forall\, t \in [\xi_j, \xi_{j+1}] \text{ and } j = 1, ..., g] \geq 1-\alpha \quad (1.8)$$

Similarly, if condition (ii) is satisfied, it follows that

$$\lim_{n\to\infty} P[l^{(\mathrm{l})}(t) - \frac{c_1}{2}(\xi_{j+1}-t)(t-\xi_j) \leq \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\beta}(t) \leq u^{(\mathrm{l})}(t) - \frac{c_1}{2}(\xi_{j+1}-t)(t-\xi_j),$$
$$\forall\, t \in [\xi_j, \xi_{j+1}] \text{ and } j = 1, ..., g] \geq 1-\alpha \quad (1.9)$$

The aforementioned procedure has some significant limitations common to Bonferroni style techniques. Bonferroni intervals, especially when many are required, tend to be quite conservative in practice, which is certainly a serious issue in this context given the requirement of choosing an "appropriately sized" grid of points for this procedure. To keep the procedure minimally conservative, as small a grid as possible would be desired; however, from (1.8) [or (1.9)] it is clear that smaller grid sizes produce "unsmooth" bands. In order to produce smooth bands, which is quite desirable with respect to our desire to produce smooth estimates of $\boldsymbol{\beta}(t)$, a large grid is required. Hence the procedure would be based on a large number of intervals of the form of (1.6), accentuating the conservative nature of the procedure. As evidence of this, Wu *et al.* (1998) includes a Monte Carlo study of their procedure in which the coverage probabilities for nominal 95% bands are estimated to be near 1.

The objective of this thesis is to develop a procedure for constructing confidence bands based on estimates of the form of (1.5) that are less conservative and more efficient than those put forth by Wu *et al.* (1998). The method of construction is based on overlaying the non-parametric estimate (1.5) with a parametric function, deriving the distribution of the estimated parameters in the overlaid function (based on the asymptotic distribution of (1.5)), and building confidence regions using a method developed by Naiman (1986). A simulation study will be performed in order to assess the relative performance of the proposed method against that of Wu *et al.* In addition, investigations will also focus on the potential use of the overlaid function as an additional smooth, which may overcome some of the difficulties associated with bandwidth selection.

CHAPTER TWO

ASYMPTOTIC DISTRIBUTION OF KERNEL ESTIMATOR

As was previously mentioned, kernel estimates of $\beta(t)$ over an interval are constructed via a set of the pointwise estimates given in equation (1.5). This seemingly trivial fact is actually a major contributor to the difficulties faced by the Bonferroni method discussed in chapter one; essentially, the dimension of the problem is quite large. In order to lower the dimension of the problem while still preserving the nature of the kernel estimate, a parametric function will be fit to the set of pointwise kernel estimates for each coefficient function. In this discussion, a polynomial will serve as the parametric function and will be fit using generalized least squares (GLS) techniques. The details of the fitting process are taken up in the next chapter; however, here it is important to note that using GLS will require some knowledge of the joint distribution of the kernel estimates given in (1.5). What follows is a summary of the results due to Wu, Chiang, and Hoover (1998) pertaining to the asymptotic distribution of estimates of the form of (1.5).

To begin, for the random error term in (1.2) define

$$\sigma^2(t_0) = E[\epsilon^2(t_0)],\ \ \rho_\epsilon(t_1, t_2) = E[\epsilon(t_1)\epsilon(t_2)]$$

and

$$\rho_\epsilon(t_0) = \lim_{\delta \to 0} E[\epsilon(t_0 + \delta)\epsilon(t_0)].$$

The definitions of $\sigma^2(t_0)$ and $\rho_\epsilon(t_0)$ may seem redundant; however, the two quantities need not be equal. A common example of where this inequality would arise is a case where the entire error process, $\epsilon(t)$, contains a stationary error process together with measurement errors that are independent at different time points. Here $\rho_\epsilon(t_0)$ would capture the variance of the stationary error process while $\sigma^2(t_0)$ describes the variance of the entire error term.

Next, with regard to the covariates in (1.2), define for $l, r = 0, 1, ..., k$

$$\eta_{lr}(t_0) = E[X_{il}(t_{ij})X_{ir}(t_{ij})|t_{ij} = t_0],$$

$$\eta_{lr}(t_1, t_2) = E[X_{il}(t_{ij_1})X_{ir}(t_{ij_2})|t_{ij_1} = t_1, t_{ij_2} = t_2]$$

and

$$E_{\boldsymbol{X}\boldsymbol{X}^\intercal}(t_0) = \begin{bmatrix} \eta_{00}(t_0) & \eta_{01}(t_0) & \dots & \eta_{0k}(t_0) \\ \eta_{10}(t_0) & \eta_{11}(t_0) & \dots & \eta_{1k}(t_0) \\ \dots & & \dots & \dots \\ \eta_{k0}(t_0) & \eta_{k1}(t_0) & \dots & \eta_{kk}(t_0) \end{bmatrix}.$$

Also, with respect to assumptions a) and b) of Wu *et al.* (1998, p. 1391) define

$$h_0 = N^{\frac{1}{5}}h \qquad \lambda = N^{-\frac{6}{5}}\sum_{i=1}^{n} m_i^2.$$

Further, for the chosen kernel function, the following parameters are needed:

$$\mu_1(K) = \int u^2 K(u)\, du \quad \text{and} \quad \mu_2(K) = \int K^2(u)\, du.$$

Finally, if the distribution of measurement times has density $f(t)$, define

$$b_l(t_0, h) = h_0^{\frac{3}{2}}\sum_{c=0}^{k}\{\mu_1(K)[\beta_c'(t_0)(\eta_{lc}'(t_0)f(t_0) + \eta_{lc}(t_0)f'(t_0)) + \tfrac{1}{2}\beta_c''(t_0)\eta_{lc}(t_0)f(t_0)]\}$$

and

$$D_{lr}(s_1, s_2, h) = \begin{cases} \eta_{lr}(s_1)[\sigma^2(s_1)f(s_1)\mu_2(K) + \lambda h_0 \rho_\epsilon(s_1) f^2(s_1)], & \text{if } s_1 = s_2 \\ \lambda h_0 \rho_\epsilon(s_1, s_2)\eta_{lr}(s_1, s_2) f(s_1) f(s_2), & \text{if } s_1 \neq s_2 \end{cases}$$

Then, for grid of time points $\boldsymbol{s} = (s_1, s_2, ..., s_q)^{\mathrm{T}}$, Wu *et al.* (1998) showed that $(Nh)^{\frac{1}{2}}\left(\widehat{\boldsymbol{\beta}}(\boldsymbol{s}, h) - \boldsymbol{\beta}(\boldsymbol{s})\right) \to \mathrm{N}_{g(\mathrm{k}+1)}(\boldsymbol{B}(\boldsymbol{s}, h), \boldsymbol{D}^*(\boldsymbol{s}, h)\,)$ in distribution as $n \to \infty$. Here, the bias term $\boldsymbol{B}(\boldsymbol{s}, h) = [\boldsymbol{B}(s_1, h), ..., \boldsymbol{B}(s_g, h)]^{\mathrm{T}}$ where

$$\boldsymbol{B}(t_0, h) = (f(t_0))^{-1} E^{-1}_{\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}}(t_0)[b_0(t_0, h), ..., b_k(t_0, h)]^{\mathrm{T}}.$$

The covariance structure, $\boldsymbol{D}^*(\boldsymbol{s}, h)$, is defined by the following:

$$\boldsymbol{D}^*(\boldsymbol{s}, h) = \begin{bmatrix} \boldsymbol{D}^*(s_1, s_1, h) & ... & \boldsymbol{D}^*(s_1, s_g, h) \\ ... & & ... \\ \boldsymbol{D}^*(s_g, s_1, h) & ... & \boldsymbol{D}^*(s_g, s_g, h) \end{bmatrix}$$

where

$$\boldsymbol{D}^*(s_{g_1}, s_{g_2}, h) = (f(s_{g_1})f(s_{g_2}))^{-1} E^{-1}_{\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}}(s_{g_1}) \boldsymbol{D}(s_{g_1}, s_{g_2}, h) E^{-1}_{\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}}(s_{g_2})$$

and

$$\boldsymbol{D}(s_{g_1}, s_{g_2}, h) = \begin{bmatrix} D_{00}(s_{g_1}, s_{g_2}, h) & ... & D_{0k}(s_{g_1}, s_{g_2}, h) \\ ... & & ... \\ D_{k0}(s_{g_1}, s_{g_2}, h) & ... & D_{kk}(s_{g_1}, s_{g_2}, h) \end{bmatrix}.$$

To fit parametric estimates to the kernel estimates and to build confidence bands for each coefficient function , the joint distribution of $\left(\widehat{\beta}_l(\boldsymbol{s}, h) - \beta_l(\boldsymbol{s})\right)$ will be extracted for $l = 0, 1, ..., k$, where $\widehat{\beta}_l(\boldsymbol{s}, h) = \left[\widehat{\beta}_l(s_1, h), ..., \widehat{\beta}_l(s_g, h)\right]^{\mathrm{T}}$ and $\beta_l(\boldsymbol{s}) = [\beta_l(s_1), ..., \beta_l(s_g)]^{\mathrm{T}}$. To achieve this, define

$$\boldsymbol{B}_{(l)}(\boldsymbol{s}, h) = (Nh)^{-\frac{1}{2}}[\boldsymbol{B}_l(s_1, h), ..., \boldsymbol{B}_l(s_g, h)]^{\mathrm{T}}$$

and

$$\boldsymbol{D}^*_{(l)}(\boldsymbol{s}, h) = (Nh)^{-1}\begin{bmatrix} \boldsymbol{D}^*_{ll}(s_1, s_1, h) & ... & \boldsymbol{D}^*_{ll}(s_1, s_g, h) \\ ... & & ... \\ \boldsymbol{D}^*_{ll}(s_g, s_1, h) & ... & \boldsymbol{D}^*_{ll}(s_g, s_g, h) \end{bmatrix}$$

for $l = 0, 1, ..., k$. It follows that $\left(\widehat{\beta}_l(\boldsymbol{s}, h) - \beta_l(\boldsymbol{s})\right) \rightarrow \mathrm{N}_g\left(\boldsymbol{B}_{(l)}(\boldsymbol{s}, h), \boldsymbol{D}^*_{(l)}(\boldsymbol{s}, h)\right)$ for $l = 0, 1, ..., k$.

CHAPTER THREE

METHOD OF CONFIDENCE BAND CONSTRUCTION

In order to develop an alternative to the methods proposed by Wu, Chiang, and Hoover (1998) as shown in (1.8) and (1.9), we will fit a parametric estimate to the non-parametric estimate given by (1.5). Specifically, to construct confidence regions for $\beta_0(t), \beta_1(t), ..., \beta_k(t)$ over some interval$[a, b]$, begin by selecting a grid of time points $\boldsymbol{s} = (s_1, s_2, ..., s_q)^{\mathrm{T}}$, such that $s_1 = a,\ s_2 = b$, and $s_{j+1} - s_j = \delta$ for all $j = 1, ..., q-1$. After constructing the estimator $\widehat{\boldsymbol{\beta}}(\boldsymbol{s}, h)$ and correcting for bias (if required) via a suitable estimate of $\boldsymbol{B}(\boldsymbol{s}, h)$, a polynomial will be fit to $\widehat{\boldsymbol{\beta}}_l(\boldsymbol{s}, h) = \left(\widehat{\beta}_l(s_1), ..., \widehat{\beta}_l(s_q)\right)^{\mathrm{T}}$ for $l = 0, 1, ..., k$. If a polynomial of degree d_l is to be fit to $\widehat{\boldsymbol{\beta}}_l(\boldsymbol{s}, h)$, define

$$\boldsymbol{S}_l = \begin{bmatrix} 1 & s_1 & s_1^2 & ... & s_1^{d_l} \\ 1 & s_2 & s_2^2 & ... & s_2^{d_l} \\ ... & & & & ... \\ 1 & s_q & s_q^2 & ... & s_q^{d_l} \end{bmatrix}.$$

Then define the polynomial overlay as

$$\widehat{p}_l(t) = \widehat{\alpha}_{l0} + \widehat{\alpha}_{l1} t + ... + \widehat{\alpha}_{ld_l} t^{d_l} \tag{3.1}$$

where

$$\widehat{\boldsymbol{\alpha}}_l = (\widehat{\alpha}_{l0}, \widehat{\alpha}_{l1}, ..., \widehat{\alpha}_{ld_l})^{\mathrm{T}} = \left(\boldsymbol{S}_l^{\mathrm{T}} \widehat{\boldsymbol{D}}_{(l)}^{*-1}(\boldsymbol{s}, h) \boldsymbol{S}_l\right)^{-1} \boldsymbol{S}_l^{\mathrm{T}} \widehat{\boldsymbol{D}}_{(l)}^{*-1}(\boldsymbol{s}, h) \widehat{\boldsymbol{\beta}}_l(\boldsymbol{s}, h) \tag{3.2}$$

and $\widehat{\boldsymbol{D}}^{*}_{(l)}(\boldsymbol{s},h)$ is a suitable estimate of $\boldsymbol{D}^{*}_{(l)}(\boldsymbol{s},h)$. It follows from standard GLS theory that $\widehat{\boldsymbol{\alpha}}_l$ has an asymptotic normal distribution whose covariance structure can be estimated by

$$\boldsymbol{V}_l = \left(\boldsymbol{S}_l^{\mathrm{T}}\widehat{\boldsymbol{D}}^{*\,-1}_{(l)}(\boldsymbol{s},h)\boldsymbol{S}_l\right)^{-1}. \tag{3.3}$$

To build confidence bands based on this polynomial overlay, an adaptation of the technique set forth by Naiman (1986) will be utilized. Naiman considers a regression scenario where

$$\mathrm{E}[Y] = \sum_{j=1}^{k} a_j f_j(x), \quad \boldsymbol{a} = (a_1, a_2, ..., a_k)^{\mathrm{T}}$$

and

$$\boldsymbol{f}(x) = (f_1(x), ..., f_k(x))^{\mathrm{T}}$$

defines a function from an interval to $\Re^k$ that is continuous, bounded away from the origin, and is piecewise differentiable with $\int_I \|\boldsymbol{f}'(x)\|^2 dx$ finite. For an estimator of $\boldsymbol{a}$, $\widehat{\boldsymbol{a}} \sim \mathrm{N}(\boldsymbol{a}, \sigma^2\boldsymbol{\Sigma})$ and $\boldsymbol{P}^{\mathrm{T}}\boldsymbol{P} = \boldsymbol{\Sigma}$, Naiman considers construction of bands of the form

$$\widehat{\boldsymbol{a}}^{\mathrm{T}}\boldsymbol{f}(x) \pm c \cdot s \cdot p(x) \text{ for } x \in I \tag{3.4}$$

where $p(x) = \|\boldsymbol{Pf}(x)\|$ and s is the usual estimate of σ. In order to choose an appropriate c, Naiman demonstrates that a lower bound for the coverage probability of (3.4) is given by

$$1 - \int_0^{\frac{1}{c}} \min\left\{ F_{k-2,2}\left[\frac{2((ct)^{-2}-1)}{k-2}\right] \times \frac{\Lambda(\gamma)}{\pi} + F_{k-1,1}\left[\frac{((ct)^{-2}-1)}{k-1}\right], 1\right\} f_T(t)\, dt \tag{3.5}$$

where f_T denotes the density of a random variable T such that $kT^2 \sim F_{\nu,k}$ (ν being the degrees of freedom associated with s) and $\Lambda(\gamma) = \int_I \|\gamma'(x)\|\, dx$ for $\gamma(x) = \|\boldsymbol{Pf}(x)\|^{-1}\boldsymbol{Pf}(x)$.

If we assume $\epsilon(t)$ in (1.2) does not contain measurement errors and is stationary, we have $\sigma^2(t_0) = \rho_\epsilon(t_0)$, $\sigma^2(t_0) = \sigma^2\, \forall\, t$, and $\rho_\epsilon(t_1, t_2) = \sigma^2\rho(t_1, t_2)$ where $\rho(t_1, t_2)$ is the correlation between $\epsilon(t_1)$ and $\epsilon(t_2)$. This allows σ^2 to be factored out of the covariance structure given in (3.3). Then (3.2), assuming correction for the bias of the kernel estimate has been made, satisfies the distributional requirements on $\widehat{\boldsymbol{a}}$ asymptotically. Also, for $t \in [a, b]$ with $a > 0$, the standard polynomial basis clearly satisfies the conditions on $\boldsymbol{f}(x)$. Hence, Naiman's technique will be employed to construct confidence bands for $\beta_l(t)$, $l = 0, 1, ..., k$, based on the polynomial estimates given by (3.1) and the estimated asymptotic covariance structure in (3.3). In particular, for some estimate $\widehat{\sigma}^2$ of σ^2 and for $l = 0, .., k$, $\left(\widehat{\sigma}^2\right)^{-1}\boldsymbol{V}_l$ is taken as $\boldsymbol{\Sigma}$ for this adaptation of Naiman's procedure and $\boldsymbol{g}_l(t) = (1, t, t^2, ..., t^{d_l})^{\mathrm{T}}$ filling the role of $\boldsymbol{f}$.

The parameter ν in Naiman's procedure refers to the degrees of freedom associated with the variance estimate. Very little is known about the properties of variance estimates for the varying coefficient model; hence, a conservative estimate will be made. Although observations on an individual subject are correlated, it is clear that we will have a minimum of *n* total degrees of freedom, from which we will subtract the number of parameters required to fit the model. So for each coefficient function, the appropriate critical value, c_l, is computed from (3.5) using $\nu = n - \sum_{l=0}^{k}(d_l + 1)$. Then the confidence band

$$\widehat{\boldsymbol{\alpha}}_l^{\mathrm{T}} \boldsymbol{g}_l(t) \pm c_l \cdot \widehat{\sigma} \cdot p_l(t)$$

is constructed with $p_l(t) = \|\boldsymbol{P}_l \boldsymbol{g}_l(t)\|$ where $\boldsymbol{P}_l^{\mathrm{T}} \boldsymbol{P}_l = \left(\widehat{\sigma}^2\right)^{-1} \boldsymbol{V}_l$.

In order to complete the procedure, estimates of the parameters in $\boldsymbol{B}(\boldsymbol{s}, h)$ and $\boldsymbol{D}^*(\boldsymbol{s}, h)$ need to be determined. Wu *et al.* (1998) provide estimators for several of the parameters. To estimate the density for the distribution of measurement times, $f(t)$, a standard kernel density estimate is proposed

$$\widehat{f}(t_0) = (Nh)^{-1} \sum_{i=1}^{n} \sum_{j=1}^{m_i} K\left(\tfrac{t_0 - t_{ij}}{h}\right).$$

An estimate of $\eta_{lr}(t_0)$ is given by

$$\widehat{\eta}_{lr}(t_0) = \left(Nh\widehat{f}(t_0)\right)^{-1} \sum_{i=1}^{n} \sum_{j=1}^{m_i} X_{il}(t_{ij}) X_{ir}(t_{ij}) K\left(\tfrac{t_0 - t_{ij}}{h}\right).$$

Hence, we may estimate $E_{\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}}(t_0)$ by

$$\widehat{E}_{\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}}(t_0) = \begin{bmatrix} \widehat{\eta}_{00}(t_0) & \widehat{\eta}_{01}(t_0) & \ldots & \widehat{\eta}_{0k}(t_0) \\ \widehat{\eta}_{10}(t_0) & \widehat{\eta}_{11}(t_0) & \ldots & \widehat{\eta}_{1k}(t_0) \\ \ldots & & \ldots & \ldots \\ \widehat{\eta}_{k0}(t_0) & \widehat{\eta}_{k1}(t_0) & \ldots & \widehat{\eta}_{kk}(t_0) \end{bmatrix}.$$

Derivative estimates will be constructed by a secant approximation. For example, for our grid of time points $\boldsymbol{s} = (s_1, s_2, \ldots, s_g)^{\mathrm{T}}$, an estimate of $\beta_l'(s_v)$ for $v = 2, \ldots, g-1$ is given by

$$\widehat{\beta}_l'(s_v) = \frac{\widehat{\beta}_l(s_{v+1}) - \widehat{\beta}_l(s_{v-1})}{s_{v+1} - s_{v-1}}.$$

Other required derivative estimates, $\widehat{\beta}_l''(s_v)$, $\widehat{f}'(s_v)$ and $\widehat{\eta}_{lr}'(s_v)$, are all defined in a similar fashion.

Under the assumption of a stationary error process, the estimator for σ^2 provided by Wu *et al.* simplifies to

$$\widehat{\sigma}^2_{\text{Wu}} = \frac{1}{N}\sum_{i=1}^{n}\sum_{j=1}^{m_i}\widehat{\epsilon}_{ij}^2$$

where $\widehat{\epsilon}_{ij} = Y_{ij} - \boldsymbol{X}_{ij}^{\mathrm{T}}\widehat{\boldsymbol{\beta}}(t_{ij})$, $j = 1, \ldots, m_i$ and $i = 1, \ldots, n$. This estimator has the unfortunate property of generally underestimating the value of σ^2. A somewhat more conservative estimate can be devised by first considering subjects separately. An estimate of σ^2 from an individual subject's time series would typically take the form

$$\widehat{\sigma}^2_{(i)} = \frac{1}{m_i - 1}\sum_{j=1}^{m_i}\widehat{\epsilon}_{ij}.$$

Since the variance is the same for each subject, we arrive at our final estimate by using the above estimator pooled across subjects

$$\widehat{\sigma}^2 = \frac{1}{N-n}\sum_{i=1}^{n}\sum_{j=1}^{m_i}\widehat{\epsilon}_{ij}.$$

This estimator generally provides a slight overestimate of the actual variance and will be employed in the simulations discussed in later chapters.

Finally, an estimate of $\rho(t_1, t_2)$ is required. For this, a variogram (see Diggle, Liang, and Zeger (1998)) approach will be utilized. The variogram is defined as

$$\nu(u) = \tfrac{1}{2}E\big[\{Y(t) - Y(t-u)\}^2\big],\ u > 0$$

for a stochastic process $Y(t)$. If $Y(t)$ is a stationary process, the variogram is related to the correlation function by

$$\nu(u) = \sigma^2[1 - \rho(u)]$$

An estimate of the variogram can be computed from squared differences between residuals

$$v_{ijk} = \tfrac{1}{2}(\widehat{\epsilon}_{ij} - \widehat{\epsilon}_{ik})^2$$

and their corresponding time differences

$$u_{ijk} = t_{ij} - t_{ik},\ j > k.$$

If the observation times are relatively regular, there should be multiple observations for several values of u. Then, for a value of u present in the sample, $\widehat{\nu}(u)$ is taken to be the average of the v_{ijk} at that particular u. Values of $\widehat{\nu}(u)$ can be estimated by interpolation or smoothing for values of u not present in the sample. Hence, our estimate for $\rho(t_1, t_2)$ is given, for $u = |t_1 - t_2|$, by

$$\widehat{\rho}(u) = 1 - \tfrac{\widehat{\nu}(u)}{\widehat{\sigma}^2}.$$

CHAPTER FOUR
SIMULATION STUDY

To investigate the viability of the proposed procedure for finite samples, several simulation studies are undertaken. Estimated coverage probabilities and average bands are displayed for both the proposed method and the Bonferroni method of Wu, Chiang, and Hoover (1998) in order to facilitate comparison between the two.

4.1 Simulation case one

This case closely follows the simulation presented by Wu *et al.* In this study, the covariate $\boldsymbol{X} = (1, X_1, X_2, X_3)^{\mathrm{T}}$ is taken as time independent; which, even though the model allows covariates to be functions of time, is very common to applications discussed in the current literature. Here X_1 and X_2 are each Bernoulli random variables with $p = 0.5$ and $X_3 \sim N(0, \frac{1}{4})$ with X_1, X_2, and X_3 mutually independent. The covariate functions are defined as follows:

$$\beta_0(t) = 15 + 20\sin\left(\tfrac{t\pi}{60}\right) \qquad \beta_1(t) = 4 - \left(\tfrac{t-20}{10}\right)^2$$

$$\beta_2(t) = 2 - 3\cos\left(\tfrac{(t-25)\pi}{15}\right) \qquad \beta_3(t) = \ -5 + \tfrac{(30-t)^3}{5000}.$$

The random error term, $\epsilon(t)$, is taken as a mean 0 Gaussian process, independent of $\boldsymbol{X}$, with

$$\mathrm{Cov}[\epsilon_{i_1}(t_{i_1 j_1}), \epsilon_{i_2}(t_{i_2 j_2})] = \begin{cases} 4\, e^{-|t_{i_1 j_1} - t_{i_2 j_2}|} & \text{if } i_1 = i_2 \\ 0 & \text{if } i_1 \neq i_2 \end{cases}.$$

Finally, time points are constructed so that the initial measurement time, t_{i1} for $i = 1, \ldots, n$, is a random variable uniformly distributed on $(0, 1)$. Remaining measurement times are determined from the initial measurement time by $t_{ij} = t_{i1} + (j-1)$, $j = 2, ..., 30$, with each of these having a 60% probability of being missing.

After generating the random components to suit the above criteria for $n = 200$ subjects, observations are generated by substituting those results and the chosen coefficient functions into model (1.2). Both the Bonferroni method and the method proposed in this paper are used to construct bands at a nominal confidence level of 95%. In each case, the degrees for the polynomial fits were chosen to be $d_0 = 5,\ d_1 = 2,$ $d_2 = 5,\ d_3 = 3$. While the choices for d_1 and d_3 are obvious, the choices for d_0 and d_2 are made by considering the maximum value of the remainder in various Taylor expansions. In the sense of choosing the smallest degree for the polynomial that also gives a reasonable amount of error in approximating the actual function, these choices are optimal. Obviously, choices of polynomial fits made in this fashion would be difficult to implement in practice. Later simulation studies will attempt to investigate problems that could potentially arise from this.

The procedure is performed for several different values of the bandwidth, h, using the standard Gaussian kernel. Bandwidths were selected to be near the typical value of h_{CV}, which is approximately 1.0 in this case. Both methods use a grid of 90 points, constructed by dividing each unit interval in $(0, 30)$ into thirds. Average bands and estimated coverage probabilities are based on 500 replications of the procedure, giving a maximum standard error of 0.0097 for the estimated coverage probabilities.

Table 4.1 shows that the proposed method is conservative with respect to the nominal confidence level of 95%, but is less conservative than the Bonferroni method. The average bands, shown in figures 4.1.1-4.1.20, clearly show that the proposed method has a substantial advantage in indicating the true precision of the estimates of the coefficient functions. This would indicate a substantial advantage for the proposed method in drawing inferences on the true nature of the relationship between the covariate effect and time. A later study will attempt to illustrate this advantage more concretely. (Note: The legend that accompanies figure 4.1.1 applies to all figures in 4.1-4.6)

Table 4.1 Estimated coverage probabilities on $t \in [1, 29]$ for case 1

h	Method	β_0	β_1	β_2	β_3
0.6	Proposed	0.968	0.978	0.966	0.970
	Bonferroni	1.000	1.000	1.000	1.000
0.8	Proposed	0.976	0.972	0.980	0.990
	Bonferroni	1.000	1.000	1.000	1.000
1.0	Proposed	0.976	0.988	0.972	0.982
	Bonferroni	1.000	1.000	1.000	1.000
1.2	Proposed	0.994	0.982	0.982	0.984
	Bonferroni	1.000	1.000	1.000	1.000
1.4	Proposed	0.986	0.986	0.962	0.986
	Bonferroni	1.000	1.000	1.000	1.000

Figure 4.1.1 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 0.6$, case 1

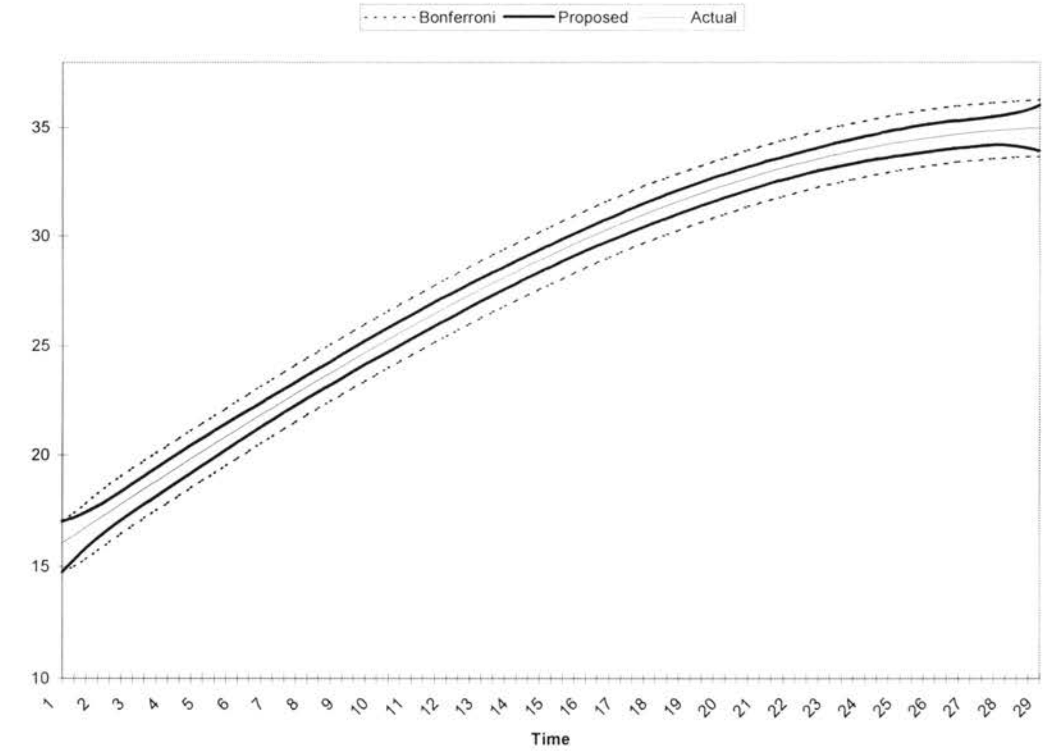


Figure 4.1.2 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 0.6$, case 1

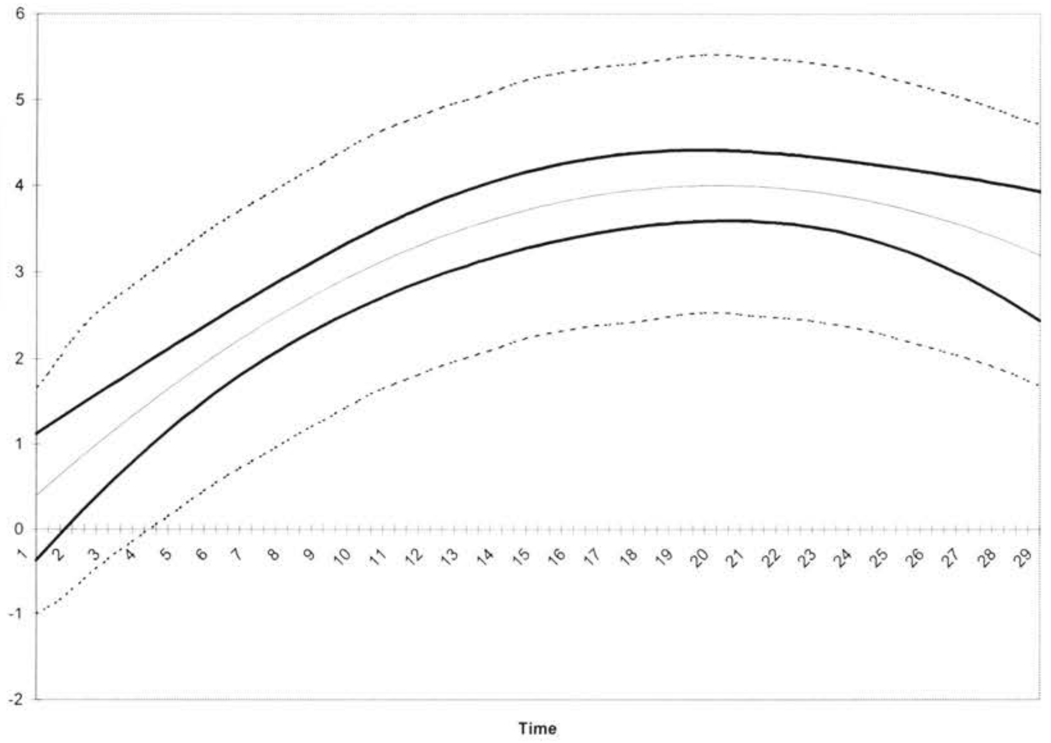


Figure 4.1.3 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 0.6$, case 1

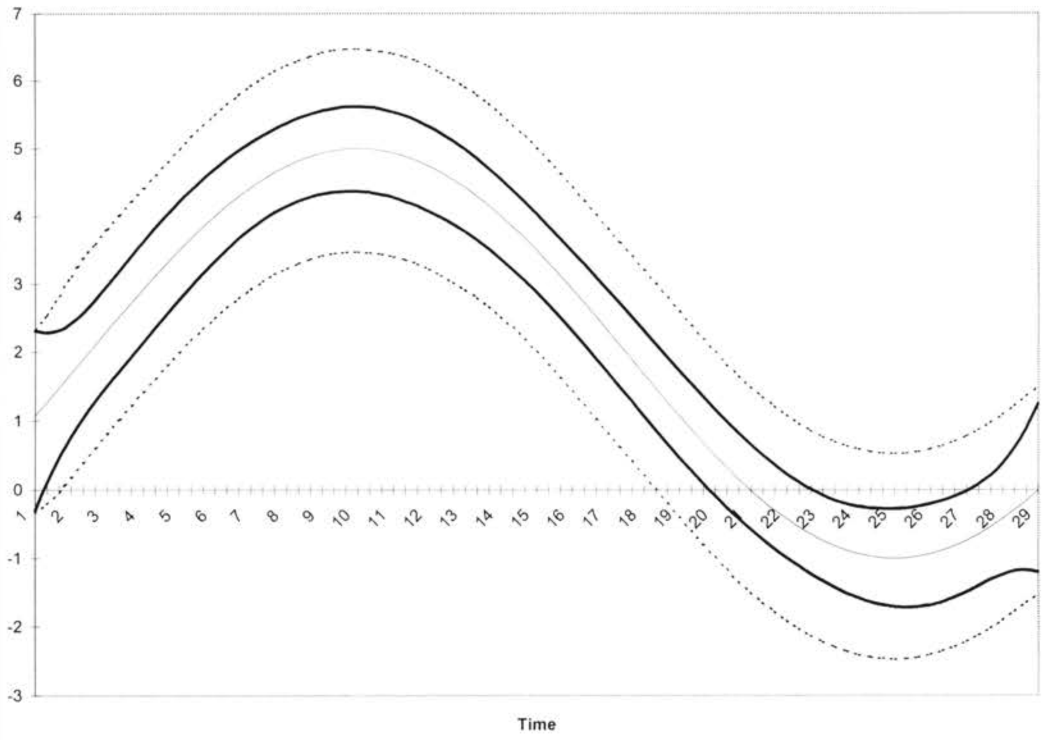


Figure 4.1.4 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 0.6$, case 1

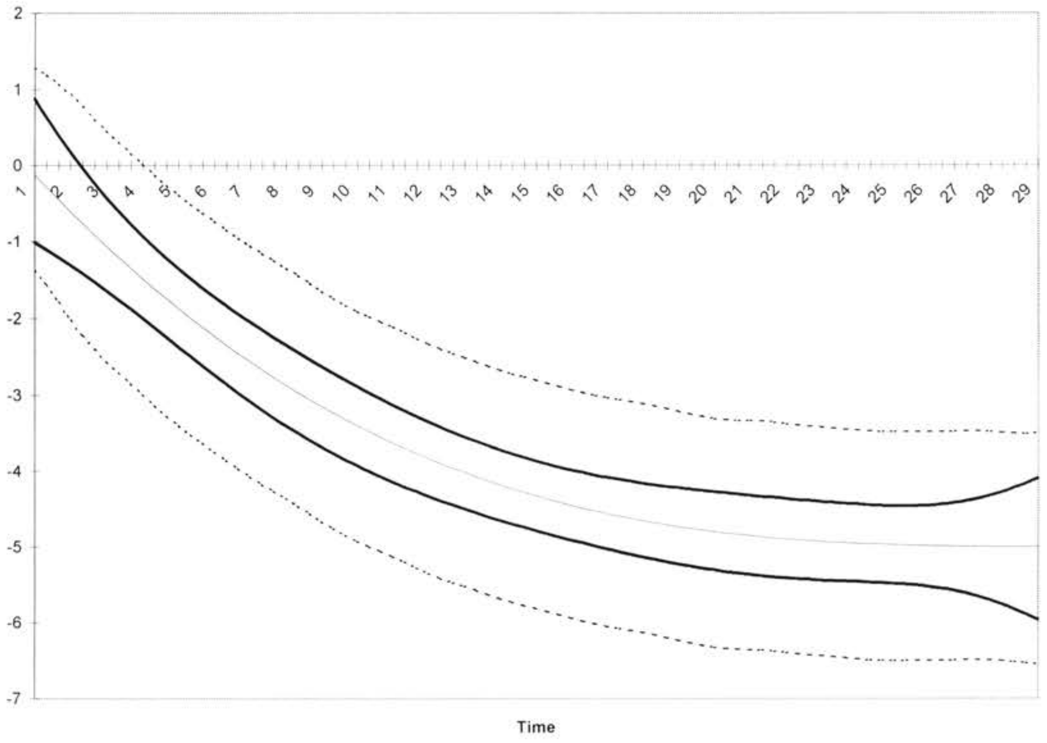


Figure 4.1.5 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 0.8$, case 1

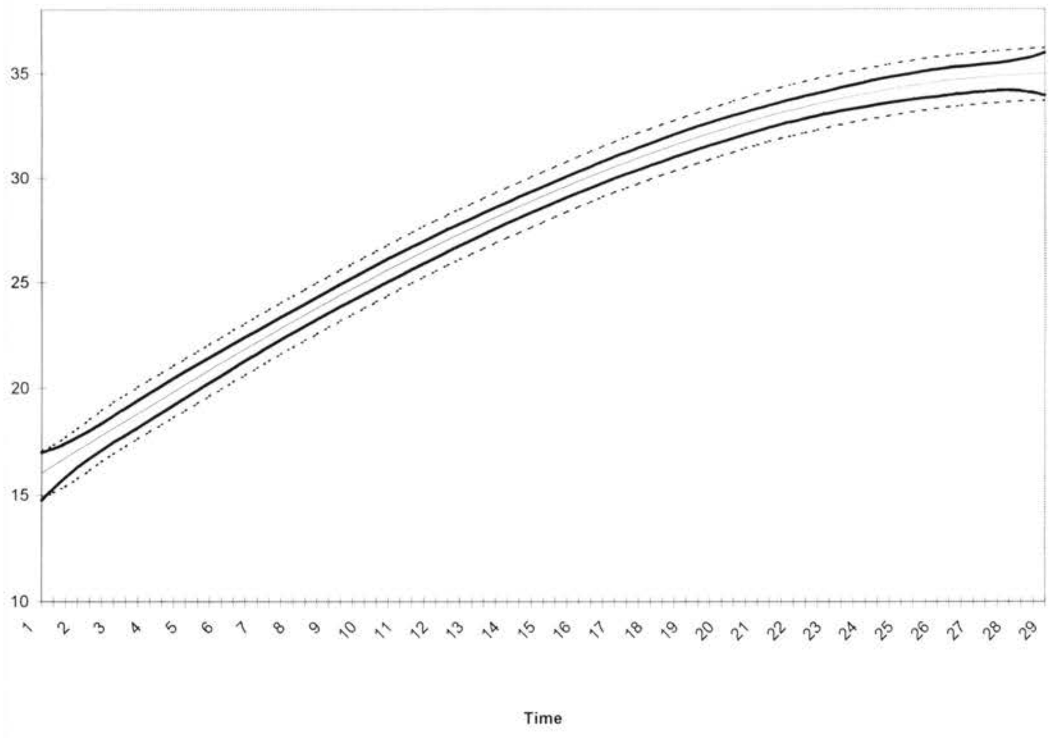


Figure 4.1.6 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 0.8$, case 1

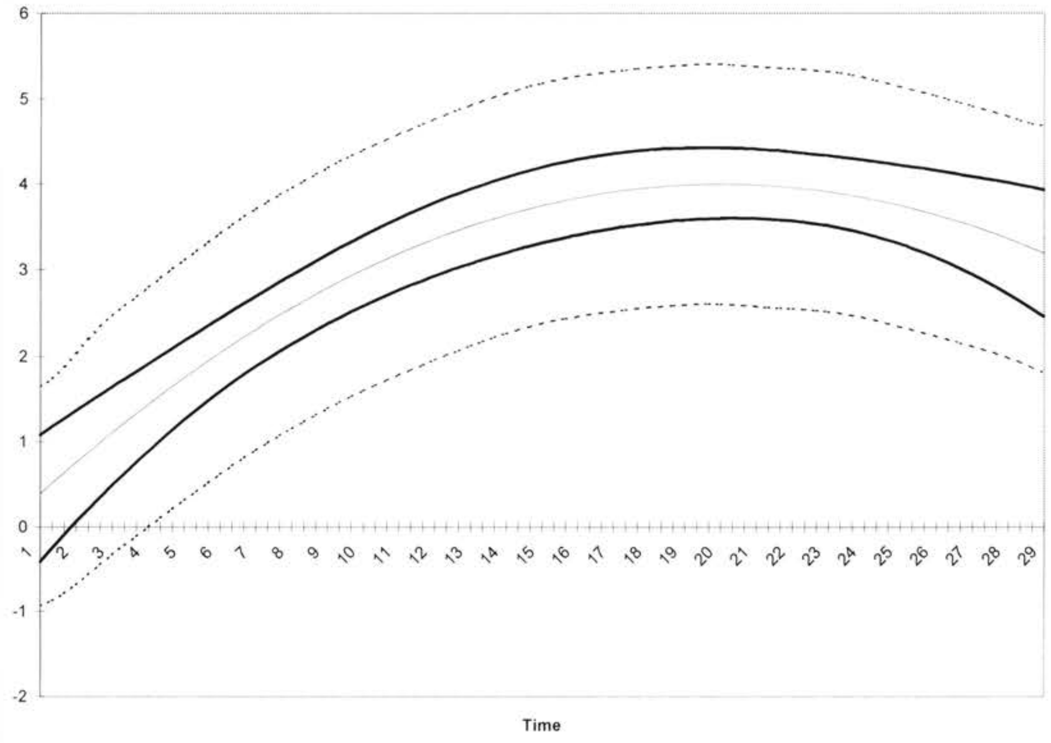


Figure 4.1.7 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 0.8$, case 1

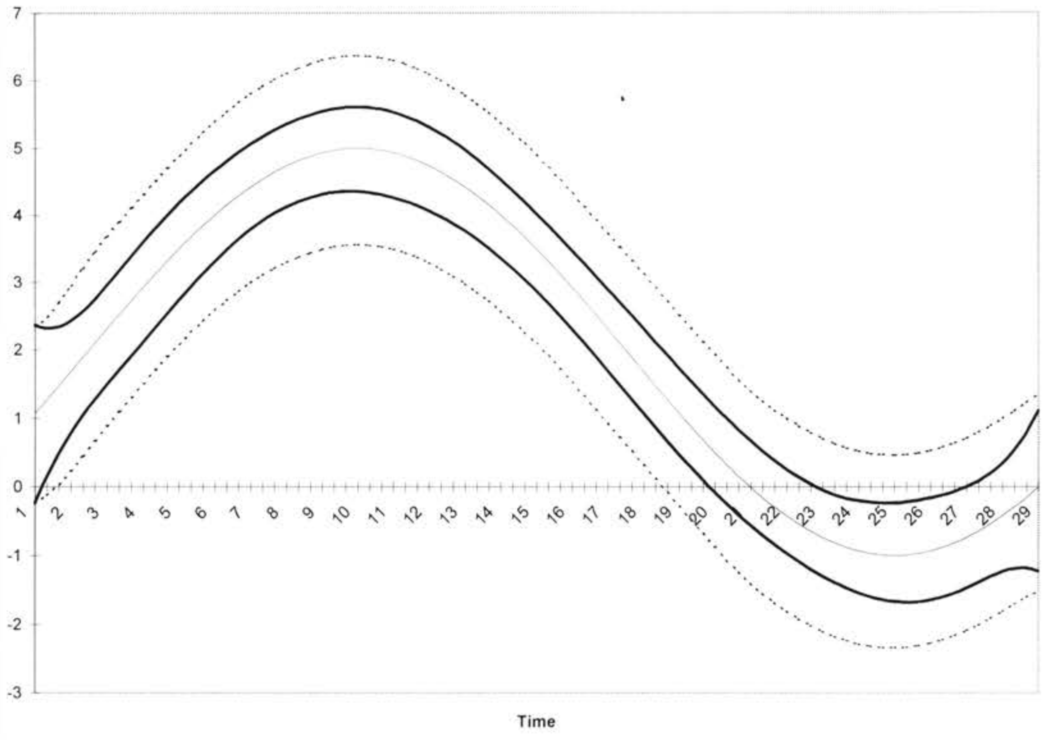


Figure 4.1.8 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 0.8$, case 1

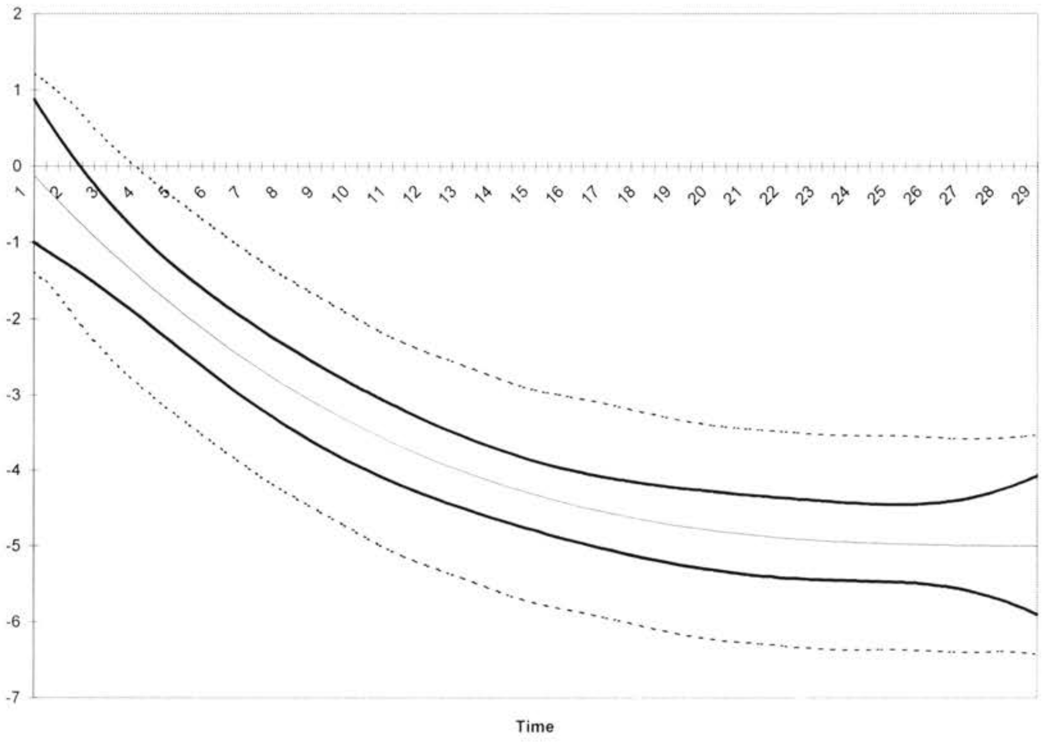


Figure 4.1.9 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 1.0$, case 1

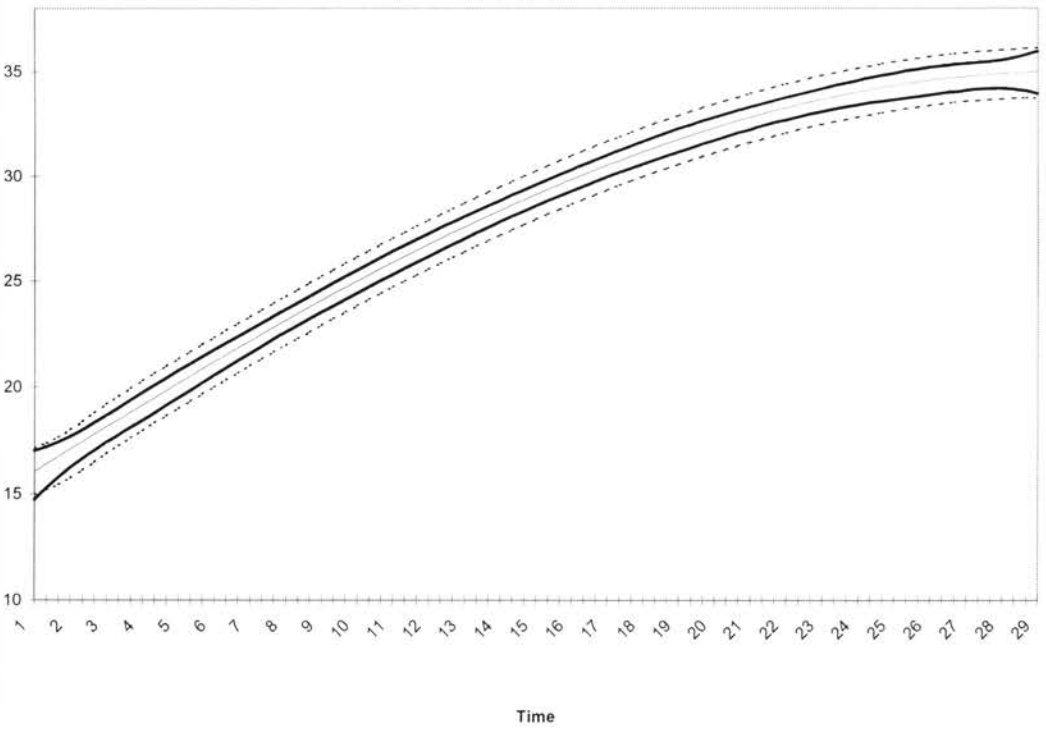


Figure 4.1.10 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 1.0$, case 1

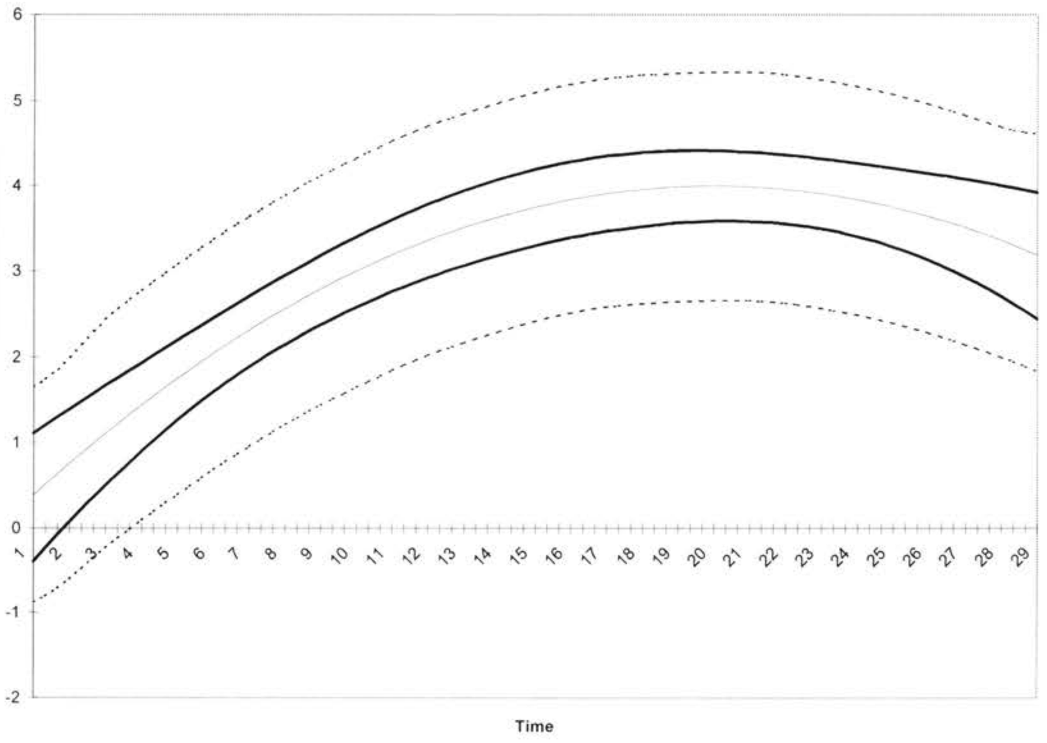


Figure 4.1.11 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 1.0$, case 1

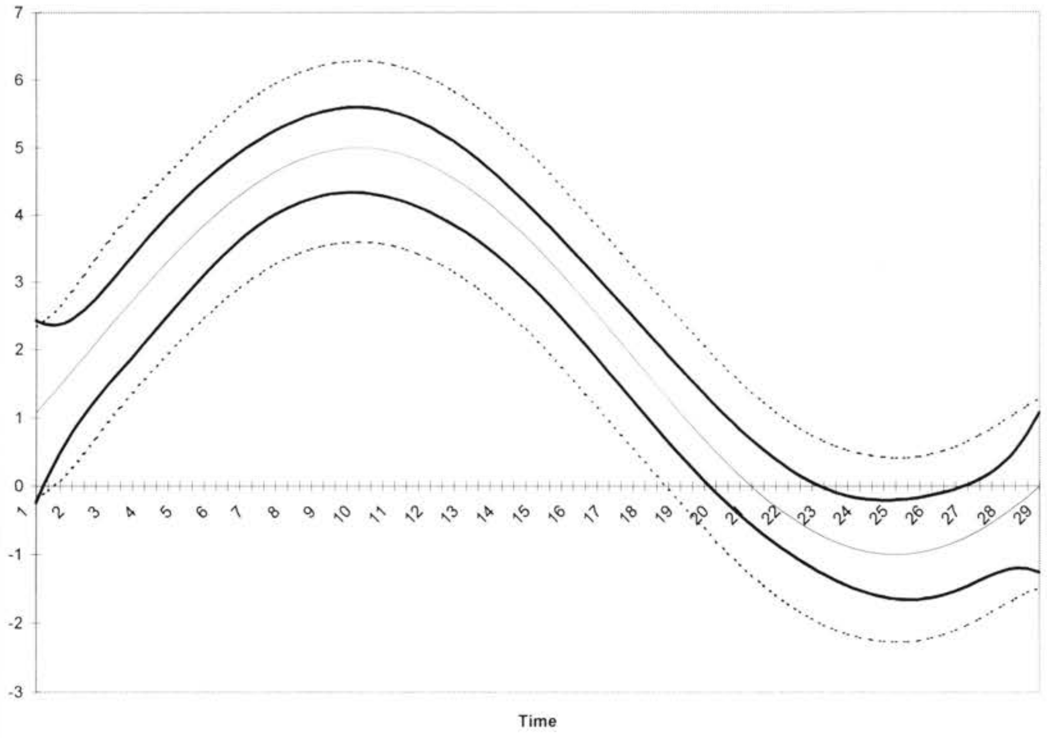


Figure 4.1.12 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 1.0$, case 1

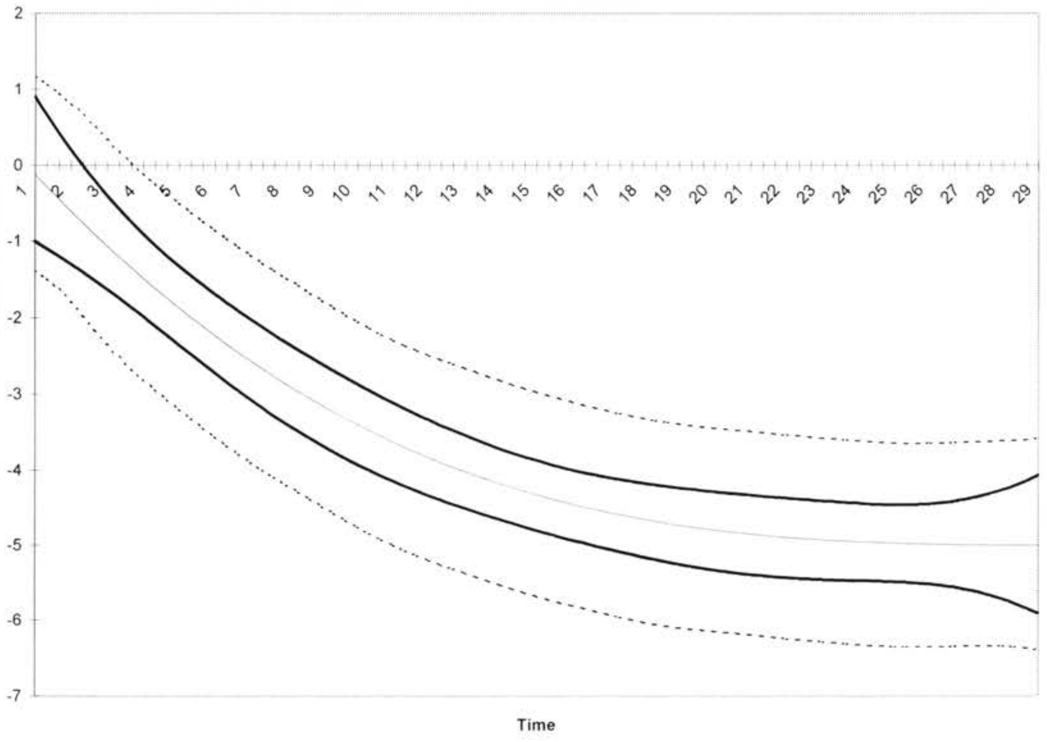


Figure 4.1.13 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 1.2$, case 1

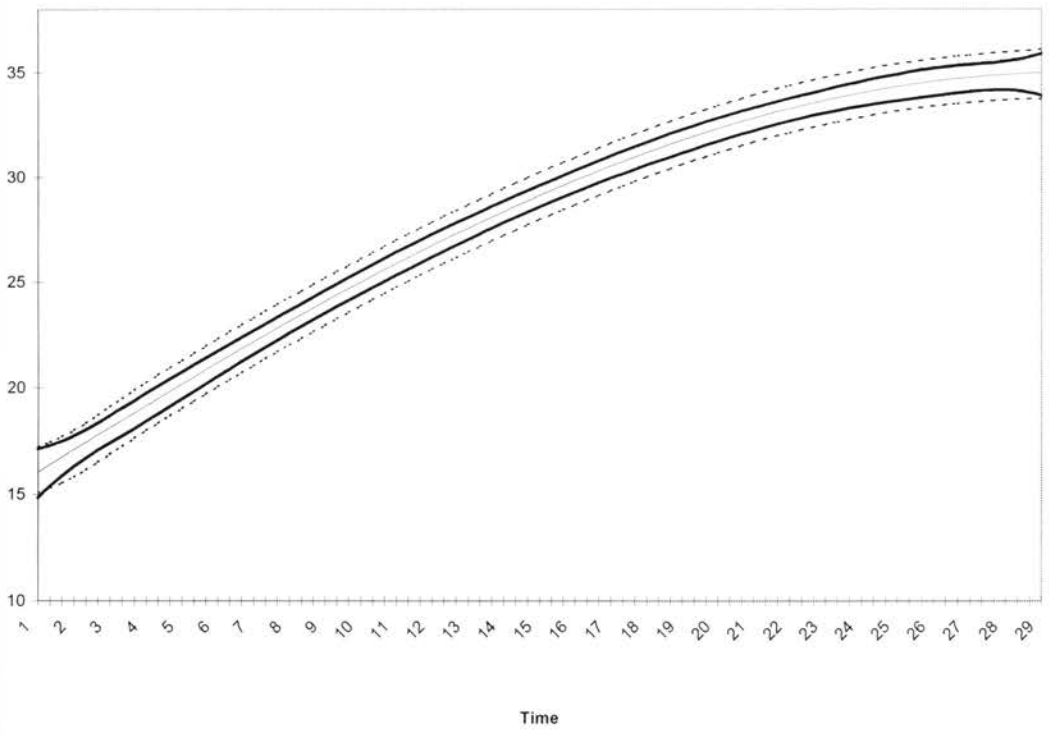


Figure 4.1.14 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 1.2$, case 1

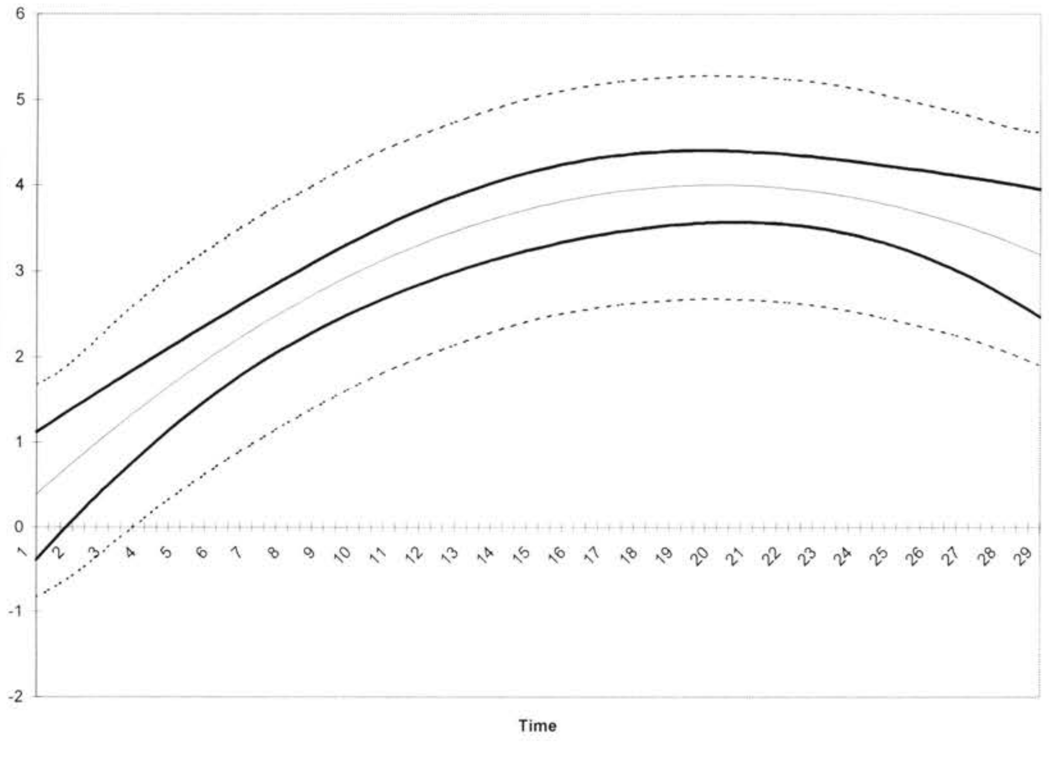


Figure 4.1.15 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 1.2$, case 1

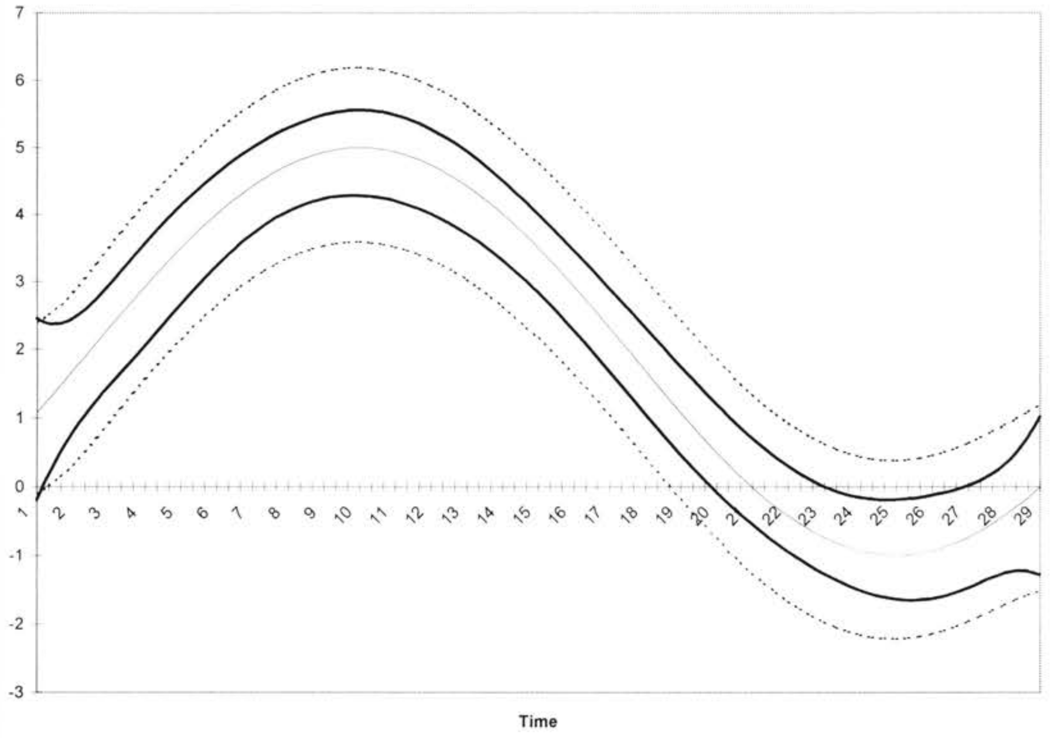


Figure 4.1.16 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 1.2$, case 1

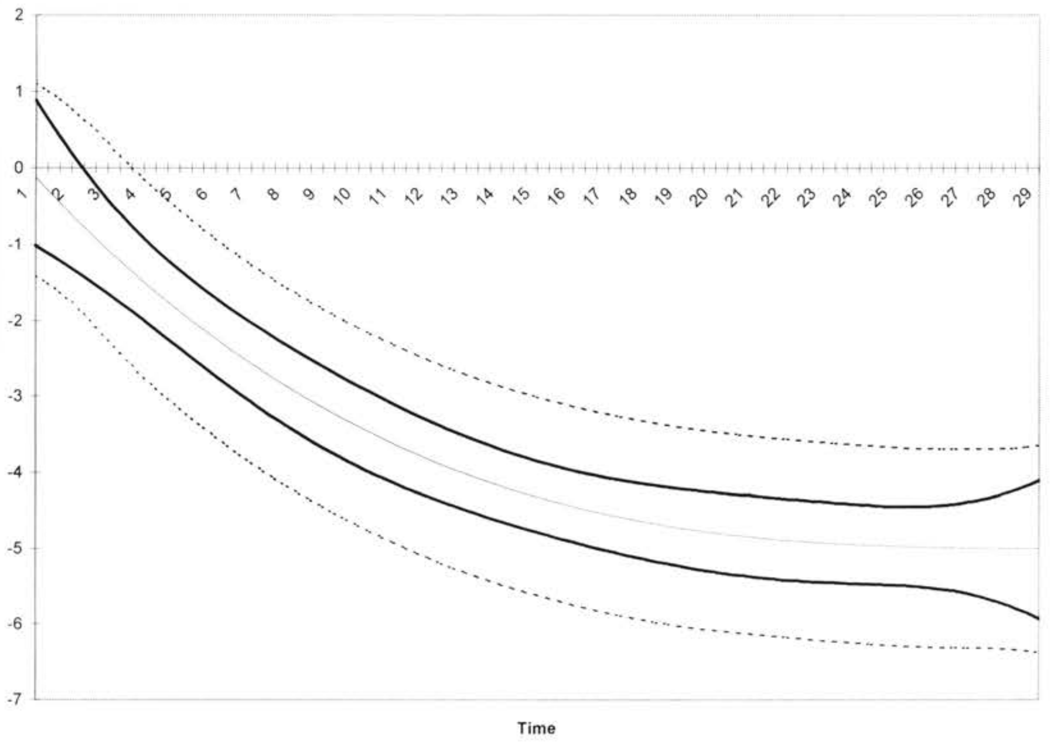


Figure 4.1.17 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 1.4$, case 1

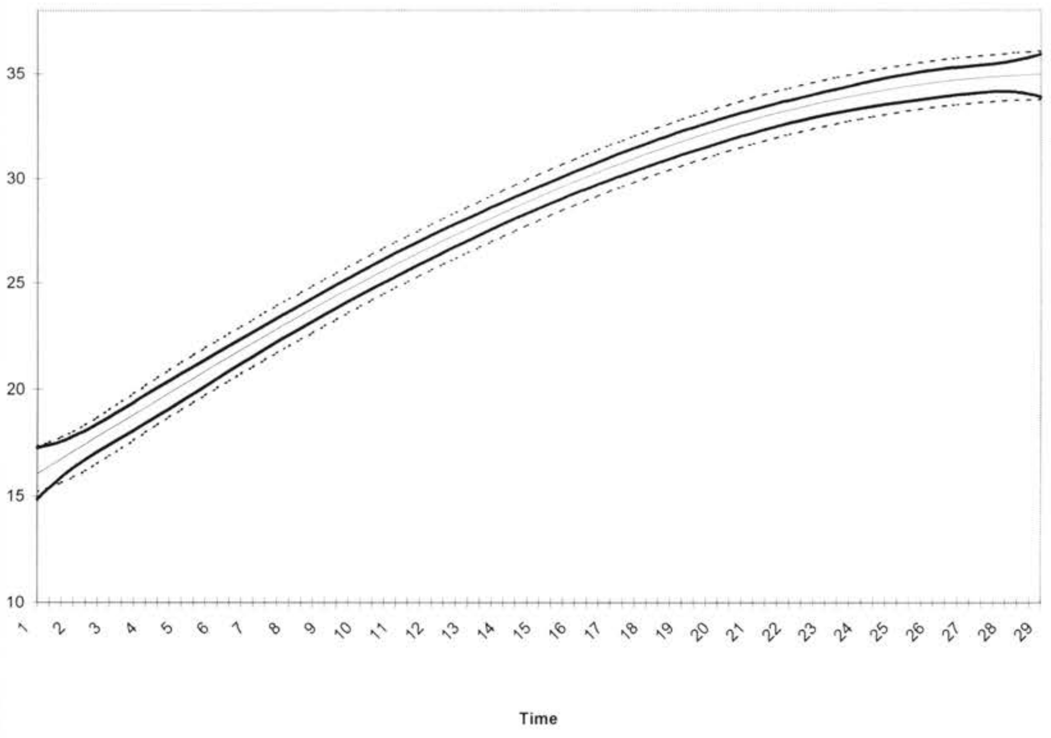


Figure 4.1.18 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 1.4$, case 1

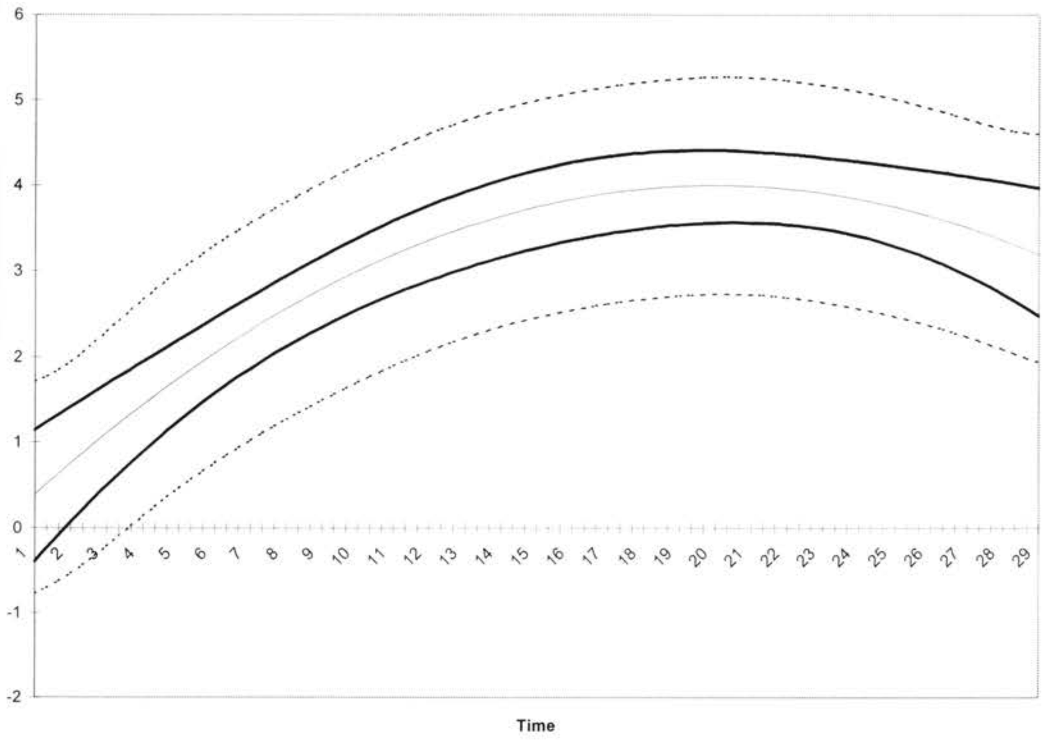


Figure 4.1.19 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 1.4$, case 1

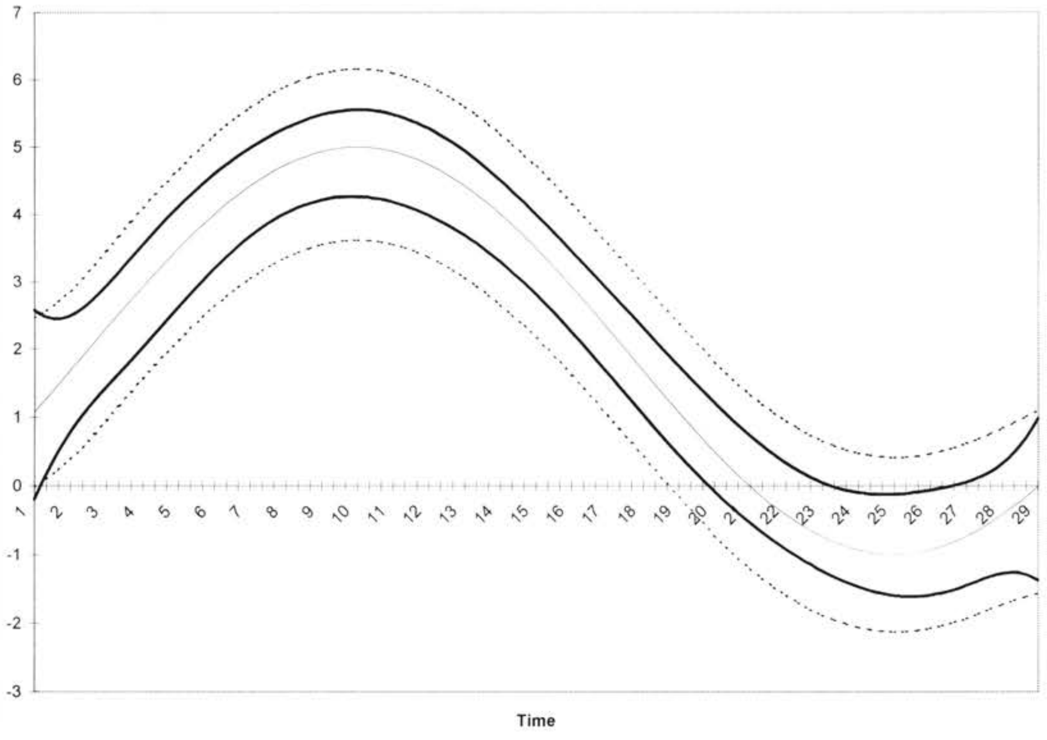
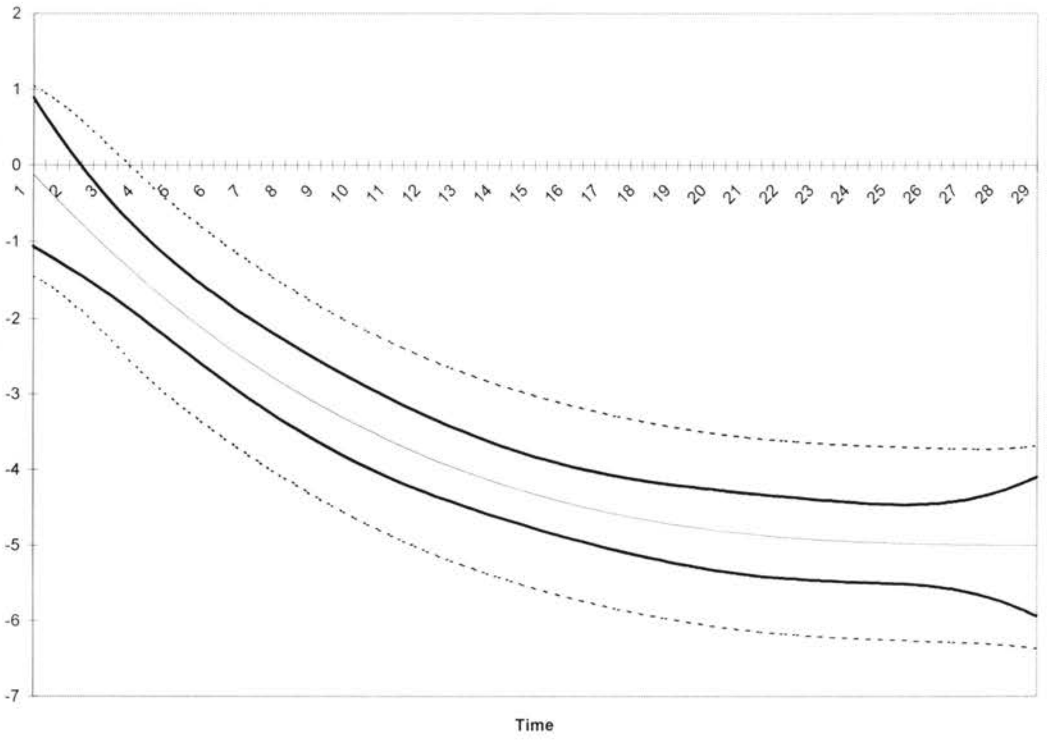


Figure 4.1.20 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 1.4$, case 1



4.2 Simulation case two

This simulation case is precisely the same as case one, except the number of subjects in the study was halved; hence, $n = 100$ for this simulation study. Once again, the proposed method appears to be conservative, but far less so than the Bonferroni method of Wu *et al.* Also, as can be seen from comparing figures 4.1.1-4.1.20 and figures 4.2.1-4.2.20, the width of the bands increases for this decrease in sample size, which is what one would expect. Yet the proposed method maintains its advantage in producing substantially narrower bands, which is especially clear in banding $\beta_1(t)$ (see figures 4.2, 4.6, 4.10, 4.14, 4.18). If one is interested in deciding if $\beta_1(t)$ is truly time dependent, the Bonferroni bands for $\beta_1(t)$ provide no conclusive evidence of that fact since time invariant functions (*i.e.* horizontal lines) would included among the possibilites defined by those bands. However, the bands constructed by the proposed method give strong evidence that $\beta_1(t)$ is, in fact, time dependent. Further examples of this type of inference are taken up in section 4.6.

Table 4.2 Estimated coverage probabilities on $t \in [1, 29]$ for case 2

h	Method	β_0	β_1	β_2	β_3
0.6	Proposed	0.962	0.962	0.954	0.972
	Bonferroni	1.000	0.998	1.000	1.000
0.8	Proposed	0.968	0.972	0.962	0.970
	Bonferroni	1.000	1.000	1.000	1.000
1.0	Proposed	0.972	0.978	0.974	0.972
	Bonferroni	1.000	1.000	1.000	1.000
1.2	Proposed	0.982	0.982	0.980	0.978
	Bonferroni	1.000	1.000	1.000	1.000
1.4	Proposed	0.988	0.986	0.972	0.982
	Bonferroni	1.000	1.000	1.000	1.000

Figure 4.2.1 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 0.6$, case 2

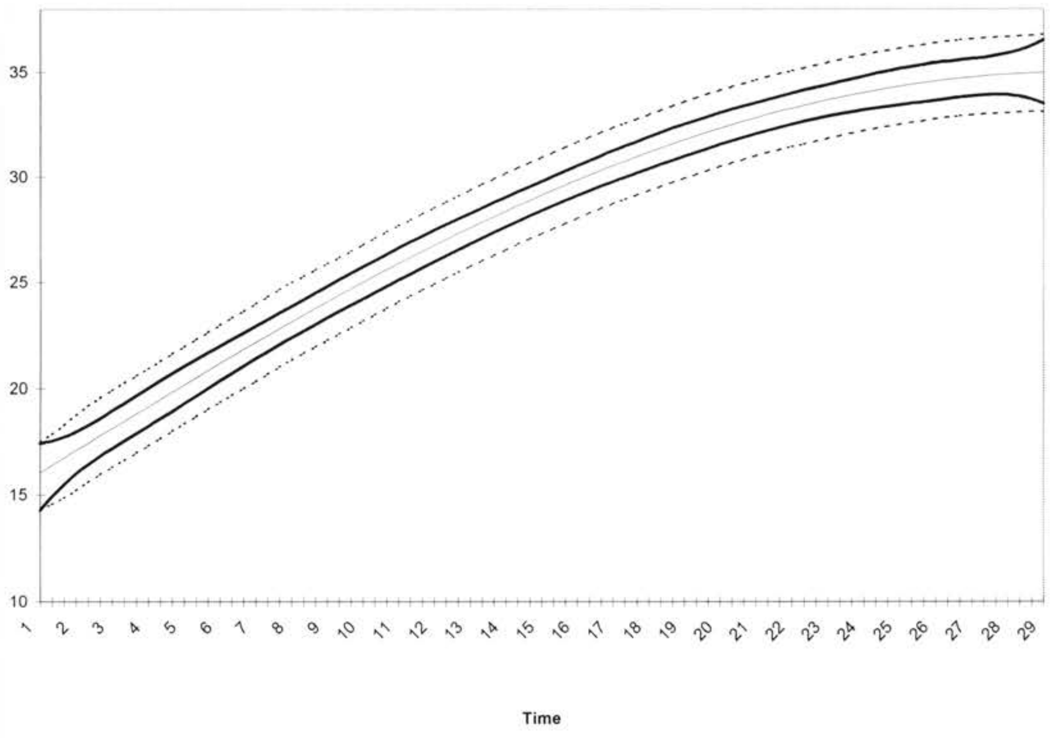


Figure 4.2.2 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 0.6$, case 2

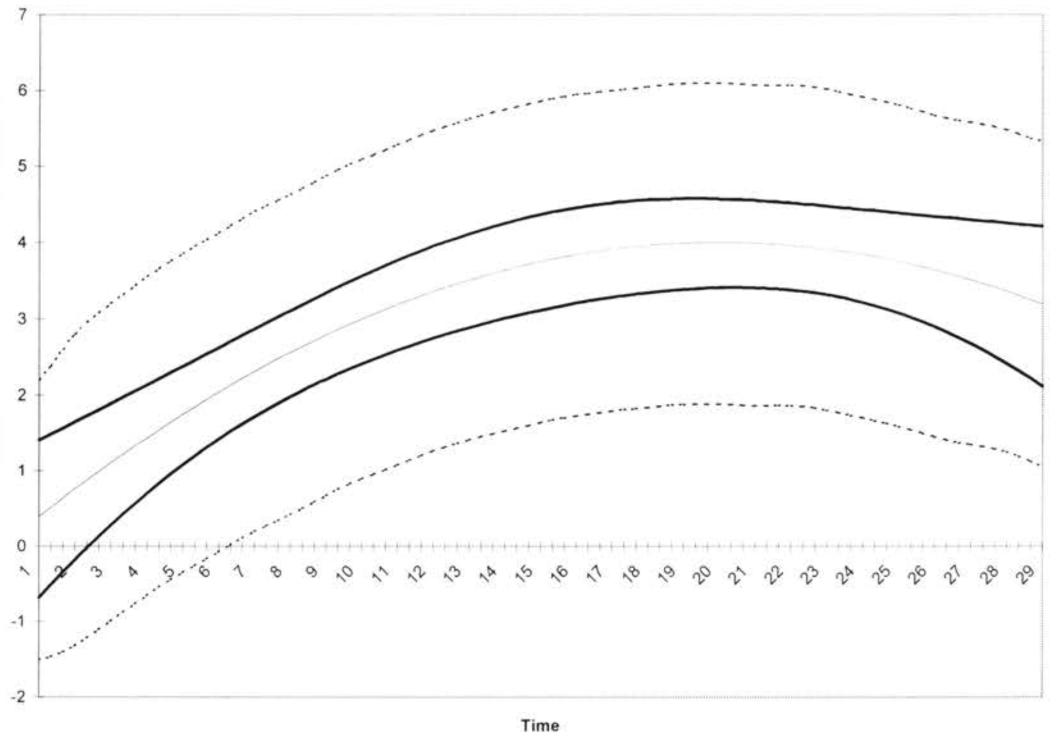


Figure 4.2.3 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 0.6$, case 2

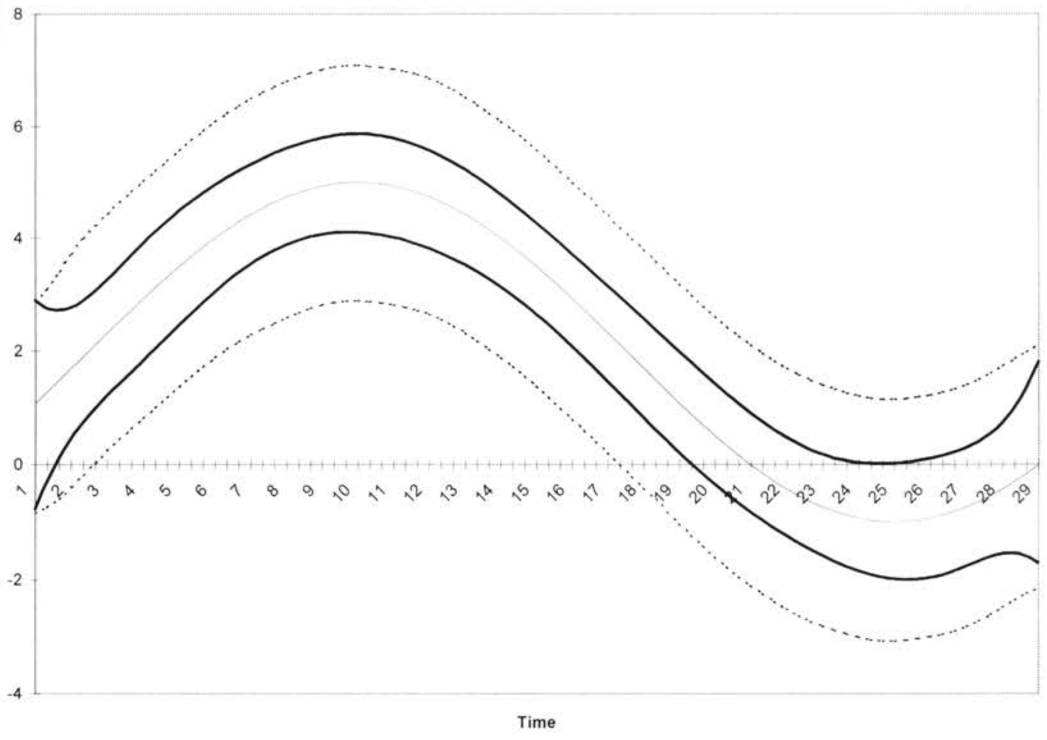


Figure 4.2.4 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 0.6$, case 2

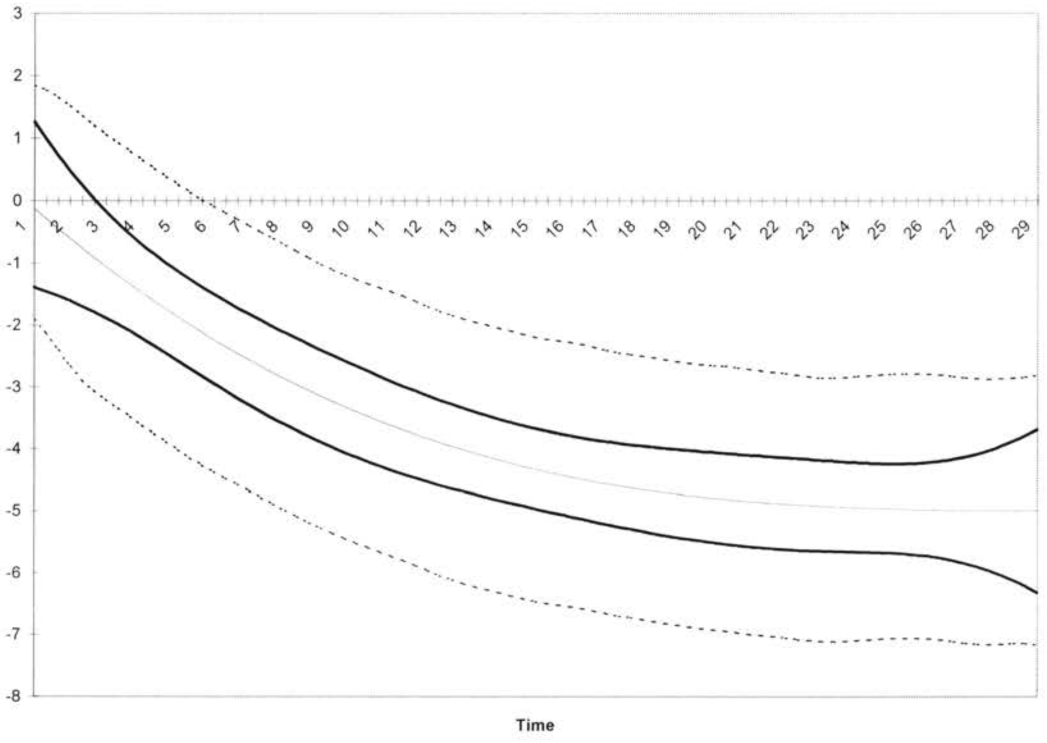


Figure 4.2.5 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 0.8$, case 2

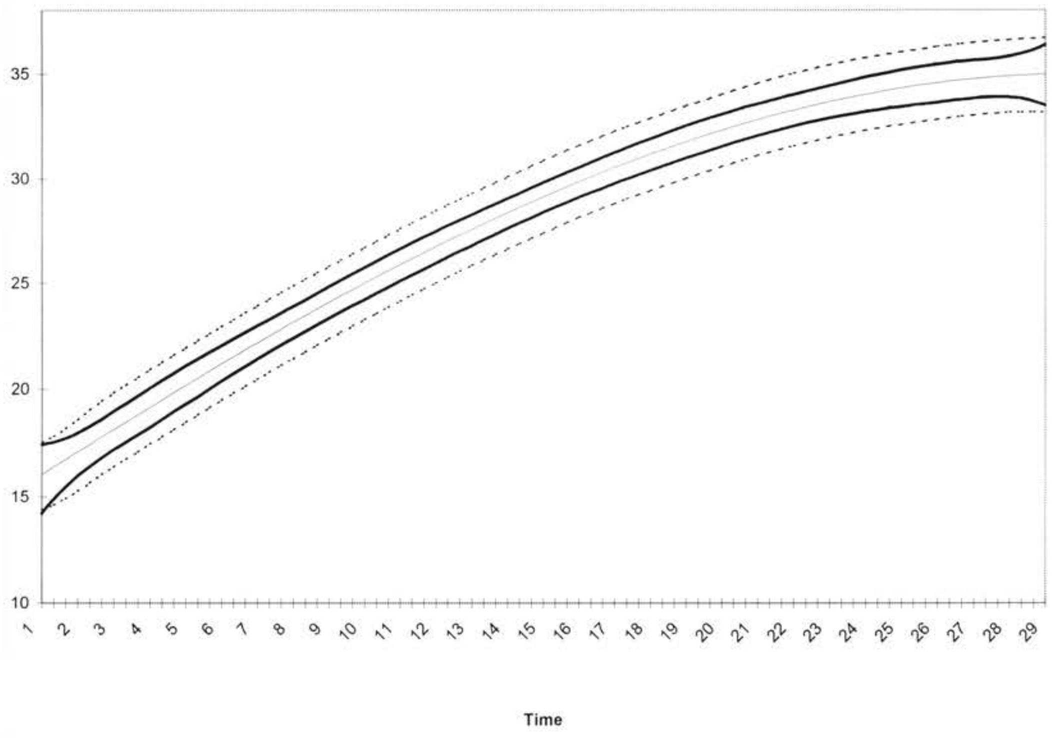


Figure 4.2.6 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 0.8$, case 2

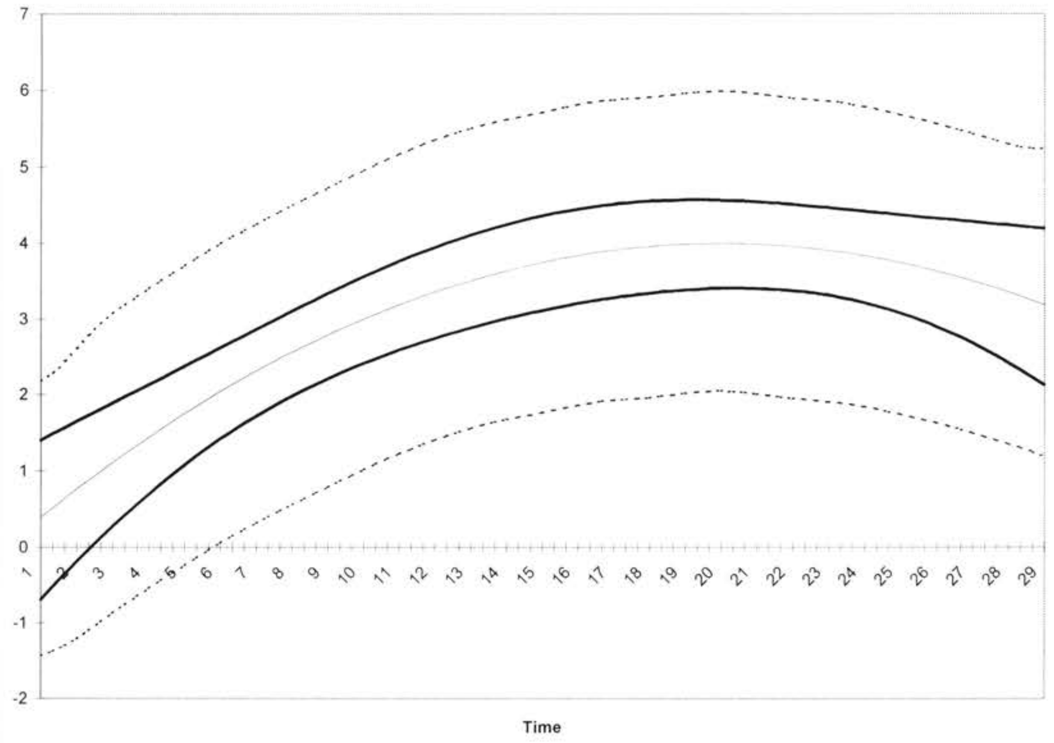


Figure 4.2.7 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 0.8$, case 2

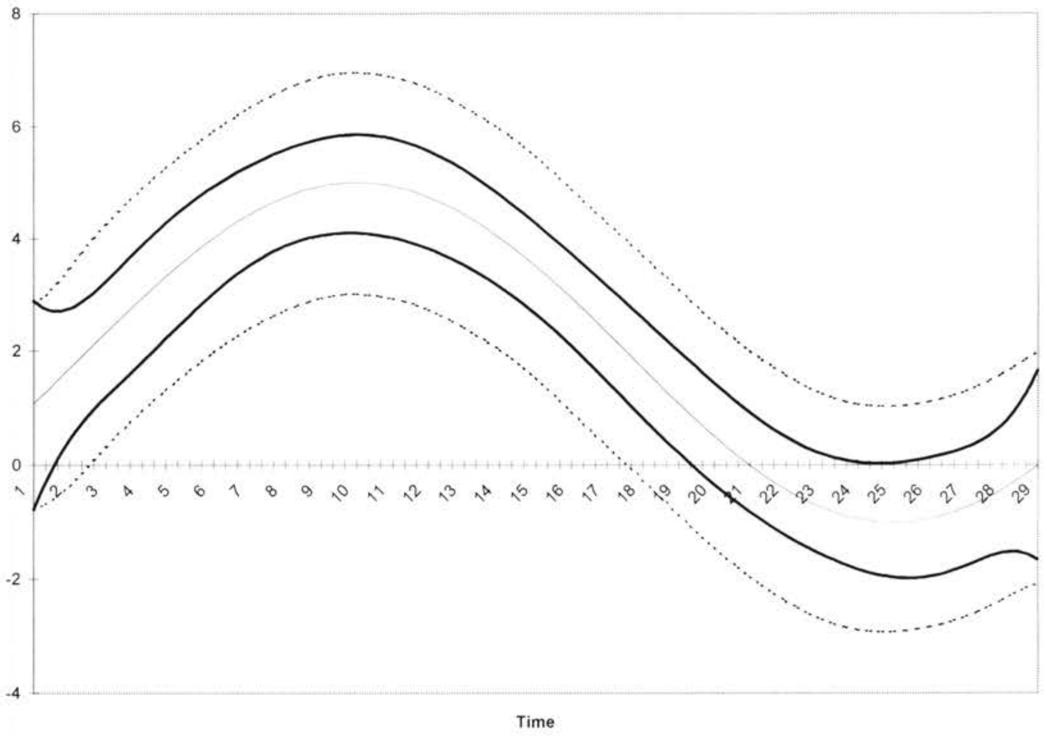


Figure 4.2.8 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 0.8$, case 2

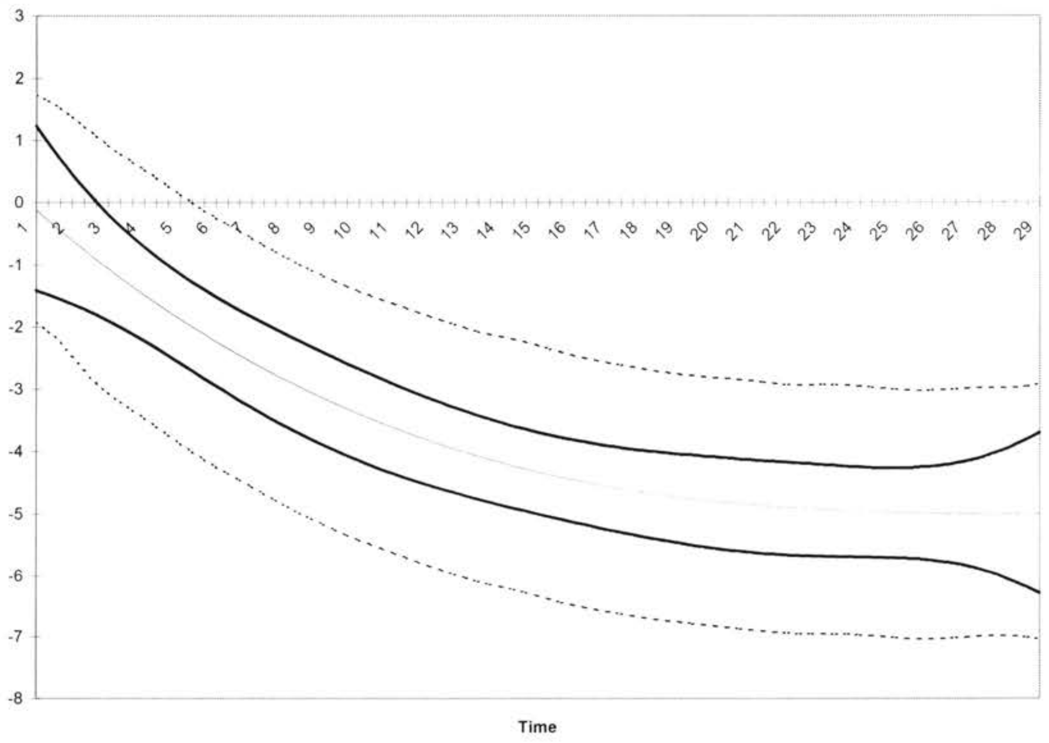


Figure 4.2.9 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 1.0$, case 2

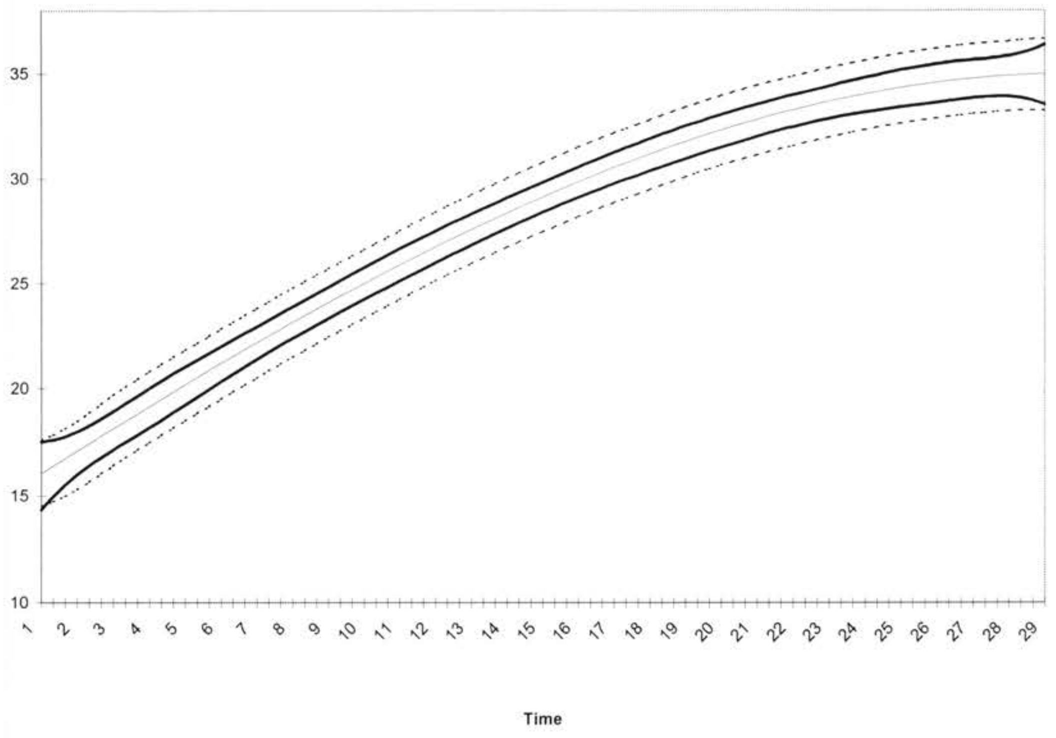


Figure 4.2.10 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 1.0$, case 2

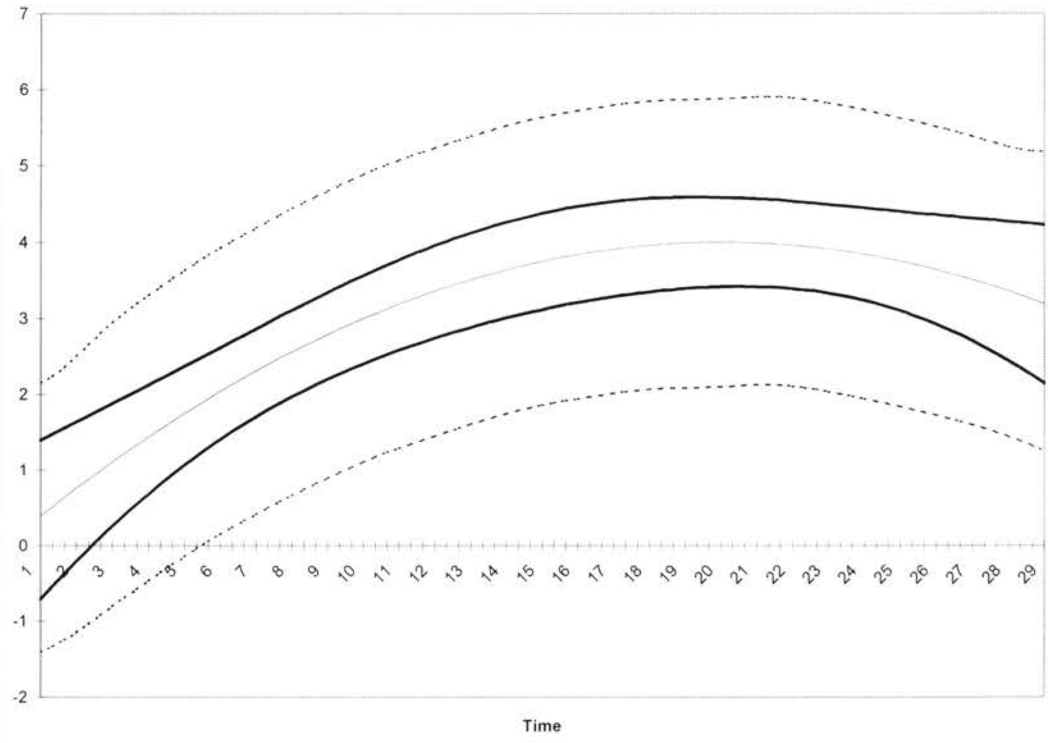


Figure 4.2.11 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 1.0$, case 2

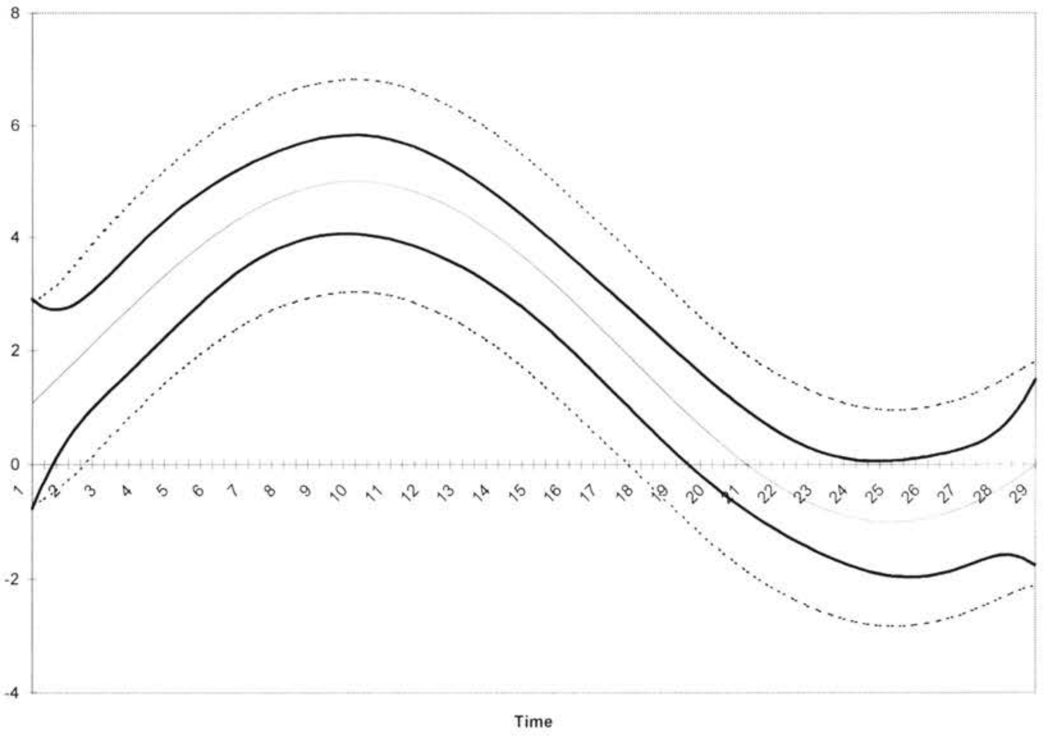


Figure 4.2.12 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 1.0$, case 2

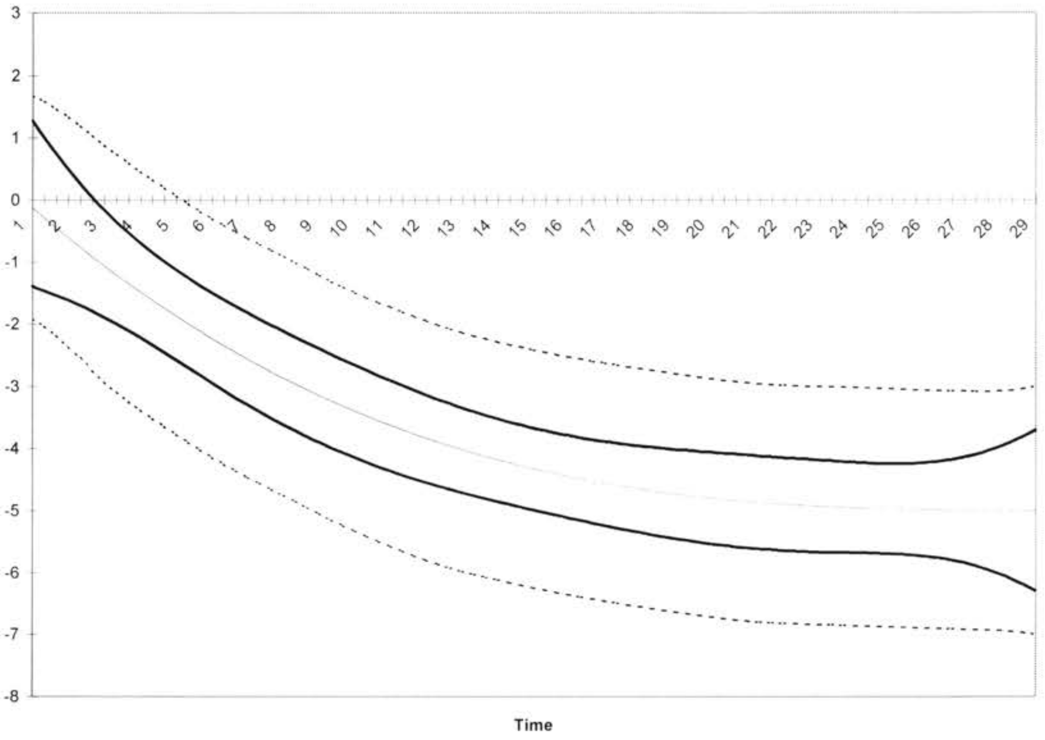


Figure 4.2.13 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 1.2$, case 2

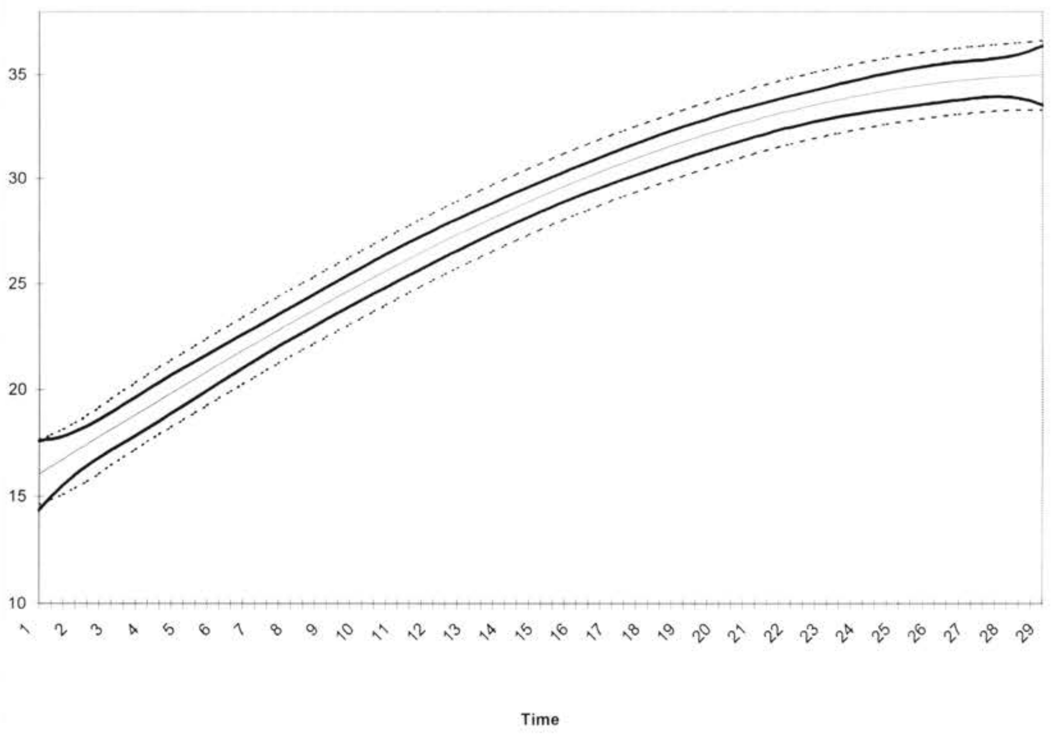


Figure 4.2.14 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 1.2$, case 2

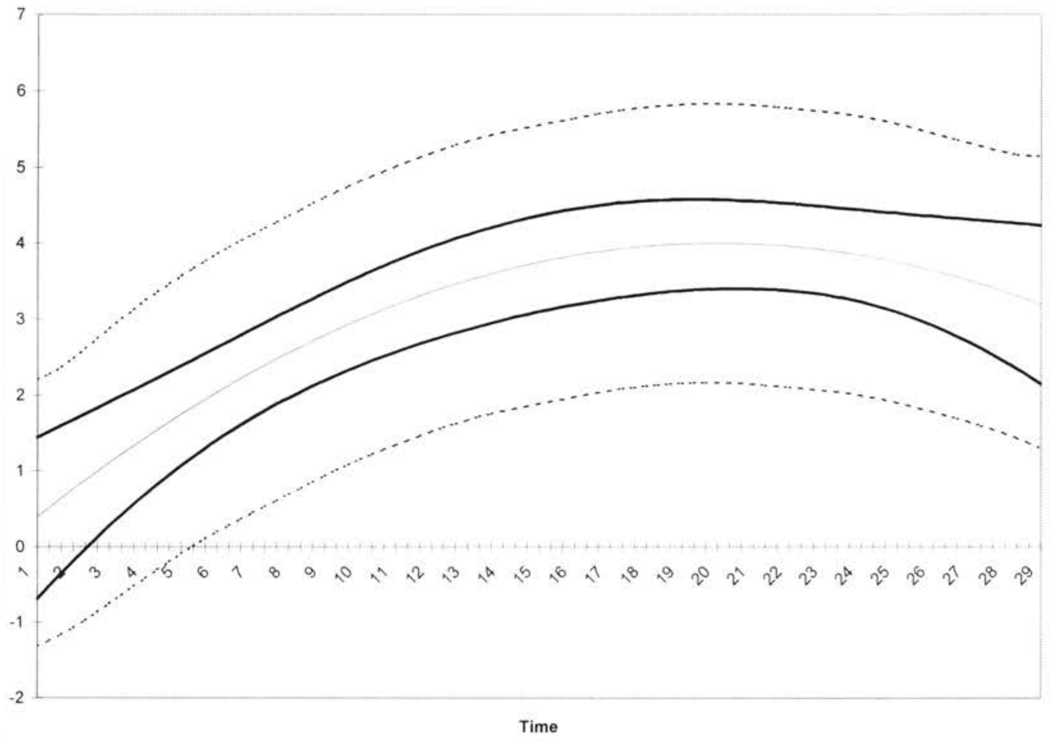


Figure 4.2.15 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 1.2$, case 2

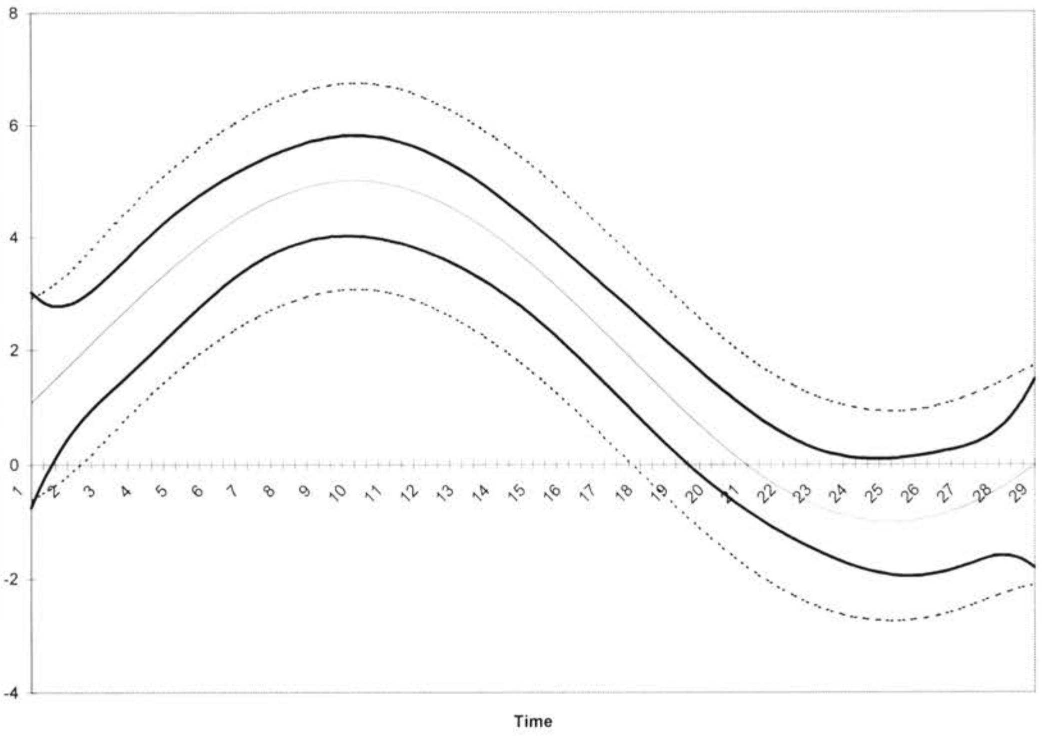


Figure 4.2.16 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 1.2$, case 2

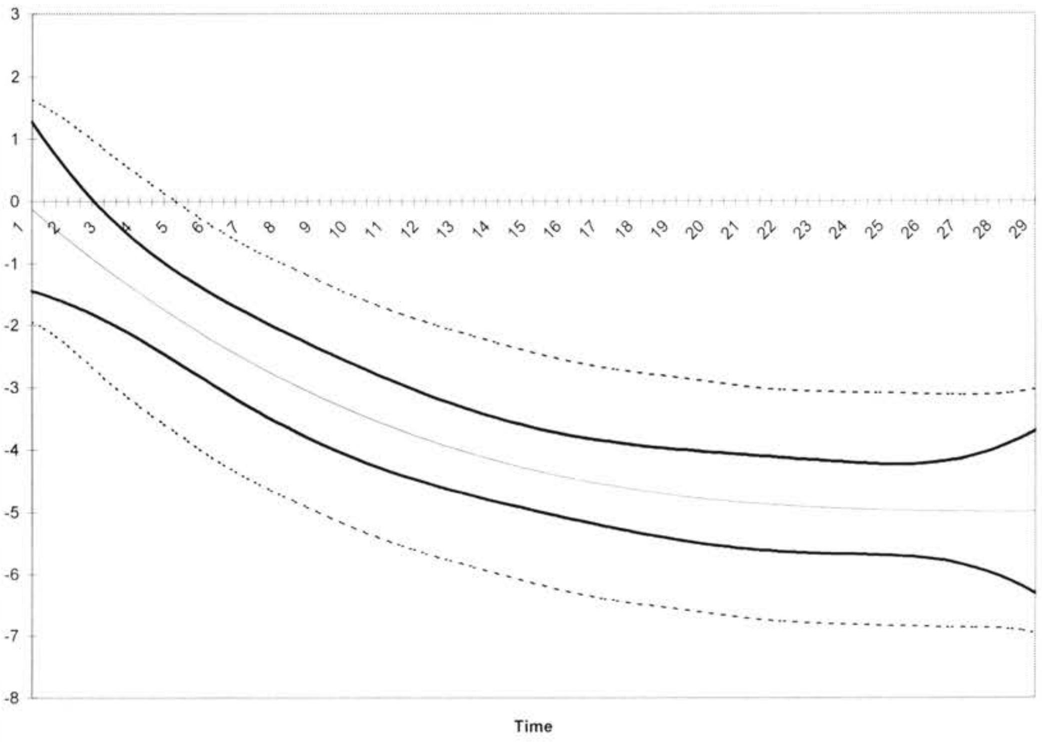


Figure 4.2.17 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 1.4$, case 2

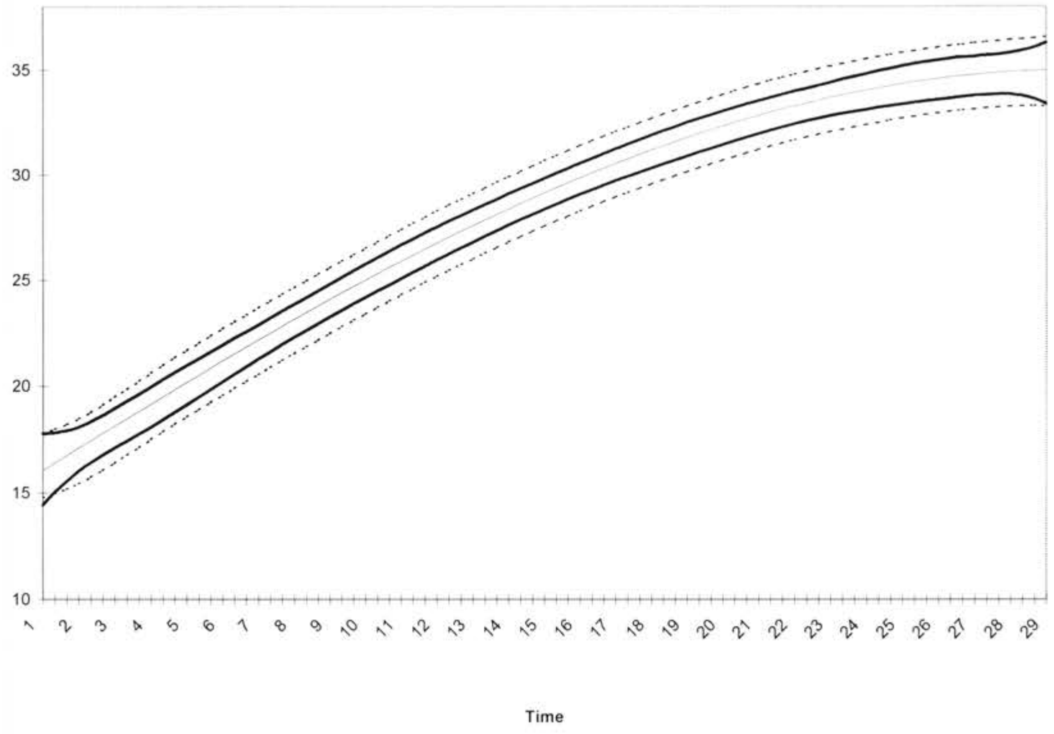


Figure 4.2.18 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 1.4$, case 2

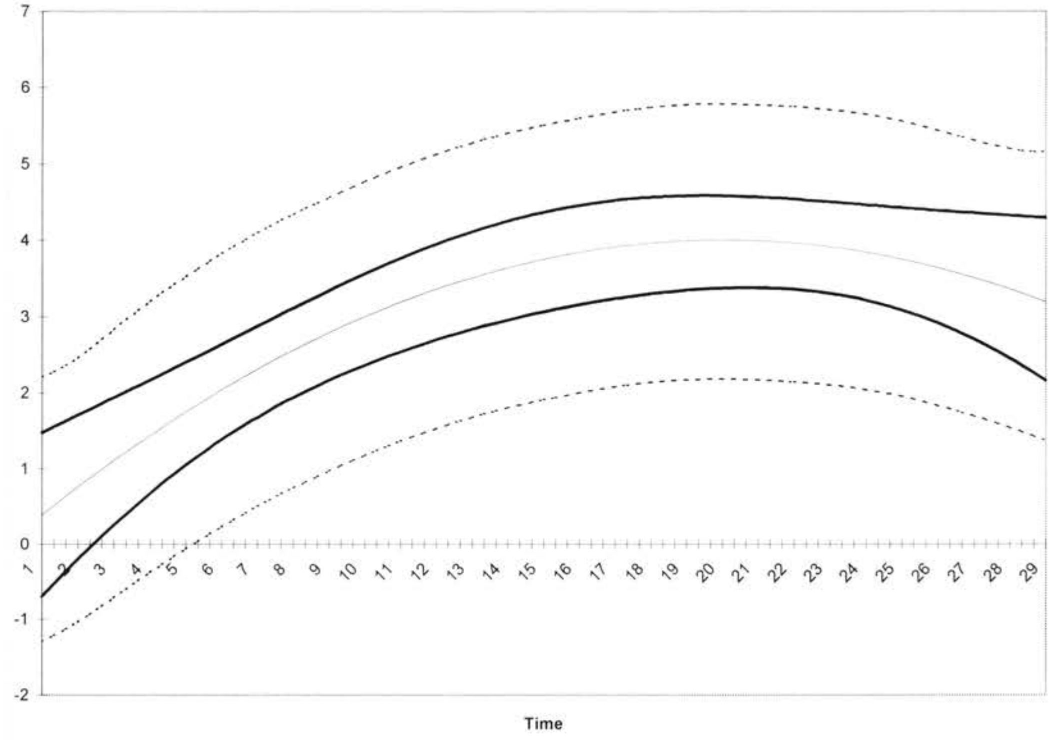


Figure 4.2.19 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 1.4$, case 2

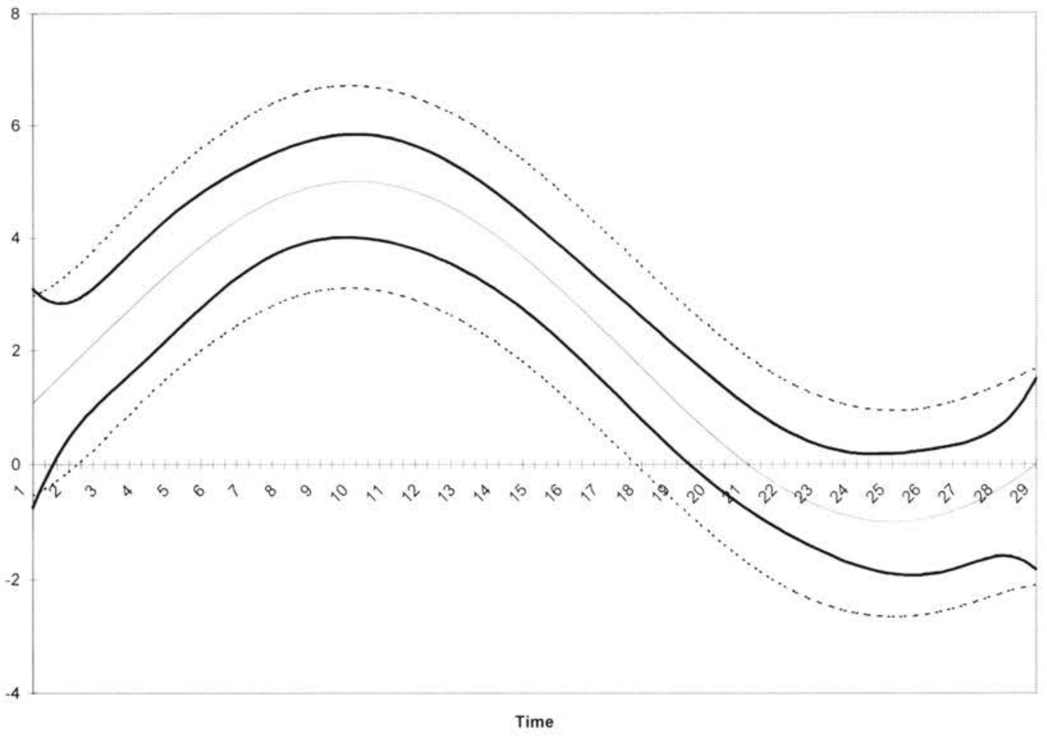
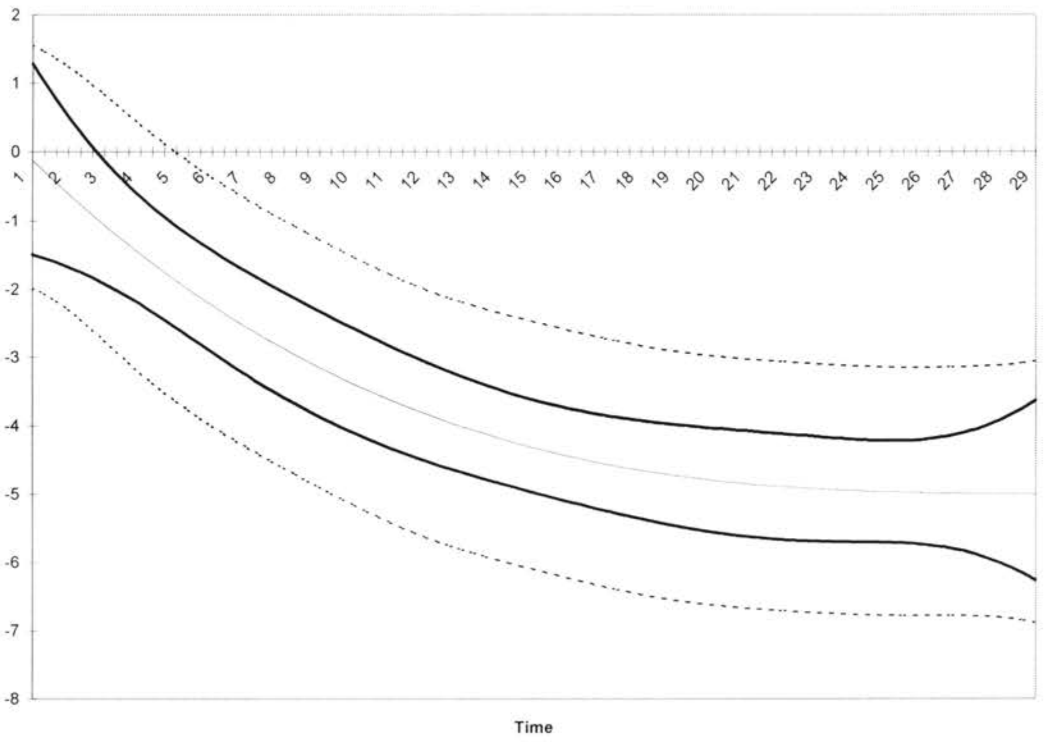


Figure 4.2.20 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 1.4$, case 2



4.3 Simulation case three

This simulation case is also derived from case one; however, the number of subjects in the study was halved, as in case two, so $n = 100$. Also, the distribution of the error term is altered so that

$$\mathrm{Cov}[\epsilon_{i_1}(t_{i_1j_1}), \epsilon_{i_2}(t_{i_2j_2})] = \begin{cases} 4\, e^{-\frac{1}{2}|t_{i_1j_1} - t_{i_2j_2}|} & \text{if } i_1 = i_2 \\ 0 & \text{if } i_1 \neq i_2 \end{cases}.$$

Essentially, this alteration of the error term causes the intra-subject correlation to increase for a given separation of observations, and an increase in intra-subject correlation generally requires additional smoothing. From table 4.3, one can see that the proposed procedure may not be conservative when minimal smoothing is done; however, it is also important to note that the results are still reasonable with respect to the nominal 95% rate. For more substantial amounts of smoothing, the proposed method appears to again be conservative, and again far less so than the Bonferroni method of Wu *et al.* And, as in previous cases, the proposed method maintains its advantage in producing substantially narrower bands. Again, this appears to be especially clear in banding $\beta_1(t)$.

Table 4.3 Estimated coverage probabilities on $t \in [1, 29]$ for case 3

h	Method	β_0	β_1	β_2	β_3
0.6	Proposed	0.946	0.968	0.956	0.946
	Bonferroni	0.998	0.998	0.994	0.998
0.8	Proposed	0.964	0.958	0.968	0.972
	Bonferroni	1.000	1.000	1.000	0.998
1.0	Proposed	0.962	0.954	0.966	0.962
	Bonferroni	0.998	1.000	1.000	1.000
1.2	Proposed	0.964	0.970	0.976	0.962
	Bonferroni	1.000	1.000	0.998	1.000
1.4	Proposed	0.984	0.964	0.964	0.974
	Bonferroni	1.000	1.000	1.000	1.000

Figure 4.3.1 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 0.6$, case 3

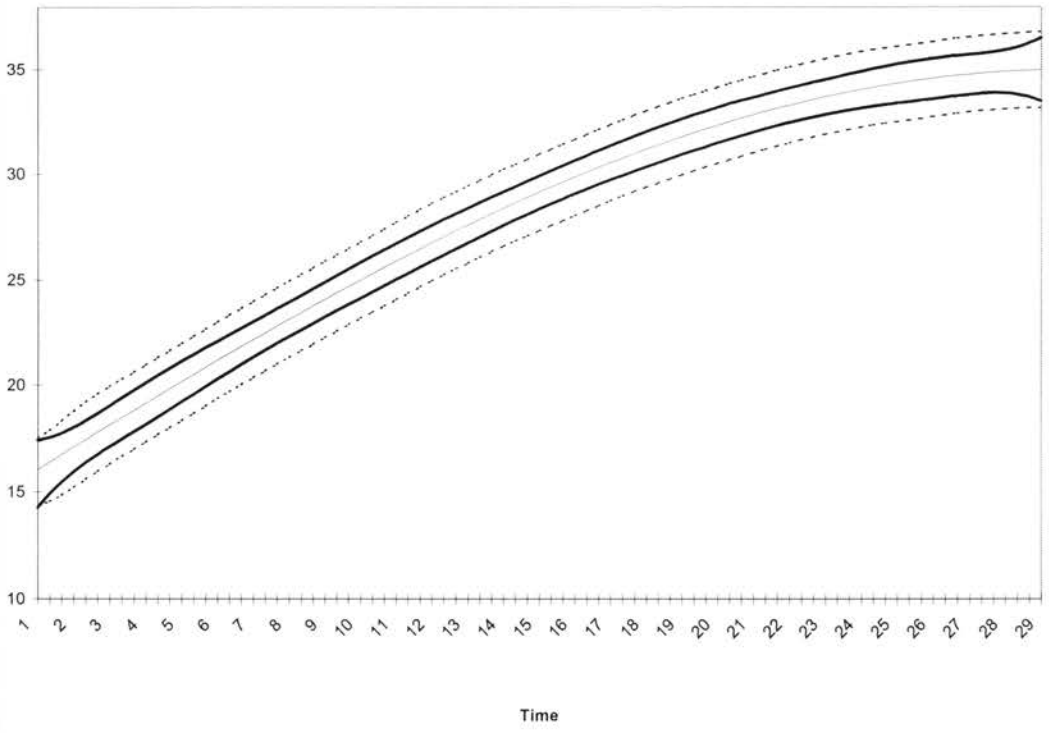


Figure 4.3.2 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 0.6$, case 3

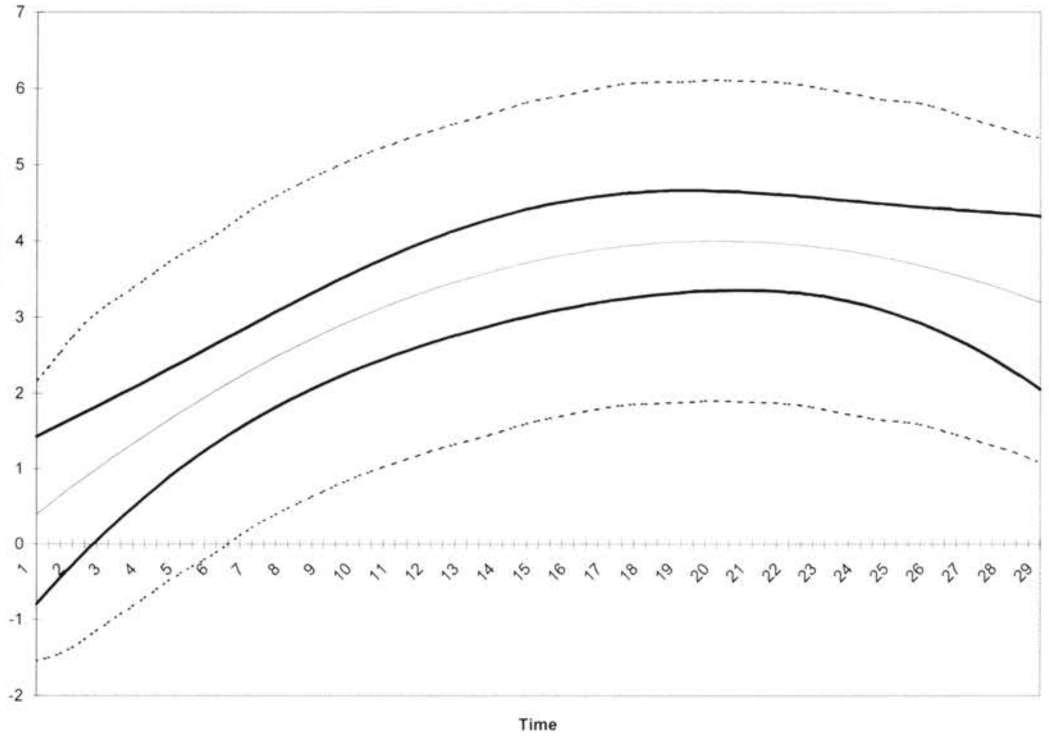


Figure 4.3.3 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 0.6$, case 3

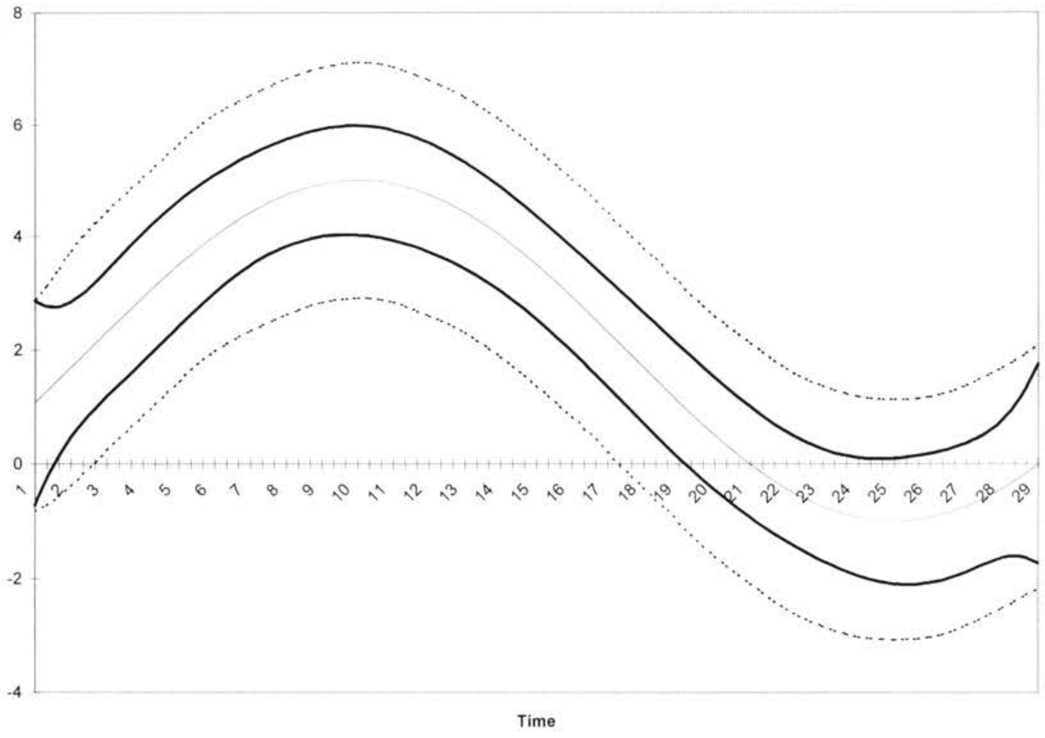


Figure 4.3.4 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 0.6$, case 3

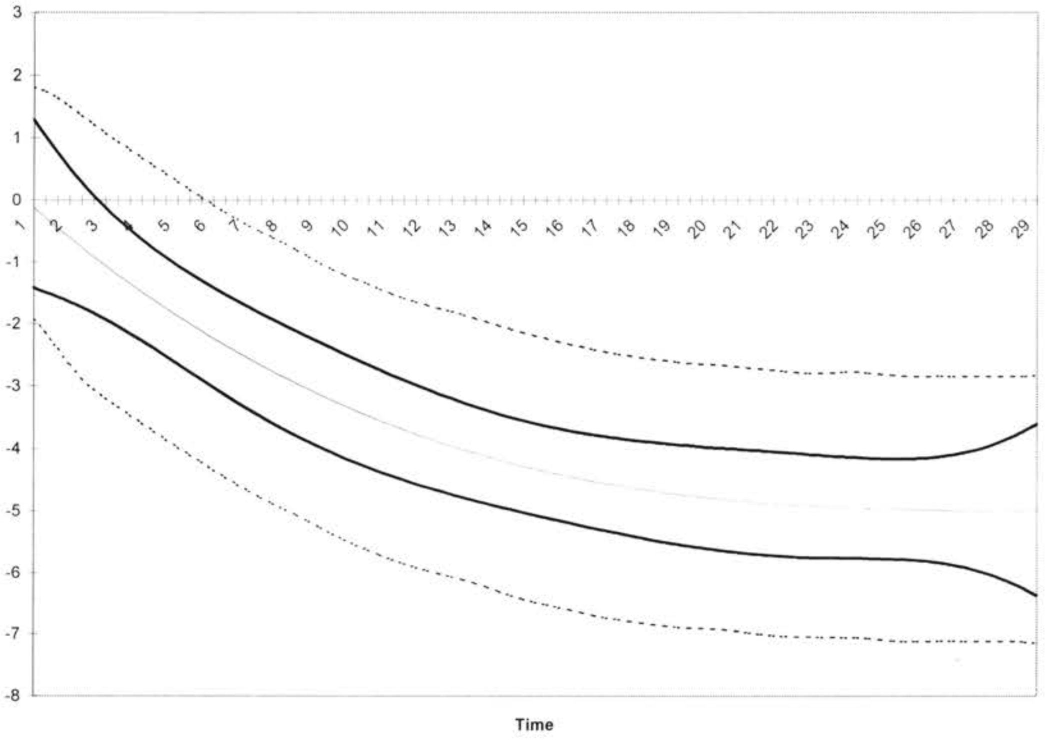


Figure 4.3.5 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 0.8$, case 3

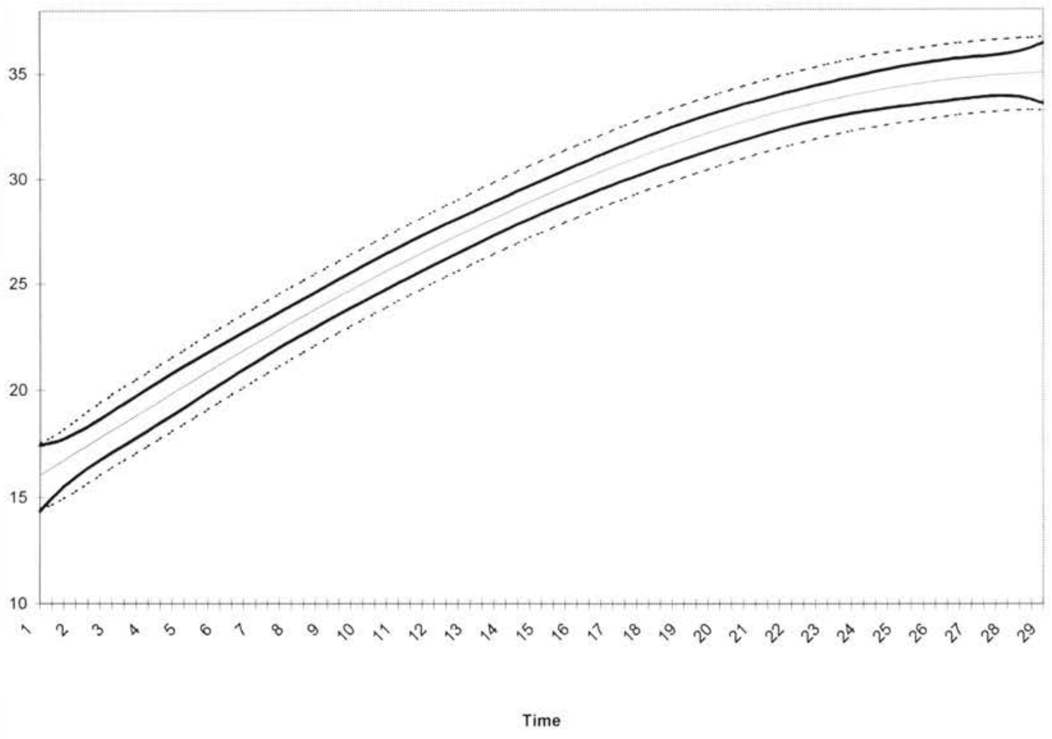


Figure 4.3.6 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 0.8$, case 3

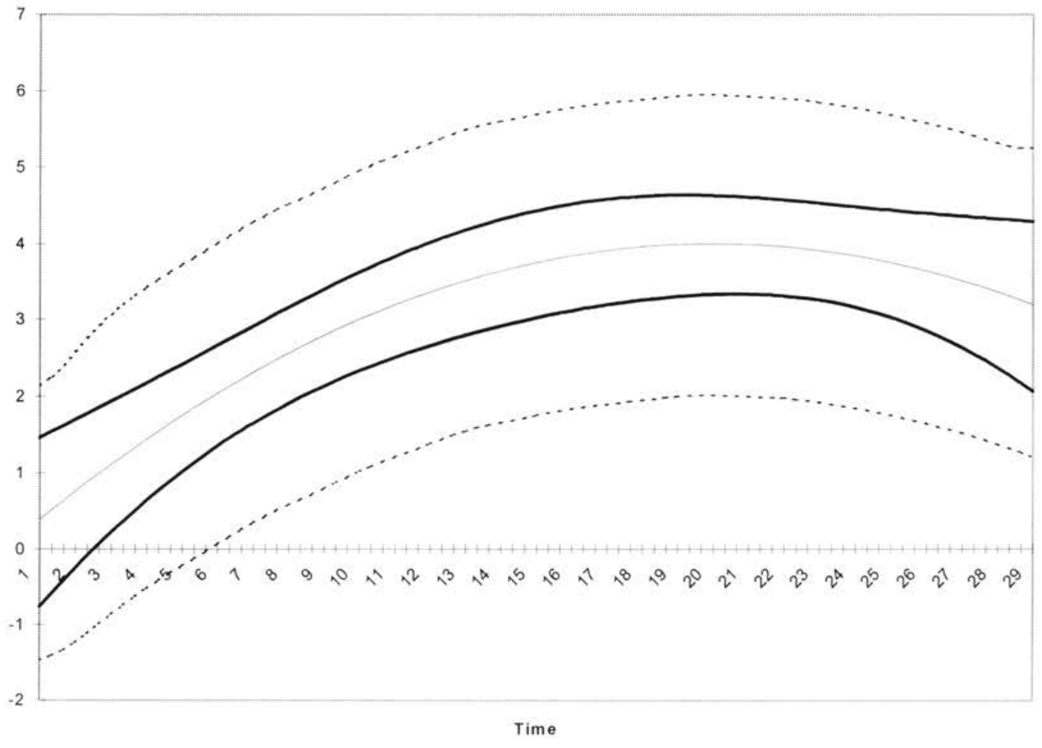


Figure 4.3.7 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 0.8$, case 3

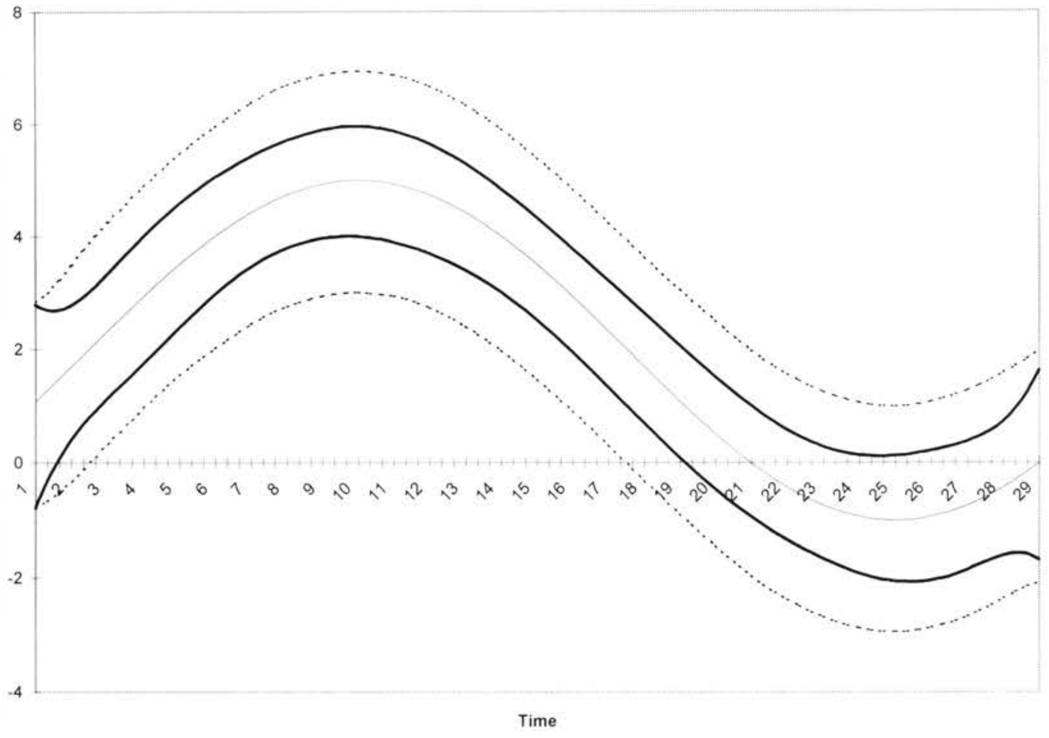


Figure 4.3.8 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 0.8$, case 3

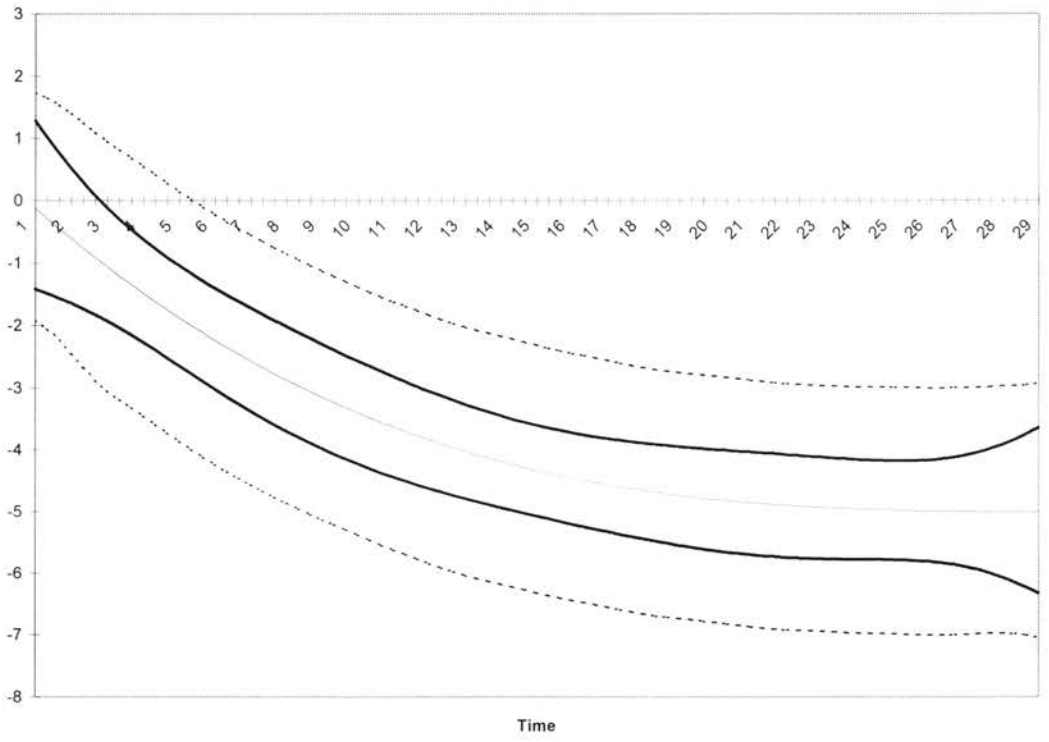


Figure 4.3.9 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 1.0$, case 3

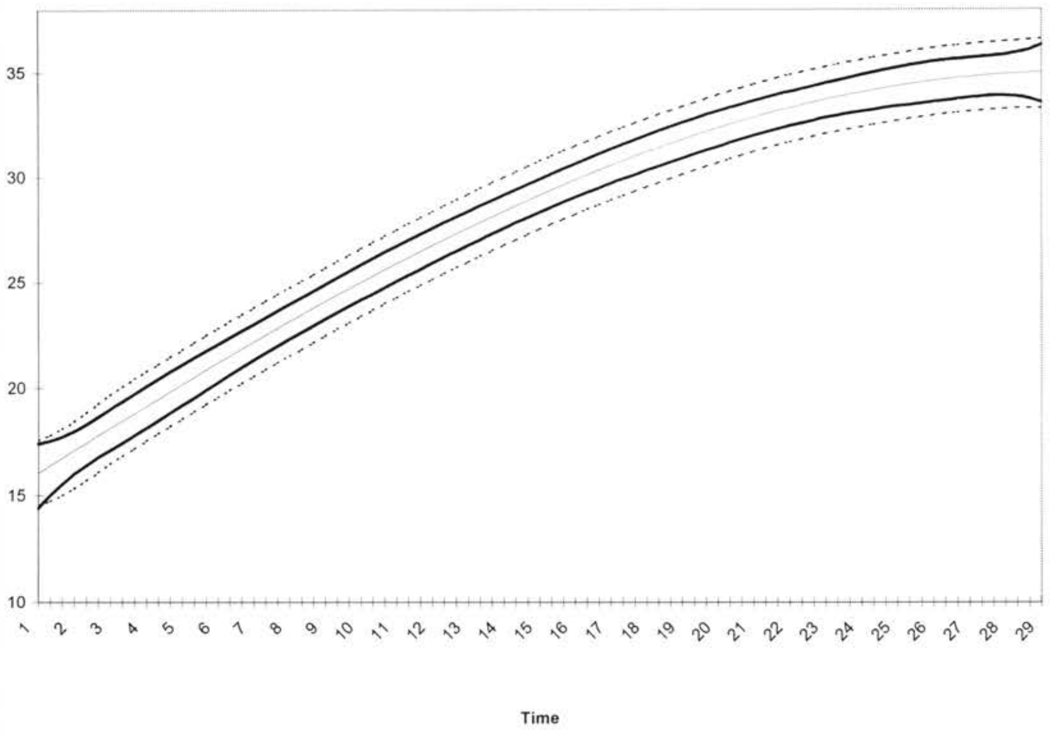


Figure 4.3.10 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 1.0$, case 3

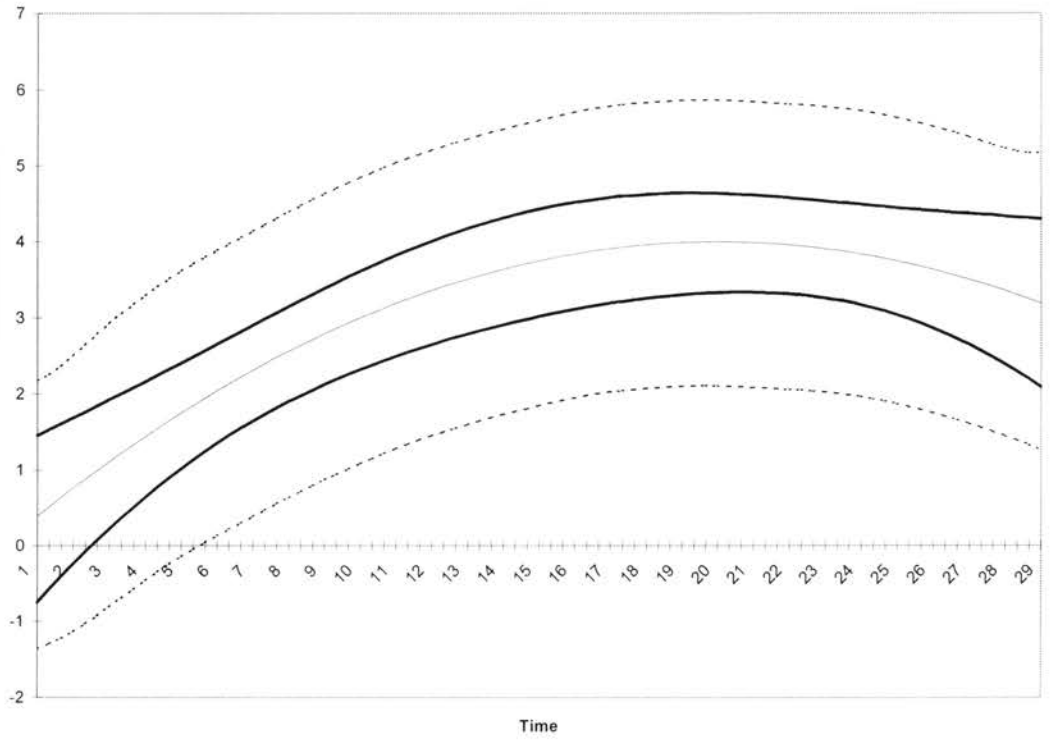


Figure 4.3.11 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 1.0$, case 3

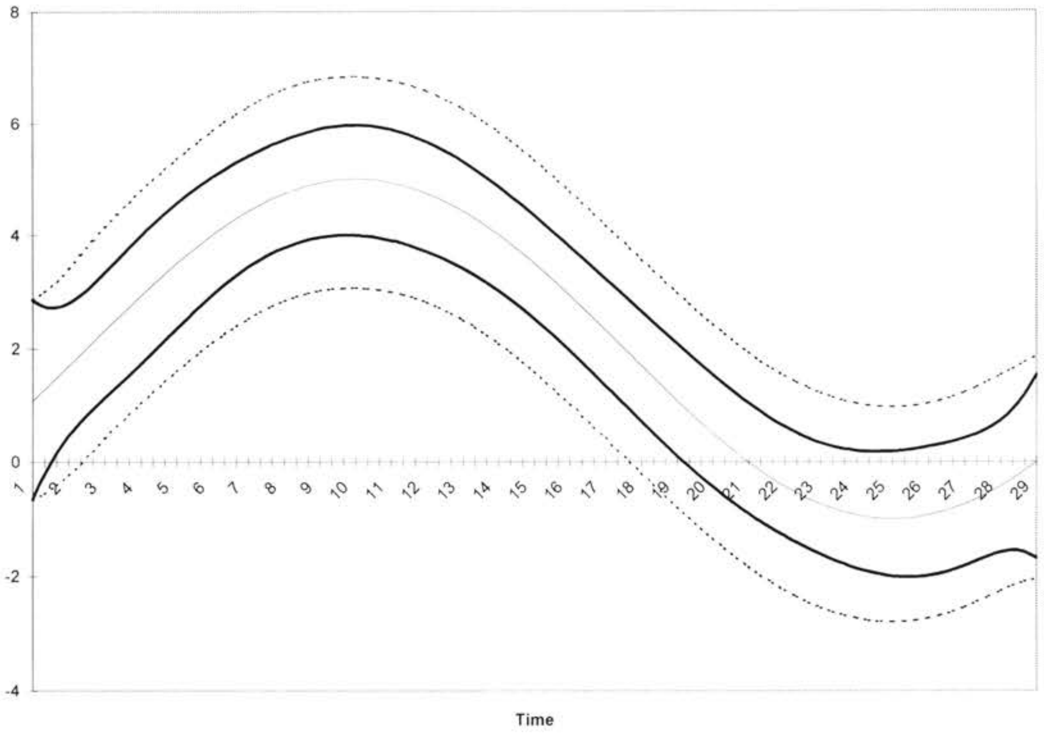


Figure 4.3.12 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 1.0$, case 3

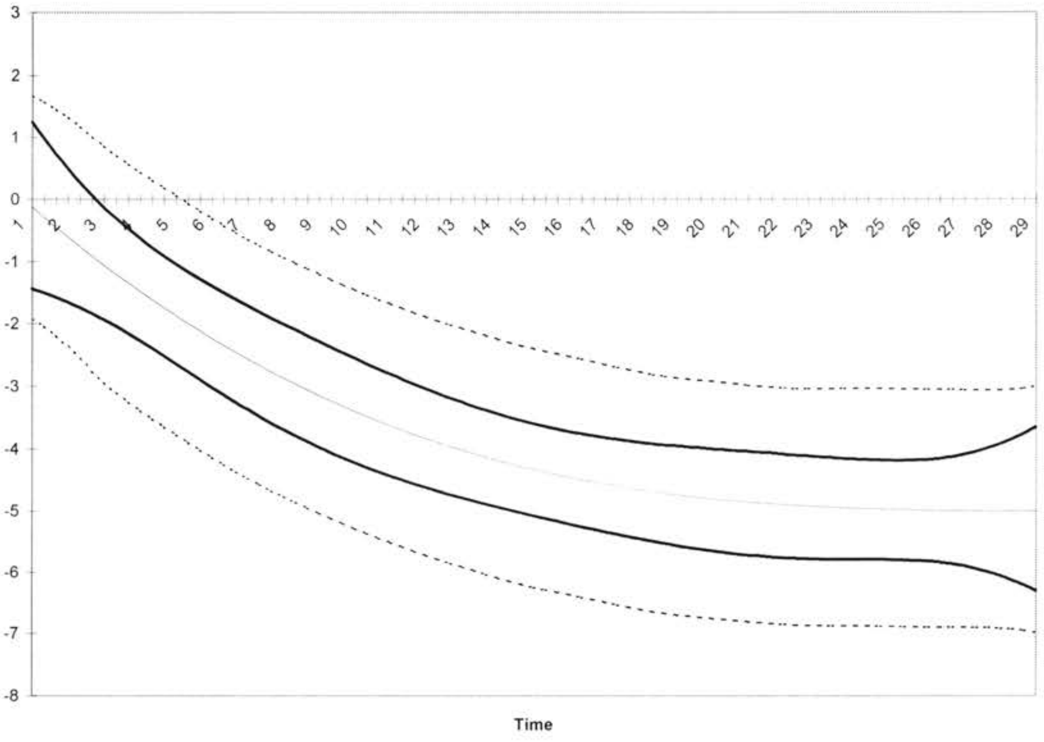


Figure 4.3.13 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 1.2$, case 3

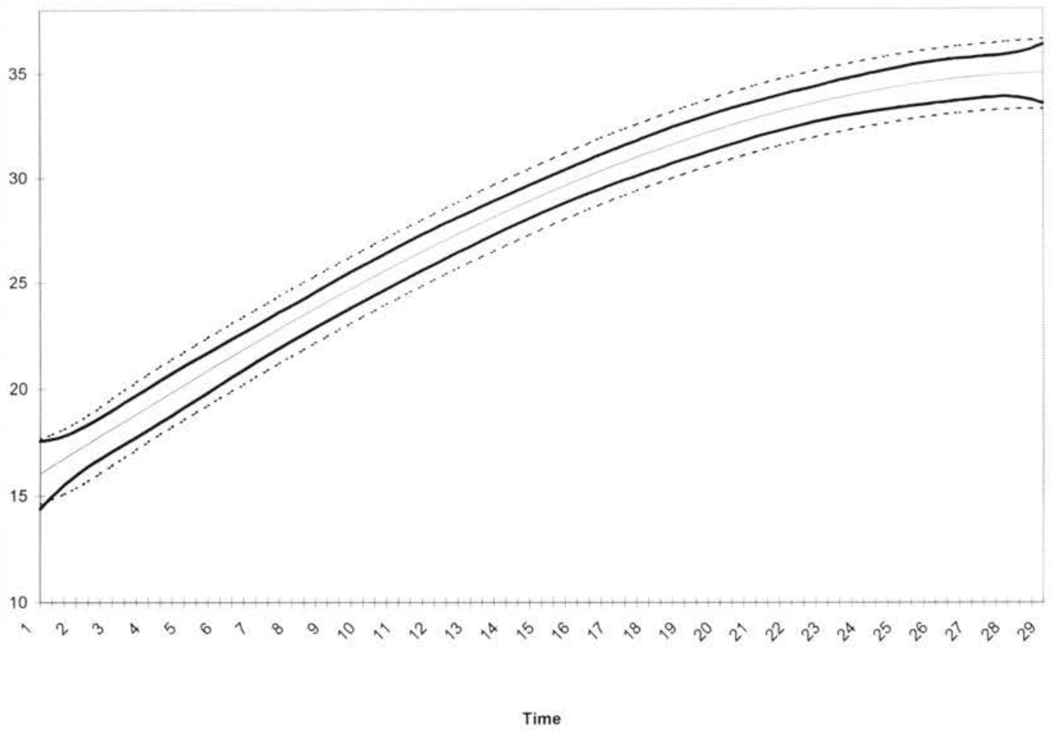


Figure 4.3.14 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 1.2$, case 3

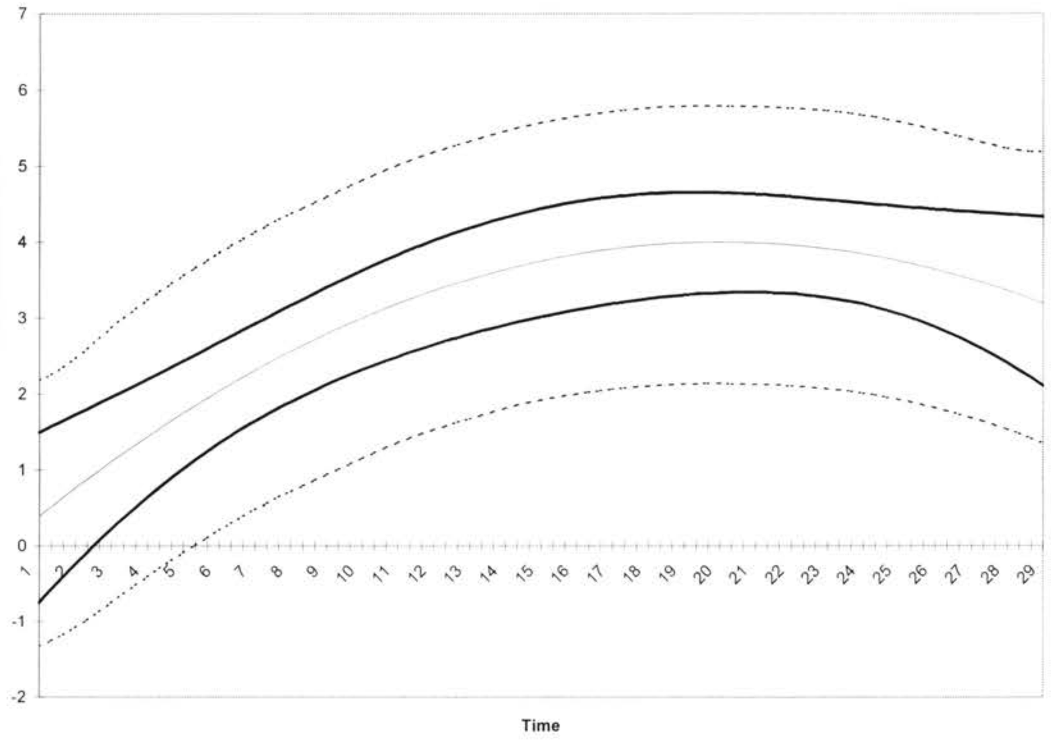


Figure 4.3.15 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 1.2$, case 3

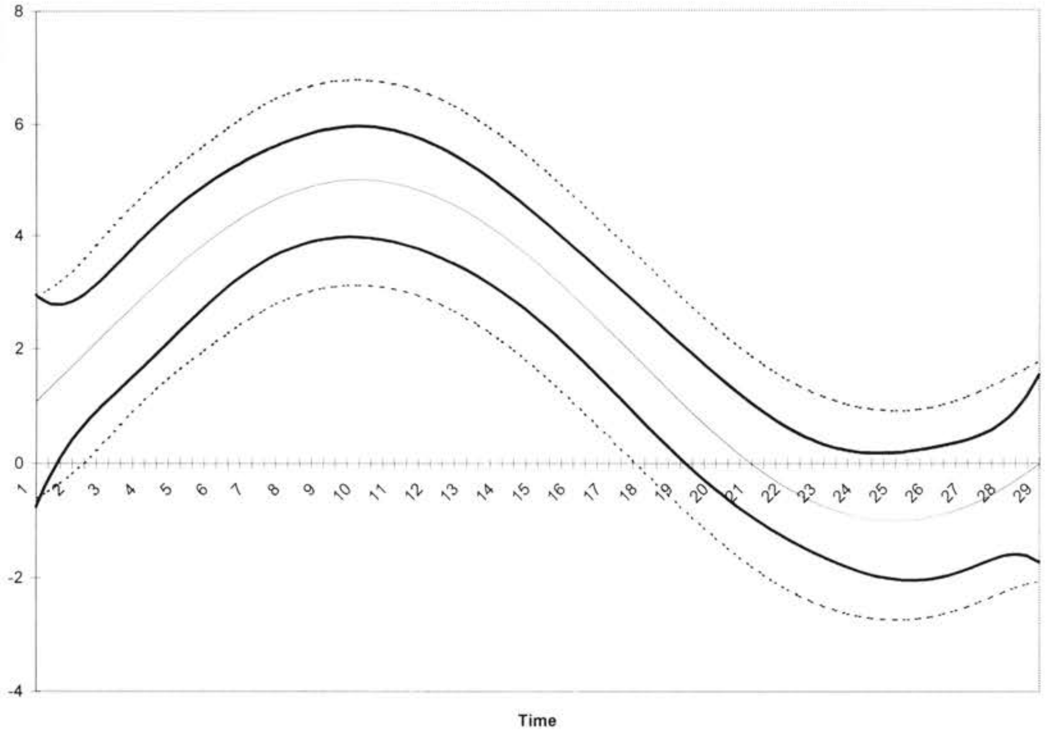


Figure 4.3.16 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 1.2$, case 3

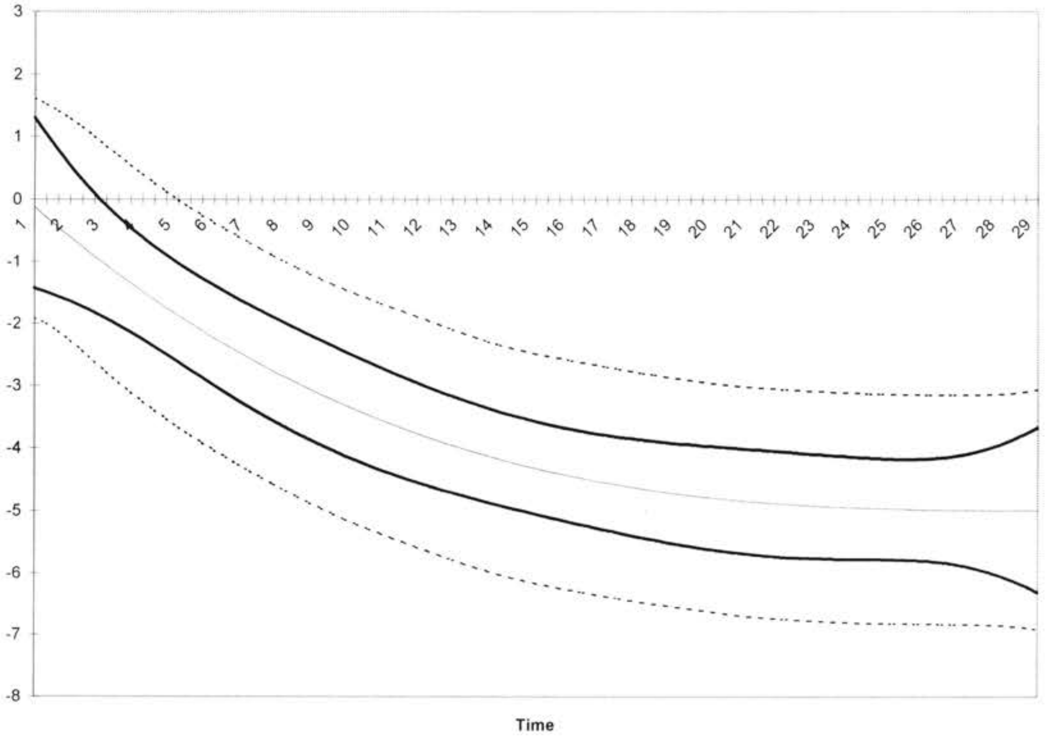


Figure 4.3.17 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 1.4$, case 3

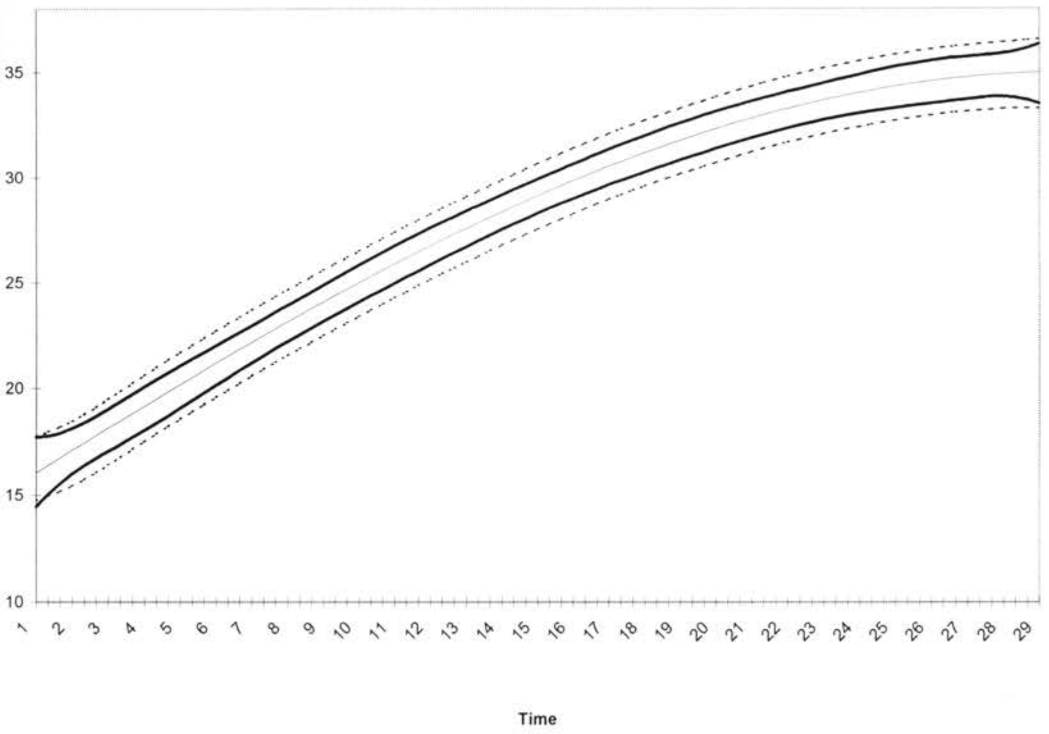


Figure 4.3.18 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 1.4$, case 3

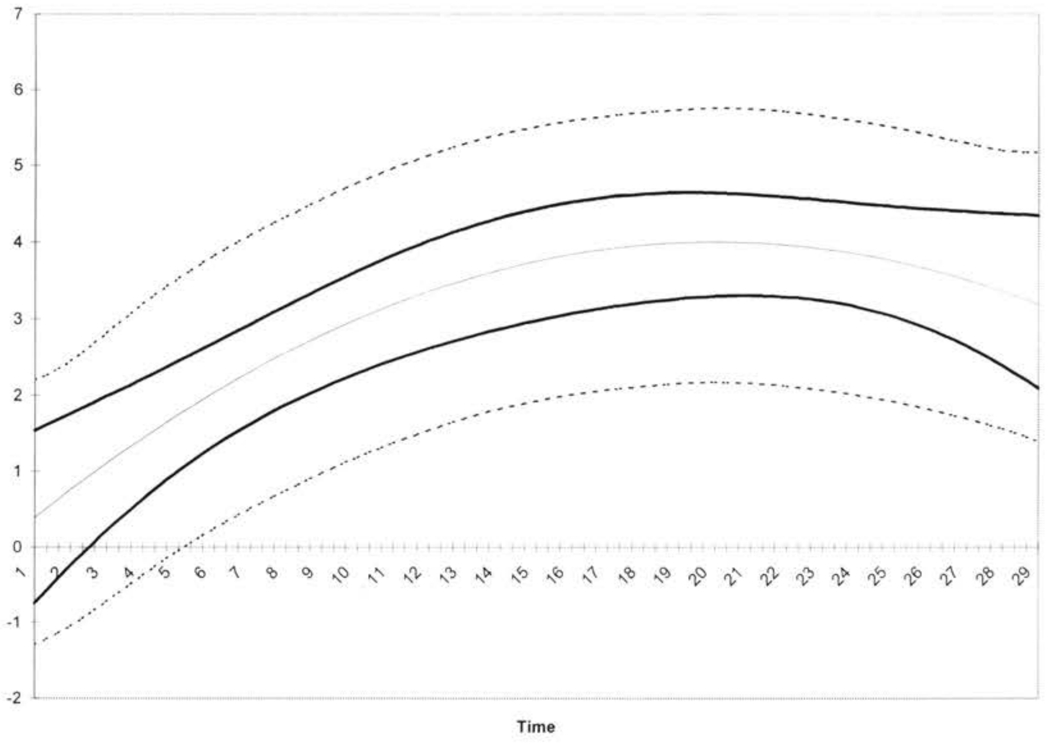


Figure 4.3.19 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 1.4$, case 3

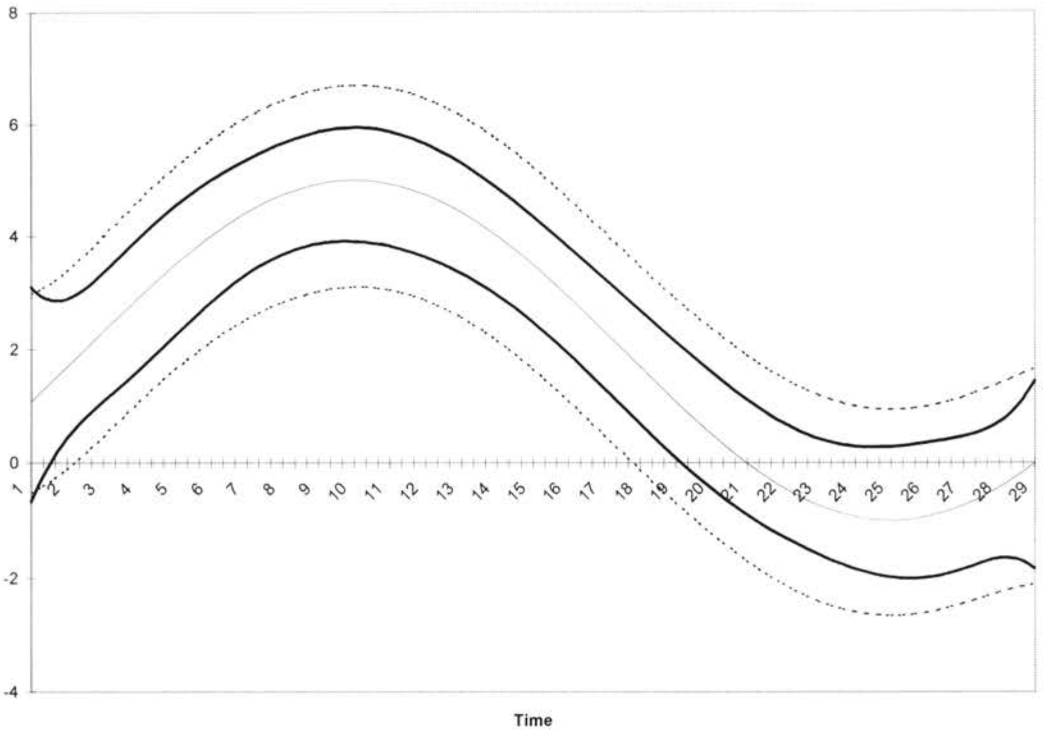
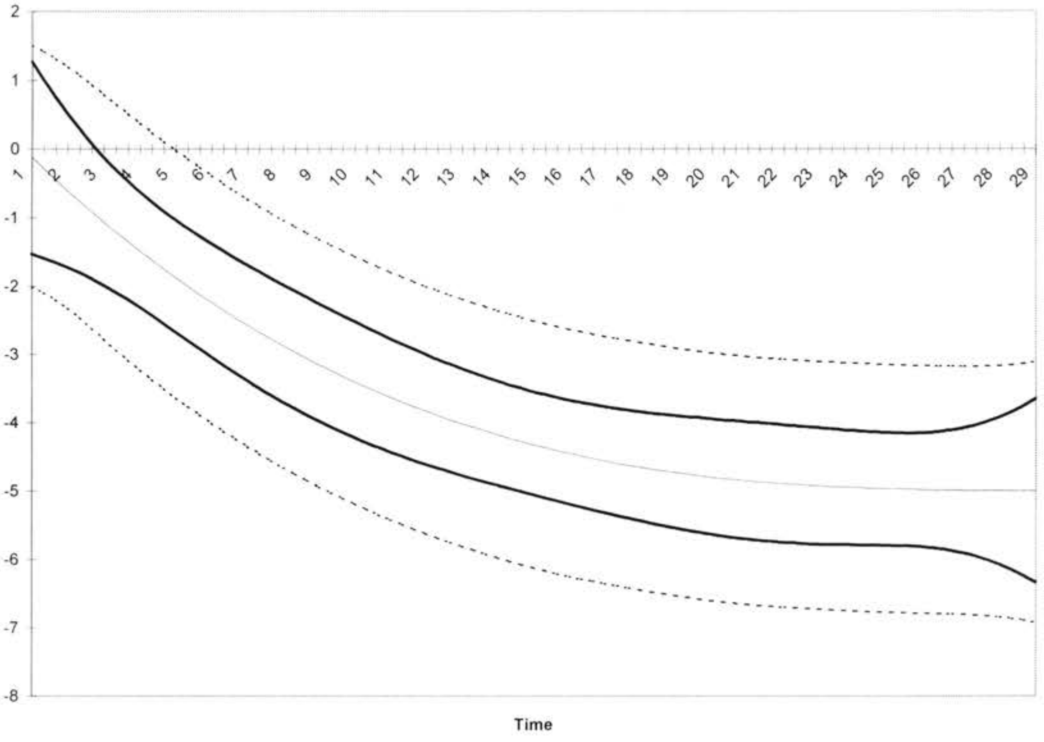


Figure 4.3.20 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 1.4$, case 3



4.4 Simulation case four

In this simulation, an effort was made to study the effects of choosing polynomial fits of a greater degree than necessary. The scenario is precisely the same as that of case two except that the degrees chosen for the polynomial fits for $\beta_0(t)$, $\beta_1(t)$, $\beta_2(t)$, $\beta_3(t)$ are 7, 4, 7 and 5, respectively. Comparing these results with those of section 4.2, one can see that the proposed method appears to be slightly more conservative in this case and produces slightly wider bands on average under these conditions. Essentially, fitting the higher degree polynomials "wastes" degrees of freedom, *i.e.* extra parameters are fit without benefit. Yet, the proposed procedure still maintains its overall superiority in comparison to the Bonferroni method.

Table 4.4 Estimated coverage probabilities on $t \in [1, 29]$ for case 4

h	Method	β_0	β_1	β_2	β_3
0.6	Proposed	0.978	0.982	0.976	0.970
	Bonferroni	1.000	1.000	0.998	0.998
1.0	Proposed	0.982	0.966	0.980	0.978
	Bonferroni	1.000	1.000	1.000	1.000
1.4	Proposed	0.996	0.986	0.988	0.982
	Bonferroni	1.000	1.000	1.000	1.000

Figure 4.4.1 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 0.6$, case 4

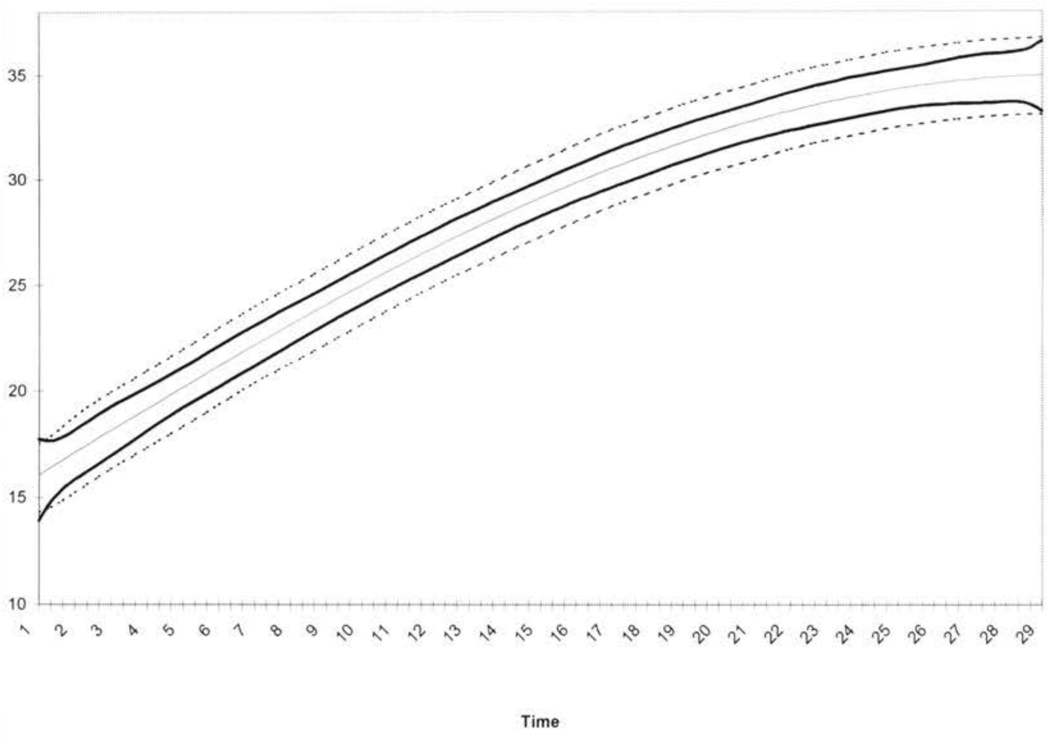


Figure 4.4.2 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 0.6$, case 4

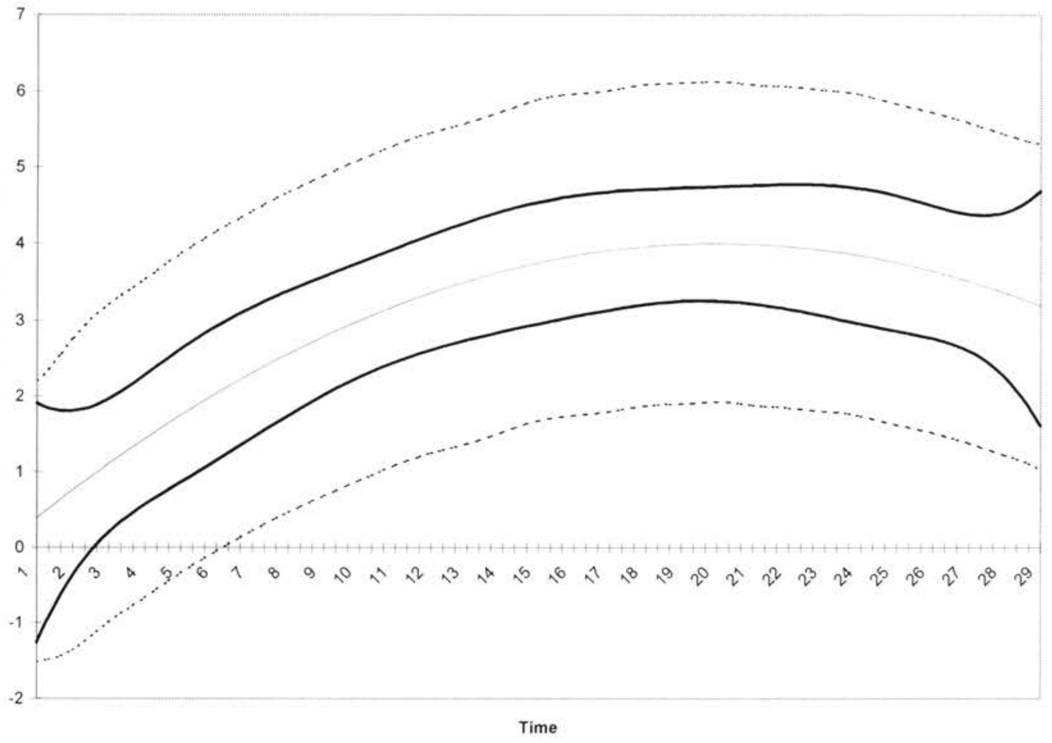


Figure 4.4.3 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 0.6$, case 4

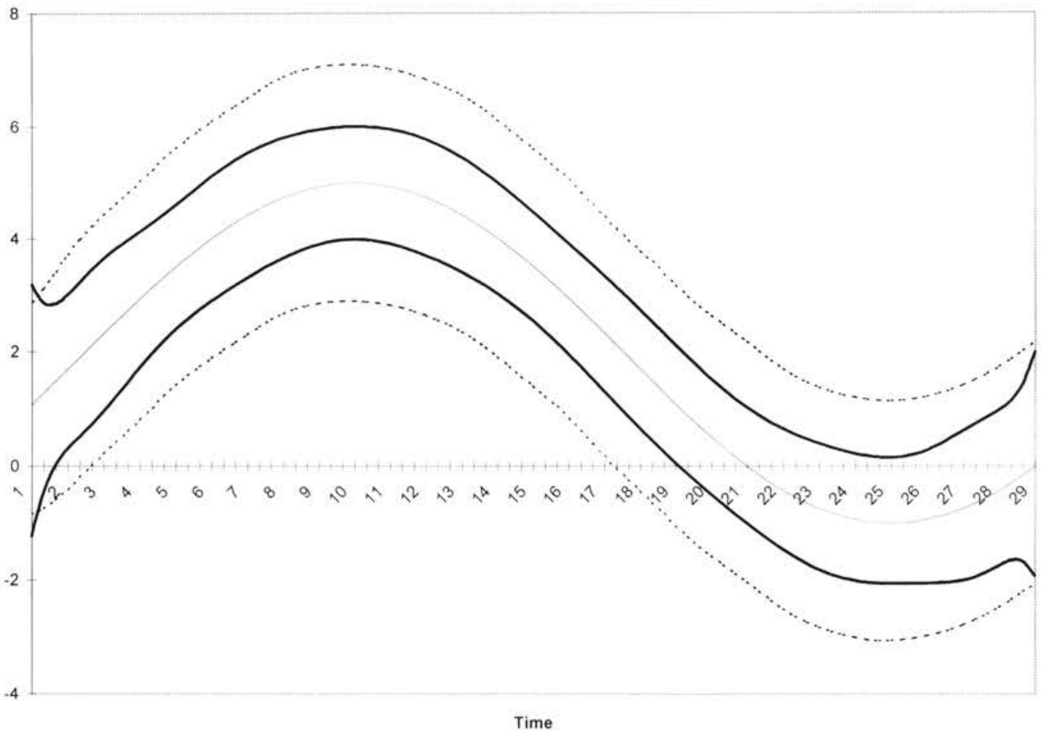


Figure 4.4.4 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 0.6$, case 4

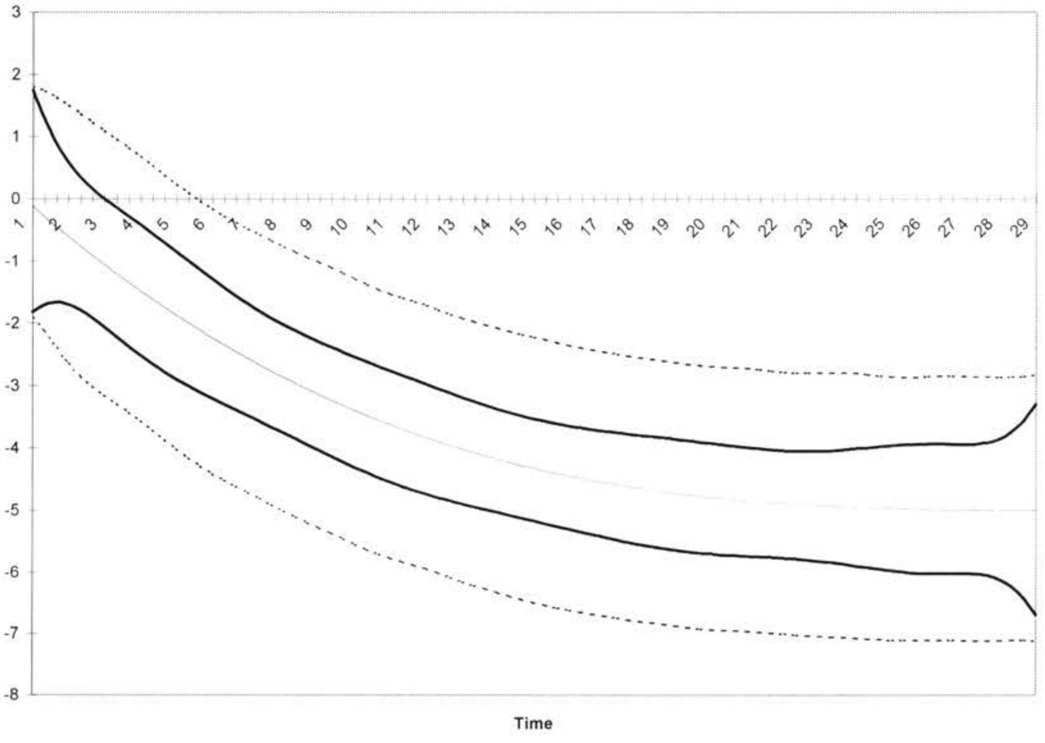


Figure 4.4.5 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 1.0$, case 4

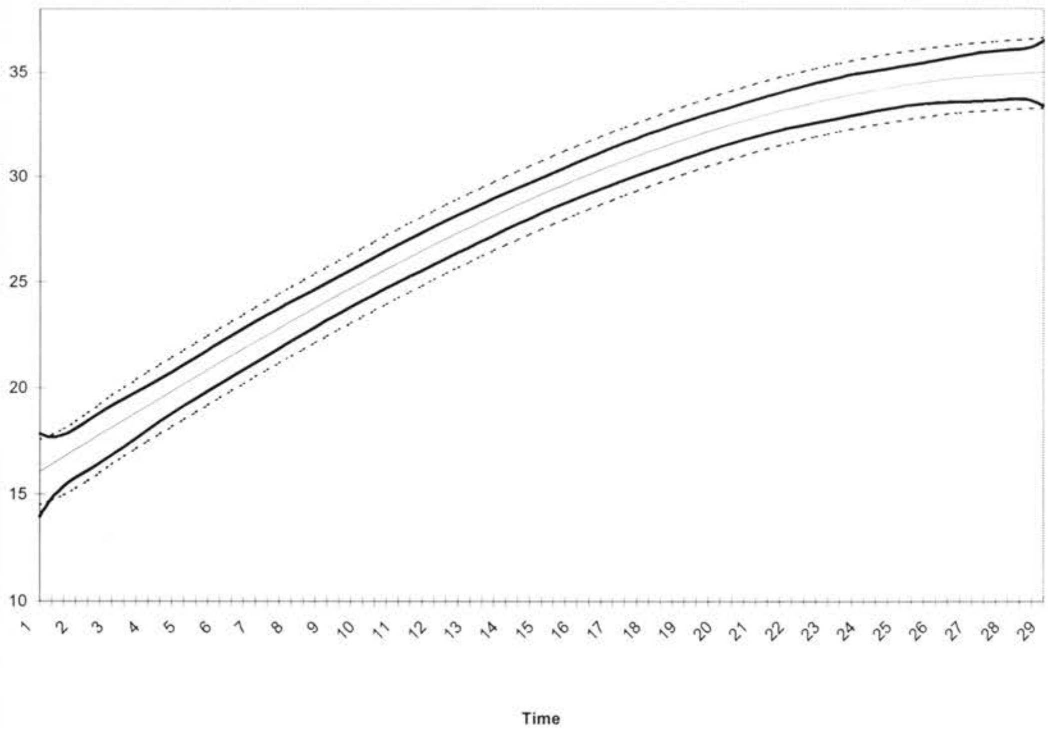


Figure 4.4.6 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 1.0$, case 4

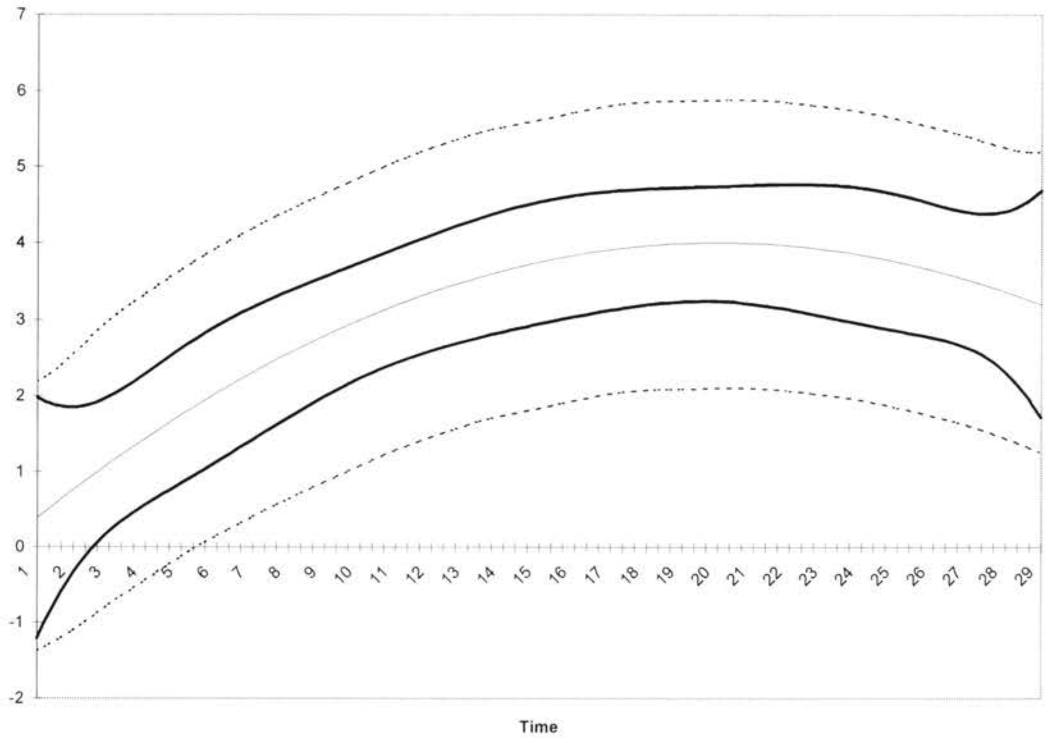


Figure 4.4.7 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 1.0$, case 4

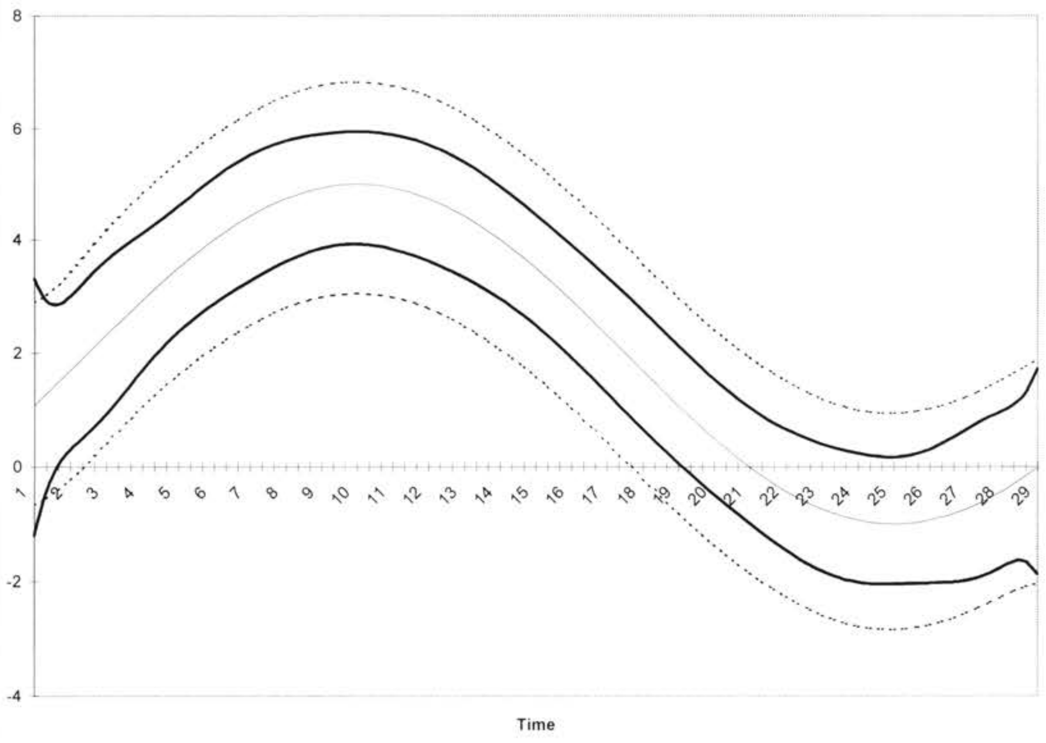


Figure 4.4.8 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 1.0$, case 4

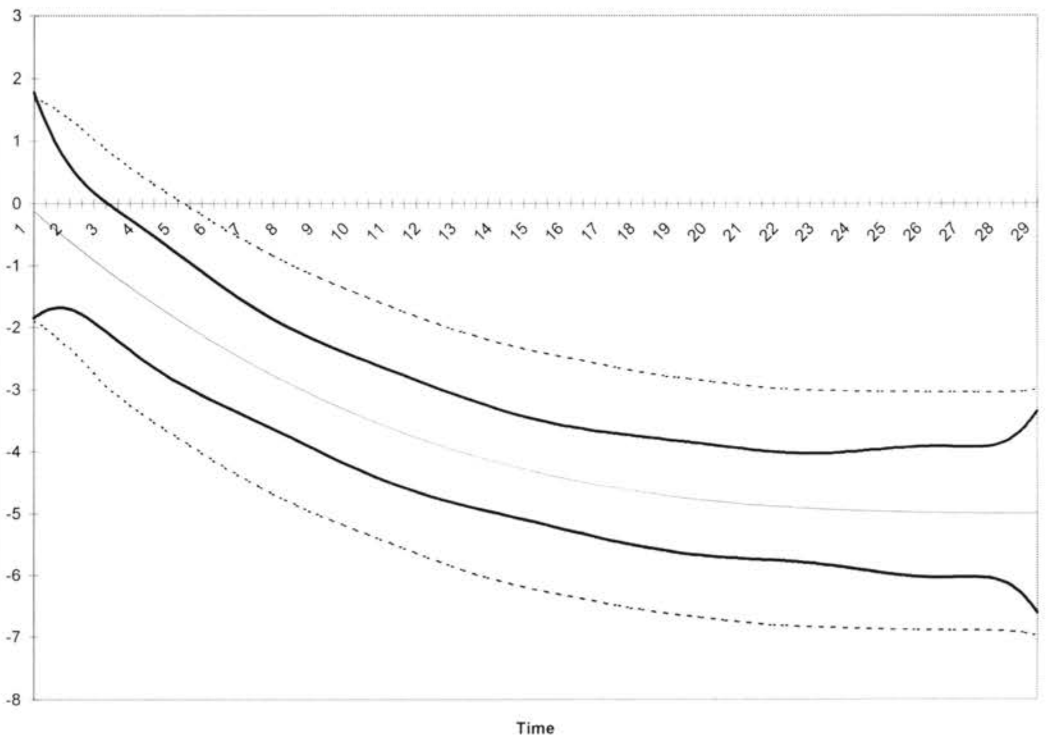


Figure 4.4.9 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 1.4$, case 4

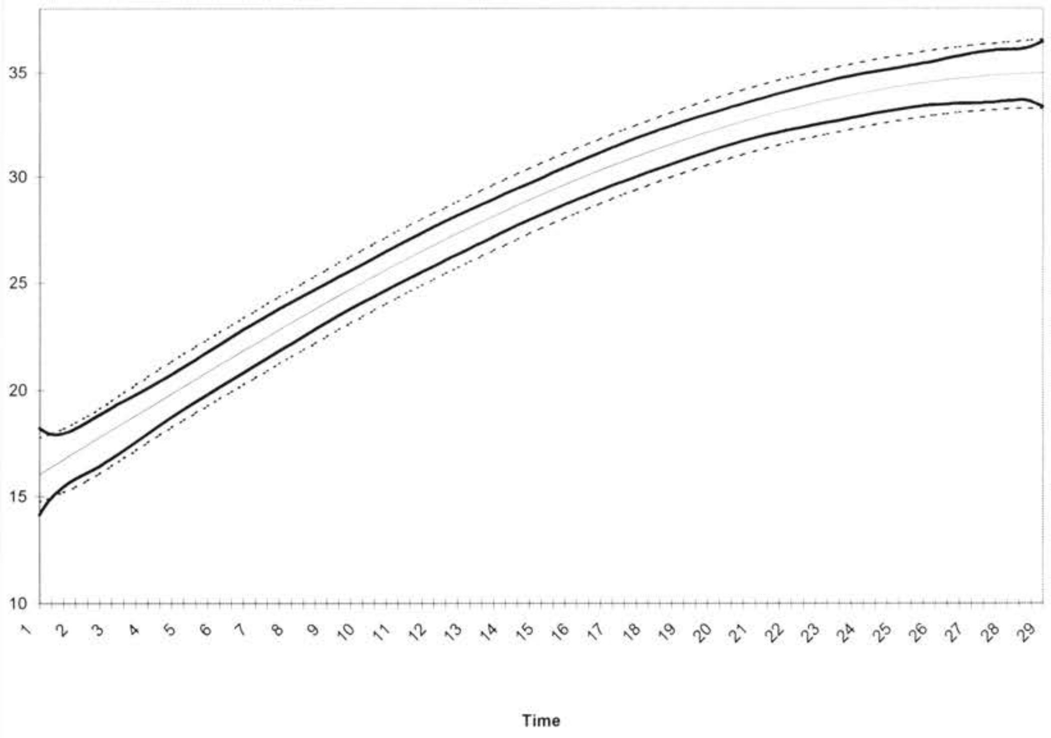


Figure 4.4.10 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 1.4$, case 4

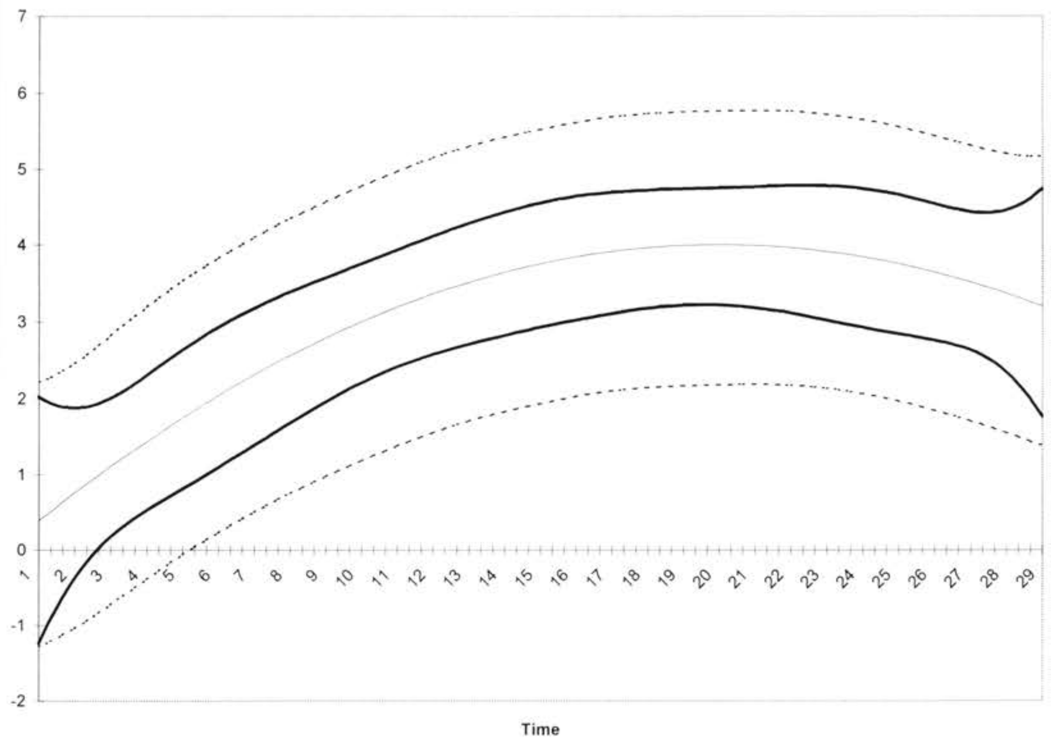


Figure 4.4.11 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 1.4$, case 4

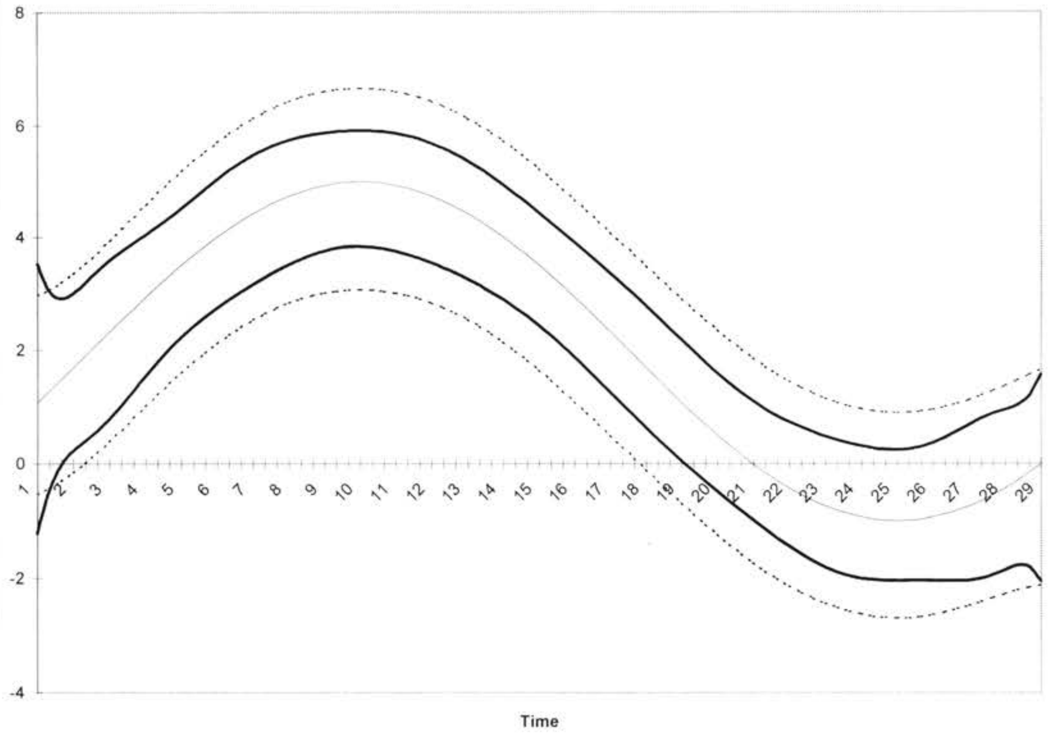
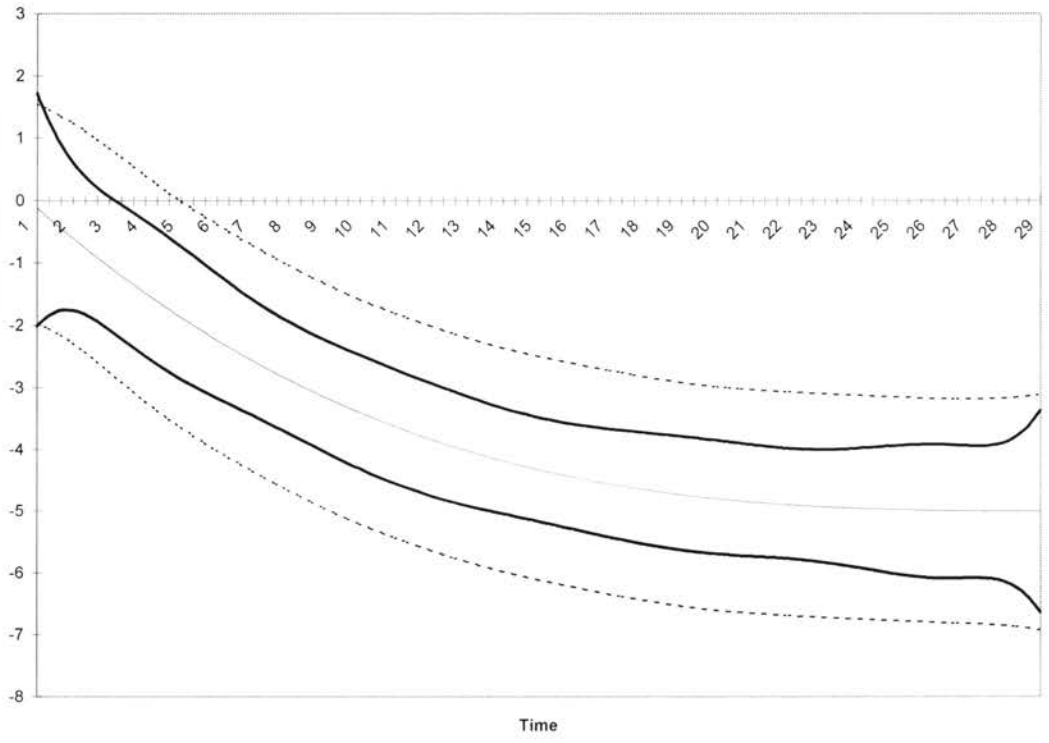


Figure 4.4.12 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 1.4$, case 4



4.5 Simulation case five

This simulation is also an effort to study the effects of choosing polynomial fits of a greater degree than necessary. The scenario is precisely the same as that of case three except that the degrees chosen for the polynomial fits for $\beta_0(t)$, $\beta_1(t)$, $\beta_2(t)$, $\beta_3(t)$ are 7, 4, 7 and 5, respectively. Comparing these results with those of section 4.3, one can see that the proposed method appears to be slightly more conservative in this case and produces slightly wider bands on average under these conditions. Again, fitting the higher degree polynomials "wastes" degrees of freedom, *i.e.* extra parameters are fit without benefit. However, the result is far from disasterous, the proposed method still outperforms the Bonferroni technique of Wu, *et al.* (1998).

Table 4.5 Estimated coverage probabilities on $t \in [1, 29]$ for case 5

h	Method	β_0	β_1	β_2	β_3
0.6	Proposed	0.966	0.954	0.970	0.974
	Bonferroni	0.998	1.000	1.000	1.000
1.0	Proposed	0.980	0.964	0.976	0.970
	Bonferroni	1.000	1.000	1.000	1.000
1.4	Proposed	0.988	0.976	0.988	0.970
	Bonferroni	1.000	1.000	1.000	1.000

Figure 4.5.1 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 0.6$, case 5

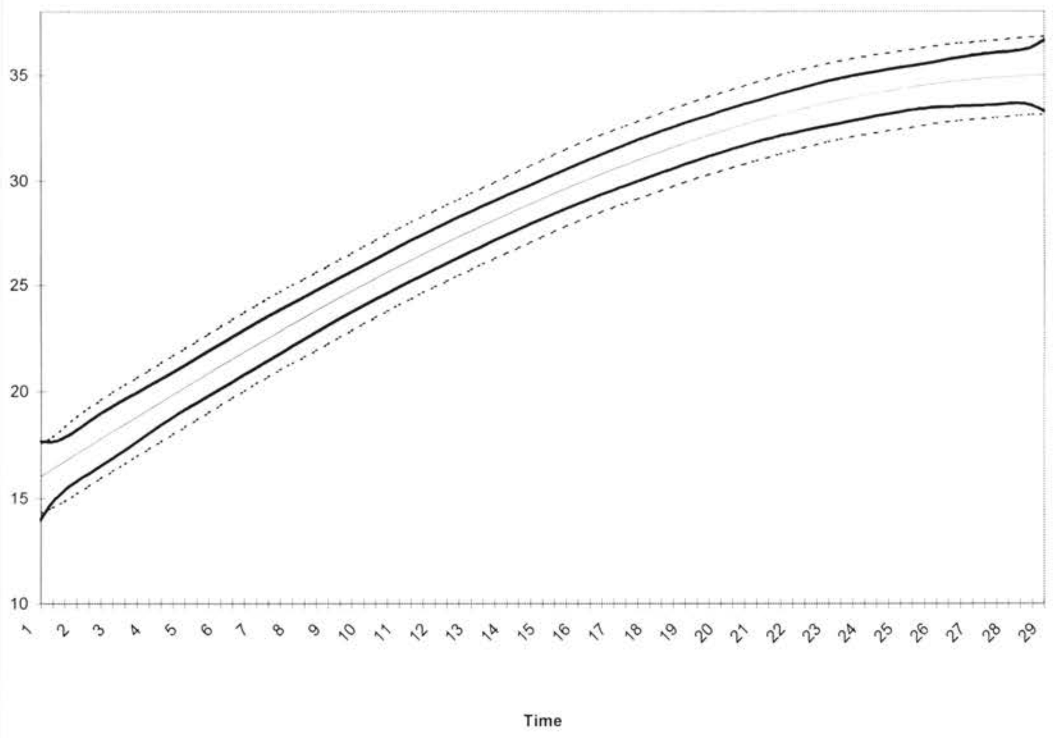


Figure 4.5.2 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 0.6$, case 5

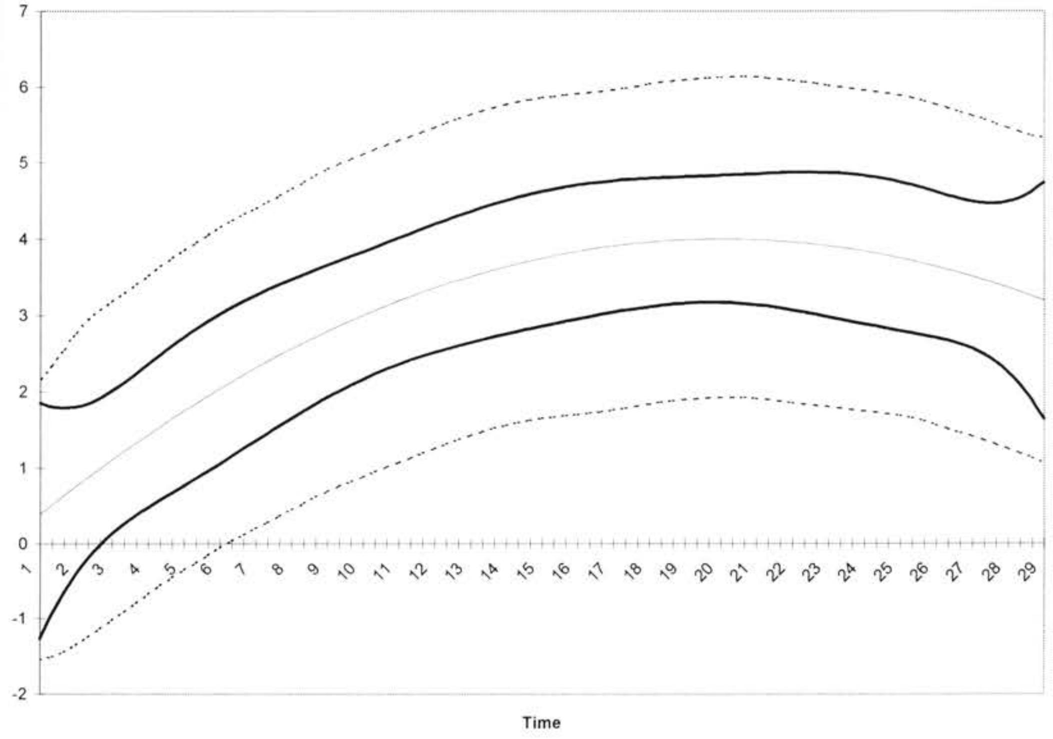


Figure 4.5.3 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 0.6$, case 5

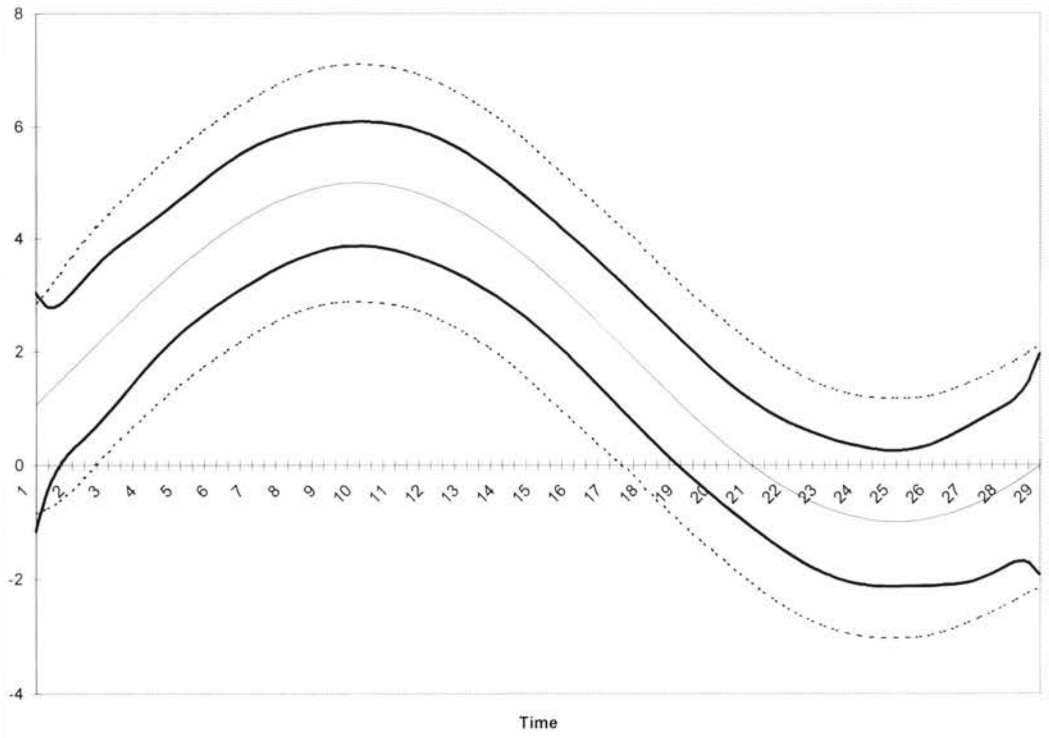


Figure 4.5.4 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 0.6$, case 5

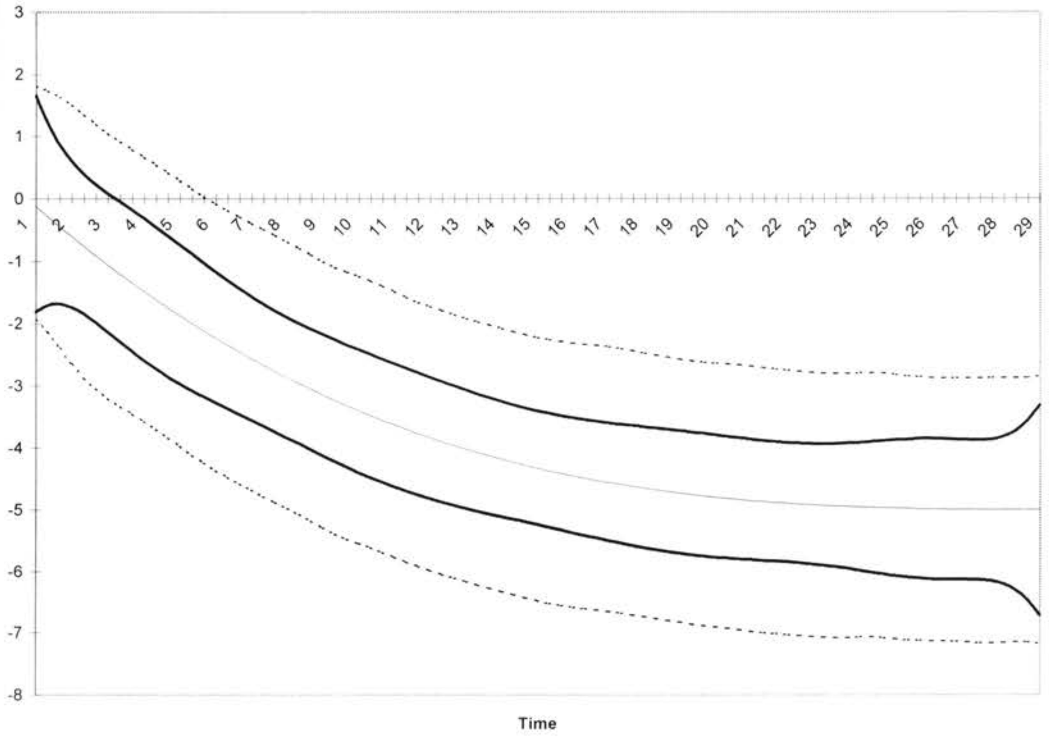


Figure 4.5.5 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 1.0$, case 5

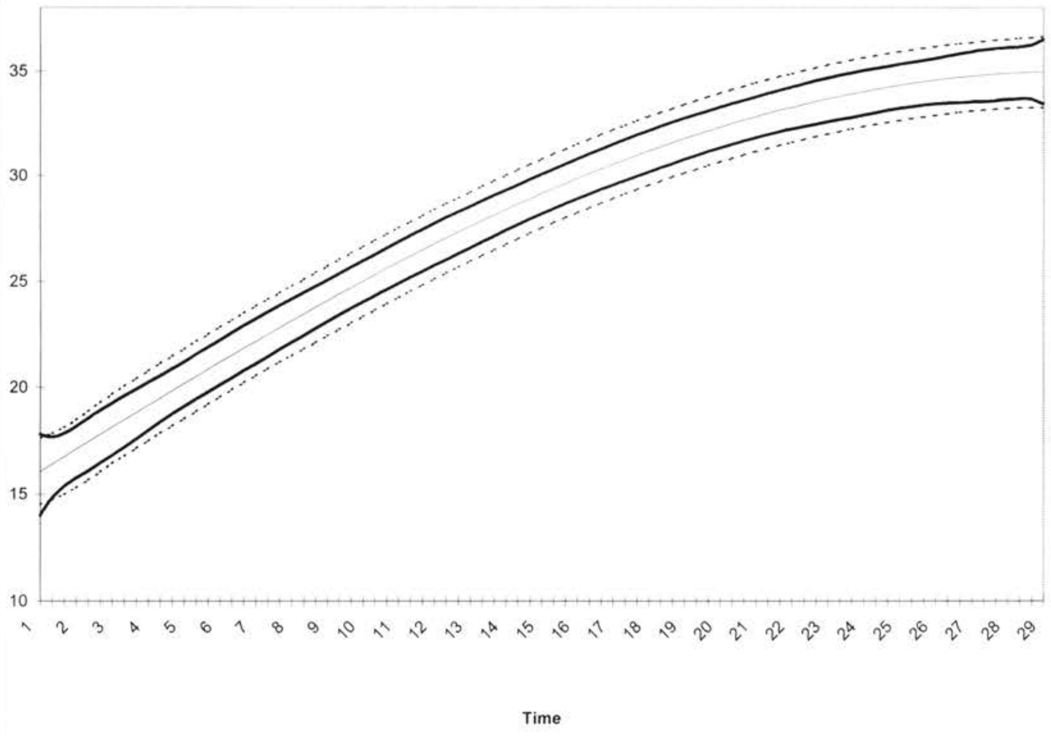


Figure 4.5.6 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 1.0$, case 5

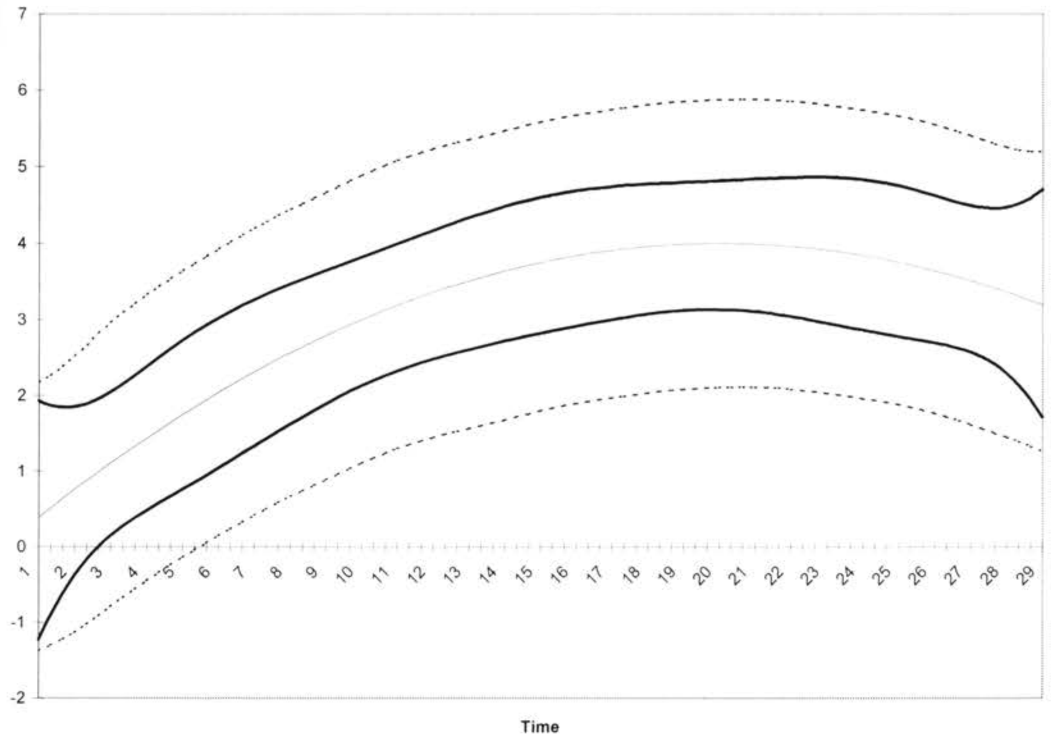


Figure 4.5.7 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 1.0$, case 5

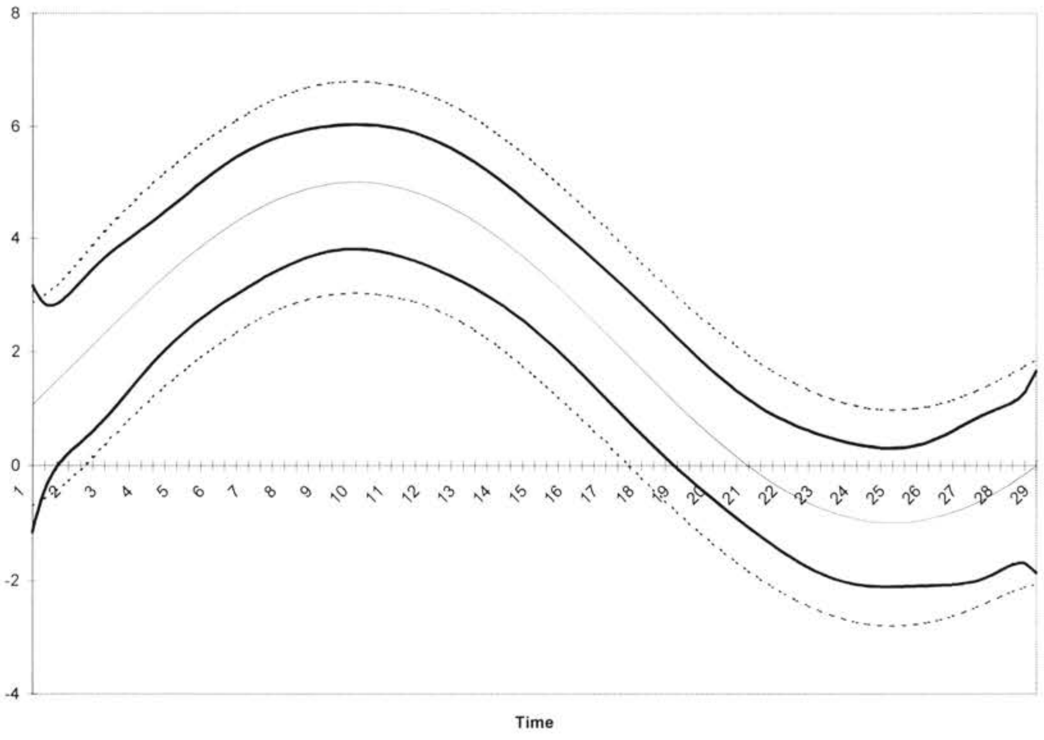


Figure 4.5.8 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 1.0$, case 5

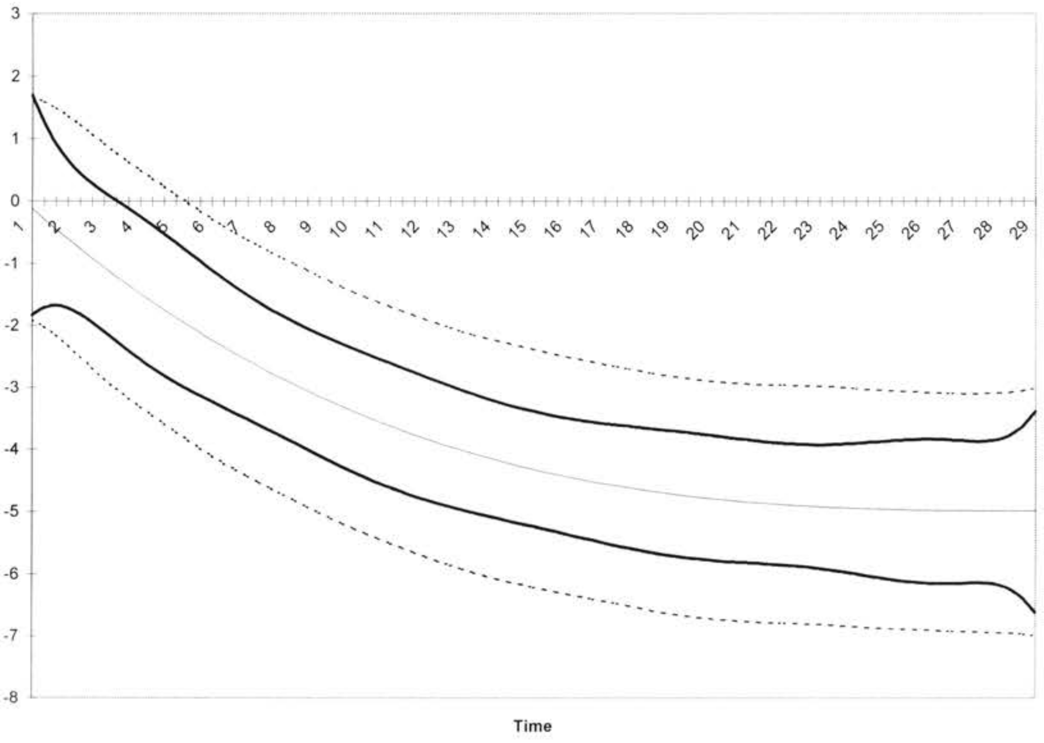


Figure 4.5.9 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 1.4$, case 5

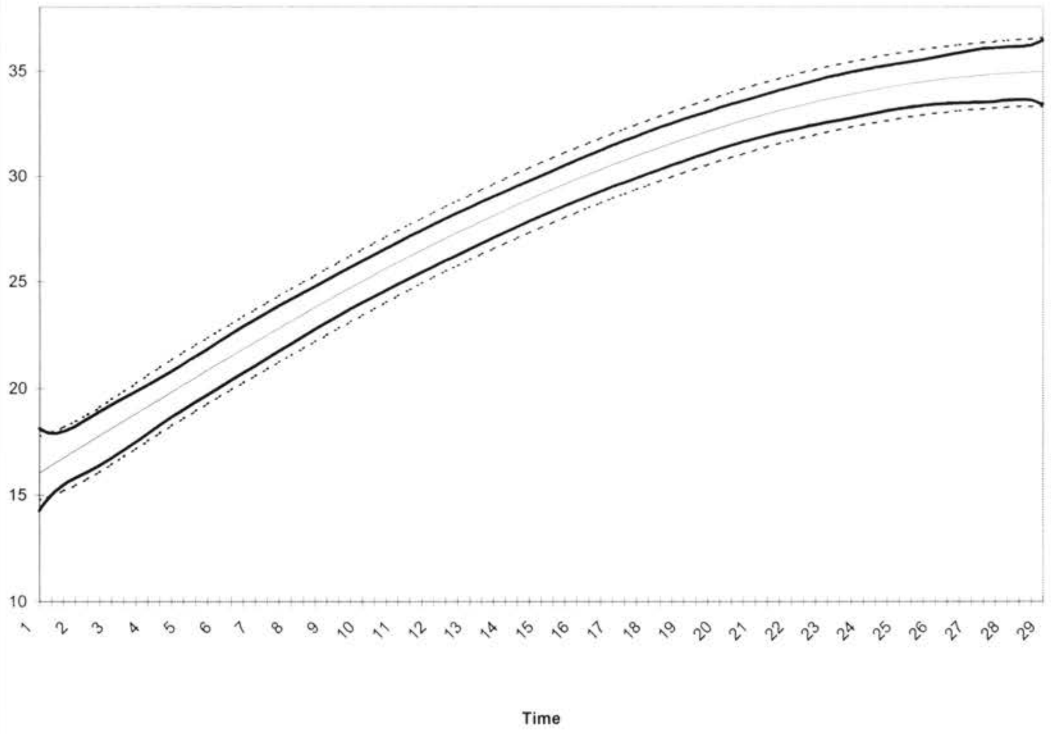


Figure 4.5.10 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 1.4$, case 5

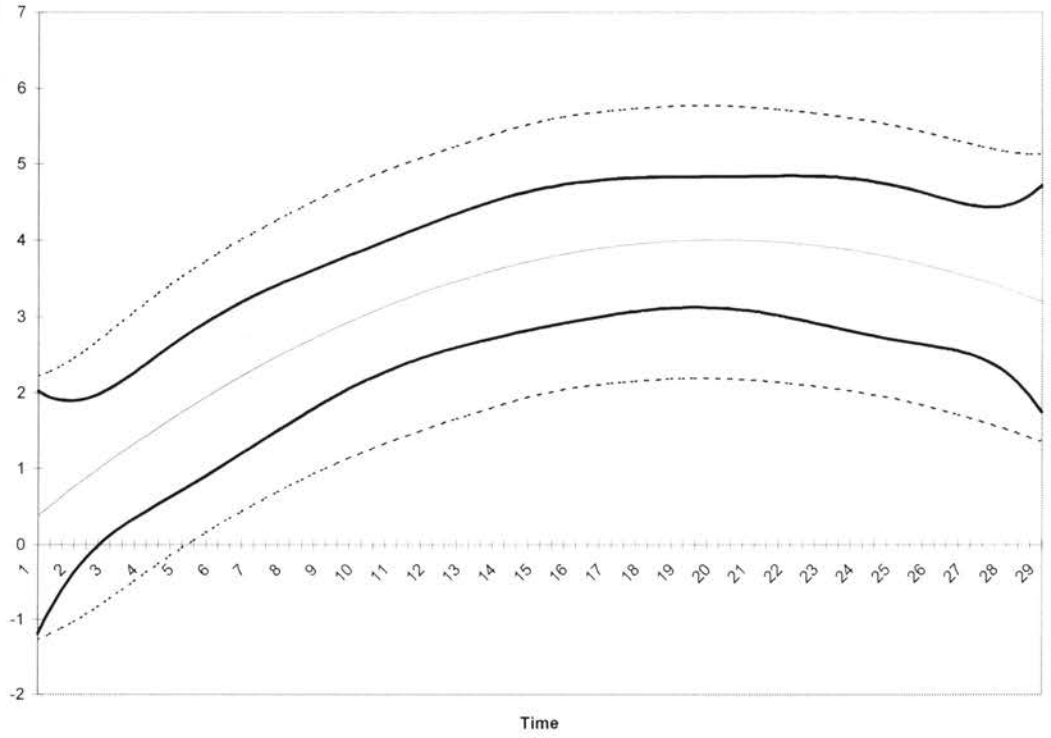


Figure 4.5.11 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 1.4$, case 5

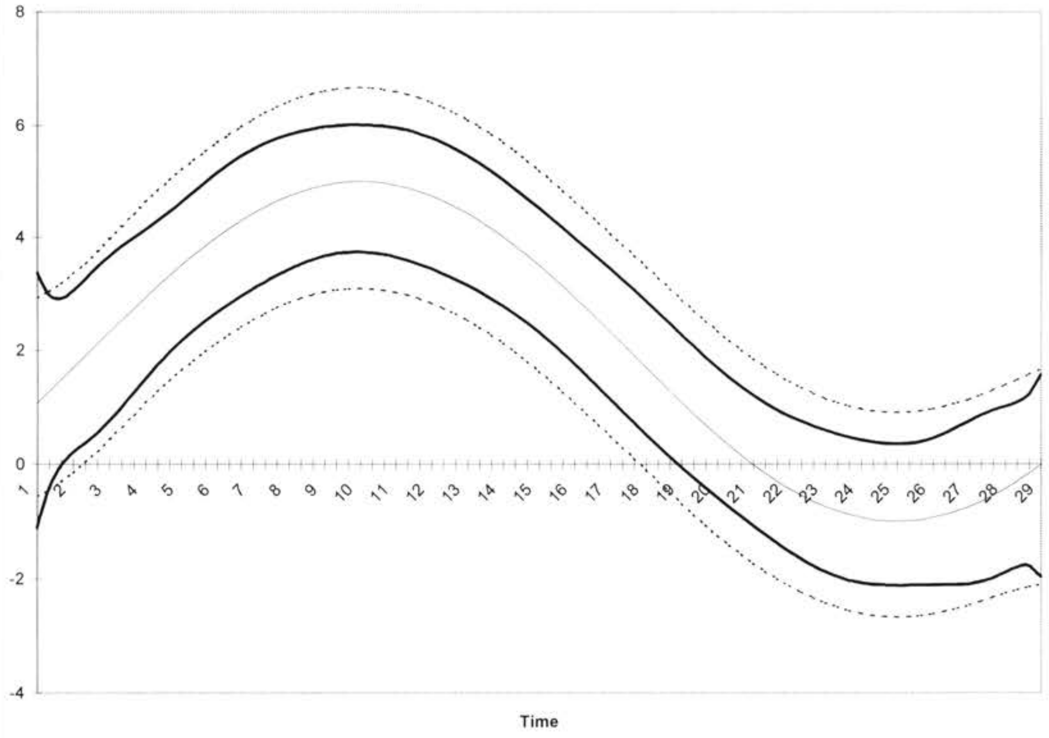
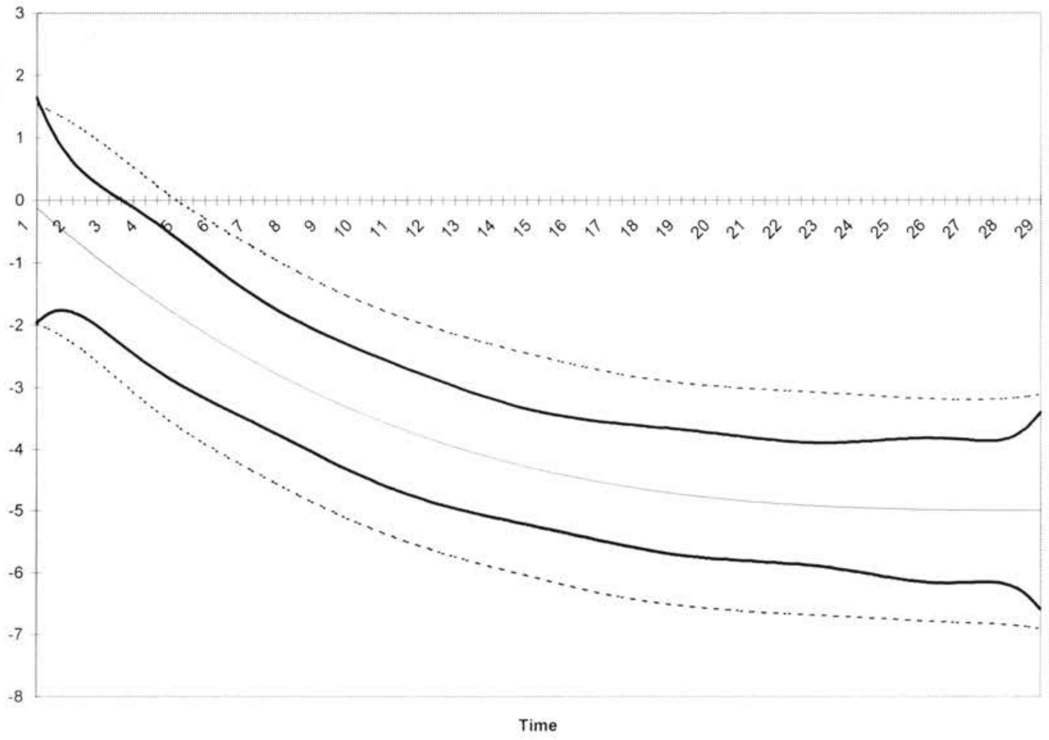


Figure 4.5.12 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 1.4$, case 5



4.6 Simulation case six

This simulation is an effort to make a full illustration of the advantage of the proposed procedure. The scenario follows case one except for the number of subjects, here $n = 150$, and the coefficient functions. Here the coefficent functions are

$$\beta_0(t) = 15 + 20\sin\left(\tfrac{t\pi}{60}\right) \qquad \beta_1(t) = 4 - \left(\tfrac{t-15}{9}\right)^2$$
$$\beta_2(t) = 2 + \tfrac{5}{4}\sin\left(\tfrac{t\pi}{15}\right) \qquad \beta_3(t) = e^{0.05t},$$

and the degrees fit are 5, 2, 5 and 2 respectively. So, the intercept function is the same, while the remaining coefficient functions possess somewhat more subtle behavior than those in the previous cases. The end result, viewing figures 4.6.2-4.6.4 and 4.6.6-4.6.8, is that the bands produced by the procedure of Wu *et al.* (1998) allow for the possibility of each of β_1, β_2 and β_3 to be time invariant, which they clearly are not. However, the proposed procedure is able to capture at least some of the nature of the relationship between the covariate effects and time, clearly indicating the advantage of being able to build tighter, less conservative bands.

Table 4.6 Estimated coverage probabilities on $t \in [1, 29]$ for case 6

h	Method	β_0	β_1	β_2	β_3
1.6	Proposed	0.986	0.978	0.990	0.984
	Bonferroni	1.000	1.000	1.000	1.000
2.0	Proposed	0.982	0.982	0.994	0.990
	Bonferroni	0.996	1.000	1.000	1.000

Figure 4.6.1 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 1.6$, case 6

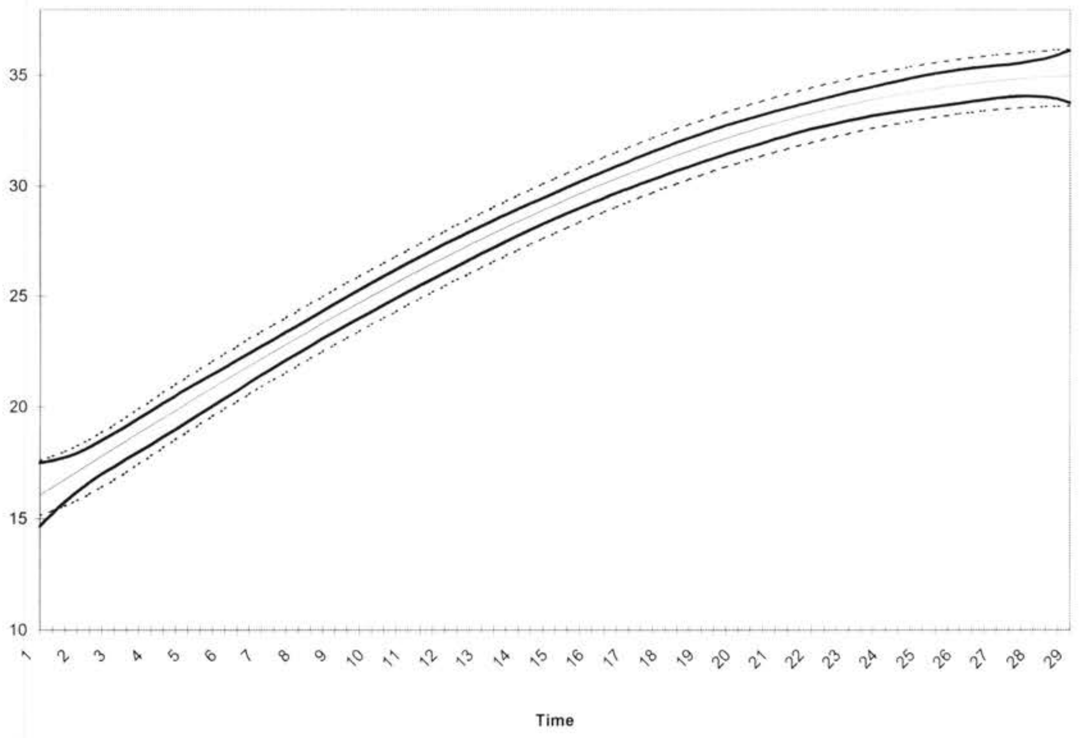


Figure 4.6.2 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 1.6$, case 6

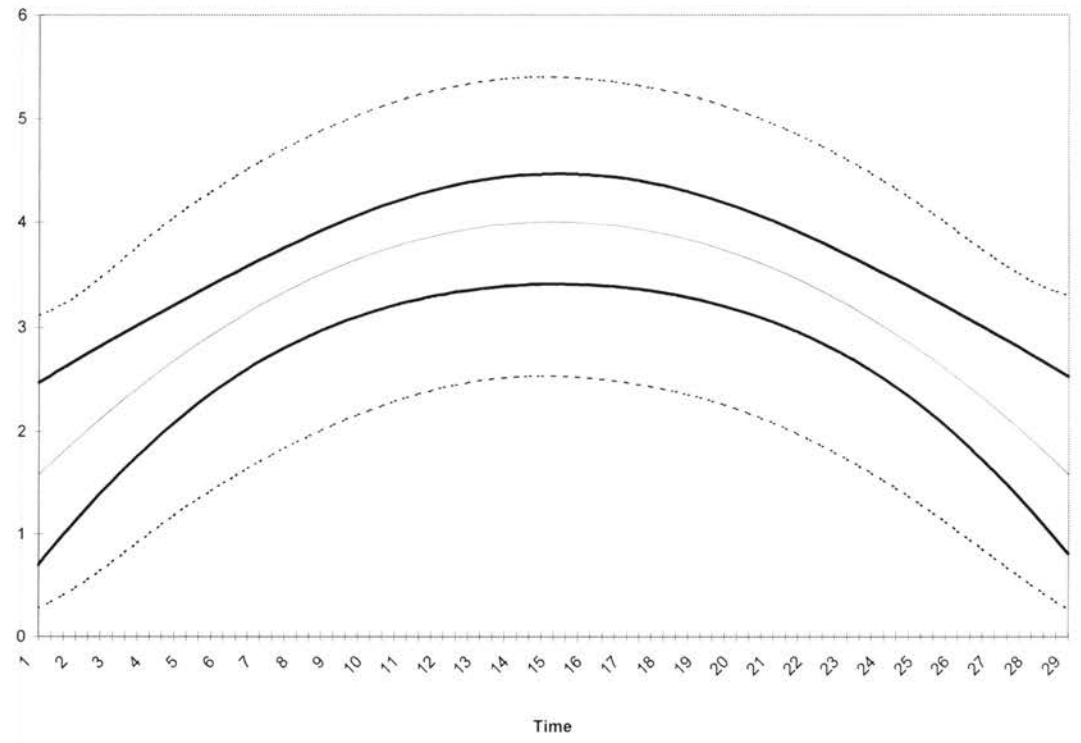


Figure 4.6.3 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 1.6$, case 6

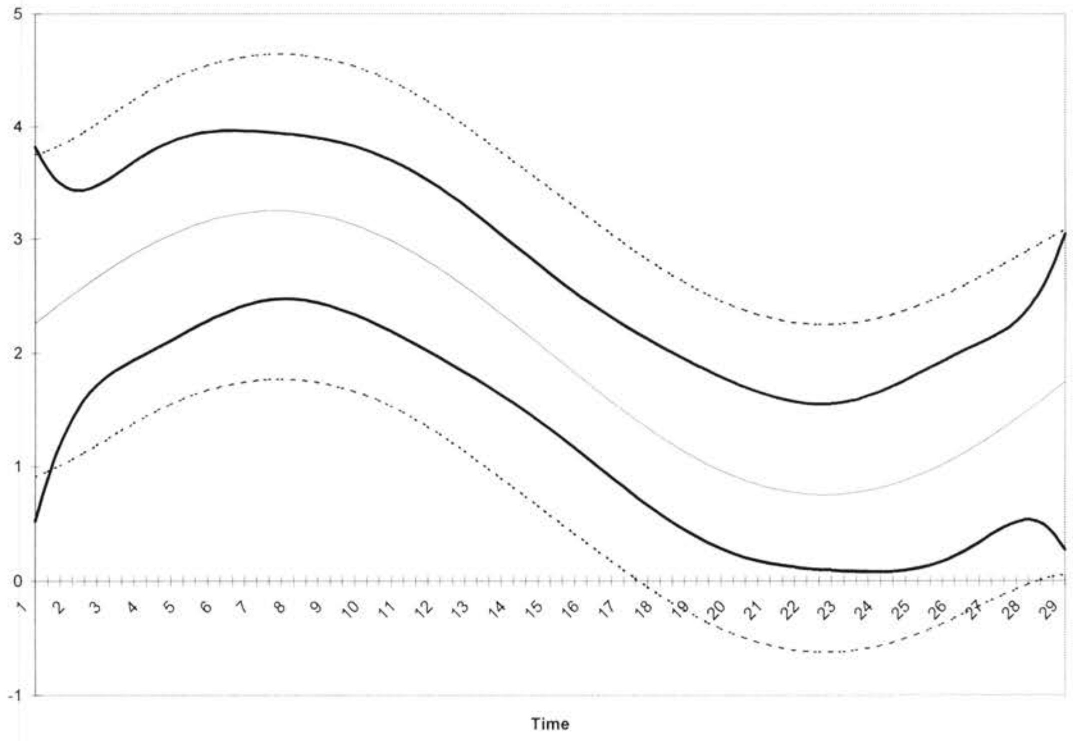


Figure 4.6.4 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 1.6$, case 6

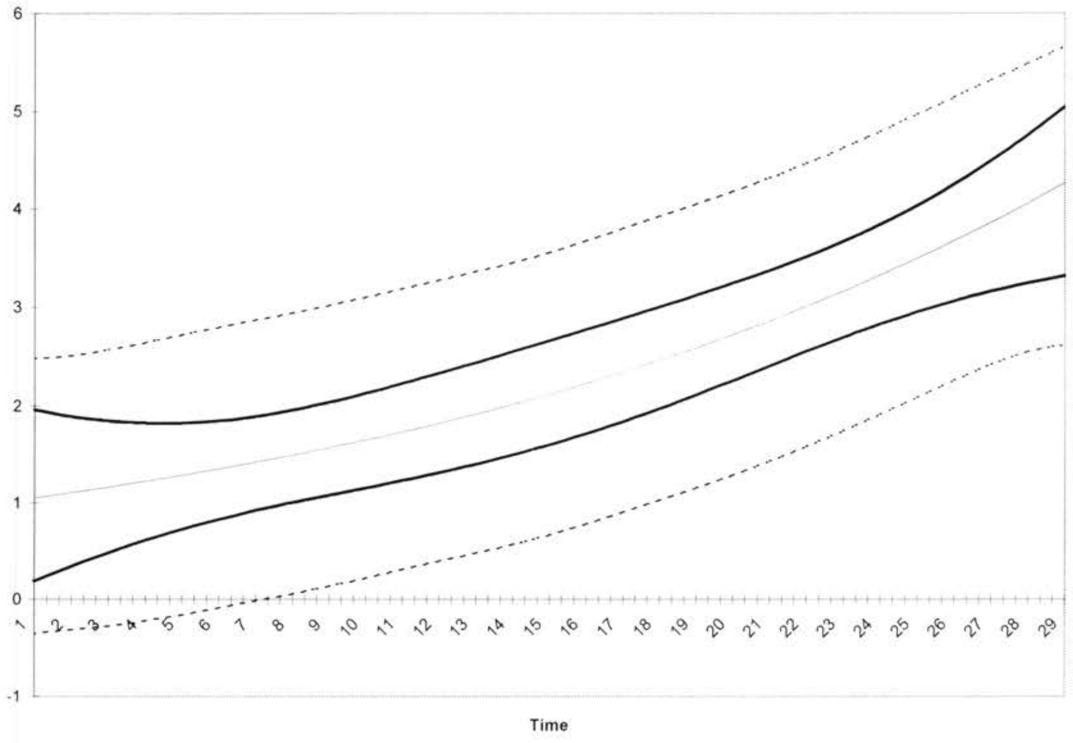


Figure 4.6.5 Average Bonferroni and proposed bands for $\beta_0(t)$, $h = 2.0$, case 6

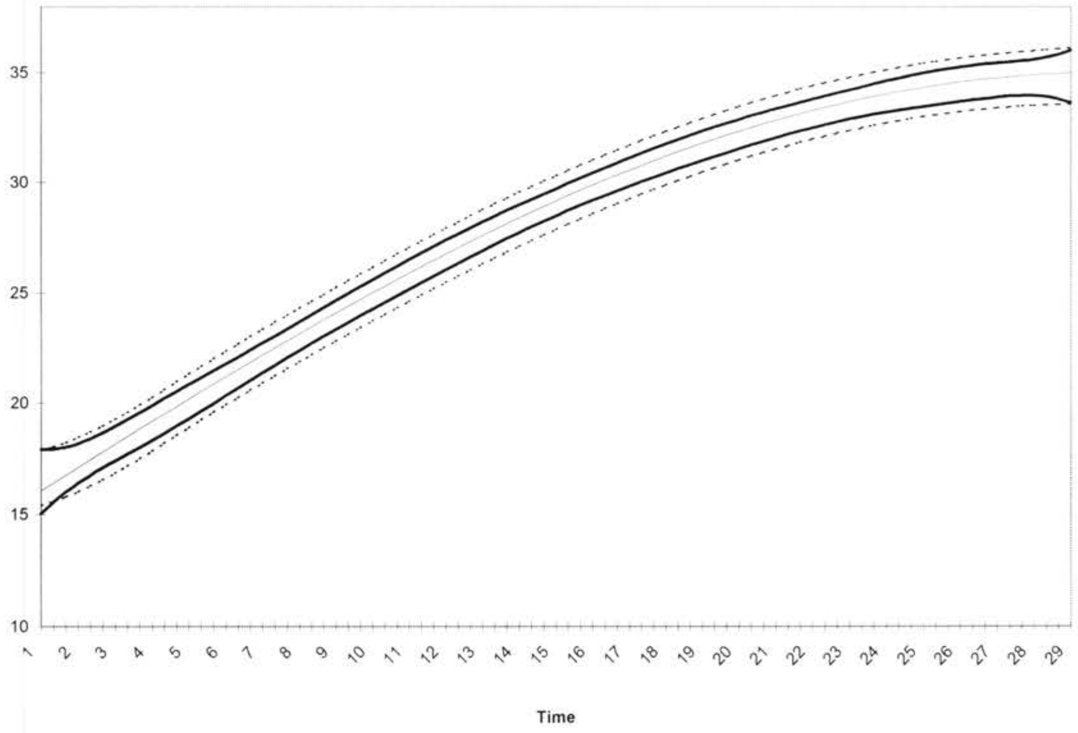


Figure 4.6.6 Average Bonferroni and proposed bands for $\beta_1(t)$, $h = 2.0$, case 6

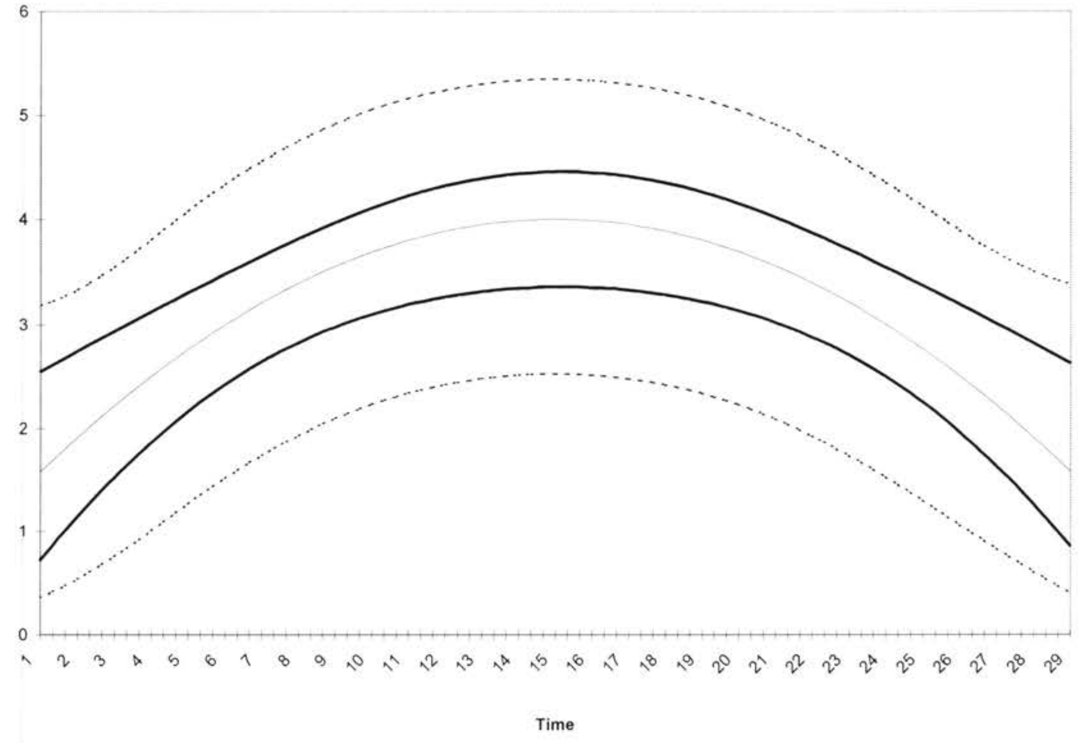


Figure 4.6.7 Average Bonferroni and proposed bands for $\beta_2(t)$, $h = 2.0$, case 6

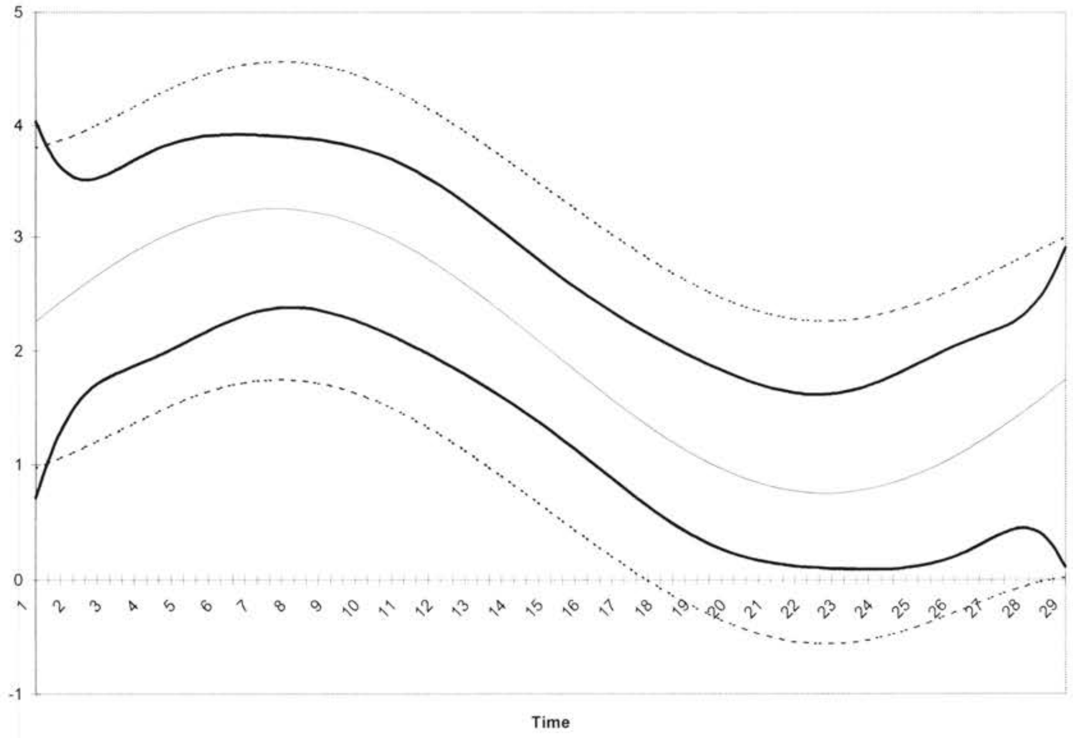
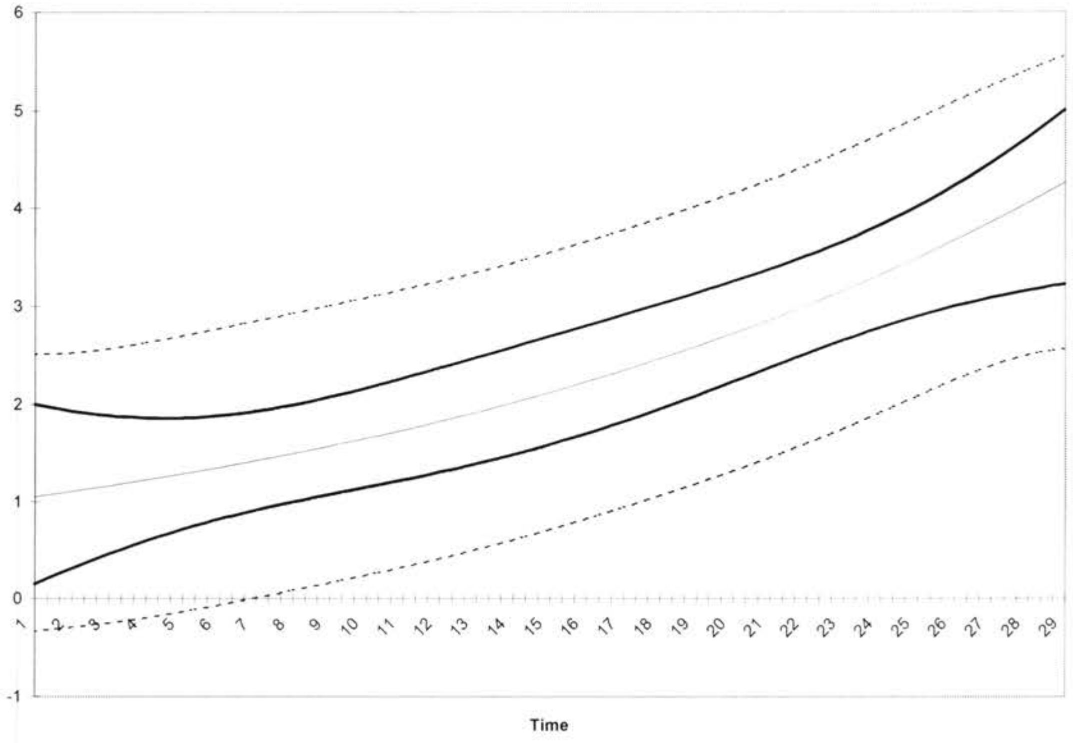


Figure 4.6.8 Average Bonferroni and proposed bands for $\beta_3(t)$, $h = 2.0$, case 6



4.7 Degree selection

For a scenario exactly like the case six simulation study, here individual estimates of the varying coefficient model are considered in an effort to study degree selection. In the following figures, the actual function, the kernel estimate and the polynomial fit for the degree defined in 4.6 are shown. Figures 4.7.1-4.7.4 are based on an estimate produced using the cross-validated (CV) bandwidth, approximately 1.17 in this case. As one can see the CV bandwidth leads to under-smoothed estimates for most of the coefficient functions, making it difficult, and also undesirable, to produce a polynomial overlays preserving their nature.

Figure 4.7.1 Kernel and polynomial estimate for $\beta_0(t)$ based on CV bandwidth

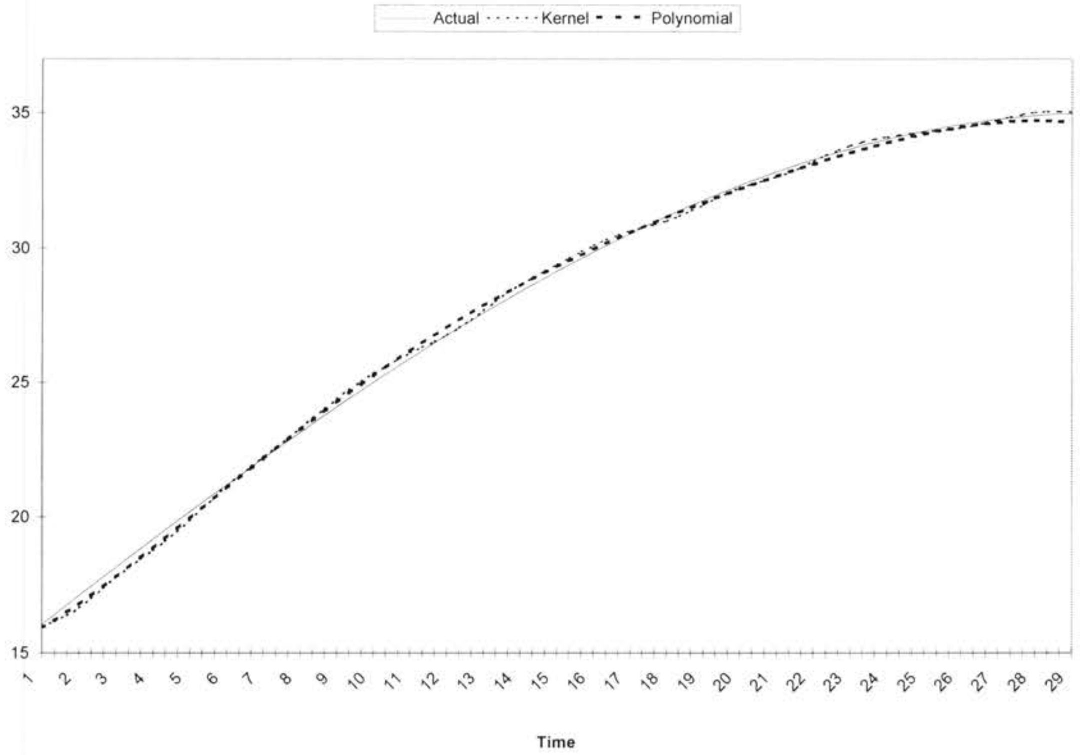


Figure 4.7.2 Kernel and polynomial estimate for $\beta_1(t)$ based on CV bandwidth

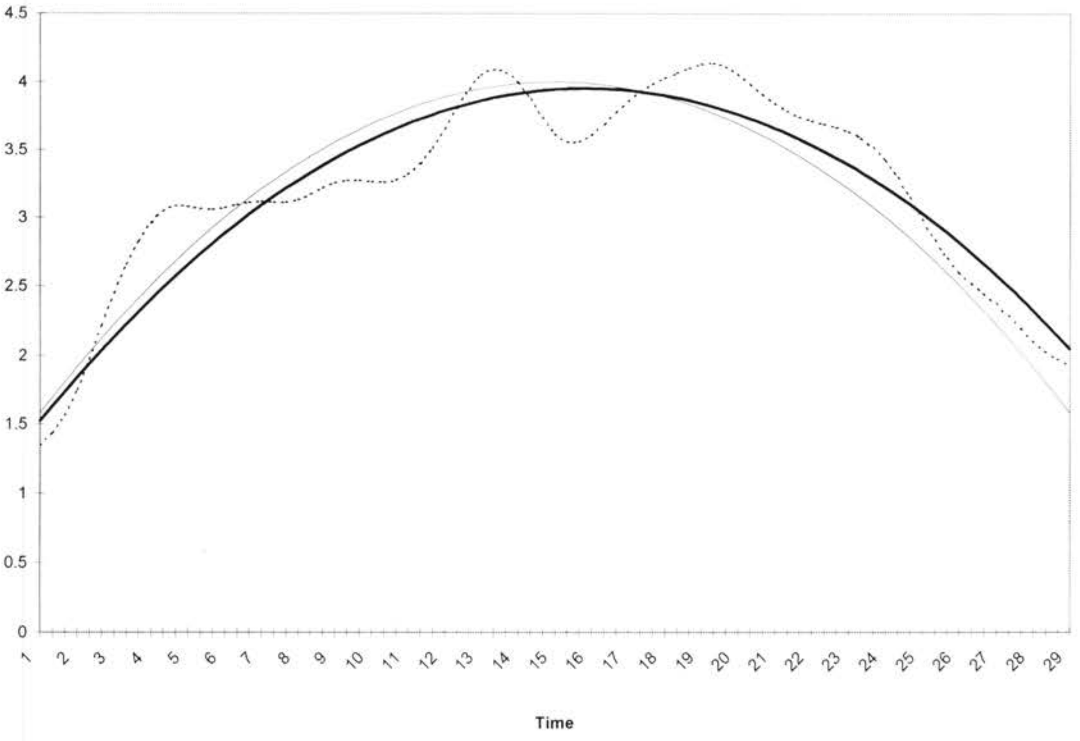


Figure 4.7.3 Kernel and polynomial estimate for $\beta_2(t)$ based on CV bandwidth

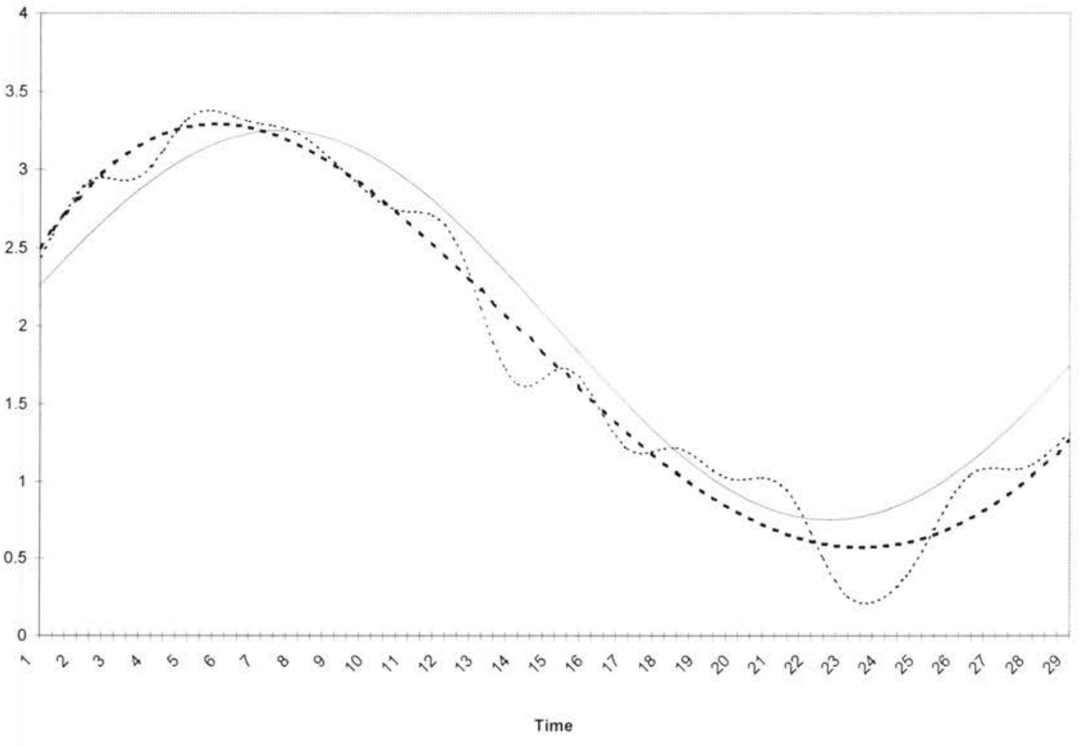
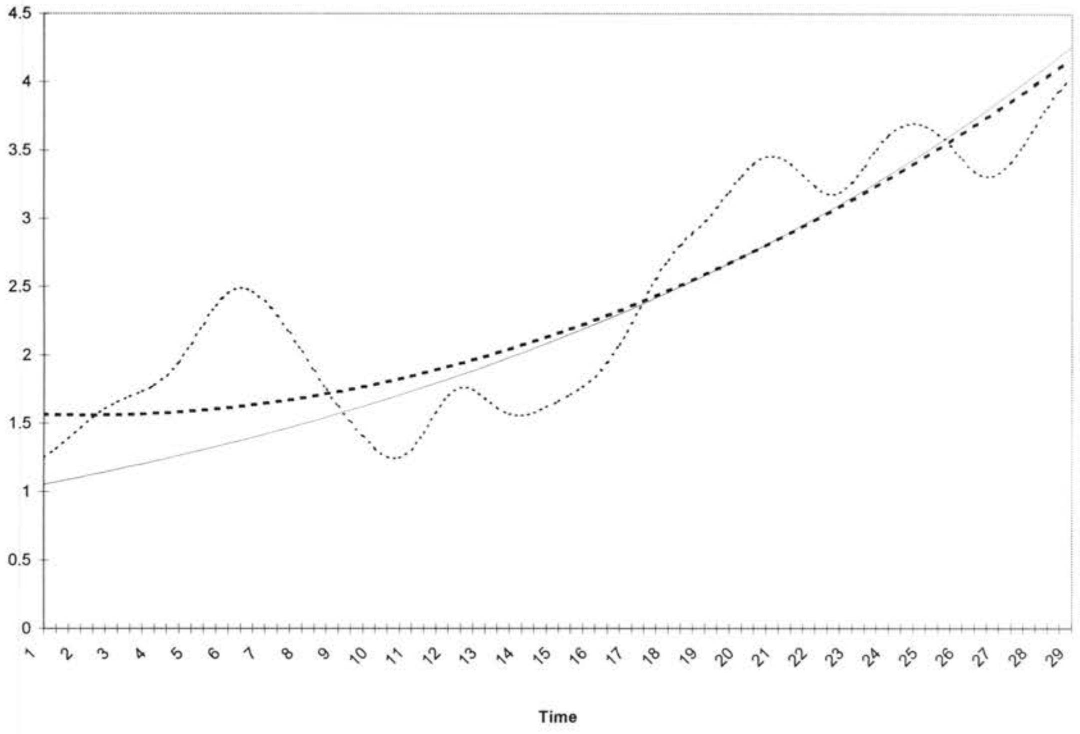


Figure 4.7.4 Kernel and polynomial estimate for $\beta_3(t)$ based on CV bandwidth



Even for twice the CV bandwidth, undersmoothness may still be a problem. And, as was noted in chapter one, since the kernel estimates only allow for one smoothing parameter, there may often be situations where proper smoothing cannot be obtained for all coefficient functions. This leaves a bit of a quandry when trying to make degree selections for the polynomial overlay. Judging by figures 4.7.1-4.7.8, a good rule of thumb for selection of polynomial overlays would be: If there appears to be a need for more smoothing, then there probably is. In practice, making an attempt to smooth out the remaining disturbances in the kernel estimates based on the CV bandwidth for β_1 and β_2 (figures 4.7.2 and 4.7.3) would seem reasonable; however, the disturbances in the kernel estimate of β_3 are much greater. Here one might over-parameterize the polynomial overlay in an effort to preserve something of the first "bump" (figure 4.7.4) in the kernel estimate. We have seen in simulation cases four and five that this over-parameterization is undesirable but not disasterous. In the end, it would seem prudent to do a fair amount of smoothing with the kernel estimate (maybe as much as 200% of the CV bandwidth) and use the polynomial overlay in an effort to smooth out any remaining suspicious behavior.

Figure 4.7.5 Kernel and polynomial estimate for $\beta_0(t)$, 200% of CV bandwidth

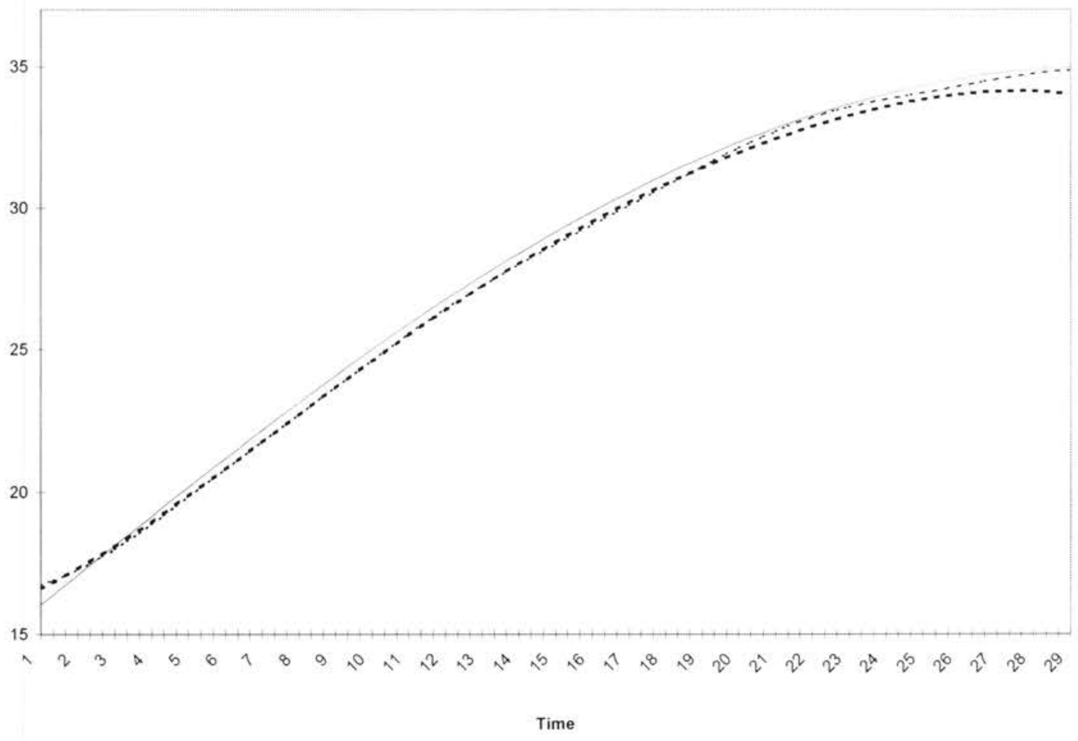


Figure 4.7.6 Kernel and polynomial estimate for $\beta_1(t)$, 200% of CV bandwidth

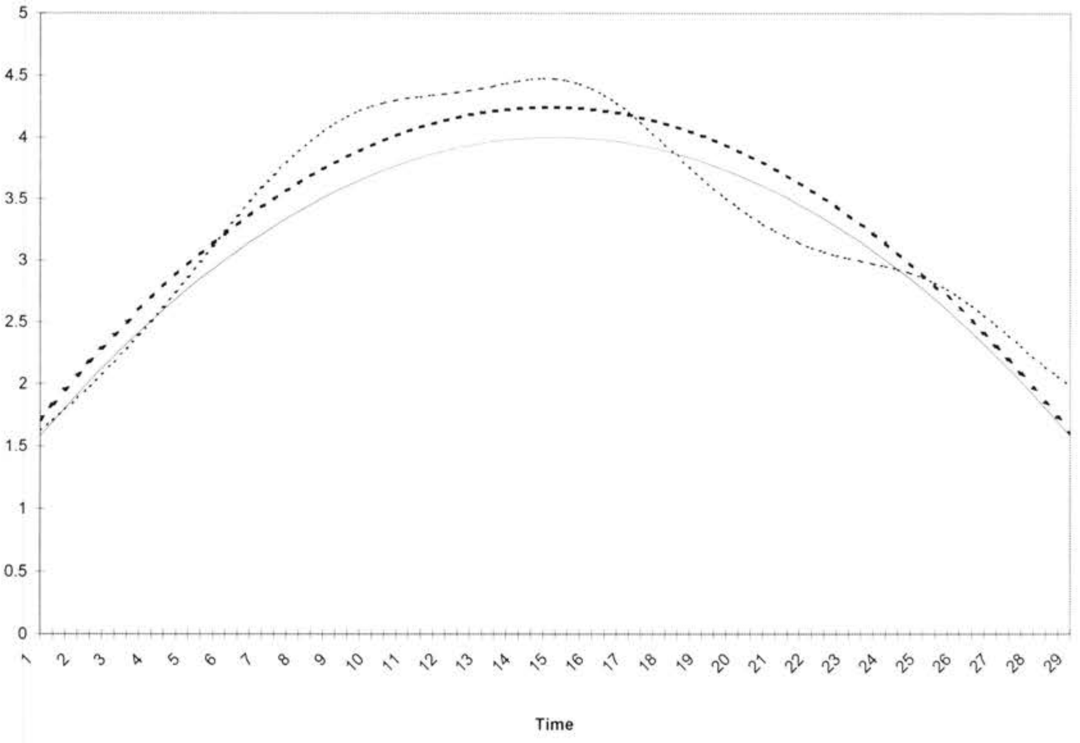


Figure 4.7.7 Kernel and polynomial estimate for $\beta_2(t)$, 200% of CV bandwidth

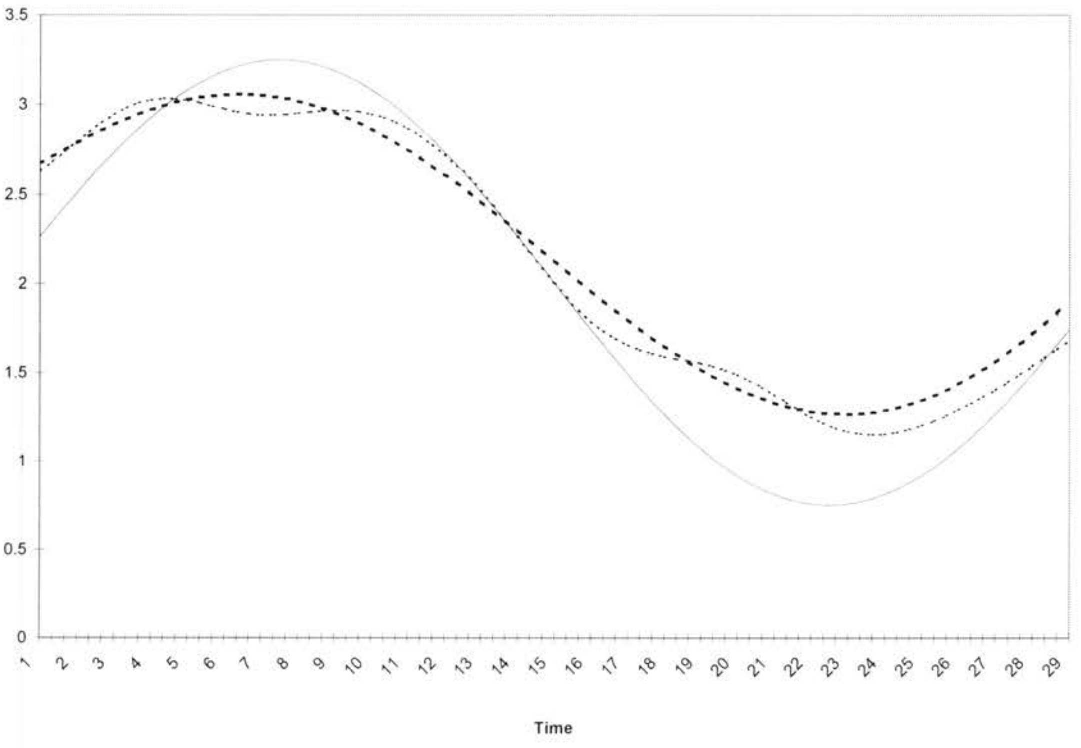
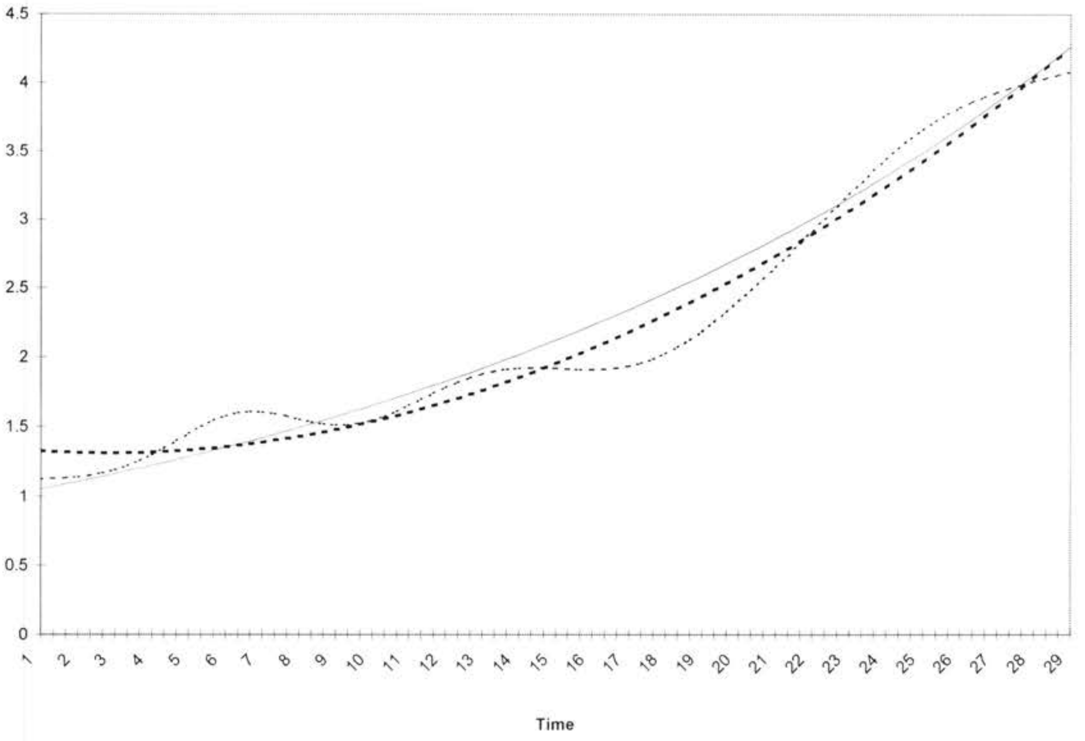


Figure 4.7.8 Kernel and polynomial estimate for $\beta_3(t)$, 200% of CV bandwidth



CHAPTER FIVE

CONCLUDING REMARKS

The method of constructing confidence bands, as set forth in chapter three, for the coefficient functions of a varying coefficient model appears to be superior to the method of Wu, Chiang and Hoover (1998) for a variety of situations. In all scenarios undertaken in chapter four, the proposed method is less conservative than the method of Wu *et al.* which allows for greater ability in discerning the true nature of the coefficient functions.

It is important to note that several avenues of development may allow for improvement to the proposed procedure, both in terms of practical and technical criteria. Investigation into the explicit properties of the variance estimate may allow for a more accurate assessment of the degrees of freedom available for the banding procedure. It would also be beneficial to devise an adaptation of the current method to situations where errors were non-stationary. Also, development of criteria and automatic selection procedures for choosing the degree of the polynomial fit would be desirable.

BIBLIOGRAPHY

Altman, N. S. (1990), "Kernel Smoothing of Data With Correlated Errors," *Journal of the American Statistical Association*, 85, 749-759.

Brumback, B. A. and Rice, J. A. (1998) "Smoothing Spline Models for the Analysis of Nested and Crossed Samples of Curves," *Journal of the American Statistical Association*, 93, 961-976.

Diggle, P. J., Liang, K. and Zeger, S. (1998) *Analysis of Longitudinal Data*, Oxford Science Publications.

Fan, J. and Zhang, J. T. (1998), "Functional Linear Models for Longitudinal Data," unpublished manuscript.

Hart, J. D. (1991), "Kernel Regression Estimation With Time Series Errors," *Journal of the Royal Statistical Society*, Ser. B, 53, 173-187.

Hart, J. D. and Wehrly, T. E. (1986), "Kernel Regression Estimation Using Repeated Measurements Data," *Journal of the American Statistical Association*, 81, 1080-1088.

Hastie, T. J. and Tibshirani, R. J. (1993), "Varying-Coefficient Models," *Journal of the Royal Statistical Society*, Ser. B, 55, 757-796.

Hoover, D. R., Rice, J. A., Wu, C. O., and Yang, L. P. (1998), "Nonparametric Smoothing Estimates of Time-Varying Coefficient Models with Longitudinal Data," *Biometrika*, 85, 809-822.

Knafl, G., Sacks, J., and Ylvisaker, D. (1985) "Confidence Bands for Regression Functions", *Journal of the American Statistical Association*, 80, 683-691.

Moyeed, R. A. and Diggle, P. J. (1994), "Rates of Convergence in Semi-Parametric Modeling of Longitudinal Data," *Australian Journal of Statistics*, 36, 75-93.

Naiman, D. Q. (1986), "Conservative Confidence Bands in Curvilinear Regression," *The Annals of Statistics*, 14, 896-906.

Ramsay, J. O. and Dalzell, C. J. (1991), "Some Tools for Functional Data Analysis," *Journal of the Royal Statistical Society*, Ser. B, 53, 539-572.

Wu, C. O., Chiang, C. T., and Hoover, D. R. (1998), "Asymptotic Confidence Regions for Kernel Smoothing of a Varying-Coefficient Model with Longitudinal Data," *Journal of the American Statistical Association*, 93, 1388-1402.

Zeger, S. L. and Diggle, P. J. (1994), "Semiparametric Models for Longitudinal Data with Application to CD4 Cell Numbers in HIV Seroconverters," *Biometrics*, 50, 689-699.

APPENDIX

SIMULATION CODE

What appears here is an example of the FORTRAN 90 code used to generate the simulation results seen in chapter four (this particular example is from simulation case six). As per FORTRAN 90 convention, the exclamation point (!) indicates the beginning of a commment line and the ampersand (&) indicates the continuation of a line of code over more than one line of text. It is important to note that, although not used for this particular example and with the relevant lines set as inactive comment lines, the program includes the code for choosing the cross-validation bandwidth.

```
Program Simulation

Use MSIMSL

Implicit None
        !Counter definitions
Integer :: i,j,k,l,r,g,g1,g2,differ,status,status3,piv,power,irule
Integer :: funcnumb,iteration,icode
Common / fcnindex / funcnumb
        !Coverage indicators and probabilities
Integer :: cover0, cover1, cover2, cover3
Integer :: covertot0, covertot1, covertot2, covertot3
Integer :: cover0b, cover1b, cover2b, cover3b
Integer :: covertot0b, covertot1b, covertot2b, covertot3b
Integer :: boncover0, boncover1, boncover2, boncover3
Integer :: boncovertot0, boncovertot1, boncovertot2, boncovertot3
Double Precision :: mycovprob0, mycovprob1, mycovprob2, mycovprob3
Double Precision :: mycovprob0b, mycovprob1b, mycovprob2b, mycovprob3b
Double Precision :: boncovprob0, boncovprob1, boncovprob2, boncovprob3
Double Precision :: actual
        !Number of covariate functions in the model
Integer, Parameter :: covariates=4
        !Number of time points where measurements are taken and subjects
Integer, Parameter :: measurements=30
Integer, Parameter :: subjects=150
Double Precision, Parameter :: pi=3.14159265359
        !Grid and iteration definitions
Integer, Parameter :: scale=3, repititions=500
Integer, Parameter :: endgrid=scale*measurements
        !Coefficient functions
```

```
Double Precision :: BETA0, BETA1, BETA2, BETA3
        !Other required functions
Double Precision, External :: CVCRIT, NGAMPR, GAM, COVPROB, NAIMARG
        !Variables used to generate errors
Double Precision, Allocatable, Dimension(:) :: errors                 !i.i.d. standard normal
errors
Double Precision, Allocatable, Dimension(:) :: err              !Errors for chosen structure
Double Precision :: variance=4                                          !Variance of error
Double Precision, Allocatable, Dimension(:,:) :: covblock  !Covariance of error structure
Double Precision, Allocatable, Dimension(:,:) :: p,pblock   !Factor of covariance
        !Matrix with columns corresponding to covariate values for each subject
Double Precision, Dimension(covariates,subjects) :: x=1
        !Holders for randomly generated covariates
Integer, Dimension(subjects) :: value
Double Precision,Dimension(subjects) :: value2
        !Design matrix for each subject
Double Precision, Dimension(measurements,covariates,subjects) :: design=0
        !Function Estimates
Double Precision, Dimension(measurements,subjects,covariates) :: est=0
Double Precision, Dimension(scale*measurements,covariates)
::b=0,bprime,bdblprime,bias
        !Variables used to generate measurement times
Double Precision, Dimension(measurements+1,subjects) :: t=0
        !Missing data indicators
Double Precision misses(measurements)
        !Number of observations per subject
Integer, Dimension(subjects) :: m=measurements
        !Simulated observations
Double Precision, Dimension(measurements,subjects) :: y=0
Common / observations / m, t, design, y
        !Computational variables and parameter estimates
Double Precision, Dimension(measurements,subjects) :: res
Double Precision, Dimension(subjects,measurements,measurements) :: v
Integer, Dimension(subjects,measurements,measurements) :: u
Double Precision, Dimension(measurements-1) :: vario, corr
Double Precision, Dimension(covariates, covariates, scale*measurements) :: etahat
Double Precision, Dimension(covariates, covariates) :: tempcovxcov, etahatinv,
etahatinv2
Double Precision, Dimension&
&(covariates,covariates,scale*measurements,scale*measurements) :: D, Dstr
Double Precision, Dimension(scale*measurements,scale*measurements,covariates)&
& :: varcov, varcovinv
Double Precision, Dimension(covariates) :: caplambda
Common / arclength / caplambda
Integer :: df
```

```
Common / degfree / df
Double Precision, Dimension(scale*measurements) :: s, fhat, fprime
Double Precision, Dimension(scale*measurements,covariates) :: myupper, mylower
Double Precision, Dimension(scale*measurements,covariates) :: bonupper, bonlower
Double Precision, Dimension(scale*measurements,covariates) :: myuppertot, mylowertot
Double Precision, Dimension(scale*measurements,covariates) :: bonuppertot,
bonlowertot
Double Precision, Dimension(scale*measurements,covariates) :: avgupper, avglower
Double Precision, Dimension(scale*measurements,covariates) :: &
&avgbonupper,avgbonlower
Double Precision, Allocatable, Dimension(:,:) :: xmat
Double Precision, Dimension(covariates, covariates) :: amat, ainv
Double Precision, Dimension(covariates) :: bmat
Double Precision, Dimension(10,10,covariates) :: covmat, faccov
Common / factmat / faccov
Double Precision, Dimension(10,covariates) :: estcoef
Integer, Dimension(covariates) :: degree
Common / powers / degree
Integer :: total
Integer :: position, count, dist
Double Precision :: bonalpha, prcentile, zscore, bonerr
Double Precision :: hcv, totss, totobssq, sumsq, vartot,mu2,low,high,tol,sum
Double Precision :: sigsq, lambda, hnaught, densitysum, arg, kern, kernsum,total2
Double Precision :: lower,upper,errabs,errrel,errest,biastemp,biastot
Double Precision :: beg,finish,aberr,relerr,estimate,temperr,stderr,errterm
Integer :: maxf
Double Precision, Dimension(covariates)::critval
Double Precision, Allocatable, Dimension(:) :: tempmat2, tempmat3, basis, tempvec
Double Precision, Allocatable, Dimension(:,:) :: tempmat, tempcovmat, tempmat4

!Initialize
myuppertot=0
mylowertot=0
bonuppertot=0
bonlowertot=0
covertot0=0
covertot1=0
covertot2=0
covertot3=0
covertot0b=0
covertot1b=0
covertot2b=0
covertot3b=0
boncovertot0=0
boncovertot1=0
```

```
boncovertot2=0
boncovertot3=0
iteration=1

Do While (iteration < repititions+1)

x=1
design=0
est=0
b=0
t=0
y=0
m=measurements
vario=0
corr=0
etahat=0
etahatinv=0
etahatinv2=0
tempcovxcov=0
D=0
Dstr=0
varcov=0
varcovinv=0
caplambda=0
fhat=0
fprime=0
myupper=0
mylower=0
bonupper=0
bonlower=0
amat=0
ainv=0
bmat=0
covmat=0
faccov=0
estcoef=0
df=0
bonerr=0
sigsq=0
lambda=0
hnaught=0
critval=0

        !!!!!Begin generation of data!!!!!!!!!!!!!!
```

```
Call RNSET (0)
!Sets seed to system clock

        !Generate random binary covariates
Do k=2, covariates-1
        Call RNBIN(subjects, 1, .5, value)
x(k,1:subjects)=value
End Do

        !Generate normal covariates
Call DRNNOA(subjects, value2)
        x(covariates,1:subjects)=.5*value2


        !Time point generation
Do i=1, subjects
        t(1,i)=DRNUNF()                                 !Generate initial time points
        Do j=1, measurements
                t(j,i)=t(1,i)+(j-1)             !Generate remaining "scheduled" time points
        End Do
End Do

        !Generate random "missing indicators"
Do i=1, subjects
        Call DRNUN(measurements, misses)
        Do j=measurements, 2, -1
                If (misses(j) < .6) Then
                        m(i)=m(i)-1                     !Update number of observations
                        Do k=j, measurements
                                t(k,i)=t(k+1,i)         !Remove missing observations
                        End Do
                End If
        End Do
End Do

        !Compute total number of observations
total=0
Do i=1, subjects
        total=total+m(i)
End Do

        !Build errors
Allocate (errors(total), STAT=status)                   !Allocate space for i.i.d. errors

Call DRNNOA(total, errors)                              !Generate i.i.d std. normal errors
```

```
        !Allocate space for factored covariance str.
Allocate (p(total,total), STAT=status3)

position=0
Do i=1, subjects
        Allocate (covblock(m(i),m(i)),pblock(m(i),m(i)))
        covblock=0.0
        pblock=0.0
        Do j=1, m(i)
                Do k=1, m(i)
                covblock(j,k)=variance*DEXP(-1.0*DABS(t(j,i)-t(k,i)))
                End Do
        End Do
        Call DCHFAC (m(i), covblock, m(i), 100*DMACH(4), piv, pblock, m(i))
        p(position+1:position+m(i),position+1:position+m(i))=pblock(1:m(i),1:m(i))
        position=position+m(i)
        Deallocate (covblock,pblock)
End Do


Allocate (err(total))                                   !Allocate space for actual errors


        !Compute errors from factorization and i.i.d errors
Call DMURRV (total, total, p, total, total, errors, 2, total, err)
Deallocate(p)

        !Build simulated observations
position=0
Do i=1, subjects
        Do j=1, m(i)

y(j,i)=BETA0(t(j,i))+BETA1(t(j,i))*x(2,i)+BETA2(t(j,i))*x(3,i)+BETA3(t(j,i))*x(4,i)+&
                err(position+j)
        End Do
        position=position+m(i)
End Do
        !!!!!End of data generation!!!!!!!



        !Build design matricies for each subject
Do i=1, subjects
        Do j=1, m(i)
```

```
                design(j,:,i)=x(:,i)
        End Do
End Do

        !Choose cross-validation bandwidth
!low=0.75E0
!high=2.05E0
!tol=1.0E-1
!Call DUVMGS(CVCRIT,low,high,tol,hcv)
!Write (*,*) hcv

hcv=1.6

        !Degrees for polynomial fits
degree(1)=5
degree(2)=2
degree(3)=5
degree(4)=2

        !Build Estimates at Design Points
Do i=1,subjects
        Do j=1,m(i)
        Call BUILDA(subjects, covariates, measurements, m, design, t(j,i), t, hcv, amat)
        Call BUILDB(subjects, covariates, measurements, m, design, t(j,i), t, hcv, y,
bmat)
        Call DLINRG(covariates, amat, covariates, ainv, covariates)
        est(j,i,:)=MATMUL(ainv, bmat)
        End Do
End Do

        !Build Residuals
res=0
Do i=1, subjects
        Do j=1, m(i)
        res(j,i)=y(j,i)-DOT_PRODUCT(x(:,i), est(j,i,:))
        End Do
End Do

        !Variance Estimate
totss=0
totobssq=0
Do i=1, subjects
        sumsq=DOT_PRODUCT(res(1:m(i),i),res(1:m(i),i))
        totss=totss+sumsq
        totobssq=totobssq+m(i)**2
```

```
End Do
sigsq=totss/(1.0*(total-subjects))
total2=1.0*total
lambda=totobssq*(total2**(-1.2))
hnaught=hcv*(total2**(.2))

        !Estimate coefficient functions
Do g=1,endgrid
        amat=0
        bmat=0
        ainv=0
        s(g)=(1.0*g)/(1.0*scale)
        Call BUILDA(subjects, covariates, measurements, m, design, s(g), t, hcv, amat)
        Call BUILDB(subjects, covariates, measurements, m, design, s(g), t, hcv, y, bmat)
        Call DLINRG(covariates, amat, covariates, ainv, covariates)
        b(g,:)=MATMUL(ainv, bmat)
        !Estimate observation time density
        densitysum=0
        Do i=1,subjects
                Do j=1,m(i)
                        arg=(s(g)-t(j,i))/hcv
                        kern=(1.0/DSQRT(2*pi))*DEXP(-.5*(arg**2))
                        densitysum=densitysum+kern
                End Do
        End Do
        fhat(g)=(1/(DBLE(total)*hcv))*densitysum
End Do

        !Estimate derivative
Do k=1,covariates
        Do g=2,endgrid-1
                bprime(g,k)=(b(g+1,k)-b(g-1,k))/(s(g+1)-s(g-1))
        End Do
End Do

        !Estimate second derivative
Do k=1,covariates
        Do g=3,endgrid-2
                bdblprime(g,k)=(bprime(g+1,k)-bprime(g-1,k))/(s(g+1)-s(g-1))
        End Do
End Do

        !Estimate derivative of time point density
Do g=2,endgrid-1
        fprime(g)=(fhat(g+1)-fhat(g-1))/(s(g+1)-s(g-1))
```

```
End Do

v=0
u=0
        !Build correlation function estimate
Do i=1, subjects
        Do j=1, m(i)-1
                Do l=j+1, m(i)
                        v(i,j,l)=.5*(res(j,i)-res(l,i))**2
                        u(i,j,l)=NINT(t(l,i)-t(j,i))
                End Do
        End Do
End Do

Do differ=1, measurements-1
        vartot=0
        count=0
        Do i=1,subjects
                Do j=1, m(i)-1
                        Do l=j+1, m(i)
                                If (u(i,j,l)==differ) Then
                                        vartot=vartot+v(i,j,l)
                                        count=count+1
                                End If
                        End Do
                End Do
        End Do
        If (count==0) Then
                count=1
                vartot=0
        End If
        vario(differ)=vartot/(1.0*count)
        corr(differ)=1.0-(vario(differ)/sigsq)
End Do

        !Estimate covariance structure for kernel estimate
mu2=0.5*(1.0/DSQRT(pi))
Do g=1,endgrid
        Do l=1,covariates
                Do r=l,covariates
                        sum=0
                        Do i=1,subjects
                                kernsum=0
                                Do j=1,m(i)
                                        arg=(s(g)-t(j,i))/hcv
                                        kern=(1.0/DSQRT(2*pi))*DEXP(-.5*(arg**2))
                                        kernsum=kernsum+kern
                              End Do !j
                              sum=sum+x(l,i)*x(r,i)*kernsum
                    End Do !i
                    etahat(l,r,g)=(1.0/(fhat(g)*DBLE(total)*hcv))*sum
                    etahat(r,l,g)=etahat(l,r,g)

D(l,r,g,g)=(etahat(l,r,g))*(fhat(g)*mu2+lambda*hnaught*(fhat(g)**2))
                    D(r,l,g,g)=D(l,r,g,g)
          End Do !r
     End Do !l
     Call DLINDS(covariates, etahat(:,:,g), covariates, etahatinv, covariates)
     tempcovxcov=MATMUL(etahatinv,D(:,:,g,g))
     tempcovxcov=MATMUL(tempcovxcov,etahatinv)
     Dstr(:,:,g,g)=(1.0/(fhat(g)**2))*tempcovxcov
End Do !g

Do g1=1,endgrid
     Do g2=g1+1,endgrid
          dist=NINT(s(g2)-s(g1))
          Do l=1, covariates
               Do r=l, covariates
                    D(l,r,g1,g2)=corr(dist)*lambda*hnaught*&
                    &((etahat(l,r,g1)+etahat(l,r,g2))/2)*fhat(g1)*fhat(g2)
                    D(r,l,g1,g2)=D(l,r,g1,g2)
               End Do !r
          End Do !l
          Call DLINDS(covariates, etahat(:,:,g1), covariates, etahatinv, covariates)
          Call DLINDS(covariates, etahat(:,:,g2), covariates, etahatinv2, covariates)
          tempcovxcov=MATMUL(etahatinv,D(:,:,g1,g2))
          tempcovxcov=MATMUL(tempcovxcov,etahatinv2)
          Dstr(:,:,g1,g2)=(1.0/(fhat(g1)*fhat(g2)))*tempcovxcov
          tempcovxcov=MATMUL(etahatinv2,D(:,:,g1,g2))
          tempcovxcov=MATMUL(tempcovxcov, etahatinv)
          Dstr(:,:,g2,g1)=(1.0/(fhat(g1)*fhat(g2)))*tempcovxcov
     End Do !g2
End Do !g1


     !Extract var-covariance matrix for kernel estmates
Do k=1,covariates
     Do g1=1,endgrid
          Do g2=g1,endgrid
               varcov(g1,g2,k)=Dstr(k,k,g1,g2)/(1.0*total*hcv)
```

```
                    varcov(g2,g1,k)=varcov(g1,g2,k)
                End Do
        End Do
        Call DLINRG(endgrid,varcov(1:endgrid,1:endgrid,k),endgrid,&
        &varcovinv(1:endgrid,1:endgrid,k),endgrid)
End Do

        !Estimate bias

Do k=1,covariates
        Do g=3,endgrid-2
                biastot=0
                Do l=1,covariates

biastemp=etahat(k,l,g)*(bprime(g,l)*fprime(g)+0.5*bdblprime(g,l)*fhat(g))
                biastot=biastot+biastemp
                End Do
                bias(g,l)=biastot*(hnaught**(1.5))/(DSQRT(1.0*total*hcv))
        End Do
End Do

b=b-bias

        !Build parametric fits
Do k=1, covariates
        Allocate (xmat(endgrid,degree(k)+1))
        Do g=3,endgrid-2
                Do power=1,degree(k)+1
                        xmat(g,power)=s(g)**(power-1)
                End Do
        End Do
        Allocate(tempmat(endgrid-4,degree(k)+1),&
        &tempmat2(endgrid-4),tempmat3(degree(k)+1),&
        &tempcovmat(degree(k)+1,degree(k)+1))
        Allocate(tempmat4(degree(k)+1,endgrid-4))
        tempmat=MATMUL(varcovinv(3:endgrid-2,3:endgrid-2,k),xmat(3:endgrid-2,:))
        tempmat4=TRANSPOSE(xmat(3:endgrid-2,:))
        tempcovmat=MATMUL(tempmat4,tempmat)
        tempmat2=MATMUL(varcovinv(3:endgrid-2,3:endgrid-2,k),b(3:endgrid-2,k))
        tempmat3=MATMUL(tempmat4,tempmat2)
        Call ERSET(0,0,0)
        Call DLINDS(degree(k)+1,tempcovmat,degree(k)+1,&
        &covmat(1:degree(k)+1,1:degree(k)+1,k),degree(k)+1)
        icode=IERCD()
        If (icode .EQ. 2) Then
```

```
                Write(*,*) 'Error'
                Deallocate(tempmat,tempmat2,tempmat3,tempcovmat,tempmat4,xmat)
                Goto 100
        End IF

estcoef(1:degree(k)+1,k)=MATMUL(covmat(1:degree(k)+1,1:degree(k)+1,k),tempmat3)
        Deallocate(tempmat,tempmat2,tempmat3,tempcovmat,tempmat4,xmat)
End Do


        !Factor covariance structure of polynomial coefficient
Do k=1,covariates
        Call DCHFAC (degree(k)+1,covmat(1:degree(k)+1,1:degree(k)+1,k),&
        &degree(k)+1, .00001*DMACH(4),&
        & piv, faccov(1:degree(k)+1,1:degree(k)+1,k), degree(k)+1)

End Do

Call ERSET(0,1,1)

        ! Compute lambdas
lower=1.0
upper=29.0
errabs=0.01
errrel=0.01
irule=2
Do funcnumb=1,covariates
        Call DQDAG(NGAMPR,lower,upper,errabs,errrel,irule,&
        &caplambda(funcnumb),errest)
End Do

df = subjects - (degree(1)+degree(2)+degree(3)+degree(4)+covariates)
        !Find critical values
Do funcnumb=1,covariates
        beg = 0.5
        finish = 10.0
        aberr = 0.0
        relerr = 0.01
        maxf = 100
        Call DZBREN(COVPROB,aberr,relerr,beg,finish,maxf)
        critval(funcnumb)=finish
End Do

        !Build proposed bands and set up averages
Do funcnumb=1,covariates
```

```
        Allocate(basis(degree(funcnumb)+1))
        Allocate(tempvec(degree(funcnumb)+1))
        Do g=scale,endgrid-scale
                Do power=1,degree(funcnumb)+1
                        basis(power)=s(g)**(power-1)
                End Do
                estimate=DOT_PRODUCT&
                &(estcoef(1:degree(funcnumb)+1,funcnumb),basis)
                tempvec=MATMUL&
                &(faccov(1:degree(funcnumb)+1,1:degree(funcnumb)+1,funcnumb),basis)
                temperr=DOT_PRODUCT(tempvec,tempvec)
                stderr=DSQRT(temperr)
                errterm=critval(funcnumb)*DSQRT(sigsq)*stderr
                myupper(g,funcnumb)=estimate+errterm
                mylower(g,funcnumb)=estimate-errterm
        End do
        Deallocate(basis,tempvec)
End Do

        !Build Bonferroni bands
bonalpha=.025/(1.0*(endgrid-2*scale+1))
prcentile=1.0-bonalpha
zscore=DNORIN(prcentile)
Do funcnumb=1,covariates
        Do g=scale,endgrid-scale
          bonerr=(1.0*total*hcv)**(-1)
          bonerr=zscore*DSQRT(bonerr*sigsq*Dstr(funcnumb,funcnumb,g,g))
          bonupper(g,funcnumb)=b(g,funcnumb)+bonerr
          bonlower(g,funcnumb)=b(g,funcnumb)-bonerr
        End Do
End Do

cover0=1
cover1=1
cover2=1
cover3=1
boncover0=1
boncover1=1
boncover2=1
boncover3=1

        !Check coverage
Do g=scale,endgrid-scale
        actual=BETA0(s(g))
        If ((actual>myupper(g,1)) .OR. (actual<mylower(g,1))) Then
```

```
                cover0=0
        End IF
        If ((actual>bonupper(g,1)) .OR. (actual<bonlower(g,1))) Then
                boncover0=0
        End IF

        actual=BETA1(s(g))
        If ((actual>myupper(g,2)) .OR. (actual<mylower(g,2))) Then
                cover1=0
        End IF
        If ((actual>bonupper(g,2)) .OR. (actual<bonlower(g,2))) Then
                boncover1=0
        End IF

        actual=BETA2(s(g))
        If ((actual>myupper(g,3)) .OR. (actual<mylower(g,3))) Then
                cover2=0
        End IF
        If ((actual>bonupper(g,3)) .OR. (actual<bonlower(g,3))) Then
                boncover2=0
        End IF

        actual=BETA3(s(g))
        If ((actual>myupper(g,4)) .OR. (actual<mylower(g,4))) Then
                cover3=0
        End IF
        If ((actual>bonupper(g,4)) .OR. (actual<bonlower(g,4))) Then
                boncover3=0
        End IF
End Do

cover0b=1
cover1b=1
cover2b=1
cover3b=1
Do g=2*scale,endgrid-2*scale
        actual=BETA0(s(g))
        If ((actual>myupper(g,1)) .OR. (actual<mylower(g,1))) Then
                cover0b=0
        End IF

        actual=BETA1(s(g))
        If ((actual>myupper(g,2)) .OR. (actual<mylower(g,2))) Then
                cover1b=0
        End IF
        actual=BETA2(s(g))
        If ((actual>myupper(g,3)) .OR. (actual<mylower(g,3))) Then
                cover2b=0
        End IF

        actual=BETA3(s(g))
        If ((actual>myupper(g,4)) .OR. (actual<mylower(g,4))) Then
                cover3b=0
        End IF
End Do


covertot0=covertot0+cover0
covertot1=covertot1+cover1
covertot2=covertot2+cover2
covertot3=covertot3+cover3

covertot0b=covertot0b+cover0b
covertot1b=covertot1b+cover1b
covertot2b=covertot2b+cover2b
covertot3b=covertot3b+cover3b


boncovertot0=boncovertot0+boncover0
boncovertot1=boncovertot1+boncover1
boncovertot2=boncovertot2+boncover2
boncovertot3=boncovertot3+boncover3

myuppertot=myuppertot+myupper
mylowertot=mylowertot+mylower
bonuppertot=bonuppertot+bonupper
bonlowertot=bonlowertot+bonlower


Write (*,*) iteration

iteration=iteration+1

100 Deallocate (errors,err)

End Do !iteration

        !Compute coverage probability
mycovprob0=DBLE(covertot0)/DBLE(repititions)
```

```
mycovprob1=DBLE(covertot1)/DBLE(repititions)
mycovprob2=DBLE(covertot2)/DBLE(repititions)
mycovprob3=DBLE(covertot3)/DBLE(repititions)

mycovprob0b=DBLE(covertot0b)/DBLE(repititions)
mycovprob1b=DBLE(covertot1b)/DBLE(repititions)
mycovprob2b=DBLE(covertot2b)/DBLE(repititions)
mycovprob3b=DBLE(covertot3b)/DBLE(repititions)

boncovprob0=DBLE(boncovertot0)/DBLE(repititions)
boncovprob1=DBLE(boncovertot1)/DBLE(repititions)
boncovprob2=DBLE(boncovertot2)/DBLE(repititions)
boncovprob3=DBLE(boncovertot3)/DBLE(repititions)

        !Compute average bands
avgupper=(1.0/DBLE(repititions))*myuppertot
avglower=(1.0/DBLE(repititions))*mylowertot

avgbonupper=(1.0/DBLE(repititions))*bonuppertot
avgbonlower=(1.0/DBLE(repititions))*bonlowertot


        !Write results
Open(1, FILE='test.dat',STATUS='replace')
Do g=scale,endgrid-scale
        actual=BETA0(s(g))
                Write (1, 10)
s(g),avgbonlower(g,1),avglower(g,1),actual,avgupper(g,1),avgbonupper(g,1)
                10 Format(F8.4,F8.4,F8.4,F8.4,F8.4,F8.4)
End Do
Close(1)

Open(2, FILE='test2.dat',STATUS='replace')
Do g=scale,endgrid-scale
        actual=BETA1(s(g))
                Write (2, 11)
s(g),avgbonlower(g,2),avglower(g,2),actual,avgupper(g,2),avgbonupper(g,2)
                11 Format(F8.4,F8.4,F8.4,F8.4,F8.4,F8.4)
End Do
Close(2)

Open(3, FILE='test3.dat',STATUS='replace')
Do g=scale,endgrid-scale
        actual=BETA2(s(g))
```

```
                Write (3, 12)
s(g),avgbonlower(g,3),avglower(g,3),actual,avgupper(g,3),avgbonupper(g,3)
                12 Format(F8.4,F8.4,F8.4,F8.4,F8.4,F8.4)
End Do
Close(3)

Open(4, FILE='test4.dat',STATUS='replace')
Do g=scale,endgrid-scale
        actual=BETA3(s(g))
                Write (4, 13)
s(g),avgbonlower(g,4),avglower(g,4),actual,avgupper(g,4),avgbonupper(g,4)
                13 Format(F8.4,F8.4,F8.4,F8.4,F8.4,F8.4)
End Do
Close(4)

Open(5, FILE='test5.dat',STATUS='replace')
                Write (5, 14) mycovprob0,mycovprob1,mycovprob2,mycovprob3
                Write (5, 14) mycovprob0b,mycovprob1b,mycovprob2b,mycovprob3b
                Write (5, 14) boncovprob0,boncovprob1,boncovprob2,boncovprob3
                14 Format(F8.4,F8.4,F8.4,F8.4)
Close(5)

End Program



!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Subroutines and functions!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        !Coefficient functions
Double Precision Function BETA0(s)
        Implicit None
        Double Precision, Intent(IN) :: s
        Double Precision, Parameter :: pi=3.14159265359
        BETA0 = 15.0 + 20.0*DSIN(s*pi/60.0)
End Function

Double Precision Function BETA1(s)
        Implicit None
        Double Precision, Intent(IN) :: s
        BETA1 = 4.0 - ((s - 15.0)/9.0)**2
End Function

Double Precision Function BETA2(s)
        Implicit None
        Double Precision, Intent(IN) :: s
        Double Precision, Parameter :: pi=3.14159265359
        BETA2 = 2.0 + 1.25*DSIN(s*pi/15.0)
End Function

Double Precision Function BETA3(s)
        Implicit None
        Double Precision, Intent(IN) :: s
        BETA3 = DEXP(0.05*s)
End Function
        !End of coefficient functions

        !Derivative of  Naiman's gamma
Double Precision Function NGAMPR(x2)
        Implicit None
        Double Precision, Intent(IN) :: x2
        Integer :: funcnumb
        Common / fcnindex / funcnumb
        Integer, Dimension(4) :: degree
        Common / powers / degree
        Integer d, korder
        Common / location / d
        Double Precision :: bgstep,tol,temp
        Double Precision, Allocatable, Dimension(:) :: gamderiv
        Double Precision, External :: GAM
        Double Precision DDERIV

        Allocate(gamderiv(degree(funcnumb)+1))
        korder=1
        bgstep=0.1
        tol=0.05
        Do d=1,degree(funcnumb)+1
                gamderiv(d)=DDERIV(GAM,korder,x2,bgstep,tol)
        End Do
        temp=DOT_PRODUCT(gamderiv,gamderiv)
        NGAMPR=DSQRT(temp)
        Deallocate(gamderiv)
End Function

        !Naiman's gamma
Double Precision Function GAM(x2)
        Implicit None
        Double Precision, Intent(IN) :: x2
        Integer :: funcnumb
        Common / fcnindex / funcnumb
        Integer, Dimension(4) :: degree
        Common / powers / degree
        Double Precision, Dimension(10,10,4) :: faccov
        Common / factmat / faccov
        Integer d,deg
        Common / location / d
        Double Precision, Allocatable, Dimension(:) :: tempgam,basis
        Double Precision tempdot,tempnorm

        Allocate(basis(degree(funcnumb)+1),tempgam(degree(funcnumb)+1))
        Do deg=1,degree(funcnumb)+1
                basis(deg)=x2**(deg-1)
        End Do
        tempgam=MATMUL&
        &(faccov(1:degree(funcnumb)+1,1:degree(funcnumb)+1,funcnumb),basis)
        tempdot=DOT_PRODUCT(tempgam,tempgam)
        tempnorm=DSQRT(tempdot)
        tempgam=(1.0/tempnorm)*tempgam
        GAM=tempgam(d)
        Deallocate(basis,tempgam)
End Function

        !Compute Naiman's integral
Double Precision Function COVPROB(c)
        Implicit None
        Double Precision, Intent (IN) :: c
        Double Precision, External :: NAIMARG
        Double Precision :: low,high,errab,relerr,cover,esterr
        Common / critical / high
        Integer :: rule
        rule=2
        low=0.0
        high=1.0/c
        errab=0.01
        relerr=0.01
        Call DQDAG(NAIMARG,low,high,errab,relerr,rule,cover,esterr)
        COVPROB=.05-cover
End Function

        !Argument in Naiman's integral
Double Precision Function NAIMARG(x3)
        Implicit None
        Integer, Parameter :: covariates=4
        Double Precision, Parameter :: pi=3.14159265359
        Double Precision, Intent(IN) :: x3
        Double Precision :: high, c
        Common / critical / high
        Integer :: df
        Common / degfree / df
        Integer :: funcnumb
        Common / fcnindex / funcnumb
        Integer, Dimension(4) :: degree
        Common / powers / degree
        Double Precision, Dimension(covariates) :: caplambda
        Common / arclength / caplambda
        Double Precision :: temp1,temp2,temp3,temp4,sum1,const
        Double Precision :: tempa,tempb,tempc,tempd,tempe,tempf
        Double Precision :: tot1,tot2,DGAMMA,DFDF
        c=1.0/high
        temp1=2.0*((c*x3)**(-2)-1.0)/DBLE(degree(funcnumb)+1-2.0)
        temp2=DFDF(temp1,&
        &DBLE(degree(funcnumb)+1-2),DBLE(2.0))*caplambda(funcnumb)/pi
        temp3=((c*x3)**(-2)-1.0)/DBLE(degree(funcnumb)+1-1.0)
        temp4=DFDF(temp3,DBLE(degree(funcnumb)+1-1),DBLE(1.0))
        sum1=temp3+temp4
        If (sum1 > 1.0) Then
                const=1.0
        Else
                const=sum1
        End If

        tempa=DLOG(DGAMMA(DBLE(df+degree(funcnumb)+1)/2.0))
        tempb=DLOG(DGAMMA(DBLE(df)/2.0))
        tempc=DLOG(DGAMMA(DBLE(degree(funcnumb)+1)/2.0))
        tempd=(DBLE(df)/2.0)*DLOG(DBLE(df))
        tempe=DBLE(df-1)*DLOG(x3)
        tempf=-(DBLE(df+degree(funcnumb)+1)/2.0)*DLOG(1.0+DBLE(df)*(x3**2))
        tot1=tempa-tempb-tempc+tempd+tempe+tempf+DLOG(DBLE(2.0))
        tot2=DEXP(tot1)
        NAIMARG=const*tot2
End Function

        !Cross-validation criteria
Double Precision Function CVCRIT(h)
        Implicit None
        Double Precision, Intent(IN) :: h
        Integer, Parameter :: covariates=4, subjects=150, measurements=30
        Integer, Dimension(subjects) :: m
        Double Precision, Dimension(measurements+1,subjects) :: t
        Double Precision, Dimension(measurements,covariates,subjects) :: design
        Double Precision, Dimension(measurements,subjects) :: y
        Common / observations / m, t, design, y
```

```
    Double Precision, Allocatable, Dimension(:,:) :: kermatrix, jmat
    Double Precision, Dimension(covariates,1) :: sum1,tempsum1,temp1,temp2
    Double Precision :: tempx,res,temp3,DBLINF
    Double Precision, Dimension(covariates,covariates) :: amat, a_noti_inv
    Integer i,j,k
    temp3=0
    Do i=1, subjects
            Do j=1, m(i)
                    sum1=0
                    Do k=1, subjects
                    Allocate (kermatrix (m(k),m(k)), jmat (covariates, m(k)))
                    Call KERNBUILD(m(k), t(j,i), measurements, t(:,k), h, kermatrix)
                    Call DMXTYF(m(k), covariates, design(1:m(k),:,k), m(k), m(k),&
                            & m(k), kermatrix, m(k), covariates, m(k), jmat, covariates)
                    tempsum1=0
                    Call DMRRRR(covariates, m(k), jmat, covariates, m(k), 1,&
                            & y(1:m(k),k), m(k), covariates, 1, tempsum1, covariates)
                                    sum1=sum1 + tempsum1
                                    Deallocate (kermatrix, jmat)
                            End Do !k
                    Allocate (kermatrix (m(i),m(i)), jmat (covariates, m(i)))
                    jmat=0
                    temp1=0
                    temp2=0
                    Call KERNBUILD(m(i), t(j,i), measurements, t(:,i), h, kermatrix)
                    Call DMXTYF(m(i), covariates, design(1:m(i),:,i), m(i), m(i),&
                            & m(i),kermatrix, m(i), covariates, m(i), jmat, covariates)
                    Call DMRRRR(covariates, m(i), jmat, covariates, m(i), 1,&
                            & y(1:m(i),i), m(i), covariates, 1, temp1, covariates)
                    temp2=sum1-temp1

                    Call BUILDA(subjects, covariates, measurements, m, design,&
                            & t(j,i), t, h, amat)
                    Call BUILDINV(t(j,i), h,covariates,m(i),amat,design(1:m(i),:,i),&
                            & t(1:m(i),i), a_noti_inv)
                    tempx=DBLINF(covariates, covariates, a_noti_inv, covariates,&
                            & design(j,:,i), temp2)
                    res=(y(j,i)-tempx)**2
                    temp3=temp3+res
                    Deallocate (kermatrix, jmat)
            End Do !j
    End Do !i
    CVCRIT=temp3
End Function
```

```
        !Kernel matrix builder
Subroutine KERNBUILD(a, time, measurements, mespts, h, kernel)
        Implicit None
        Integer :: i
        Integer, Intent(IN) :: a, measurements          !a is the dimension of kernel matrix
        Double Precision, Intent(IN) :: time, h         !time=current time point
        Double Precision, Dimension(measurements+1) :: mespts
        Double Precision, Dimension(a,a), Intent(OUT) :: kernel     !kernel mat.
        Double Precision :: argmt, c
        Double Precision :: pi=3.14159265359
        kernel=0
        Do i=1, a
                argmt=(time-mespts(i))/h
                c=1.0/SQRT(2*pi)
                kernel(i,i)=c*DEXP(-.5*(argmt**2))
        End Do
End Subroutine

        !Matrix A as defined by Hoover
Subroutine BUILDA(subjects, covariates, measurements, m, design, time, t, h, amat)
        Implicit None
        Integer :: i
        Integer, Intent(IN) :: subjects, covariates, measurements
        Integer, Dimension(subjects), Intent(IN) :: m
        Double Precision, Dimension(measurements,covariates,subjects),&
                &Intent(IN) :: design
        Double Precision, Intent(IN) :: time, h
        Double Precision, Dimension(measurements+1,subjects), Intent(IN) :: t
        Double Precision, Dimension(covariates,covariates), Intent(OUT) :: amat
        Double Precision, Allocatable, Dimension(:,:) :: temp, kernel
        Double Precision, Dimension(covariates,covariates) :: temp2

        amat=0
        temp2=0
        Do i=1, subjects
                Allocate (kernel(m(i),m(i)))
                Allocate (temp(covariates,m(i)))
                Call KERNBUILD(m(i), time, measurements, t(:,i), h, kernel)
                Call DMXTYF(m(i),covariates,design(1:m(i),:,i), m(i), m(i), m(i),
kernel,&
                        & m(i), covariates, m(i), temp, covariates)
                Call DMRRRR( covariates, m(i), temp, covariates, m(i),covariates,&
                        & design(1:m(i),:,i), m(i), covariates, covariates, temp2,
covariates)
                amat=amat+temp2
                Deallocate(kernel)
                Deallocate(temp)
        End Do
End Subroutine

Subroutine BUILDB(subjects, covariates, measurements, m, design, time, t, h, y, bmat)
        Implicit None
        Integer :: i, index
        Integer, Intent(IN) :: subjects, covariates, measurements
        Integer, Dimension(subjects), Intent(IN) :: m
        Double Precision, Dimension(measurements,covariates,subjects),&
                & Intent(IN) :: design
        Double Precision, Intent(IN) :: time, h
        Double Precision, Dimension(measurements+1,subjects), Intent(IN) :: t
        Double Precision, Dimension(measurements,subjects), Intent(IN) :: y
        Double Precision, Dimension(covariates), Intent(OUT) :: bmat
        Double Precision, Allocatable, Dimension(:,:) :: temp, kernel
        Double Precision, Dimension(covariates) :: temp2
        index=1
        bmat=0
        temp2=0
        Do i=1, subjects
                Allocate (kernel(m(i),m(i)))
                Allocate (temp(covariates,m(i)))
                Call KERNBUILD(m(i), time, measurements, t(:,i), h, kernel)
                Call DMXTYF(m(i),covariates,design(1:m(i),:,i), m(i), m(i), m(i),
kernel,&
                        & m(i), covariates, m(i), temp, covariates)
                Call DMURRV(covariates,m(i), temp, covariates, m(i), y(1:m(i),i),
index,&
                        & covariates, temp2)
                bmat=bmat+temp2
                Deallocate(kernel)
                Deallocate(temp)
        End Do
End Subroutine

        !Build inverse of A
        !!!Call as BUILDINV(t(j,i), h, covariates, m(i), amat, design(1:m(i),:,i),
t(1:m(i),i),
        !!!a_noti_inv)
Subroutine BUILDINV(time, h, covariates, m, amat, subjmat, subjtimes, a_noti_inv)
        Implicit None
        Double Precision, Intent(IN) :: time, h
        Integer, Intent(IN) ::covariates, m
        Double Precision, Dimension(covariates,covariates), Intent(IN) :: amat
        Double Precision, Dimension(m, covariates), Intent(IN) :: subjmat
        Double Precision, Dimension(m), Intent(IN) :: subjtimes
        Double Precision, Dimension(covariates,covariates), Intent(OUT) :: a_noti_inv
        Double Precision, Dimension(m,m) :: kernel
        Double Precision, Dimension(covariates, m) :: temp
        Double Precision, Dimension(covariates, covariates) :: temp2, temp3

        Call KERNBUILD(m, time, m, subjtimes, h, kernel)
        Call DMXTYF(m,covariates,subjmat,m,m,m,kernel,m,&
                &covariates,m,temp,covariates)
        Call DMRRRR(covariates, m, temp, covariates, m, covariates, subjmat, m,&
                & covariates, covariates, temp2, covariates)
        temp3 = amat - temp2
        Call DLINRG(covariates, temp3, covariates, a_noti_inv, covariates)
End Subroutine
```

VITA

James Edward Blum

Candidate for the Degree of

Doctor of Philosophy

Thesis: CONSERVATIVE CONFIDENCE REGIONS FOR KERNEL ESTIMATES OF THE VARYING COEFFICIENT MODEL

Major Field: Statistics

Biographical:

Personal Data: Born in Paw Paw, Michigan, On April 10, 1970, the son of Robert and Jo Ann Blum.

Education: Graduated from Lawton High School, Lawton, Michigan in May 1988. Received a Bachelor of Arts degree in Education from the University of Michigan, Ann Arbor, Michigan in August 1992. Received a Master of Science degree in Mathematics from Oklahoma State University, Stillwater, Oklahoma in July 1998. Completed the requirements for the Doctor of Philosophy degree with a major in Statistics at Oklahoma State University in December, 2000.

Experience: Employed as an instructor by Southwestern Michigan College from August 1992 to August 1995. Employed by Oklahoma State University as a teaching assistant in both the Department of Mathematics and the Department of Statistics, August 1995 to present.

Professional Membership: American Statistical Association.