MULTISCALE MULTIVARIATE FUNCTIONAL PRINCIPAL COMPONENT ANALYSIS WITH AN APPLICATION TO MULTIVARIATE LONGITUDINAL CARDIAC SIGNALS

by

Andrew N. Potter

B.A., Cornell University, 2007

Submitted to the Graduate Faculty of

the Graduate School of Public Health in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2016

UNIVERSITY OF PITTSBURGH

GRADUATE SCHOOL OF PUBLIC HEALTH

This dissertation was presented

by

Andrew N. Potter

It was defended on

December 2nd, 2016

and approved by

Stewart J. Anderson, PhD, Professor, Department of Biostatistics, Graduate School of Public Health, University of Pittsburgh

Robert T. Krafty, PhD, Associate Professor, Department of Biostatistics, Graduate School of Public Health, University of Pittsburgh

Ying Ding, PhD, Assistant Professor, Department of Biostatistics, Graduate School of Public Health, University of Pittsburgh

Kehui Chen, PhD, Assistant Professor, Department of Statistics, Dietrich School of Arts and Sciences, University of Pittsburgh

Jeffery J. Teuteberg, MD, Associate Professor of Medicine, Heart and Vascular Institute, University of Pittsburgh Medical Center

Dissertation Director: Stewart J. Anderson, PhD, Professor, Department of Biostatistics, Graduate School of Public Health, University of Pittsburgh

MULTISCALE MULTIVARIATE FUNCTIONAL PRINCIPAL COMPONENT ANALYSIS WITH AN APPLICATION TO MULTIVARIATE LONGITUDINAL CARDIAC SIGNALS

Andrew N. Potter, PhD

University of Pittsburgh, 2016

Abstract

Circadian cycles in humans are an important health indicator in cardiovascular disease. With recent developments in ventricular assist devices (VADs), continuous recording of cardiac circadian cycles in cohorts of heart failure patients is now possible for the entire life of the implant. Specifically, VADs continuously record multivariate data on blood flow and device status providing a unique longitudinal view of circadian cycles in these cohorts.

Our statistical challenge is to simultaneously model the cohort average pump output (PO) and pulsatility (PI) circadian cycle measurements and patient specific longitudinal evolution of his/her circadian cycle. While functional principal components analysis (FPCA) methods exist for the analysis of univariate longitudinal functional data with this structure, these techniques do not address bivariate functional data.

We first divide time into two time scales: "fast" (circadian) and "slow" (longitudinal). We assume that the data are generated by smooth functions of time and extend FPCA to include both time scales. Use of a marginal model separates the estimation and inference for the two time scales. On the circadian time scale, we use wavelet based FPCA to estimate the cohort mean cycle and subject specific cycles. Confidence bands for the cohort mean and other estimates are calculated with a bootstrap. On the longitudinal time scale, a second FPCA step captures the subject specific longitudinal evolution. Furthermore, using data from VAD patients, we implement our method to characterize the population circadian

cycle and identify regions of high between-subject variability in both the fast and slow time scales.

Our model provides a novel approach for analyzing multivariate circadian cycles. This work opens new avenues to understand the relationship between circadian cycles in simultaneously recorded cardiovascular measurements. The public health significance is that care can be improved with better understanding of the longitudinal course of these patients.

Keywords: Functional Data Analysis, Discrete Wavelet Transformation, Marginal Covariance Kernel, Physiological Signal Analysis, Circadian Cycle, Chronobiology.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

As with any large project or anything worth doing, my dissertation research has been a long and winding journey. The completion of this journey would not have been possible without the support of those around me. First I would like to thank my guide on this journey, my advisor, Dr. Stewart Anderson, who has provided essential support at every step along the way. My committee members, Dr. Robert Krafty, Dr. Jeffrey Teuteberg, Dr. Ying Ding, and Dr. Kehui Chen, have each provided expert guidance to improve the quality of my dissertation. I would also like to thank Dr. Jonathan Holtz, Dr. Robert Kormos, and Dr. Jeffrey Teuteberg who each provided the clinical touch to my dissertation and keep my research grounded in clinical relevance. One of the joys of a statistician's job is the ability to collaborate with people with different backgrounds and viewpoints. I would like to provide a special thanks to Dr. Jonathon Holtz for providing the clinical data and motivations for which this project would not have been possible. I enjoyed developing as a student with him and exploring the possibilities of medical research. I would like to thank Dr. Joyce Chang for providing guidance during teaching and providing an area that makes it not only possible to meet with clinicians, but to form lasting collaborations with doctors.

I would also like to thank for emotional support of my family and friends who provided essential support during the process. My parents provided warm meals and an essential break when I felt overwhelmed during this process. The last person I would like to thank my fiancé, Michelle, for providing me with support through the process, especially copy editing my dissertation, and helping me achieve my goals.

1.0 INTRODUCTION

Multivariate cardiac signals measured over a circadian (daily) cycle are frequently encountered in the study of the long term health of heart failure patients. In previous chronobiological studies, both blood pressure and heart rate show circadian variation in healthy humans (Millar-Craig et al., 1978; Lombardi et al., 1992; van de Borne et al., 1992; Takeda and Maemura, 2011). In cardiovascular disease (CVD), including all types of CVD from mild hypertension to end stage heart failure (HF) and myocardial infarction (MI), disruption of the circadian cycle is associated with both increased risk of CVD as well as a symptom of CVD itself (Millar-Craig et al., 1978; Lombardi et al., 1992; Takeda and Maemura, 2011). During recovery from a MI, the amplitude of circadian variation in heart rate is depressed (Lombardi et al., 1992). The same pattern is seen in both heart rate and blood pressure circadian cycles in HF patients. Van de Borne et al. (1992) demonstrate that a blood pressure circadian cycle returns to normal in heart transplant patients within seven months post transplant. Due to the recent increase in use of left ventricular assist devices (VADs), a need for understanding the circadian cycles in these patients has arisen.

A VAD is a life-saving medical device consisting a pump implanted in a patient's chest (Slaughter et al., 2010). Blood flows into the pump through an inflow cannula in the left ventricle and out into the ascending aorta and to the rest of the body. Blood is pumped using a continuously spinning impeller pump. The blood flow depends on both the pump speed and the patients heart function.

With the appearance of continuous logging of multiple device and hemodynamic parameters (power, pump output [PO], pulsatility [PI], speed) in modern VADs, the ability to simultaneously study the daily evolution of the circadian cycle in these variables becomes possible. The two variables of clinical interest here are PO and PI. PO is a measure of

blood flow through the VAD. PI is a measure of blood flow variability. In patients with a VAD, both the characteristic shape and longitudinal evolution of the circadian cycles in PO, PI, and power are hypothesized to be important markers of long term health (Slaughter et al., 2010; Suzuki et al., 2014). However, no studies have yet linked this circadian variation to health outcomes. In one of the earliest studies of circadian cycles in VADs, Slaughter et al. (2010) shows that a circadian cycle in VAD power is present in VAD patients by end the of the first month post implant. Suzuki et al. (2014) report monthly variation in VAD power circadian cycle parameters such as amplitude and phase. These studies have analyzed changes in the circadian cycle by comparing an estimated parameter or sample means across several time points. Despite this recent research, the clinical meaning of these cycles in VAD patients is poorly understood.

As part of a larger study of circadian waveform behavior and its continuous longitudinal evolution in VAD patients, we propose a new functional data analysis method, multivariate multiscale functional principal component analysis (MMFPCA). Analyzing a small pilot dataset, our goal is to formulate answers to the clinical questions about a cohort of patients with VAD data such as: "Does a circadian cycle exist for VAD patients?" "If so, what is its shape?" "How do PO and PI jointly vary across a day?" Our pilot data consist of VAD clinical log files for nine patients sampled from a larger database at a major academic medical center. Each log file contains bivariate measurements of PO and PI recorded every 15 minutes for the entire life of the VAD implant. Data are downloaded at clinic visits but contains missing periods as data is only stored for 30 days before being over written. In our cohort, the length of follow-up ranged from 20 to 79 days after gaps were excluded.

Example data from a single patient are presented in Figure 1. This figure shows 20 days of data on the PO and PI waveforms with visually strong and correlated circadian cycles. For this patient, the daily minimum of PO is about 5000 mL/min, and the daily maximum is over 7000 mL/min. Similarly, the PI circadian variation ranges from 4000 mL to 9000 mL. Appendix A contains plots of the remaining eight patients. Circadian variation is seen in the other patients.

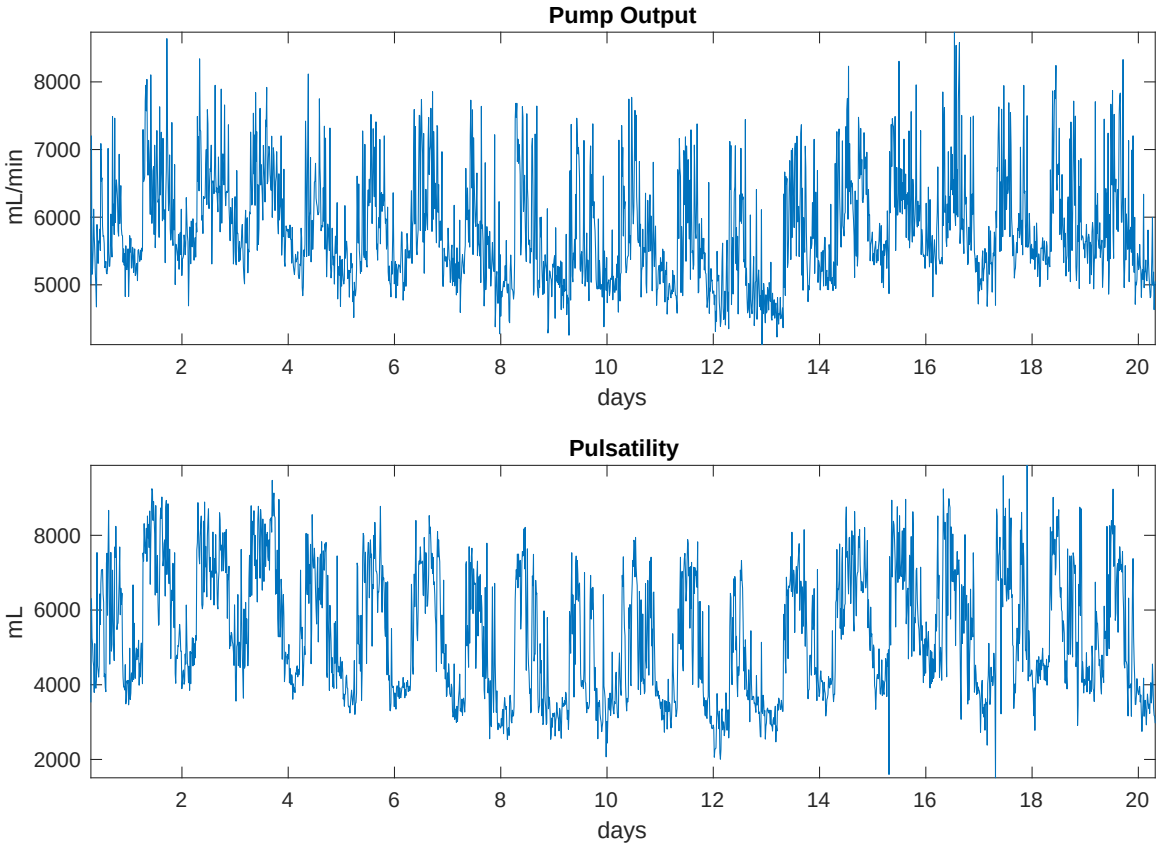

Figure 1: Example VAD log file data.

Sample profile for 20 days of pump output and pulsatility for a representative patient.

Traditionally, the data are often analyzed using spectral or frequency domain methods such as periodograms and extensions. In the study of circadian cycles (chronobiology), the cosinor model, a parametric model that assumes the circadian cycle takes on a cosine shape, is often used. An excellent review by Refinetti et al. (2007) covers both the spectral analysis and cosinor approach in depth. While frequency domain approaches are briefly covered, this dissertation focuses mainly on the analysis of the functional aspects of our motivating data.

Our statistical approach begins by viewing the VAD data as noisy observations on multivariate functions of the two scales of time, the important features of these data occur over two independent time scales: the circadian cycle defining a "fast" time scale, t, and its longitudinal evolution defining a "slow" time scale s. However, the data are originally collected as a function, $\mathscr{F}(\tau)$, of a single time variable, τ, requiring a transformation onto the two time scales, (t, s). We introduce a novel statistical framework to jointly model both fast and slow time scales and the multivariate structure by modeling $\mathscr{F}(\tau)$ using a function $\boldsymbol{F}(t)$ for the fast time cycles that is repeatedly observed over a range of days. The longitudinal evolution over s is modeled with a function $\boldsymbol{G}(s)$. This multiple scale approach was motivated by the two time solution method used in solving non-linear ordinary differential equations (ODE) (Fink et al., 1974; Strogatz, 1994). This framework formalizes the intuitively appealing approach dividing a continuously recorded signal into one period long blocks.

Instead of the challenging estimation of $\mathscr{F}(\tau)$ for the entire cohort, the multiple time scale changes the estimation problem to the tractable analysis of repeated observations of $\boldsymbol{F}(t)$. Several recent papers address the modeling of repeatedly observed functions. These methods are based on the functional extension of principal component analysis, FPCA, which models random functions using an empirical basis function expansion. The longitudinal functional principal component analysis (LFPCA), introduced by Greven et al. (2010), models the repeatedly observed functions using the FPCA equivalent of a linear mixed model, the longitudinal evolution is required to be linear. Park and Staicu (2015) extend LFPCA to the case when the longitudinal evolution follows an unknown function. This approach is effective when the longitudinal follow-up information is sparsely observed. LFPCA is one specific model that uses the marginal FPCA approach discussed in Chen et al. (2016). Another approach is repeated function functional principal component analysis (RF-FPCA)

introduced by Chen and Müller (2012). RF-FPCA models the univariate repeated functional observations by correlated conditional FPCA for each time unit of longitudinal follow-up. RF-FPCA also uses a non-parametric form for the effect of longitudinal time.

In order to model our VAD data, LFPCA and RF-FPCA must be both extended to multivariate observations that are densely sampled on both fast and slow time scales. Accordingly, our proposed method is inspired by RF-FPCA's approach for dense longitudinal data and LFPCA's marginal decomposition of the covariance structure (Chen et al., 2016; Park and Staicu, 2015; Greven et al., 2010). The marginal covariance model is used as a starting point as it easily extends to the multivariate case, unlike the conditional covariance model used in RF-FPCA.

1.1 RESEARCH QUESTIONS

The following research aims will be addressed in this dissertation:

1. Develop a multivariate functional principal component analysis (MFPCA) to analyze the daily cyclic and long term behavior of a population of VAD patients when all patients show a daily cycle.
2. Characterize the performance of the new method using large sample theory and finite sample simulation studies.
3. Analyze the motivating VAD cohort with the new technique.

2.0 EXPLORATORY DATA ANALYSIS OF THE VENTRICULAR ASSIST DEVICE CIRCADIAN PATTERNS

In this chapter, we present results from an exploratory data analysis of the waveforms seen in the ventricular assist device (VAD) data presented in Figure 1. Throughout this chapter, consider the sample grid for each patient τ_ℓ with $\ell = 1, 2, \ldots, n \times J_i$. Here, n is the total number of samples per day and J_i is the numbers of that each patient i is observed. Our exploration of the patterns in the VAD clinical log files consists of using periodograms to search for signals with a period of one day. Also, we break both PO and PI into day long blocks. Then, surface plots of the raw data aid the visualization of any circadian pattern, and its longitudinal evolution. In addition, both the daily marginal patient specific means and the marginal patient specific longitudinal evolution are examined.

2.1 PERIODOGRAM ANALYSIS OF THE VAD DATA

Because one of the motivating clinical questions is to characterize the presence of a circadian pattern in the VAD patient population, we use a periodogram to examine the strength of any periodic signal in each patients data. At each frequency, ω_g, a periodogram $\hat{\mathcal{P}}_i(\omega_g)$ represents the sample amplitudes of sine and cosine functions oscillating at ω_g (Shumway and Stoffer, 2011). If a patient has a circadian cycle, a strong peak is expected a 1 cycle/day.

Unless the circadian cycle is represented by a pure cosine signal, peaks at the higher order harmonic frequencies will also be present. A harmonic frequency is any frequency that is an integer multiple of fundamental frequency, 1 cycle/day, in our case. Visual examination of the harmonics aids in deciding if spectral analysis of repeated functional data analysis is appropriate for out VAD data.

As $\hat{\mathcal{P}}_i(\omega_g)$ is an inconsistent estimate of the power spectrum, a smoothed estimate is needed. In our case, we used a Welch periodogram, which smooths $\hat{\mathcal{P}}_i(\omega_g)$ by sub-setting $\mathscr{F}_i(\tau_\ell)$ into non-overlapping blocks (Welch, 1967). Then a periodogram is estimated for each block. Finally, these are averaged together yielding a smoothed estimate.

The Welch periodograms of PO and PI are shown in Figure 2 for a representative patient. Additional periodograms for the other eight patients can be found in Appendix B. In Figure 2, a peak at 1 cycle/day in both PO and PI shows that both PO and PI have a circadian cycle. As the periodogram contains significant peaks at harmonics of 1 cycle/day, the circadian cycle has a shape that cannot be accurately captured as a linear combination of only sine and cosine functions with daily periods. Additionally, an increase of power (≈ 80 dB) in the very low frequency end of the spectrum indicates that the random process is likely non-stationary. The other eight patients show a similar pattern.

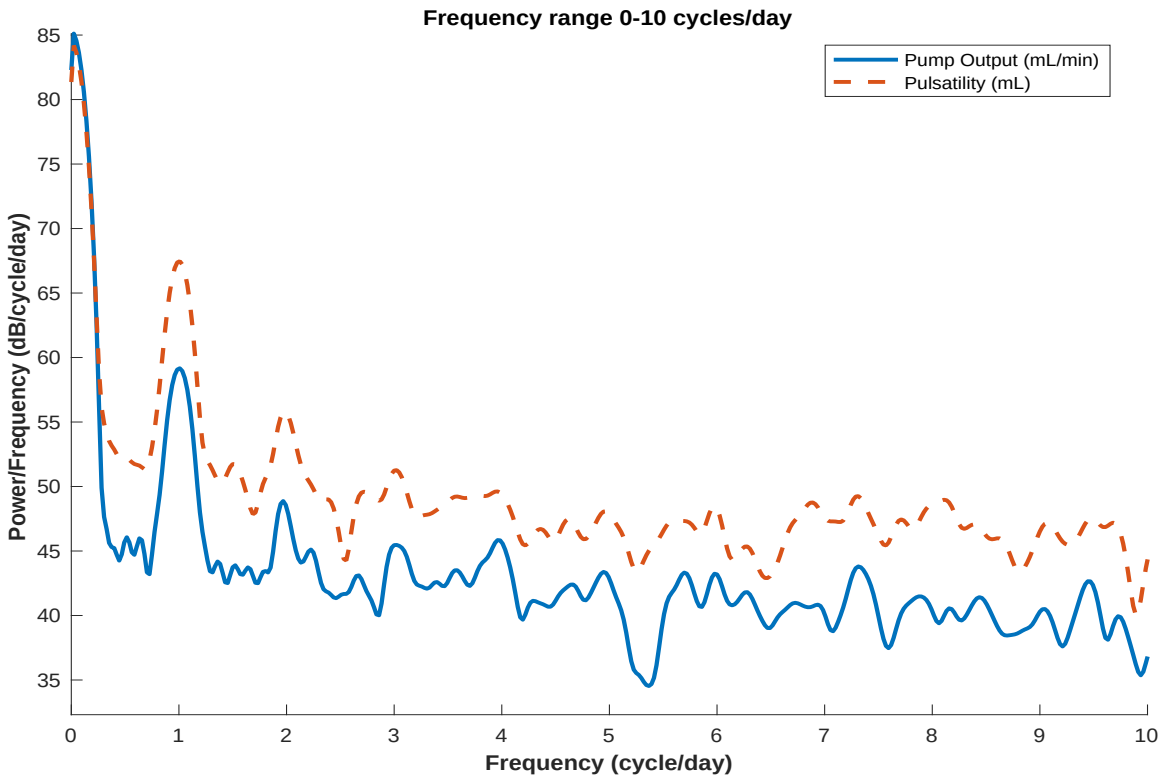

Figure 2: Welch Periodogram from a representative patient showing circadian behavior.

2.2 TIME DOMAIN EXPLORATORY DATA ANALYSIS

As each patient has a circadian cycle, the data is now split into non-overlapping day long blocks. Figure 3 presents 20 days pump output plotted in one day long blocks for the same patient as in Figure 1. In Figure 3, different colors represent different days. For this patient, circadian variation is seen in each seen for all 20 days. Also, we observed that the average PO is lower during the days 10-15 than during the first ten days or last five days. Figure 1 also shows that measurement error has two major regimes (low during sleep and high during the day). Therefore, level-dependent noise is present in the data.

Even without any smoothing, both the circadian cycle and its longitudinal evolution can be simultaneously visualized. The PO circadian cycle consists of two potentially smooth regions, sleeping and waking, with an abrupt change during the morning hours. Examining

the patient's raw average circadian cycles for both PO and PI, see Figures 4 and 5, circadian variation is clearly seen in all patients. While there may exist a smooth circadian cycle for any patient, abrupt jumps are seen in most patients, e.g., the time period from 0700h-1100h in the bottom center patients shows a rapid increase in PO, then decrease. The PI circadian cycle, see Figure 5, shows a similar and mostly likely correlated pattern. Also, all patients have a rapidly changing PO and PI during the morning wake-up period. Outside of this period, the PO and PI levels change slower. In Figures 6 and 7, each days average PO or PI is plotted against calendar day. In the top left plot of Figures 6 and 7, both PO and PI are seen to decline from their highest level at day 1 to their lowest level at day 12. During these 12 days, PO declines by approximately 1000 mL/min and PI by 2000 mL. Then both sharply rebound by day 15. Several other patients show day-to-day fluctuations of a similar magnitude. In contrast, the plots in the lower right show little longitudinal variation in flow.

Therefore, we focus on introducing estimation that are able to model functions with multiple types of smoothness and does not fail is the presence of non-white noise. Several candidate estimation techniques include the use of wavelet basis functions and varying bandwidth smoothers. We use wavelet basis functions throughout the rest of this dissertation to address these features of the data as the use of a wavelet basis has better performance in data with regions of rapid and slow variation.

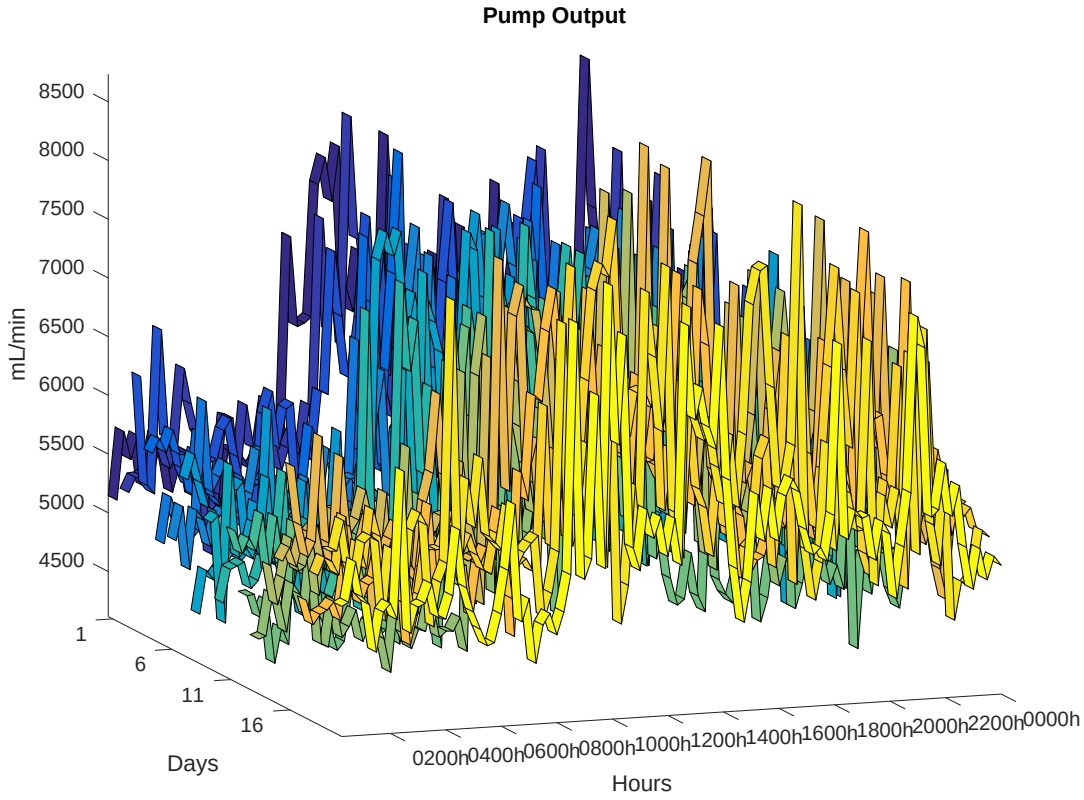

Figure 3: Raw surface plot for 20 days of pump output for a representative patient.

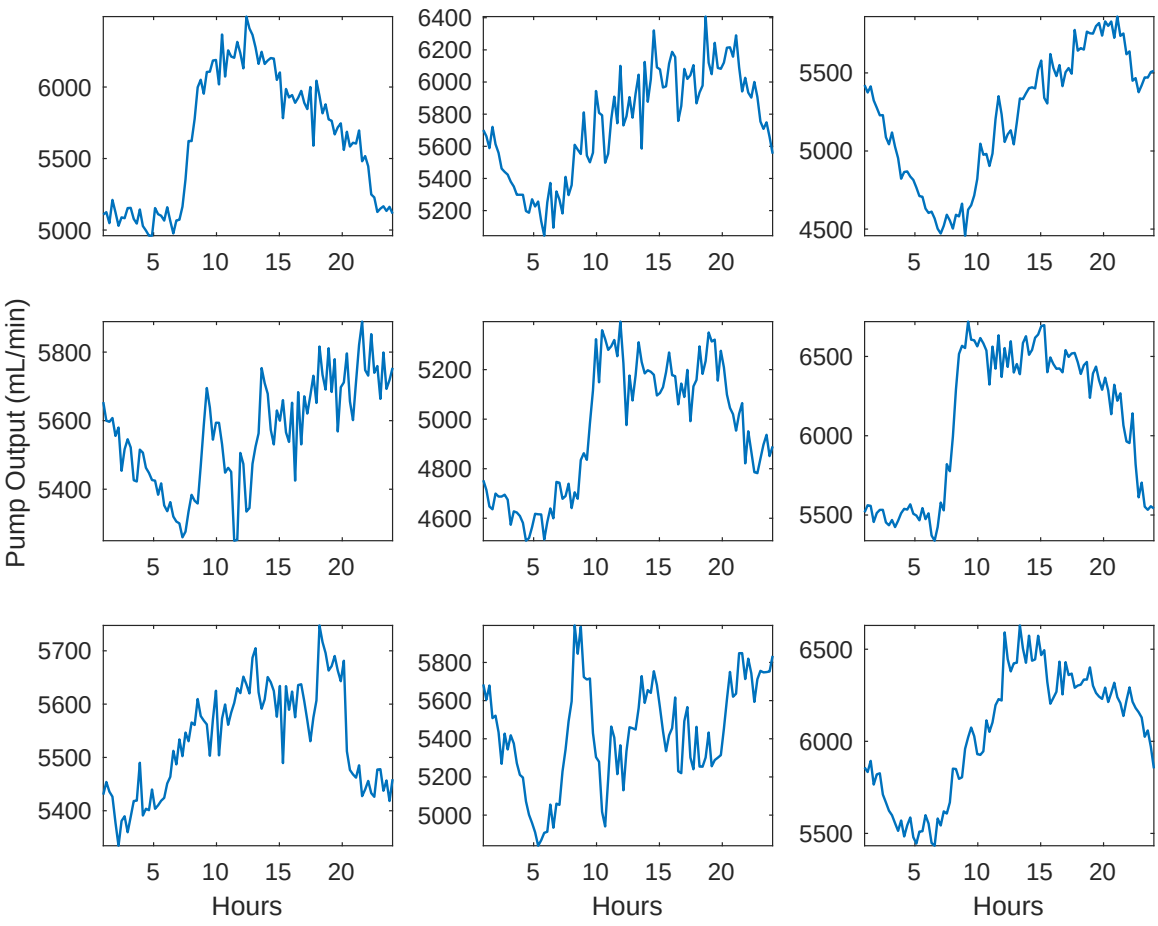

Figure 4: Raw mean pump output circadian cycles for nine patients.

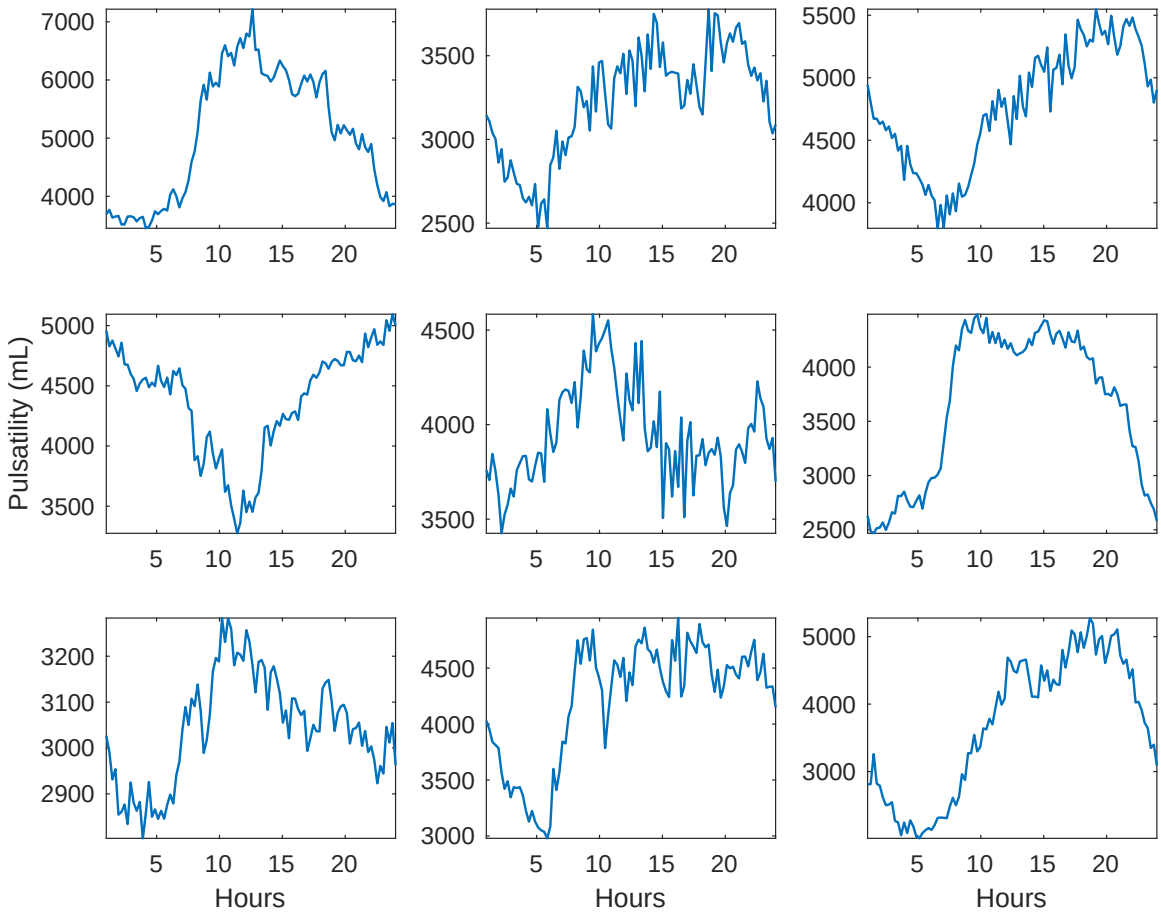

Figure 5: Raw mean puslatility circadian cycles for nine patients.

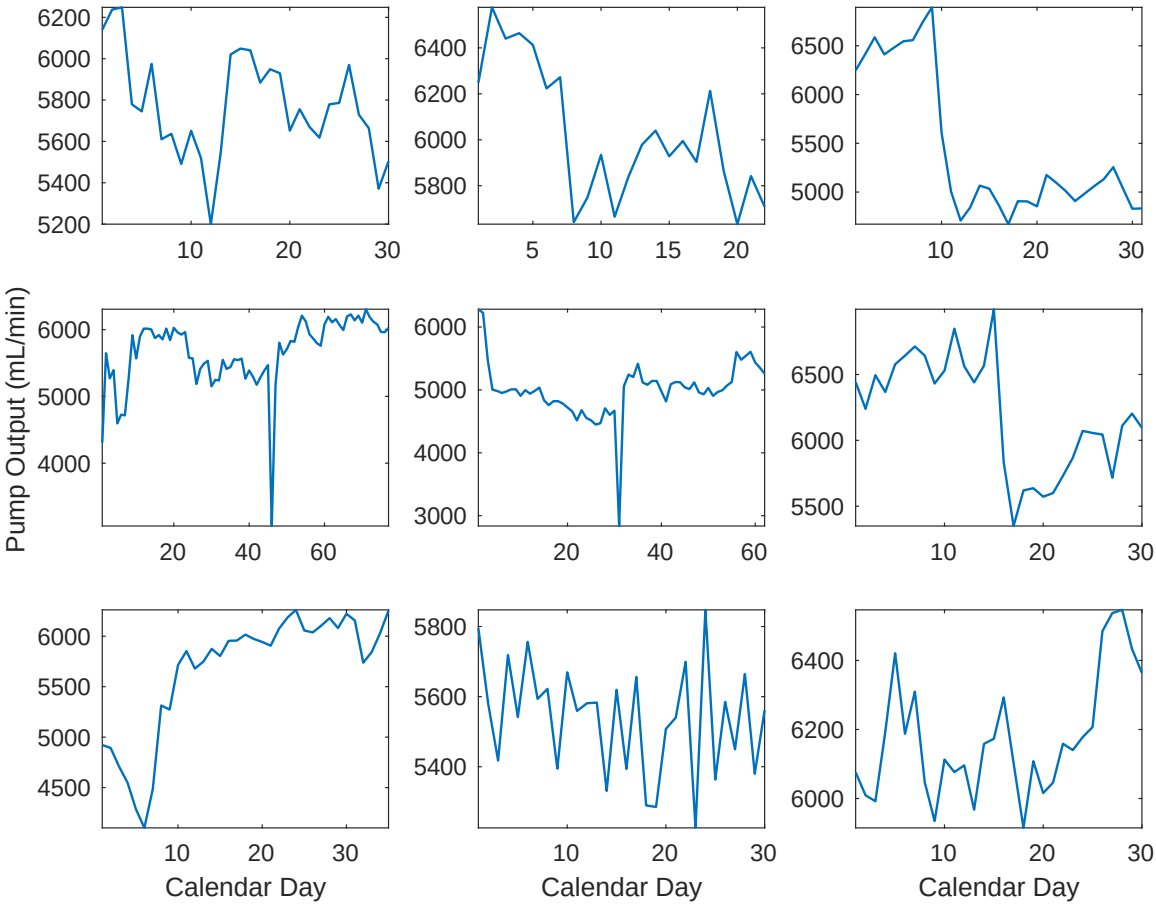

Figure 6: Raw mean pump output longitudinal evolution for nine patients.

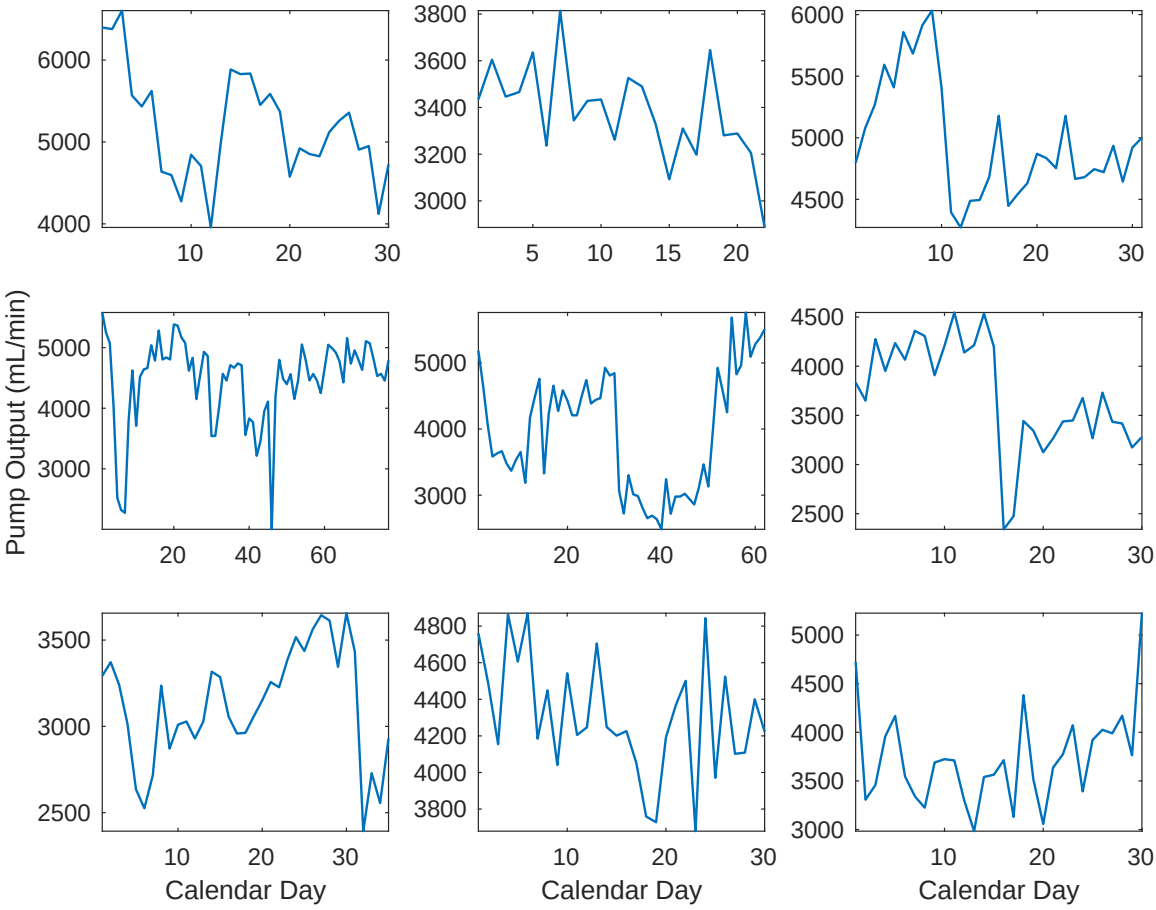

Figure 7: Raw mean puslatility longitudinal evolution for nine patients.

3.0 CURRENT FPCA METHODS AND THEORY

Before reviewing the existing FPCA methods, we introduce several important sampling schemes on both the fast and slow time scales. On the fast time scale, all methods discussed and developed assume that the data points are equally spaced and have the same number of samples each day. Therefore, the observed fast time data points form a dense regular grid t_l with $l = 1, \ldots, n$. The slow time sampling can take on several types of sampling. It is worth exploring four different types of longitudinal sampling encountered in LFD. If we denote each repeated observation time of s_{ij} for the i^{th} subject, s_{ij}'s fall into one of several main cases. The first case is dense regular follow-up where the s_{ij} are sampled for all values of $j \in [0, J_i]$. In this case, each subject can have the same length of follow-up, i.e., $J_1 = J_2 = \cdots = J_N = J$ (balanced design) or unbalanced follow-up with no missing data. In both of the these cases, the total number of observations, $N \times J$ or $\sum_i J_i$ diverges as $J_i \to \infty$. A third case is dense regular follow-up with missing data. Alternatively, the follow-up can have sparse irregular sampling with subjects having different lengths of follow-up J_i and random gaps between each s_{ij}. Here, $\sum_i J_i < \infty$ as $J_i \to \infty$. Equipped with the sampling schemes, we now embark on an overview of existing FPCA techniques for both cross-section (single observation) functional data and longitudinal (repeated observation) functional data.

3.1 CROSS-SECTIONAL FUNCTIONAL DATA ANALYSIS WITH FPCA

Before illustrating analysis of longitudinal FPCA, we overview FPCA for a single observation of univariate or multivariate functions on N subjects. We term this analysis cross-section FPCA to distinguish it from longitudinal or repeated FPCA. The observed data consist of

functions $y_i(t)$ observed on $i = 1, \ldots, N$ subjects at equally spaced time points $t_l = t_1, \ldots, t_n$. For example, a study may be interested in asking what is the population average VAD PO curve during the first day post hospitalization and how PO varies betweens patients.

These data are modeled assuming that they are generated by a smooth function and observed with noise (Ramsay and Silverman, 2005). Therefore, we start all FDA with the model

$$y_i(t) = F_i(t) + \epsilon_i(t), \tag{3.1}$$

where $F_i(t)$ is a subject specific random function with mean function $\mu(t)$ and covariance function $\mathcal{K}(t, t')$, and noise process $\epsilon_i(t)$. Defining centered subject specific random functions as $X_i(t) = F_i(t) - \mu(t)$, the covariance function is the expectation of product of centered random functions,

$$\mathcal{K}(t, t') = \text{cov}(X_i(t), X_i(t')) = E\big[X_i(t)X_i(t')\big]. \tag{3.2}$$

This formulation leads to the model

$$y_i(t) = \mu(t) + X_i(t) + \epsilon_i(t). \tag{3.3}$$

At the observed time points t_l, Ramsay and Silverman (2005) argue that the sample mean vector

$$\hat{\mu}(t_l) = \frac{1}{N}\sum_{i=1}^{N} y_i(t_l) \tag{3.4}$$

and sample covariance matrix

$$\hat{\mathcal{K}}(t_l, t_{l'}) = \frac{1}{N-1}\sum_{i=1}^{N}\big(y_i(t_l) - \hat{\mu}(t_l)\big)\big(y_i(t_{l'}) - \hat{\mu}(t_{l'})\big) \tag{3.5}$$

are reasonable but noisy estimators for the population mean and covariance. These estimates are used in FPCA. As both (3.4) and (3.5) still contain noise, a regularization step is included in FPCA. While the data can be pre-smoothed, we recommend that regularization takes place during FPCA to both prevent over-smoothing and reduce to the computational burden. Details of several smoothing procedures are discussed in Chapter 4.

In this dissertation, all unknown functions such as $\mu(t)$ can be accurately represented by a basis function expansion. If $\phi_p(t)$ form a set of P orthogonal basis functions on the domain $t \in [0, \mathcal{T}]$, then $\mu(t) = \sum_{p=1}^{P} \xi_p \phi_p(t)$ where ξ_p are fixed basis coefficients. Similarly,

a random function $X_i(t)$ has the approximate expansion $X_i(t) \approx \sum_{p=1}^{P} \xi_{ip}\phi_p(t)$ where the ξ_{ip} are random basis coefficients. If the basis functions are known functions such as B-splines or trigonometric functions, the coefficients are estimates using techniques such as smoothing or p-splines, kernel density estimators, or wavelets, see Ramsay and Silverman (2005). If the basis functions are unknown, FPCA is used to determine a set of basis functions that minimizes the square error of the approximation of $\hat{X}_i(t)$. As FPCA forms the basis of several important LFDA methods, we provide an in-depth review of FPCA.

The condition of least square approximation error implies that a FPCA basis can be found using the least square objective function, (Ramsay and Silverman, 2005). For a population of N subjects, the objective function is

$$\begin{aligned} H &= \sum_{i=1}^{N} \|X_i(t) - \hat{X}_i(t)\|_2^2 \\ &= \sum_{i=1}^{N} \left\| X_i(t) - \sum_{p}^{\infty} \xi_{ip}\phi_p(t) \right\|_2^2, \end{aligned} \tag{3.6}$$

where $\|\cdot\|_2$ is the $\mathbb{L}^2$-norm for functions defined on $\mathbb{L}^2$. This norm has the integral representation

$$\|X_i(t)\|_2^2 = \int_{\mathcal{T}} X_i(t)^2 \, \mathrm{d}t,$$

for a function $X_i(t)$ defined on a domain, $\mathcal{T}$. Ramsay and Silverman (2005) show that the set of basis functions that minimizes (3.6) has the additional property that it maximizes the amount of variation explained in the random functions $X_i(t)$.

3.2 THEORECTICAL SUPPORT OF FPCA VIA THE KARHUNEN-LOÈVE THEOREM AND MERCER'S THEOREM

The theoretical support for FPCA is the Karhunen-Loève expansion and Mercer's theorem (Happ and Greven, 2015). Mercer's theorem allows for the eigen-decomposition of a covariance function $\mathcal{K}(t, t')$ into eigenvalues λ_p and eigenfunctions $\phi_p(t)$. Under the assumption

that $\mathcal{K}(t,t')$ is square integrable, Mercer's Theorem states that

$$\mathcal{K}(t,t') = \sum_{p=1}^{\infty} \lambda_p \phi_p(t) \phi_p(t'), \tag{3.7}$$

with the eigenvalues and eigenfunctions being solutions to

$$\int_{\mathcal{T}} \mathcal{K}(t,t') \phi_p(t') \, \mathrm{d}t' = \lambda_p \phi_p(t). \tag{3.8}$$

When Mercer's theorem holds, the Karhunen-Loève theorem states that a random function has a basis expansion

$$X_i(t) = \sum_{p=1}^{\infty} \xi_{ip} \phi_p(t), \tag{3.9}$$

where the basis coefficient or principal component (PC) scores are

$$\xi_{ip} = \int_{\mathcal{T}} X_i(t) \phi_p(t) \, \mathrm{d}t \tag{3.10}$$

with $\xi_{ip} \sim N(0, \lambda_p)$ that are uncorrelated for different p. For a subject i, the predicted response function is

$$y_i(t) = \mu(t) + \sum_{p} \xi_{ip} \phi_p(t).$$

For multivariate functions, the case when $\boldsymbol{y}_i(t)$ consists of D simultaneous observations instead of 1 observation, analysis the mean function $\boldsymbol{\mu}(t)$ is unchanged and is conducted in a component-wise fashion. However, analysis of multivariate random functions $\boldsymbol{X}_i(t)$ requires an adjustment to the covariance function $\mathcal{K}(t,t')$, Karhunen-Loève theorem and Mercer's theorem. For multivariate data, the covariance function changes to

$$\mathcal{K}(t,t') = E\big[\boldsymbol{X}_i(t) \boldsymbol{X}_i^t(t')\big]. \tag{3.11}$$

Accordingly, the eigenfunctions from equations (3.7) and (3.9) change to $\boldsymbol{\phi}_p(t)$. However, the eigenvalues and the PC scores remain scalars. This small change in the theory will cause challenges in FPCA estimation, discussed in Section 5.2.1. These theorems provide a firm foundation for the creation of a data driven basis set that has minimal least square approximation error.

3.3 LONGITUDINAL FUNCTIONAL DATA ANALYSIS WITH FPCA

Often, the function of interest is observed not just once, but repeatedly observed over multiple days, weeks, years. For example, the VAD power circadian cycle is recorded for each day that the patient has a VAD implanted. Therefore, we introduce both how functions are observed longitudinally and detail theory and application of two useful existing models: LFPCA and RF-FPCA. While both LFPCA and RF-FPCA are able to analyze longitudinal functional data (LFD), they differ in approach, interpretation, and computing time.

Both methods start with the idea that the LFD form a two-dimensional surface on the domain (t, s). They model the cohort mean surface $\mu(t, s)$ and provide two different FPCA models for the random functions $X_i(t, s)$. For both RF-FPCA and LFPCA, the basic model for LFD when both t and s are continuous variables is

$$\begin{aligned} y_i(t,s) &= \mathscr{F}_i(t,s) + \epsilon_i(t,s) \\ &= \mu(t,s) + X_i(t,s) + \epsilon_i(t,s), \end{aligned} \tag{3.12}$$

where $y_i(t, s)$ are the observed data, $\mathscr{F}_i(t, s)$ is a random surface with mean surface $\mu(t, s)$ and mean zero subject specific surface $X_i(t, s) = \mathscr{F}_i(t, s) - \mu(t, s)$ with covariance function $\mathcal{K}(t, t', s, s') = \text{cov}\left[X_i(t, s), X_i(t', s')\right]$, and $\epsilon_i(t, s)$ is the error which is specified for each model. In (3.12), the random functions have a two dimensional Karhunen-Loève expansion

$$X_i(t,s) = \sum_{p=1}^{\infty} \xi_{ip}\phi_p(t,s) \tag{3.13}$$

where $\phi_p(t, s)$ are the eigensurfaces of

$$\mathcal{K}(t,t',s,s') = \text{cov}\left[X_i(t,s), X_i(t',s')\right] = \sum_{p=1}^{\infty} \lambda_p\phi_p(t,s)\phi_p(t',s') \tag{3.14}$$

with eigenvalues, $\lambda_1 > \lambda_2 > \cdots > \lambda_\infty$. The PC scores are found with the double integral

$$\xi_{ip} = \int_{\mathcal{S}}\int_{\mathcal{T}} X_i(t,s)\phi_p(t,s)\,\mathrm{d}t\,\mathrm{d}s.$$

Conducting FPCA on the full covariance, $\mathcal{K}(t, t', s, s')$, is challenging as a four dimensional smoothing step is required. Instead both RF-FPCA and LFPCA conduct a two step analysis of the covariance.

As the mean surface is estimated by well accepted techniques, both LFPCA and RF-FPCA focus on modeling the random processes $X_i(t, s)$, see details in Section 4.2. RF-FPCA models the conditional stochastic processes $X_i(t|s)$ with a set of conditional FPCA models that depends upon the longitudinal observation. These expansions and the associated basis functions are dependent on longitudinal time. The longitudinal dependence in the $X_i(t|s)$ is modeled through a second stage FPCA. In contrast, LFPCA uses a longitudinally constant set of basis functions for $X_i(t, s)$ with the longitudinal dependence induced through time varying coefficients, $\xi_{ip}(s_{ij})$ when s_{ij} form a sparse sample (case 4).

3.3.1 Repeated Function FPCA

Chen and Müller (2012) model $X_i(t|s)$ by a two step Karhunen-Loève expansion instead where the function $X(t, s)$ is observed repeated for fixed s leading to the conditional representation $X(t|s)$. For the i^{th} subject,

$$F_i(t|s) = \mu(t|s) + \sum_{p=1}^{\infty} \xi_{ip}(s)\phi_p(t|s) \tag{3.15}$$

where for fixed s, $\phi_p(\cdot|s)$ are the eigenfunctions of the covariance function $\mathcal{K}(t, t'|s) = \text{cov}\left[X(t|s), X(t'|s)\right]$ and $\xi_{ip}(s)$ are the corresponding PC scores which are random functions of s with mean zero and $\text{var}\left[\xi_{ip}(s)\right] = \lambda_p(s)$ for fixed s.

Now a second Karhunen-Loève expansion is applied to the $\xi_{ip}(s)$ representing these functions as

$$\xi_{ip}(s) = \sum_{q=1}^{\infty} \zeta_{iqp}\psi_{qp}(s) \tag{3.16}$$

where $\psi_{qp}(s)$ are the "second level" eigenfunctions and the mean zero ζ_{iqp} are "second level" PC scores. The associated covariance operator has kernel $\mathcal{K}(s, s') = \text{cov}\left[\xi_{ip}(s), \xi_{ip}(s')\right]$.

Then, they combine the Karhunen-Loève expansions in Equation (3.15) and Equation (3.16) leading to the final model

$$F_i(t|s) = \mu(t|s) + \sum_{p=1}^{\infty}\sum_{q=1}^{\infty}\zeta_{iqp}\psi_{qp}(s)\phi_p(t|s) \tag{3.17}$$
$$= \mu(t|s) + \sum_{p=1}^{\infty}\sum_{q=1}^{\infty}\zeta_{iqp}\varphi_{qp}(t|s)$$

where $\varphi_{qp}(t|s) = \psi_{qp}(s)\phi_p(t|s)$. The surfaces, $\varphi_{qp}(t|s)$,describe how the FPCA basis varies over time.

In order to use the model in (3.17), we need to both truncate the infinite sums as well as estimate a smooth mean function, $\hat{\mu}(t|s)$, and covariance functions $\hat{\mathcal{K}}(t,t'|s)$. Chen and Müller (2012) recommend setting the size of each FPCA basis is determined using fraction of variance explained (FVE), although both AIC and BIC also give reasonable results. The estimation of the mean and covariance depends upon both the sampling in both t and s. Chen and Müller (2012) assume that sampling in the $t-$direction to be on a dense regular grid. All cases of longitudinal sampling can be analyzed with RF-FPCA by changing the FPCA algorithm.

3.3.2 Longitudinal Functional Principal Component Analysis

When the subjects have sparse longitudinal sampling, LFPCA can be used instead of RF-FPCA. In contrast to RF-FPCA, LFPCA uses a longitudinally constant set of basis functions, $\phi_p(t)$, with time varying coefficients, $\xi_{ip}(s_{ij})$, where s_{ij} are the subject specific longitudinal observations, (Park and Staicu, 2015; Greven et al., 2010). Park and Staicu (2015) introduce a modified version of (3.12) for sparse follow-up:

$$y_i(t, s_{ij}) = \mu(t, s_{ij}) + X_i(t, s_{ij}) + \epsilon_i(t, s_{ij}), \tag{3.18}$$

where all terms are defined as previously but evaluated at the time points s_{ij}. However, the random functions, $X_i(t, s_{ij})$, have different Karhunen-Loève expansion,

$$X_i(t, s_{ij}) = \sum_{p=1}^{\infty}\xi_{ip}(s_{ij})\phi_p(t),$$

with a marginal covariance $\mathcal{K}(t,t') = \int\int \mathcal{K}(t,t',s,s')\,\mathrm{d}s\,\mathrm{d}s'$. Chen et al. (2016) refers to this type of covariance kernel as a marginal kernel. Using the same strategy of a double FPCA, the full LFPCA model is

$$y_i(t,s_{ij}) = \mu(t,s_{ij}) + \sum_{p=1}^{\infty}\sum_{q=1}^{\infty}\zeta_{iqp}\psi_{qp}(s_{ij})\phi_p(t) + \epsilon_i(t,s_{ij}), \tag{3.19}$$

where $\psi_{qp}(s_{ij})$ are the eigenfunctions of the longitudinal covariance function

$$\mathcal{K}_p(s_{ij},s'_{ij}) = \sum_q \lambda_{qp}\psi_{qp}(s_{ij})\psi_{qp}(s'_{ij}),$$

$$\zeta_{iqp} = \int \xi_{ip}(s_{ij})\psi_{qp}(s_{ij})\,\mathrm{d}s,$$

$\epsilon_i(t,s_{ij})$ is the error term, and all else is defined as previous. In any application, the infinite sums are truncated using FVE or another method.

As these models consist of three parts, mean function (surface), a random function for the domain $\mathcal{T}$, and a longitudinal random function on $\mathcal{S}$, the estimation methods considered in Section 4.4 are broken down in the same fashion. For both RF-FPCA and LFPCA, the mean surface estimation and both FPCA estimation steps are estimated with existing algorithms. However, both methods introduce new techniques to handle the estimation of the $\xi_{ip}(s)$. These topics are the focus of the next Chapter.

4.0 CURRENT FPCA ESTIMATION ALGORITHMS

In Chapter 3, we introduced several models for analyzing longitudinal functional data and the theoretical backing. Before any of these techniques are useful for analyzing data, an estimation procedure is needed. This chapter consists of three main sections: function estimation with a spline basis, function estimation with a wavelet basis and FPCA estimation. Splines are introduced as they are commonly used and necessary for understanding existing techniques. In addition, wavelets are introduced as they are used for the estimation algorithm of the new method, see Chapter 5.

4.1 ESTIMATION WITH A SPLINE BASIS

A popular functional estimation approach for estimating an unknown function such as $\mu(t)$ uses a B-spline basis. We first discuss spline estimation for functions of one domain variable before moving on to functions of two domain variables and tensor product splines. In this section, we discuss estimation of a mean curve $\mu(t)$ of data $Y_i(t_l) = \mu(t_l) + \epsilon_i(t_l)$ observed on $i = 1, \ldots, N$ subjects and at time points t_l where $l = 1, \ldots, n$. Here, $\epsilon_i(t)$ are mean zero normally distribute random variables. The covariance structure is discussed separately for each estimation method.

B-spline basis functions are a set of known polynomial functions, $e_g(t)$, of degree m with compact support that form an orthonormal basis for $\mathbb{L}^2$ on an interval $[t_0, t_{max}]$ (de Boor, 1972; Schumaker, 2007). To represent a function $\mu(t_l)$ observed at time points $t_l \in [t_0, t_{max}]$

using a spline basis, first partition of $[t_0, t_{max}]$ into G intervals such that each interval is

$$I_g = [t_g, t_{g+1}),\ g = 1, 2, \ldots, G+1.$$

The points, t_g, are called knots, and successive basis functions are joined at a knot (Schumaker, 2007). The basis functions, $e_g(t_l)$, have $m-1$ continuous derivatives. Using a B-spline basis, a smooth function such as $\mu(t)$ is represented by

$$\mu(t) = \sum_{g=1}^{G} a_g e_g(t), \tag{4.1}$$

where a_g are the basis coefficients, $e_g(t)$ are B-spline basis functions, and G is the number of functions.

The first step in estimating (4.1) is to select the number and location of the knots must be determined. One option is to pick a small number of knots yielding a very smooth estimate of $\mu(t)$. However, this approach fails to capture interesting behavior of the data. To make sure that the data are well described, a knot can be placed at every time point t_l for a total of n knots. However, this approach may lead to the over-fitting of the data with the estimated function $\hat{\mu}(t_l)$ passing through every $\bar{Y}_i(t_l) = \frac{\sum_i Y_i(t_l)}{N}$. To prevent this over-fitting, some type of regularization or penalty on the size of the spline coefficients must be used. Two common types of penalties are continuous penalties, often on the second derivative, that lead to smoothing splines and discrete or differencing penalties, often on the second difference, that lead to the P-spline solution (Ramsay and Silverman, 2005; Eilers and Marx, 1996). Both of techniques penalize have a similar effect to penalize the fit if the estimated function $\hat{\mu}(t_l)$ has a large global curvature measure.

To fit this spline model of the data, the least squares objective function in Equation (4.3) is minimized with respect to the coefficients a_g in (4.1)

$$SSE = \frac{1}{N} \sum_{i=1}^{N} \sum_{l=1}^{n} \left[Y_i(t_l) - \sum_{g=1}^{G} a_g e_g(t_l) \right]^2 \tag{4.2}$$

Following the approach of Ramsay and Silverman (2005), equation (4.2) is re-expressed in matrix notation

$$SSE = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{Y}_i - \boldsymbol{E}\boldsymbol{A})^t (\boldsymbol{Y}_i - \boldsymbol{E}\boldsymbol{A}), \tag{4.3}$$

where $\boldsymbol{Y}_i$ is a data column vector, $\boldsymbol{E}$ is a matrix of basis functions, and $\boldsymbol{A}$ is a matrix of coefficients. The weighted least squares criteria can also be used if the error structure is believed to not be i.i.d normal with a covariance matrix, $\boldsymbol{\Sigma}$, leading to the addition of a weight, $\boldsymbol{W} = \boldsymbol{\Sigma}^{-1}$, to (4.3),

$$SSE_w = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{Y}_i - \boldsymbol{E}\boldsymbol{A})^t\boldsymbol{W}^{-1}(\boldsymbol{Y}_i - \boldsymbol{E}\boldsymbol{A}).$$

The smoothing spline model is a common choice for fitting (4.3). A smoothing spline sets the number of basis functions equal to the number of observations, $G = n$, and places a knot at each time point t_l. If this model is directly applied to (4.3), this spline model is over-parametrized as the number of basis coefficients equals the number of data points. To prevent over-fitting of the model, roughness penalties are incorporated into the least squares objective function (Silverman, 1985). The penalized least squares criteria is

$$SSE_{PEN} = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{Y}_i - \boldsymbol{E}\boldsymbol{A})^t\boldsymbol{W}^{-1}(\boldsymbol{Y}_i - \boldsymbol{E}\boldsymbol{A}) + \lambda\boldsymbol{A}^t\boldsymbol{D}\boldsymbol{A} \tag{4.4}$$

Here, λ is a smoothing parameter that controls how much impact the penalty term $\boldsymbol{A}^t\boldsymbol{D}\boldsymbol{A}$ has on the fit and $\boldsymbol{D}$ is a matrix of derivatives of $e_g(t_l)$. Often the 2^{nd} derivatives are used leading to a cubic smoothing spline model. When $\lambda = 0$, the spline fit by (4.4) interpolates the data, and when $\lambda \to \infty$, the spline fit by (4.4) is a linear least squares regression of the data $\boldsymbol{Y}$. (4.4) has the effect of shrinking the coefficients a_g towards a linear fit while allowing for important curvature in $\mu(t)$. Ramsay and Silverman (2005) provide a solution to (4.4) in terms of a smoothing matrix

$$\boldsymbol{S}(\lambda) = \boldsymbol{E}\big(\boldsymbol{E}^t\boldsymbol{W}\boldsymbol{E} + \lambda\boldsymbol{D}\big)^{-1}\boldsymbol{E}^t\boldsymbol{W}. \tag{4.5}$$

For observations, $\boldsymbol{Y}_i$, the underlying smooth function $\mu(t_l)$ is estimated by

$$\hat{\boldsymbol{\mu}}_\lambda = \boldsymbol{S}(\lambda)\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{Y}_i.$$

Notice that $\hat{\boldsymbol{\mu}}_\lambda$ is still a function of the unknown smoothing parameter λ. Estimation of $\hat{\lambda}$ depends on the type of error process $\epsilon_i(t_l)$. We consider two types of error structures, white noise and colored noise. We also discuss estimation of $\hat{\lambda}$ in each case.

4.1.1 Smoothing Parameter Estimation

Since all of the above estimation techniques depend on a tuning parameter, λ, the parameter must either be set by hand or estimated from the data using a pre-specified criterion. As manual control of the tuning parameter is impractical, automated techniques are used. These techniques are based on minimizing the mean square error (MSE) $E\|\hat{\mu}_\lambda(t_l) - \mu(t_)\| = \epsilon_i(t_l)$ for the spline estimation. An estimate $\hat{\lambda}$ that minimizes the MSE is considered optimal.

If the error process $\epsilon_i(t_l)$ is assumed to be a white noise process, the MSE can be estimated using the leave one observation out cross validation estimator (CV). The CV estimator of the MSE is

$$\begin{aligned} CV(\lambda) &= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{n}\sum_{l=1}^{n}\left[Y_i(t_l) - \hat{\mu}_\lambda^{(-l)}(t_l)\right]^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{n}\sum_{l=1}^{n}\frac{\left[Y_i(t_l) - \hat{\mu}_\lambda(t_l)\right]^2}{\left[1 - \boldsymbol{S}(\lambda)_{ll}\right]^2}, \end{aligned} \tag{4.6}$$

where $\bar{y}_l$ is the observed data and $\hat{\mu}_\lambda^{(-l)}(t_l)$ is the estimated value of the $\hat{\mu}$ estimated without the l^{th} observed point (Hastie et al., 2009; Gu, 2013).

The quantity in (4.6) can also estimated by generalized cross-validation (GCV) introduced by Craven and Wahba (1979). Unlike CV score, GCV is invariant to orthonormal transformations of the data (Gu, 2013; Craven and Wahba, 1979). The GCV criteria is a function of the smoother matrix $\boldsymbol{S}(\lambda)$ and for n points is computed by

$$GCV(\lambda) = \frac{1}{N}\sum_{i=1}^{N}\frac{n\boldsymbol{Y}_i^t[\boldsymbol{I} - \boldsymbol{S}(\lambda)]^2\boldsymbol{Y}_i}{\left\{\operatorname{tr}[\boldsymbol{I} - \boldsymbol{S}(\lambda)]\right\}^2}. \tag{4.7}$$

Because (4.7) only depends on the trace of $\boldsymbol{S}(\lambda)$, the GCV score is slightly more efficient to compute than the CV score. GCV smoothing parameter estimation assumes that the $\epsilon_i(t_l)$ are independent and identically distributed mean zero normal random variables, $\epsilon_i(t_l) \sim N(0, \sigma^2)$. GCV fails in the case of colored noise because $\epsilon_i(t_l)$ is not i.i.d. $N(0, \sigma^2)$.

4.2 ESTIMATION WITH A TENSOR PRODUCT SPLINE BASIS

In the analysis of longitudinal functional data, an important step is the estimation of the unknown mean surface, $\bar{y}(t,s) = \mu(t,s) + \epsilon(t,s)$. Basis functions of t and s are needed. One choice is the thin-plate spline (TPS); however, the penalty that leads to the TPS is complicated and time intensive to evaluate for bivariate splines, and practically intractable in higher dimensions. Therefore, the tensor product spline basis is introduced represent functions of multiple variables. A tensor product spline represents $\mu(t,s)$ by observed on a regular grid in t, s by

$$\mu(t,s) = \sum_{\substack{1\leq g\leq G \\ 1\leq h\leq H}} a_{g,h} e_g(t) e_h(s) \tag{4.8}$$

where $e_g(t)$ are the G B-splines in the t direction, $e_h(s)$ are the H B-splines in the s direction, and $a_{g,g}$ are the spline coefficients (Xiao et al., 2013). Similar to the single variable smoothing case, a penalized estimation procedure is used to prevent over fitting. The fitting criteria is

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{n}\sum_{j=1}^{J}\left[y_{ijl} - \sum_{g=1}^{G}\sum_{g=1}^{G} a_{g,h} e_g(t_l) e_h(s_j)\right]^2 + \text{Pen}(\lambda_t, \lambda_s), \tag{4.9}$$

where the first term is the multiple variable sum of squares and the second term is a multi-variable penalty term (Park and Staicu, 2015). Any penalty consists of at least a t direction penalty and a s direction penalty (Xiao et al., 2013; Park and Staicu, 2015). In Park and Staicu (2015), they use a $t-$direction penalty

$$\text{Pen}_t = \lambda_t \int\int \left\{\frac{\partial^2 \mu(t,s)}{\partial t^2}\right\} \mathrm{d}t\, \mathrm{d}s$$

and a $s-$direction penalty

$$\text{Pen}_s = \lambda_s \int\int \left\{\frac{\partial^2 \mu(t,s)}{\partial s^2}\right\} \mathrm{d}t\, \mathrm{d}s.$$

Alternatively, Xiao et al. (2013) introduce the sandwich smoother, a multivariable extention of P-spline smoothers Eilers and Marx (2003); Xiao et al. (2013). P-splines differ from smoothing splines as the penalty matrix is directly calculated as the 2^{nd} difference of the basis instead of the 2^{nd} derivatives. To use the sandwich smoother for estimating a mean

surface, first define $\bar{\boldsymbol{Y}} = \frac{1}{N}\sum_{i=1}^{N} y_{ijl}$ as the data matrix. Xiao et al. (2013) proposes to estimate

$$\hat{\mu} = \boldsymbol{S}_t \bar{\boldsymbol{Y}} \boldsymbol{S}_s = (\boldsymbol{S}_s \otimes \boldsymbol{S}_t)\bar{\boldsymbol{Y}} \tag{4.10}$$

where

$$\boldsymbol{S}_k = \boldsymbol{E}_k(\boldsymbol{E}_k^t \boldsymbol{E}_k + \lambda_k \boldsymbol{D}_k^t \boldsymbol{D}_k)^{-1}\boldsymbol{E}_k^t$$

is smoother matrices for the k^{th} direction, $\bar{\boldsymbol{Y}}$ is the data matrix, $\boldsymbol{D}_k$ is a differing matrix of order m_k, $\otimes$ is the tensor product, and all other terms defined as previous. For two domain variables, this formulation leads to the penalty term

$$\boldsymbol{P} = \lambda_t \boldsymbol{E}_s^t \boldsymbol{E}_s \otimes \boldsymbol{D}_t^t \boldsymbol{D}_t + \lambda_s \boldsymbol{E}_t^t \boldsymbol{E}_t \otimes \boldsymbol{D}_s^t \boldsymbol{D}_s + \lambda_t \lambda_s \boldsymbol{D}_s^t \boldsymbol{D}_s \otimes \boldsymbol{D}_t^t \boldsymbol{D}_t.$$

By minimizing

$$\|\bar{\boldsymbol{Y}} - \boldsymbol{E}_t \boldsymbol{A} \boldsymbol{E}_s\|_F^2 + \text{vec}(\boldsymbol{A})^t \boldsymbol{P} \text{vec}(\boldsymbol{A}),$$

where the norm is the Frobenius matrix norm, Xiao et al. (2013) introduced a fast algorithm for estimating $\hat{\mu}(t, s)$. The smoothing parameters are selected using GCV.

While smoothing splines and P-splines are commonly used for FDA, we do not use them as basis functions for the analysis of our VAD waveform data because of the noise structure and regions of rapid change. Our exploration of splines provides a gentle introduction into functional data analysis with basis functions that is expanded in the discussion of wavelets. Additionally, these spline estimators are used in LFPCA and explanation of LFPCA estimation is clarified by a clear understanding of both smoothing and P-splines.

4.3 ESTIMATION WITH A WAVELET BASIS

Another important class of basis functions is the wavelet multi-resolution basis. A wavelet basis differs from the spline basis in Section 4.1 in that it has two types of basis functions that capture the behavior $\mu(t_l)$ at multiple resolutions. Scale functions $e_g(t_l)$ at scale level g describe the average behavior of $\mu(t_l)$ and wavelets $e_{gh}(t_l)$ that reflect the variation at location h on the g^{th} scale (Donoho and Johnstone, 1994; Mallat, 1999; Nievergelt, 2001).

We illustrate a wavelet basis using Haar wavelets inspired by the approach of Nievergelt (2001).

On the interval $[0, 1)$, the father Haar wavelet is defined as

$$e_{[0,1)}(t) = \begin{cases} 1 & \text{if } 0 \leq t < 1 \\ 0 & o.w. \end{cases}. \tag{4.11}$$

Nievergelt (2001) extends this definition to a half-open interval of arbitrary length $1/g$ on the g^{th} scale. For $g = 2$, these intervals form a partition $\mathbb{P}_2 = \{0, 1/2, 1\}$ of $[0, 1)$. Therefore, the Haar scale functions for $\mathbb{P}_g$ are defined by

$$e_g(t) = \begin{cases} g & \text{if } (h-1)/g \leq t < h/g \\ 0 & o.w. \end{cases}$$

for $h = 1, 2, \ldots, g$. At the g^{th} scale, the scale functions approximate a function $\mu(t)$ on $[0, 1/g)$ by its average value

$$\mu_g^* = g \int_{(h-1)/g}^{h/g} \mu(t) e_g(t)\, \mathrm{d}t.$$

While the scale functions capture the average of $\mu(t)$, they ignore any changes in $\mu(t)$ on the interval h on the g^{th} scale. Wavelet functions capture the change in $\mu(t)$ for the h^{th} interval on the g^{th}. The Haar wavelet, $e_{gh}(t)$, is defined as

$$e_{gh}(t) = e_{[(h-1)/g,(h-1)/g+1/2g)}(t) - e_{[(h-1)/g+1/2g,h/g)}(t). \tag{4.12}$$

The change in $\mu(t)$ at scale g and location h is

$$\mu_{gh}^* = g \int_{(h-1)/g}^{h/g} \mu(t) e_{gh}(t)\, \mathrm{d}t. \tag{4.13}$$

By constructing wavelets for any scale g, a Haar Wavelet basis can be constructed to approximate $\mu(t)$ to any desired detail level. While Haar wavelets are the simplest wavelets, many other wavelet functions exist. Two important wavelet bases are Daubechies wavelets and Daubechies' least asymmetric wavelets, "symlets" (Mallat, 1999). These wavelets do not exist in closed form but can be calculated recursively. In addition, the coefficient vector, μ_{gh}^*, is sparse if $\mu(t)$ is a smooth function (Mallat, 1999; Johnstone, 2013).

The basic plan for wavelet estimation of an unknown function is to first use the discrete wavelet transform (DWT). The DWT transforms a vector of data $\bar{\boldsymbol{Y}}$ from the time domain into the wavelet basis functions defined above (wavelet domain) via a linear transformation,

$$\bar{\boldsymbol{Y}}^* = \boldsymbol{W}\bar{\boldsymbol{Y}} \tag{4.14}$$

where $\boldsymbol{W}$ is the transform matrix defined by the integral transformation in (4.13) and $\bar{\boldsymbol{Y}}^*$ is the vector of wavelet coefficients. When the $\bar{\boldsymbol{Y}}$ are modeled as noisy observations of an unknown function, $\mu(t_l)$ with errors $\epsilon(t_l)$, the wavelet domain model is

$$\bar{Y}^*_{gh} = \mu^*_{gh} + \epsilon^*_{gh}$$

where μ^*_{gh} is the vector of wavelet coefficients for the unknown function, ϵ^*_{gh} is the wavelet transform of the noise, and all else is defined as previous (Johnstone, 2013). Assuming that $\boldsymbol{\epsilon} \sim N(\boldsymbol{0}, \boldsymbol{\Sigma})$, it is easily seen that $\boldsymbol{\epsilon}^* \sim N(\boldsymbol{0}, \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{W}^t)$. When $\boldsymbol{\Sigma} = \sigma^2\boldsymbol{I}$, Donoho and Johnstone (1995) show that the coefficient $\hat{\mu}^*_{gh}$ is estimated well with a threshold estimator, $\eta(\bar{Y}^*_{gh}, \varpi)$. A threshold estimator keeps all $\bar{Y}^*_{gh}$ that are large and sets all other $\bar{Y}^*_{gh}$ following a threshold rule. Many types of thresholds exist in the literature; however, we focus on the soft threshold, $\eta_S(\bar{Y}^*_{gh}, \varpi) = \text{sgn}(\bar{Y}^*_{gh})(|\bar{Y}^*_{gh}| - \varpi)_+$, and the hard threshold $\eta_H(\bar{Y}^*_{gh}, \varpi) = \bar{Y}^*_{gh} I(|\bar{Y}^*_{gh}| \geq \varpi)$. Here, $\text{sgn}(\cdot)$ takes on the sign of its argument, $(\cdot)_+$ is non-zero when its argument is positive, and $I(\cdot)$ is the indicator function.

Several techniques estimate $\hat{\varpi}$ from the data. The universal threshold $\hat{\varpi}_U = \hat{\sigma}\sqrt{2\log n}$ is the simplest choice. However, $\hat{\varpi}_U$ requires the noise to be i.i.d. normal. A more general threshold rule is derived using Stein's unbiased risk estimator (SURE) (Donoho and Johnstone, 1995; Johnstone and Silverman, 1997). The SURE method is based on finding the $\hat{\tau}$ that minimizes mean square error and selects the value $\hat{\varpi}$ that minimizes

$$\hat{H}(\varpi) = \hat{\sigma}^2 n + \sum_g \sum_h (\bar{Y}^{*,2}_{gh} \wedge \varpi^2) - 2\hat{\sigma}^2 I\{|\bar{Y}^*_{gh}| \leq \varpi\} \tag{4.15}$$

and

$$\hat{\varpi} = \arg\min \hat{H}(\varpi). \tag{4.16}$$

$\hat{\varpi}$ is used with either threshold estimator types.

In the presence of colored noise, Johnstone and Silverman (1997) modifies (4.16) to have a different threshold at each wavelet decomposition level, g. Colored or non-white noise is any case when $\boldsymbol{\Sigma} \neq \sigma^2 \boldsymbol{I}$. For example, the soft threshold estimator takes the form

$$\hat{\mu}^*_{gh} = \eta(\bar{Y}^*_{gh}, \varpi_g) = \text{sgn}(\bar{Y}^*_{gh})(|\bar{Y}^*_{gh}| - \varpi_j)_+, \tag{4.17}$$

where ϖ_g is the threshold at the g scale, and all else is defined as previous. Johnstone and Silverman (1997) adjust the threshold such that

$$\hat{\varpi}_g = \sigma_g \hat{\varpi}(\bar{Y}^*_g / \sigma_g)$$

for the levels in the specific wavelet decomposition. The modification of the threshold is the only adjustment needed for using a wavelet threshold estimator with colored noise because threshold estimation is a coordinate-wise operation (Johnstone and Silverman, 1997).

4.4 CROSS-SECTIONAL FPCA ESTIMATION

Estimation algorithms for FPCA fall into one of two categories: those that smooth the estimated covariance function, $\hat{\mathcal{K}}(t_l, t_{l'})$, or those that smooth the eigenfunctions, $\hat{\phi}_p(t_l)$. Since both methods are used in the estimation of our new methods, we discuss both. The estimation algorithms are introduced assuming that $\boldsymbol{\epsilon}_i(t_l) \sim N(\mathbf{0}, \sigma^2 \boldsymbol{I})$. In each case, we comment on the robustness of each algorithm to violations of the white noise assumption as physiological signals often violate the white noise assumption.

4.4.1 FPCA via smoothed covariance functions

This section follows the approached for covariance smoothed FPCA outlined in Yao et al. (2005). Recalling from (3.5) that

$$\widetilde{\boldsymbol{K}} = (N-1)^{-1} \sum_{i=1}^{N} \boldsymbol{X}_i \boldsymbol{X}_i^t$$

is a raw estimate of $\mathcal{K}(t_l, t_{l'})$ at the observed times t_l with the column vector $\boldsymbol{X}_i = X_i(t_l)$. Here, $X_i(t_l) = y_i(t_l) - \hat{\mu}(t_l)$. Because $E(\widetilde{\boldsymbol{K}}) = \boldsymbol{K} + \sigma^2 \boldsymbol{I}$, Yao et al. (2005) recommend estimating the smoothed covariance $\hat{\boldsymbol{K}}$ by replacing the diagonal elements K_{ll} with $\check{K}_{ll} = 1/2\big(K_{l-1,l} + K_{l,l+1}\big)$, where the $K_{l-1,l}$ and $K_{l,l+1}$ are the nearest off diagonal elements. At this point, $\check{K}_{ll}$ have $n-2$ elements lacking elements K_{11} and K_{nn}. Therefore, linear extrapolation is used to calculate these elements. Any surface estimator such as the sandwich smoother is used to smooth the matrix, $\tilde{\boldsymbol{K}}$, to yield a smoothed estimate, $\hat{\boldsymbol{K}}$.

Because $\hat{\boldsymbol{K}}$ is a discretized estimate of $\mathcal{K}(t, t')$, the integral eigenvalue equation (3.8) with a matrix eigenvalue equation

$$\hat{\boldsymbol{K}} \boldsymbol{\phi}_p = \lambda_p \boldsymbol{\phi}_p, \tag{4.18}$$

where $\hat{\boldsymbol{K}}$ is a discrete version of smoothed covariance, ϕ_p is the p^{th} eigenfunction evaluated at the same points as $\hat{\boldsymbol{K}}$, and λ_p is the p^{th} eigenvalue. Ramsay and Silverman (2005) state that (4.18) can be solved using any matrix PCA or eigenvalue software. However, the results must be converted back to the functional data form using interpolation. This approach to FPCA is relativity straight forward to implement provided that $\hat{\boldsymbol{K}}$ is easy to estimate. In the case of white noise, this is true. However, if this step is challenging such as with non-white noise error structures or non-smooth regions of $\mathcal{K}(t, t')$, it is more fruitful to apply regularization to the eigenfunctions instead.

4.4.2 FPCA via smoothed eigenfunctions

While many methods exist for FPCA with smoothed eigenfunctions, see Ramsay and Silverman (2005), we focus attention on adaptive sparse PCA (ASPCA) introduced by Johnstone and Lu (2009) both for computation efficiency and robustness to violations of white noise

and smoothness in the eigenfunctions. These are always concerns in physiological signal analysis. Considering a covariance matrix, $\boldsymbol{K}$, with eigenvectors, $\boldsymbol{\phi}_p$, and eigenvalues, λ_p, ASPCA estimates the P eigenvectors $\boldsymbol{\phi}_p$ by transforming the PCA algorithm from the data basis where $\boldsymbol{K}$ is not sparse to one where $\boldsymbol{K}^* = \boldsymbol{W}\boldsymbol{K}\boldsymbol{W}^t$ is sparse, where $\boldsymbol{W}$ is a transformation matrix. Orthonormal basis vectors, $\boldsymbol{e}_g$, represent both the data $\boldsymbol{X}_i$ and PCs as: $\boldsymbol{X}_i = \sum_g X^*_{g,i}\boldsymbol{e}_g$, where $X^*_{g,i}$ is the g^{th} basis coefficient and $\boldsymbol{X}_i$ and $\boldsymbol{e}_g$ are defined as previous; $\boldsymbol{\phi}_p = \sum_g \phi^*_{g,p}\boldsymbol{e}_g$, where $\phi^*_{g,p}$ is the g^{th} basis coefficient and all other terms defined as previous. Johnstone and Lu (2009) enforce sparsity not on $\boldsymbol{K}$ but via $\boldsymbol{\phi}_p$. When only a small number of the $\phi^*_{g,p}$ are non-zero, $\boldsymbol{K}$ and $\boldsymbol{\phi}_p$ have a sparse representation on the basis $\boldsymbol{e}_g$. Specifically, the basis coefficients are required to decay at rate

$$|\phi^*_{g,p}| \leq Cg^{-1/q}, \quad g = 1, 2, \ldots.$$

Often, a wavelet basis provides the needed sparse basis as the wavelet coefficients decay same rate for smooth functions (Johnstone, 2013).

Johnstone and Lu (2009) outline an algorithm that can be applied to estimate $\hat{\boldsymbol{\phi}}$. The algorithm is outlined below for a wavelet basis with basis functions, e_g, with all data observed at times, t_l.

1. *Transform the data.* For random functions, $X_i(t_l)$,

$$X_i(t_l) = \sum_{g=1}^{G} X^*_{i,g} e_g.$$

2. *Select coefficients.* Calculate G variances, $\hat{\sigma}^2_g = \text{var}(x^*_{i,g})$, for each coefficient. Find a subset, $\mathscr{G}_v$, of the v largest variances using one of the selection methods introduced in Johnstone and Lu (2009).
3. *PCA* Estimate the v eigenvectors the $\tilde{\boldsymbol{\phi}}^*_p$ where v is size of the reduced set of coefficients.
4. *Filter the PCs.* Johnstone and Lu (2009) recommend that a hard threshold to estimate $\hat{\phi}^*_{g,p} = \eta_H(\tilde{\phi}^*_{g,p}, \varpi_p)$ for $g \in \mathscr{G}_v$ to remove any remaining noise. Here, ϖ_p is an appropriate threshold for the p^{th} eigenvector that is based on the noise structure.

5. *PC reconstruction.* Transform the PCs back to the data domain by

$$\hat{\phi}_p(t_l) = \sum_{g \in \mathscr{G}_v} \hat{\phi}^*_{g,p} e_g.$$

6. *Selection of P.* Determine the number of PCs using fractional of variance explained (FVE), see Section 4.7.

Johnstone and Lu (2009) provide two data driven approaches for finding v. The first approach selects all coordinates where $\hat{\sigma}^2_g \geq \hat{\sigma}^2(1+\alpha)$ where $0 < \alpha < 1$, σ^2 is a measure of the total variability in $x^*_{i,g}$, and σ^2_g is defined previously. The second approach selects v by finding the number of coordinates g with a variance σ^2_g greater than the expected variance without any signal (Johnstone and Lu, 2009). Johnstone and Lu (2009) recommend estimating $\hat{\sigma}^2 = \text{median}(\hat{\sigma}^2_g)$ for each of the p PCs.

4.5 ESTIMATION OF RF-FPCA

RF-FPCA is started by estimating the mean surface $\hat{\mu}(t|s)$ from (3.15). Chen and Müller (2012) suggest two different estimation procedures for RF-FPCA depending on the longitudinal sampling. For dense, balanced longitudinal design, the raw mean estimate is

$$\tilde{\mu}(t_l|s_j) = \frac{1}{N} \sum_{i=1}^{N} y_i(t_l|s_j), \tag{4.19}$$

and raw covariance estimate is

$$\tilde{\mathcal{K}}(t_l, t_{l'}|s_j) = \frac{1}{N} \sum_{i=1}^{N} y_i(t_l|s_j) y_i(t_{l'}|s_j) - \tilde{\mu}(t_l|s_j)\tilde{\mu}(t_{l'}|s_j). \tag{4.20}$$

In the presence of high measurement error, the estimates, $\tilde{\mu}(t_l|s_j)$ and $\tilde{\mathcal{K}}(t_l, t_{l'}|s_j)$, can be smoothed with any of surface estimators from Section 4.2 can be used. In the case of sparse, irregular longitudinal designs, Chen and Müller (2012) recommends use a smoothing procedure for both $\hat{\mu}(t|s)$ and $\hat{\mathcal{K}}(t, t'|s)$. The estimates $\hat{\mathcal{K}}(t, t'|s)$ are fed into the smoothed covariance FPCA method from Section 4.4.1. Estimates of the PC and PC scores $\hat{\phi}_p(t|s)$ and λ_p are obtained. A working random process $\hat{\xi}_{ip}(s_j) = \int X_i(t|s_j)\hat{\phi}_p(t|s_j)\,\mathrm{d}t$ is estimated

by numerical integration. When the longitudinal is dense and regular, further smoothing is needed for the $\hat{\xi}_{ip}(s_j)$ when σ^2 is high. $\hat{\psi}_{qp}(s)$ and λ_{qp} are estimated by solving $\hat{\boldsymbol{K}}_p\boldsymbol{\psi}_{qp} = \lambda_{qp}\boldsymbol{\psi}_{qp}$, where $\hat{\boldsymbol{K}}_p = (N-1)^{-1}\sum_i \hat{\xi}_{ip}(s_j)\hat{\xi}_{ip}(s_{j'})$. In the sparse case, other FPCA algorithms such as PACE introduced by Yao et al. (2005) is used. PACE uses conditional expectation to calculate ζ_{iqp} (Yao et al., 2005).

4.6 ESTIMATION OF LFPCA

The estimation algorithm for LFPCA is similar to RF-FPCA. However, LFPCA estimates a marginal covariance $\hat{\mathcal{K}}(t,t')$ for the t direction instead of $\max(J_i)$ conditional covariances as in RF-FPCA. Park and Staicu (2015) proposes

$$\hat{\mathcal{K}}(t,t') = \frac{\sum_i \sum_j \left[y_i(t_l, s_{ij}) - \hat{\mu}(t_l, s_{ij})\right]\left[y_i(t_{l'}, s_{ij}) - \hat{\mu}(t_{l'}, s_{ij})\right]}{\sum_i J_i}.$$

Estimation of all other quantities for LFPCA follows the same algorithm as RF-FPCA.

4.7 DETERMINING THE SIZE OF A FPCA THE BASIS

The size of each FPCA basis can be selected in multiple ways, e.g., via AIC, BIC, or FVE (Yao et al., 2005). AIC and BIC are based on the likelihood of the data. In FPCA, we use FVE as it is easily calculated and interpreted. All quantities needed to calculate FVE are estimated during FPCA. The number of PCs is selected as the smallest number of components that explains a predetermined level of variance explained. For example, the number of PCs, P, for a FPCA is found by

$$\text{FVE}_p = \frac{\sum_{r=1}^{p} \lambda_r}{\sum_{r=1}^{L} \lambda_r},$$

where λ_r is the r^{th} eigenvalue of $\mathcal{K}(t,t')$, and L is the total number of eigenvalues.

5.0 NEW METHODOLOGY

5.1 DEVELOPMENT OF THE MULTISCALE FRAMEWORK AND MODEL

Assume that τ, t and s are continuous time variables and that the observed $D-$dimensional multivariate data, $\widetilde{\boldsymbol{Y}}_i(\tau)$, is a vector of noisy measurements on the underlying functional process, $\widetilde{\mathscr{F}_i}(\tau)$, that has a periodic component of period T, for $i = 1, \ldots, N$ subjects that are followed for J_i periods. The observed data is modeled as

$$\widetilde{\boldsymbol{Y}}_i(\tau) = \widetilde{\mathscr{F}_i}(\tau) + \tilde{\boldsymbol{\epsilon}}_i(\tau) \tag{5.1}$$

where $\tilde{\boldsymbol{\epsilon}}_i(\tau) \sim N(\boldsymbol{0}, \widetilde{\boldsymbol{\Sigma}})$, $\widetilde{\boldsymbol{\Sigma}}$ is the noise covariance matrix, τ is the argument of $\widetilde{\mathscr{F}}(\cdot)$, and everything else is defined above. We assume that $\widetilde{\boldsymbol{\Sigma}}$ can either be white noise, $\widetilde{\boldsymbol{\Sigma}} = \tilde{\sigma}^2\boldsymbol{I}$ or take on a colored noise structure where the correlations between two time points τ_1 and τ_2 decay at $|\tau_2 - \tau_1|^{-\alpha}$, where $\alpha \in (0, 1)$, or faster. Johnstone and Silverman (1997) term this power law decay of correlation as "long-range" dependence. We provide more details later on the structure of $\tilde{\boldsymbol{\Sigma}}$ on the transformed time domain.

Taking inspiration from the two-time solutions to non-linear ODEs, we formally define two new variables: slow time, $s = \lfloor \tau/T \rfloor$, which counts the number of periods observed, and fast time, $t = \tau - s * T$ (Fink et al., 1974; Strogatz, 1994). Here, $t \in \mathcal{T} = [0, T)$ and $s \in \mathcal{S} = [0, J_i]$. In this dissertation, we focus on the case where T is known yielding a conditional version of (5.1),

$$\boldsymbol{Y}_i(t, s|T) = \mathscr{F}_i(t, s|T) + \boldsymbol{\epsilon}_i(t, s|T), \tag{5.2}$$

where $\mathscr{F}_i(t, s|T)$ is the same function as $\widetilde{\mathscr{F}_i}(\tau)$ but transformed on a two dimensional domain and $\boldsymbol{\epsilon}_i(t, s|T) \sim N\big[\mathbf{0}, \boldsymbol{\Sigma}(t, t', s, s'|T)\big]$. Period detection can be accomplished using either a scientific motivation or using data driven statistical methods. After selection of the dominate period, we drop the $(\cdot|T)$ for notation simplicity.

Under the assumption of colored noise and for $J_i = 2$ and $D = 2$, we assume that

$$\boldsymbol{\Sigma}(t, t', s, s') = \begin{pmatrix} \boldsymbol{\Sigma}(t, t') & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}(t, t') \end{pmatrix}, \tag{5.3}$$

where

$$\boldsymbol{\Sigma}(t, t') = \begin{pmatrix} \boldsymbol{\Sigma}^{11}(t, t') & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}^{22}(t, t') \end{pmatrix}.$$

Finally, we require the correlations between an two time points t_1 and t_2 to decay at $|t_2 - t_1|^{-\alpha}$. This covariance structure assumes that noise is not correlated across days or across outcome components.

The two-time solution to a non-linear ODE approximates the solution by assuming the solution consists of a periodic fast time function, $\boldsymbol{F}(t)$, and a slow time function, $\boldsymbol{G}(s)$, that governs how $\boldsymbol{F}(t)$ evolves across multiple periods (Strogatz, 1994). For ODEs, periodic solutions for $\boldsymbol{F}(t)$ are represented as a sum of sines and cosines. $\boldsymbol{G}(s)$ is found by solving the averaged equations, which are the average of ODE and the sine/cosine solution of $\boldsymbol{F}(t)$ over one period. In the next section, we propose a statistical multiscale model analogous to the two-time solution of ODEs.

5.1.1 Multiscale Decomposition and Marginal FPCA

The analysis of $\mathscr{F}_i(t, s)$ focuses on two key features: the patient specific marginal fast time function, $\boldsymbol{F}_i(t)$, and patient specific marginal slow time evolution, $\boldsymbol{G}_i(s)$. Before building the full model, we introduce the patient specific average fast time random function

$$\boldsymbol{F}_i(t) = \frac{1}{J_i} \int_{\mathcal{S}} \mathscr{F}_i(t, s)\, \mathrm{d}s, \tag{5.4}$$

with mean function $E\big[\boldsymbol{F}_i(t)\big] = \boldsymbol{\mu}(t)$ and random effect $\boldsymbol{X}_i(t) = \boldsymbol{F}_i(t) - \boldsymbol{\mu}(t)$. In addition, we define the patient specific marginal slow time process as

$$\boldsymbol{G}_i(s) = \frac{1}{T}\int_{\mathcal{T}} \Big[\mathscr{F}_i(t,s) - \boldsymbol{F}_i(t)\Big]\,\mathrm{d}t, \tag{5.5}$$

with mean function $E\big[\boldsymbol{G}_i(s)\big] = \boldsymbol{\nu}(s)$ and random effect $\boldsymbol{U}_i(s) = \boldsymbol{G}_i(s) - \boldsymbol{\nu}(s)$. The marginal processes, $\boldsymbol{F}_i(t)$ and $\boldsymbol{G}_i(s)$ are connected to the full random function, $\mathscr{F}_i(t,s)$, through the marginal FPCA model introduced in Chen et al. (2016). For multivariate functional data, marginal FPCA assumes that

$$\mathscr{F}_i(t,s) = \boldsymbol{\mu}(t,s) + \sum_{p=1}^{\infty} \theta_{ip}(s)\boldsymbol{\phi}_p(t), \tag{5.6}$$

where $\boldsymbol{\mu}(t,s)$ is the mean surface, $\theta_{ip}(s)$ is a random function of slow time, and $\boldsymbol{\phi}_p(t)$ are the multivariate eigenfunctions.

To link (5.4),(5.5) and (5.6), we first represent $\boldsymbol{F}_i(t)$ in terms of the marginal kernel. Let $\boldsymbol{\mu}(t) = \frac{1}{J_i}\int_{\mathcal{S}} \boldsymbol{\mu}(t,s)\,\mathrm{d}s$ and $\xi_{ip} = \int_{\mathcal{S}} \theta_{ip}(s)\,\mathrm{d}s$, the marginal fast time process has the representation

$$\begin{aligned}
\boldsymbol{F}_i(t) &= \frac{1}{J_i}\int_{\mathcal{S}} \left[\boldsymbol{\mu}(t,s) + \sum_{p=1}^{\infty}\theta_{ip}(s)\boldsymbol{\phi}_p(t)\right]\mathrm{d}s \\
&= \frac{1}{J_i}\int_{\mathcal{S}} \boldsymbol{\mu}(t,s)\,\mathrm{d}s + \frac{1}{J_i}\int_{\mathcal{S}}\sum_{p=1}^{\infty}\theta_{ip}(s)\boldsymbol{\phi}_p(t)\,\mathrm{d}s \\
&= \boldsymbol{\mu}(t) + \frac{1}{J_i}\sum_{p=1}^{\infty}\xi_{ip}\boldsymbol{\phi}_p(t).
\end{aligned} \tag{5.7}$$

Now, we identify $\boldsymbol{X}_i(t) = \sum_{p=1}^{\infty}\xi_{ip}\boldsymbol{\phi}_p(t)$ with $E\big[\boldsymbol{X}_i(t)\big] = 0$ and associated covariance operator with square integrable kernel, $\mathrm{cov}\left[\boldsymbol{X}_i(t), \boldsymbol{X}_i(t')\right] = \mathcal{K}(t,t')$. By application of Mercer's Theorem, the covariance kernel has the singular value decomposition, $\mathcal{K}(t,t') = \sum_{p=1}\lambda_p\boldsymbol{\phi}_p(t)\boldsymbol{\phi}_p^t(t')$, where λ_p is the p^{th} eigenvalue and $\boldsymbol{\phi}_p(t)$ is the p^{th} eigenfunction. Furthermore, the Karhunen-Loève theorem states that PC scores have distribution, $\xi_{ip} \sim N(0,\lambda_p)$. Also, over one period, the fast time function has constant mean value

$$\boldsymbol{\mu} + \boldsymbol{X}_i = \frac{1}{T}\int_{\mathcal{T}}\left[\boldsymbol{\mu}(t) + \sum_{p=1}^{\infty}\xi_{ip}\boldsymbol{\phi}_p(t)\right]\mathrm{d}t, \tag{5.8}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{X}_i$ are $D \times 1$ columns vectors.

The slow time marginal process is linked to marginal FPCA by

$$\begin{aligned}\boldsymbol{G}_i(s) &= \frac{1}{T}\int_{\mathcal{T}} \left\{ \mathscr{F}_i(t,s) - \left[\boldsymbol{\mu}(t) + \boldsymbol{X}_i(t)\right] \right\} \mathrm{d}t \\ &= \frac{1}{T}\int_{\mathcal{T}} \left[\boldsymbol{\mu}(t,s) - \boldsymbol{\mu}(t)\right] \mathrm{d}t + \frac{1}{T}\int_{T} \sum_{p=1}^{\infty} \left[\theta_{ip}(s) - \xi_{ip}\right] \boldsymbol{\phi}_p(t)\, \mathrm{d}t \end{aligned} \tag{5.9}$$

To identify the model in (5.9), we must assume structures for both $\boldsymbol{\mu}(t,s)$ and $\theta_{ip}(s)$. At this point, we assume that both the mean structure, $\boldsymbol{\mu}(t,s)$, and $\theta_{ip}(s)$ have an additive structure. Accordingly, we assume that $\boldsymbol{\mu}(t,s) = \boldsymbol{\mu}(t) + \boldsymbol{\nu}(s)$ implying that $\int_{\mathcal{S}} \boldsymbol{\nu}(s)\, \mathrm{d}s = 0$. If $\int_{\mathcal{S}} \boldsymbol{\nu}(s)\, \mathrm{d}s \neq 0$, then the marginal process in (5.4) does not result from the integral over s. In the case of our VAD circadian cycle data, we make the additional assumption that $\boldsymbol{\nu}(s) = 0$. The zero slow time mean assumption is reasonable because a stable patient is believed to have no slow time variation in his/her circadian cycle. Similarly, we assume that $\theta_{ip}(s) = \xi_{ip} + U_{ip}(s)$, where $U_{ip}(s)$ is a mean zero random process with covariance kernel $\mathcal{K}_p(s,s')$ and ξ_{ip} is defined as previously. Also, we assume that ξ_{ip} and $U_{ip}(s)$ are independent, and $U_{ip}(s)$ is restricted such that $\int_{\mathcal{S}} U_{ip}(s)\, \mathrm{d}s = 0$. Note that the random functions $U_{ip}(s)$ contain all information about how the circadian cycles evolve in slow time.

Therefore, both the fast time and slow time components are modeled with

$$\mathscr{F}_i(t,s) = \boldsymbol{\mu}(t) + \sum_{p=1}^{\infty} \left[\xi_{ip} + U_{ip}(s)\right] \boldsymbol{\phi}_p(t), \tag{5.10}$$

where all terms are defined as previous. The $\sum_p \xi_{ip}\boldsymbol{\phi}_p(t)$ term models the subject specific deviation circadian cycle from the mean circadian cycle, and the term $\sum_p U_{ip}(s)\boldsymbol{\phi}_p(t)$ describes the longitudinal evolution of the circadian cycle.

The slow time functions, $U_{ip}(s)$, are modeled with a second FPCA expansion such that $\mathcal{K}_p(s,s') = E\left[U_{ip}(s)U_{ip}(s')\right] = \sum_q \lambda_{qp}\psi_{qp}(s)\psi_{qp}(s')$ and

$$U_{ip}(s) = \sum_{q=1}^{\infty} \zeta_{iqp}\psi_{qp}(s), \tag{5.11}$$

where $\psi_{qp}(s)$ are the eigenfunctions of $\mathcal{K}_p(s,s')$, Q is the number of terms in the basis expansion, and $\zeta_{ipq} \sim N(\mathbf{0}, \lambda_{qp})$ where λ_{qp} are the ordered eigenvalues of $\mathcal{K}_p(s,s')$ with $\lambda_{1p} > \lambda_{2p} > \cdots > \lambda_{\infty,p} > 0$.

Combining equations (5.10) and (5.11), the complete data is modeled as

$$\boldsymbol{Y}_i(t,s) = \boldsymbol{\mu}(t) + \sum_{p=1}^{\infty} \xi_{ip}\boldsymbol{\phi}_p(t) + \sum_{p=1}^{\infty}\sum_{q=1}^{\infty} \zeta_{ipq}\psi_{pq}(s)\boldsymbol{\phi}_p(t) + \epsilon_i(t,s), \tag{5.12}$$

where all terms have been previously defined. In practice, the infinite sums are truncated to P terms for the fast time scale FPCA and Q_p terms for each of the slow time scale FPCAs using FVE. Here, P and Q_p are small positive integers often less than 10.

5.2 ESTIMATION AND INFERENCE

Fitting of a MMFPCA model proceeds in a different manner compared to the existing models, RF-FPCA, LFPCA and the marginal FPCA, because of the noise structure (Chen et al., 2016; Chen and Müller, 2012; Park and Staicu, 2015). We first estimate a fast time marginal random function, its mean, $\hat{\boldsymbol{\mu}}(t_l)$, and covariance structure, $\hat{\mathcal{K}}(t_l, t_{l'})$. After then conducting MFPCA on $\hat{\mathcal{K}}(t_l, t_{l'})$, we obtain both estimated marginal fast time PC scores and estimated daily fast time PC scores. At this stage, these daily fast time scores are used to estimate the slow time FPCA models.

For the i^{th} patient, suppose we observe data, $\boldsymbol{Y}_i(t_l, s_{ij})$, sampled at times (t_l, s_{ij}) where $l = 1, \ldots, n$, and n is number of samples/day. Also, $j = 1, \ldots, J_i$, where J_i is numbers days observed for patient i, with $i = 1, \ldots, N$. For a given day, s_{ij}, $\boldsymbol{Y}_i(t_l, s_{ij})$ is a $Dn \times 1$ vector with each outcome component stacked vertically. The next step entails defining the patient specific marginal observation vector as $\boldsymbol{Z}_i(t_l) = \frac{\sum_{s=1}^{J_i} \boldsymbol{Y}_i(t_l, s_{ij})}{J_i}$.

Assume that

$$\begin{aligned} \boldsymbol{Z}_i(t_l) &= \boldsymbol{F}_i(t_l) + \bar{\epsilon}_i(t_l) \\ &= \boldsymbol{\mu}(t_l) + \boldsymbol{X}_i(t_l) + \bar{\epsilon}_i(t_l) \end{aligned} \tag{5.13}$$

where $\boldsymbol{F}_i(t_l)$ is the underlying process at time point, t_l, and $\bar{\epsilon}_i(t)$ is the measurement error process averaged across days. Furthermore, it is easily seen that $E\big[\boldsymbol{Z}_i(t_l)\big] = \boldsymbol{\mu}(t_l)$. Under the assumption of white noise, the vector, $\boldsymbol{X}_i(t_l) = \boldsymbol{Z}_i(t_l) - \boldsymbol{\mu}(t_l)$, has distribution,

$$\boldsymbol{X}_i(t_l) \sim N\left[\boldsymbol{0}, \mathcal{K}(t_l, t_{l'}) + \frac{1}{J_i}\boldsymbol{\Sigma}(t_l, t_{l'})\right].$$

Here, $\mathcal{K}(t_l, t_{l'})$ is the marginal fast time covariance kernel evaluated at the sample times, t_l, and $\boldsymbol{\Sigma}(t_l, t_{l'})$ is the error covariance matrix. The slow time information is contained in the daily PC scores

$$U_{ip}(s_{ij}) = \int_{\mathcal{T}} \left[\boldsymbol{Y}_i(t_l, s_{ij}) - \boldsymbol{F}_i(t_l)\right] \hat{\boldsymbol{\phi}}_p(t_l)\, \mathrm{d}t, \tag{5.14}$$

where $\boldsymbol{Y}_i(t_l, s_{ij})$ is the observed data, $\boldsymbol{F}_i(t_l)$ is the underlying fast time marginal process and $\boldsymbol{\phi}_p(t_l)$ is the p^{th} eigenfunction of $\mathcal{K}(t_l, t_{l'})$. With the new random functions, the estimation procedure is organized into eight main steps as outlined below.

1. Calculate $\hat{\boldsymbol{Z}}_i(t_l) = \dfrac{\sum_{s_{ij}=1}^{J_i} \boldsymbol{Y}_i(t_l, s)}{J_i}$ from the raw data.
2. Estimate $\hat{\boldsymbol{\mu}}(t_l)$ using non-parametric regression. (Details are in Section 5.2.1.)
3. Calculate $\hat{\boldsymbol{X}}_i(t_l) = \hat{\boldsymbol{Z}}_i(t_l) - \hat{\boldsymbol{\mu}}(t_l)$.
4. Conduct FPCA on $\hat{\boldsymbol{X}}_i(t_l)$ and estimate $\hat{\boldsymbol{F}}_i(t_l) = \hat{\boldsymbol{\mu}}(t_l) + \sum_p \hat{\xi}_{ip}\hat{\boldsymbol{\phi}}_p(t_l)$. (Details in Section 5.2.1.)
5. Calculate $\hat{U}_{ip}(s_{ij}) = \int_{\mathcal{T}} \left[\boldsymbol{Y}_i(t_l, s_{ij}) - \hat{\boldsymbol{F}}_i(t_l)\right] \hat{\boldsymbol{\phi}}_p(t)\, \mathrm{d}t$.
6. Conduct FPCA on $\hat{U}_{ip}(s_{ij})$. (Details in Section 5.2.2.)
7. Reconstruct $\hat{\boldsymbol{Y}}_i(t_l, s_{ij})$.
8. Estimate confidence bands on estimates with functional bootstrap. (Details in Section 5.2.4.)

5.2.1 Estimating the Average Circadian Cycles - Fast Time Functions

The first step to apply MMFPCA to a dataset such as the VAD circadian cycle application, is to estimate both the population average circadian cycle $\hat{\boldsymbol{\mu}}(t_l)$ and the subject specific circadian cycles, $\hat{\boldsymbol{F}}_i(t_l)$, along with the additional model parameters for the fast time FPCA. Because we assume that the noise structure may have long-range dependence, we use wavelets for the fast time scale estimation as wavelets easily handle this type of noise structure (Johnstone and Silverman, 1997). Additionally, wavelets are optimal for any jumps in the data, which may happen as these patients are quite ill. This desirable property stems from the nature of the wavelet multi-resolution basis.

A multi-resolution basis represents the d^{th} component, $d = 1, 2$, of a multivariate function, $X_i^{(d)}(t_l)$, at a set of scale functions, e_g, and wavelets, e_{gh}. The scale functions capture the

average behavior of $X_i^{(d)}(t_l)$ at the g^{th} scale. At the g^{th} scale, a wavelet, e_{gh}, captures the changes in $X_i^{(d)}(t_l)$ at an interval around the h^{th} location. This construction isolates any pathological point in $X_i^{(d)}(t_l)$ to the wavelet coefficients in a small neighborhood around its location. Finally, all smoothing is conducted on the mean and covariance estimates to reduce the amount of smoothing needed and the computational burden (less smoothing steps), and to ensure that the covariance estimates are improved.

The wavelet domain fast time objective function and "raw" estimates. We chose to use symlets with four vanishing moments for the wavelet basis. On this basis, smooth functions $\boldsymbol{X}_i(t_l)$ have a sparse coefficient vector $\boldsymbol{X}_i^* = \boldsymbol{W}\boldsymbol{X}_i(t_l)$ where $\boldsymbol{W}$ is the discrete wavelet transformation (DWT) matrix that is applied to each of the $D = 2$ components separately. Also, its covariance $\boldsymbol{K}^* = \boldsymbol{W}\mathcal{K}(t_l, t_{l'})\boldsymbol{W}^t$ assumed to sparse on the wavelet domain. The transformed noise covariance is $\boldsymbol{\Sigma}^* = \boldsymbol{W}\boldsymbol{\Sigma}(t_l, t_{l'})\boldsymbol{W}^t$. For the data observed at points t_l, we define the following quantities on the wavelet domain: the transformations of the data, $\boldsymbol{Z}_i^* = \boldsymbol{W}\boldsymbol{Z}_i(t_l)$, the mean functions, $\boldsymbol{\mu}^* = \boldsymbol{W}\boldsymbol{\mu}(t_l)$, and the transformed eigenfunctions $\boldsymbol{\phi}_p^* = \boldsymbol{W}\boldsymbol{\phi}_p(t_l)$. Now, estimates on the wavelet domain are obtained by minimizing the least squares objective function,

$$H^{ft} = \sum_{i=1}^{N} \left[\boldsymbol{Z}_i^* - \left(\boldsymbol{\mu}^* + \sum_{p=1}^{P} \xi_{ip} \boldsymbol{\phi}_p^* \right) \right]^t \left[\boldsymbol{Z}_i^* - \left(\boldsymbol{\mu}^* + \sum_{p=1}^{P} \xi_{ip} \boldsymbol{\phi}_p^* \right) \right], \tag{5.15}$$

where P is determined by fraction of variance explained (FVE), see 5.2.3. Using standard normal theory, H^{ft}, is minimized with the estimates

$$\hat{\boldsymbol{\mu}}_{raw}^* = \frac{1}{N} \sum_{i=1}^{N} \hat{\boldsymbol{Z}}_i^*, \tag{5.16}$$

and

$$\hat{\boldsymbol{K}}^* = \boldsymbol{W} \left[\mathcal{K}(t_l, t_{l'}) + \frac{1}{\bar{J}} \boldsymbol{\Sigma}(t_l, t_{l'}) \right] \boldsymbol{W}^t = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{X}_i^* \left(\boldsymbol{X}_i^* \right)^t, \tag{5.17}$$

where $\boldsymbol{X}_i^* = \boldsymbol{Y}_i^* - \boldsymbol{\mu}^*$ and $\bar{J}$ is the average slow time follow-up length across all patients (Anderson, 2003).

Smoothing the population average circadian cycles. To smooth the mean estimates, the threshold wavelet regression estimator introduced in Johnstone and Silverman (1997) is used on each component separately. First, the empirical mean estimate from (5.16) is modeled as

$$\hat{\boldsymbol{\mu}}^*_{raw} = \boldsymbol{\mu}^* + \boldsymbol{\rho}, \tag{5.18}$$

where $\rho_{gh} \sim N\big[0, K^*_{gh,g'h'}/N + \Sigma^*_{gh,g'h'}/(N\bar{J})\big]$ is the wavelet transform of the noise. In (5.18), all the columns vectors have $\dim \hat{\boldsymbol{\mu}}^*_{raw} = \dim \boldsymbol{\mu}^* = \dim \boldsymbol{\rho} = Dn^*$, where n^* is the total number of wavelet coefficients. As $\boldsymbol{K}^*/(N) + \boldsymbol{\Sigma}^*/(N\bar{J})$ is not a diagonal matrix, the variances of the wavelet coefficients μ^*_g are not guaranteed to be equal for any two levels. In this case, the level-dependent threshold estimator of Johnstone and Silverman (1997) can be directly applied to estimate $\hat{\boldsymbol{\mu}}(t_l)$. We follow their recommendation and estimate the mean functions with a soft threshold estimator, $\hat{\mu}^*_{gh} = \text{sgn}(\hat{\mu}^*_{gh,raw})(|\hat{\mu}^*_{gh,raw}| - \varpi_g)_+$, where ϖ_g is the level dependent threshold, $\text{sgn}(\cdot)$ takes the sign of its argument, and $(\cdot)_+ = 0$ when its argument is negative and takes on the value of its argument otherwise. On the time domain, the estimated circadian cycles are found by back transforming the $\hat{\mu}^*_{gh}$ such that

$$\hat{\boldsymbol{\mu}}(t_l) = \boldsymbol{W}^t \hat{\boldsymbol{\mu}}^*. \tag{5.19}$$

FPCA expansion of the circadian cycles. Next, we consider estimation of the fast time eigenfunctions $\boldsymbol{\phi}_p(t_l)$, eigenvalues λ_p, and PC scores ξ_{ip}. Unlike the mean estimation, the estimation of covariance of the multivariate data is quite challenging. A solution is to combine the multivariate FPCA (MFPCA) algorithm introduced by Happ and Greven (2015) with adaptive sparse PCA (ASPCA) of Johnstone and Lu (2009). Compared to other algorithms that estimate the multivariate FPCA in a single step, the advantage of using MFPCA is it provides a rigorous approach to combine univariate FPCA without requiring the same type of regularization on each of the univariate FPCAs. For our motivating data, MFPCA allows us to focus on the selection of appropriate univariate FPCA algorithm, ASPCA. The rest of this section briefly overviews MFPCA and ASPCA, applies them to MMFPCA, considers selection of P, and the estimation of the PC scores.

For each of the d components, the estimates, $\hat{\boldsymbol{K}}^{*,(d)}$, are noisy as the $X^{*,(d)}_{il}$ are not smoothed. Therefore, the univariate FPCA algorithm should be able to filter out the noise

and be robust to violations of the white noise assumption. Also, since each FPCA problem is high dimensional, the chosen algorithm needs to regularize the estimates (Johnstone and Lu, 2009). In the case when $\boldsymbol{K}$ has a sparse representation in a basis, ASPCA both regularizes the estimates by conducting PCA on a covariance matrix of the basis coefficients with the largest variance (Johnstone and Lu, 2009). Additional filtering can be on the resulting eigenvectors. In the case of MMFPCA, the wavelet basis provides a basis on which $\boldsymbol{K}^{*,(d)}$ is sparse. Therefore, ASPCA is used to estimate $\phi_p^{(d)}(t_l)$ and $\lambda_p^{(d)}$.

MFPCA uses the structure of $\mathcal{K}(t_l, t_{l'}) = E\big[\boldsymbol{X}_i(t_l)\boldsymbol{X}_i^t(t_{l'})\big]$ to efficiently solve the multivariate eigenvalue that determines $\boldsymbol{\phi}_p(t_l)$ (Happ and Greven, 2015). MFPCA combines the results of D univariate FPCA through the correlation of the univariate PC scores. Happ and Greven (2015) purpose to estimate the multivariate eigenfunctions and PC scores by weighting the respective univariate estimates by the eigenvectors of $\hat{\boldsymbol{\kappa}}$. $\hat{\boldsymbol{\kappa}}$ is the PC score covariance matrix. Here, $\hat{\boldsymbol{\kappa}} = N^{-1}\hat{\boldsymbol{\Xi}}^t\hat{\boldsymbol{\Xi}}$ where $\dim(\boldsymbol{\Xi}) = N \times \sum_d P_d$ with the rows of $\boldsymbol{\Xi}_i = \big(\hat{\xi}_{i1}^{(1)}, \hat{\xi}_{i2}^{(1)}, \ldots, \hat{\xi}_{iP_1}^{(1)}, \ldots, \hat{\xi}_{i1}^{(D)}, \hat{\xi}_{i2}^{(D)}, \ldots, \hat{\xi}_{iP_D}^{(D)}\big)$. The the PC scores can be reordered by permuting columns to match the ordering of the eigenvalues output from our eigenvalue software. Because this column permutation is conducted via right multiplication with a permutation matrix, the MFPCA method can proceed without other changes as long as the new order is carried throughout the algorithm (Happ and Greven, 2015). Now, the multivariate eigenfunctions and FPCA scores are given by $\boldsymbol{\phi}_p^{(d)}(t_l) = \sum_{r=1}^{P_d}[\hat{\boldsymbol{c}}_p]_r^{(d)}\phi_r^{(d)}(t_l)$ and $\hat{\xi}_{ip} = \sum_{d=1}^{D}\sum_{r=1}^{P_d}[\hat{\boldsymbol{c}}_p]_r^{(d)}\hat{\xi}_{ir}^{(d)}$ (Happ and Greven, 2015), where $[\hat{\boldsymbol{c}}_p]_r^{(d)}$ is a subset of the p^{th} eigenvector of $\hat{\boldsymbol{\kappa}}$ associated with the d^{th} outcome component. For each subject, i, the multivariate predicted fast time function is $\hat{\boldsymbol{F}}_i(t_l) = \boldsymbol{\mu}(t_l) + \sum_{p=1}^{P}\hat{\xi}_{ip}\boldsymbol{\phi}_p(t_l)$, and univariate component-wise predictions are $\hat{F}_i^{(d)}(t_l) = \mu^{(d)}(t_l) + \sum_{p=1}^{P_d}\hat{\xi}_{ip}^{(d)}\phi_p^{(d)}(t_l)$.

5.2.2 Slow Time Scale Estimation

After obtaining the fast time model, the next step is to estimate the slow time random functions $U_{ip}(s_{ij})$ for each subject and multivariate principal component $\boldsymbol{\phi}_p(t_l)$. These estimates are found by adapting the univariate score combiner from Happ and Greven (2015). For each component, d, an estimate of the working random process at observed days, s_j, is obtained

by numerical integration of

$$\hat{U}_{ip}^{(d)}(s_j) = \int_{\mathcal{T}} \left[y_i^{(d)}(t_l, s_j) - \hat{F}_i^{(d)}(t_l) \right] \hat{\boldsymbol{\phi}}_p(t_l)\, \mathrm{d}t \tag{5.20}$$

for each day, s_j. At this stage, the D random processes are correlated so one calculates the multivariate daily FPCA score

$$\hat{U}_{ip}(s_j) = \sum_{d=1}^{D} \sum_{r=1}^{P_d} [\hat{\boldsymbol{c}}_p]_r^{(d)} \hat{U}_{ir}^{(d)}(s_j). \tag{5.21}$$

These estimates are now used to estimate the P multivariate slow time random process.

Armed with estimates, $\hat{U}_{ip}(s_{ij})$, the slow time model is estimated using one of several FPCA algorithms depending on the slow time sampling of the follow-up. Assuming that $\hat{U}_{ip}(s_{ij}) = U_{ip}(s_{ij}) + \epsilon_{ip}(s_{ij})$ where $\epsilon_{ip}(s_{ij}) \sim N(0, \sigma_p^2)$, the choice of FPCA algorithms depends on the longitudinal sampling. We present four important cases of the longitudinal sampling. The first case is dense regular follow-up, i.e. $J_1 = J_2 = \cdots = J_N = J$, and measurements are observed at the same time points. The second case is dense irregular follow-up with no missing data. The third case is dense regular or irregular follow-up with missing data. The final case is sparse irregular follow-up when subjects have different lengths of follow-up J_i and random gaps between each s_{ij}. This dissertation focuses on the dense regular case as the sparse irregular case is not present in the motivating data. When a sparse irregular case is encountered, we recommend application of the PACE algorithm following the work of Yao et al. (2005); Park and Staicu (2015).

In the dense regular case with slow time observations $\left\{s_j\right\}_{j=1}^{J}$, each subject has p data vectors of length J, $\hat{U}_{ip}(s_j) = \left[\hat{U}_{ip}(s_1), \hat{U}_{ip}(s_2), \ldots, \hat{U}_{ip}(s_J)\right]^t$. For these data, the observed slow time objective functions are

$$H_p^{st} = \sum_{i=1}^{N} \left[\hat{U}_{ip}(s_j) - \sum_{q=1}^{Q_p} \zeta_{ipq} \psi_{pq}(s_j) \right] \left[\hat{U}_{ipj} - \sum_{q=1}^{Q_p} \zeta_{ipq} \psi_{pq}(s_j) \right]^t, \tag{5.22}$$

where all functions are evaluated at the observed days s_j and Q_p is number of basis components as determined by FVE.

Each H_p^{st} is minimized when $\hat{\psi}_{pq}(s_j)$ are the eigenfunctions of $\boldsymbol{K}_p = \mathcal{K}_p(s_j, s_{j'})$ and $\hat{\zeta}_{ipq} = \int \hat{U}_{ip}(s_j)\hat{\psi}_{pq}(s_j)\,\mathrm{d}s$ where the integral is evaluated with numerical integration Ramsay and Silverman (2005, Chapter 8). An estimate of $\boldsymbol{K}_p$ is the raw covariance $\tilde{\boldsymbol{K}}_p = (N-1)^{-1}\sum_i \hat{\boldsymbol{U}}_{ip}\hat{\boldsymbol{U}}_{ip}^t$, which has expectation $E(\tilde{\boldsymbol{K}}_p) = \boldsymbol{K}_p + \sigma^2\boldsymbol{I}$. Following the discussion of covariance smoothing in Yao et al. (2005), we exclude the diagonal elements before smoothing. A 2-D Gaussian kernel density estimator is then used to obtain smooth estimates $\hat{\boldsymbol{K}}_p$. $\hat{\psi}_{qp}(s)$ and $\hat{\lambda}_{qp}$ are estimated using a standard software for finding eigenvalues and eigenvectors a matrix. Since the s_j's are one unit apart, no weighting to convert back to the functional space is needed. The $\hat{\zeta}_{ipq}$ are found using numerical integration. As part of signal reconstruction, predict $\tilde{U}_{ip}(s) = \sum_q \hat{\zeta}_{ipq}\psi_{qp}$.

Subject Specific Prediction. After estimating both the fast time and slow time models, subject specific predicted circadian cycles and evolution are estimated as

$$\hat{\boldsymbol{Y}}_i(t_l, s) = \hat{\boldsymbol{\mu}}(t_l) + \sum_p \left[\hat{\xi}_{ip} + \sum_q \hat{\zeta}_{iqp}\hat{\psi}_{qp}(s_j)\right]\hat{\boldsymbol{\phi}}_p(t_l). \tag{5.23}$$

5.2.3 Selection of Number of Principal Components

The size of each FPCA basis can be selected in multiple ways, e.g., via AIC, BIC, or FVE (Yao et al., 2005). In MMFPCA, FVE is used as it is easily calculated and interpreted. The number of PCs is selected as the smallest number of components that explains a predetermined level of variance explained. For example, the number of PCs, P_d, for d^{th} fast time FPCA is found by

$$\mathrm{FVE}_p^{(d)} = \frac{\sum_{r=1}^{p}\lambda_r^{(d)}}{\sum_{r=1}^{L}\lambda_r^{(d)}},$$

where $\lambda_r^{(d)}$ is the r^{th} eigenvalue of $\mathcal{K}^{(d)}(t_l, t_{l'})$, and L is the total number of eigenvalues. All other P's and Q's are determined in the same manner.

5.2.4 Bootstrap Confidence Intervals for the Fast Time Functions

In any study of repeated circadian cycle data, it is of scientific importance to determine the statistical significance of the fast time estimates. Slow time statistical inference contains several additional complications due to the number of slow time FPCA estimation steps and is discussed in future work. We declare that a circadian cycle is statistically significant if a global confidence band around the estimate has any region that does not overlap another region.

Both point-wise and global $(100-\alpha)\%$ confidence intervals are estimated using a non-parametric bootstrap (Crainiceanu et al., 2012). The N subjects are resampled with replacement B times creating the bootstrap samples. Here, a subject either contributes all of his/her data for both functional components or none. For the fast time mean functions, the point-wise bootstrap CI's are found using the normal confidence intervals, $\boldsymbol{\mu}(t_l) \pm z_{1-\alpha} \times \text{SD}(\boldsymbol{\mu}(t_l))$ on the wavelet domain. The time domain CI is found by transforming the CIs back to the time domain. In functional data, global CIs are not found as simply as the point-wise CIs. An adjustment for the number of observed time points must be made using any multiple comparison method (Pini and Vantini, 2016; Krafty and Collinge, 2013). In the case of our VAD data, we adjust the $\alpha-$level using the Bonferroni correction for simplicity leading to an $\alpha = \alpha_0/n$ where n is the number of fast time observations and α_0 point-wise $\alpha-$level. A similar approach is followed for the $\boldsymbol{\phi}_p(t_l)$. However, we recommend the use the quantile based CIs on the time domain because normality is not guaranteed. However, the calculations are conducted on the time domain.

6.0 SIMULATION STUDIES

We evaluated performance of MMFPCA using a simulation study designed to provide insight into the behavior of MMFPCA in capturing both periodic fast time behavior such as circadian cycles and assess the variability of this behavior over the slow time component. Several combinations of sample size, longitudinal follow-up, sampling rate per day, mean functions, and covariance parameters were considered. The first simulation scenario was designed to mimic our VAD data in shape of population average function as fast time sampling. In further simulations the number of subjects, fast time sampling rate, and length of longitudinal follow-up were increased to understand how MMFPCA depends on samples sizes. Additionally, the measurement error was simulated with both low noise and high noise. All simulations were conducted with $C = 250$ replications.

The stimulated data was generated with the following parameter values: 128 and 256 fast time sample time points, longitudinal follow-up of 30 and 250 days, two outcome components, with each component having a two-dimensional fast time basis, and a one dimensional slow time basis. Because $P_1 = P_2 = 2$, the multivariate fast time basis has $P = 4$. The measurement errors were simulated from normal distributions with variances, $\tilde{\sigma}$, 0.001, 0.5, 1, and 1.5. The population average function was

$$\boldsymbol{\mu}(t) = \begin{pmatrix} -5\sin\left(\frac{2\pi t}{T}\right) \\ -2.5\sin\left(\frac{2\pi t}{T}\right) \end{pmatrix}. \tag{6.1}$$

A two dimensional fast time orthonormal basis consisting of two functions functions was generated as seen in Figure 8. A one dimensional slow time orthonormal basis was generated as

$\psi(s) = -s/J + 0.5$. The fast FPCA scores were generated from bivariate normal distribution with

$$\mathbf{\Lambda}_1 = \begin{pmatrix} 4 & 1 \\ 1 & 2 \end{pmatrix}, \quad \mathbf{\Lambda}_2 = \begin{pmatrix} 3 & 0.5 \\ 0.5 & 1 \end{pmatrix},$$

with $\mathbf{\Lambda}_p$ defined to be the correlation matrix of the p^{th} principal component. For example, the first principal component has a correlation magnitude of 1. The slow time scale FPCA scores were sampled from $N(0, \lambda_{pq})$ with $\lambda_{1p} = 16$. Furthermore, both fast PC components have the same slow time evolution. For each combination of these simulation parameters, sample size of 30, 100, and 250 subjects were considered.

The data was simulated via the procedure outline below.

1. Simulate the mean functions in (6.1) for all J days and N subjects.
2. Simulate the fast time random functions, $X_i^{(d)}(t_l) = \sum_{p=1}^{P} \xi_{ip}^{(d)} \boldsymbol{\phi}_p^{(d)}(t_l)$. First, we draw N random samples from $N(\mathbf{0}, \mathbf{\Lambda}_1)$ and $N(\mathbf{0}, \mathbf{\Lambda}_2)$ for the $(\xi_{i1}^{(1)}, \xi_{i1}^{(2)})^t$ an $(\xi_{i2}^{(1)}, \xi_{i2}^{(2)})^t$ PC scores respectively. Then, the d^{th} component of $\boldsymbol{X}_i(t_l)$ is calculated by

$$X_i^{(d)}(t) = \sum_{p=1}^{2} \xi_{ip}^{(d)} \phi_p^{(d)}(t_l). \tag{6.2}$$

3. Simulate the four slow time random functions, $U_i(s_j) = \sum_{q=1}^{1} \zeta_{iq} \psi_q(s)$. Again, we draw N random variables, ζ_i from $N(0, 16)$. Then, we calculate the N slow time functions $U_i(s_j) = \zeta_i \psi(s)$. Next,

$$\sum_{p=1}^{2} U_i(s_j) \phi^{(d)}(t_l) \tag{6.3}$$

   mixes the fast and slow time scales.
4. Add measurement error by drawing $\epsilon_i(t_l, s_j)$ from $N(0, \tilde{\sigma}^2)$.
5. Simulated data for the i^{th} patient, $\boldsymbol{Y}_i(t_l, s_j)$, is calculated by summing (6.1), (6.2), (6.3), and the noise from item 4.

Estimator performance is measured with Monte Carlo mean integrated square error

$$\text{RMISE}(\theta(t)) = C^{-1} \sum_c \frac{\int_{\mathcal{T}} \left[\hat{\theta}_c(t) - \theta(t)\right]^2 \mathrm{d}t}{\int_{\mathcal{T}} \left[\theta(t)\right]^2 \mathrm{d}t}$$

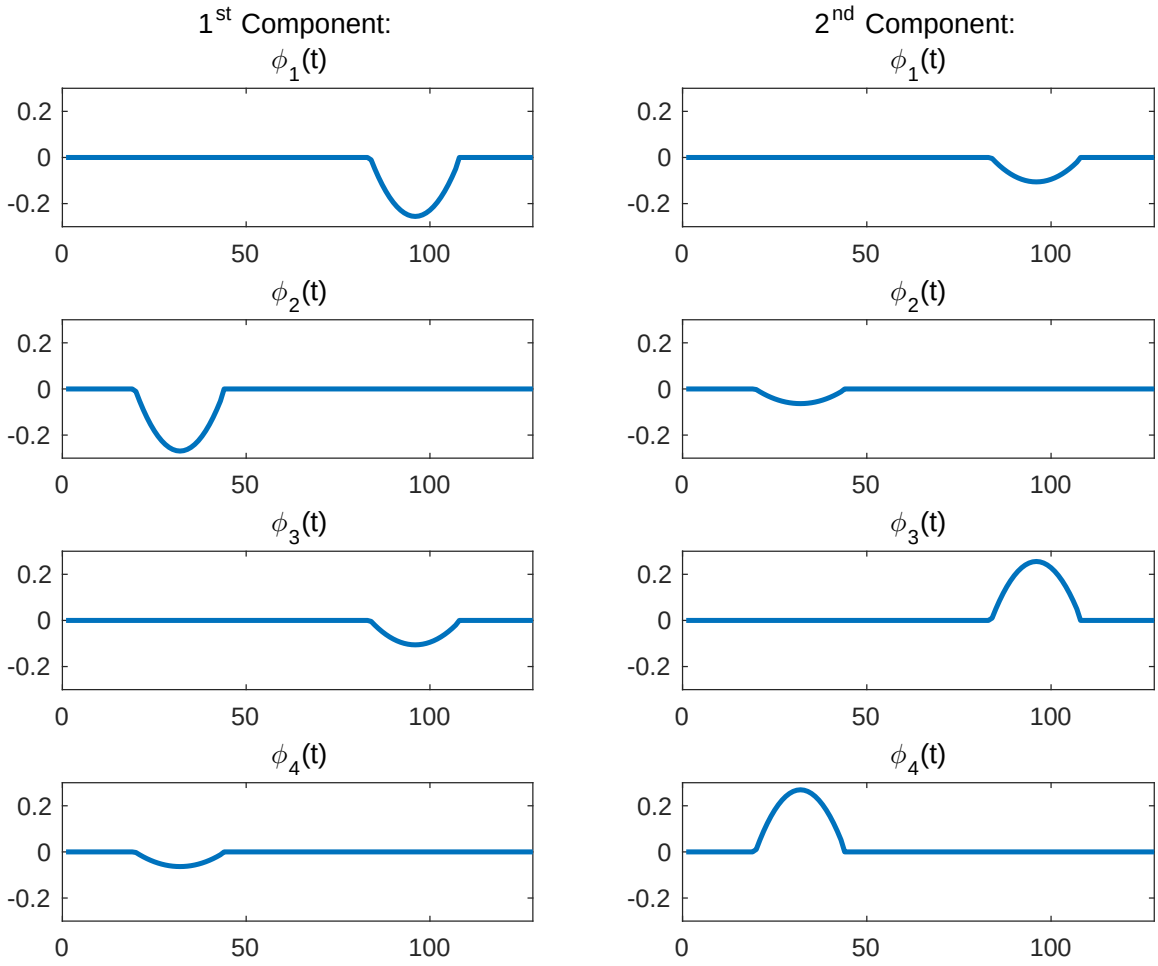

Figure 8: The fast time scale basis functions for the simulation studies.

for any function $\theta(t)$ and mean square error of the eigenvalues λ

$$\text{RMSE}(\lambda) = C^{-1} \sum_{c} \frac{(\hat{\lambda}_c - \lambda)^2}{\lambda^2}.$$

Table 1: 1000 RMISE for simulated fast time mean with 128 fast time samples.

		$\mu^{(1)}(t)$			$\mu^{(2)}(t)$		
		N = 30	N = 100	N=250	N = 30	N = 100	N=250
$J = 30$	$\tilde{\sigma} = 0.001$	0.1571	0.0523	0.0221	0.3003	0.0856	0.0340
	$\tilde{\sigma} = 0.5$	0.2706	0.0804	0.0367	0.3647	0.1255	0.0544
	$\tilde{\sigma} = 1$	0.6607	0.1824	0.0794	0.7747	0.2240	0.0938
	$\tilde{\sigma} = 1.5$	1.3172	0.3710	0.1463	1.4226	0.4190	0.1588
$J = 250$	$\tilde{\sigma} = 0.001$	0.1463	0.0460	0.0213	0.2311	0.0868	0.0335
	$\tilde{\sigma} = 0.5$	0.1590	0.0525	0.0248	0.2496	0.0854	0.0362
	$\tilde{\sigma} = 1$	0.2089	0.0665	0.0306	0.3261	0.0876	0.0400
	$\tilde{\sigma} = 1.5$	0.2684	0.0909	0.0380	0.3587	0.1211	0.0546

Table 2: 1000 RMISE for simulated fast time mean with 256 fast time samples.

		$\mu^{(1)}(t)$			$\mu^{(2)}(t)$		
		N = 30	N = 100	N=250	N = 30	N = 100	N=250
$J = 30$	$\tilde{\sigma} = 0.001$	0.0877	0.0409	0.0251	0.1351	0.0545	0.0326
	$\tilde{\sigma} = 0.5$	0.2039	0.0669	0.0367	0.2522	0.0796	0.0434
	$\tilde{\sigma} = 1$	0.5945	0.1743	0.0684	0.6540	0.1904	0.0765
	$\tilde{\sigma} = 1.5$	1.2203	0.3750	0.1438	1.2575	0.3967	0.1529
$J = 250$	$\tilde{\sigma} = 0.001$	0.0887	0.0372	0.0256	0.1436	0.0586	0.0307
	$\tilde{\sigma} = 0.5$	0.0936	0.0405	0.0267	0.1482	0.0608	0.0318
	$\tilde{\sigma} = 1$	0.1361	0.0527	0.0309	0.1718	0.0674	0.0377
	$\tilde{\sigma} = 1.5$	0.2108	0.0685	0.0377	0.2639	0.0849	0.0448

Table 3: RMSE for fast time eigenvalues simulated with 128 fast time samples.

		$J = 30$			$J = 250$		
		N = 30	N = 100	N=250	N = 30	N = 100	N=250
λ_1	$\tilde{\sigma} = 0.001$	0.0597	0.0182	0.0086	0.0695	0.0170	0.0077
	$\tilde{\sigma} = 0.5$	0.1012	0.0199	0.0106	0.0745	0.0193	0.0085
	$\tilde{\sigma} = 1$	0.2324	0.0665	0.0354	0.0693	0.0221	0.0100
	$\tilde{\sigma} = 1.5$	0.6726	0.2629	0.1654	0.0931	0.0233	0.0097
λ_2	$\tilde{\sigma} = 0.001$	0.0469	0.0170	0.0071	0.0507	0.0206	0.0069
	$\tilde{\sigma} = 0.5$	0.0509	0.0161	0.0097	0.0454	0.0159	0.0083
	$\tilde{\sigma} = 1$	0.0876	0.0557	0.0440	0.0642	0.0186	0.0081
	$\tilde{\sigma} = 1.5$	0.1709	0.2016	0.2389	0.0454	0.0218	0.0086
λ_3	$\tilde{\sigma} = 0.001$	0.0516	0.0183	0.0081	0.0469	0.0249	0.0068
	$\tilde{\sigma} = 0.5$	0.0633	0.0184	0.0093	0.0595	0.0236	0.0068
	$\tilde{\sigma} = 1$	0.0722	0.0426	0.0296	0.0513	0.0196	0.0087
	$\tilde{\sigma} = 1.5$	0.2075	0.1364	0.1377	0.0544	0.0204	0.0104
λ_4	$\tilde{\sigma} = 0.001$	0.0861	0.0179	0.0081	0.0821	0.0220	0.0084
	$\tilde{\sigma} = 0.5$	0.0723	0.0193	0.0089	0.0815	0.0200	0.0077
	$\tilde{\sigma} = 1$	0.0830	0.0442	0.0484	0.0727	0.0201	0.0078
	$\tilde{\sigma} = 1.5$	0.3231	0.2302	0.2409	0.0672	0.0196	0.0104

Table 4: RMSE for fast time eigenvalues simulated with 256 fast time samples.

		$J = 30$			$J = 250$		
		N = 30	N = 100	N=250	N = 30	N = 100	N=250
λ_1	$\tilde{\sigma} = 0.001$	0.0796	0.0190	0.0078	0.0717	0.0199	0.0091
	$\tilde{\sigma} = 0.5$	0.1093	0.0281	0.0119	0.0740	0.0225	0.0077
	$\tilde{\sigma} = 1$	0.2714	0.0776	0.0429	0.0621	0.0210	0.0103
	$\tilde{\sigma} = 1.5$	0.8181	0.3712	0.2045	0.0936	0.0262	0.0119
λ_2	$\tilde{\sigma} = 0.001$	0.0500	0.0157	0.0069	0.0481	0.0159	0.0101
	$\tilde{\sigma} = 0.5$	0.0465	0.0222	0.0122	0.0478	0.0205	0.0070
	$\tilde{\sigma} = 1$	0.1041	0.0643	0.0598	0.0416	0.0161	0.0082
	$\tilde{\sigma} = 1.5$	0.1656	0.2271	0.2957	0.0520	0.0210	0.0098
λ_3	$\tilde{\sigma} = 0.001$	0.0572	0.0178	0.0077	0.0471	0.0239	0.0077
	$\tilde{\sigma} = 0.5$	0.0583	0.0190	0.0089	0.0560	0.0190	0.0107
	$\tilde{\sigma} = 1$	0.0762	0.0438	0.0409	0.0530	0.0194	0.0077
	$\tilde{\sigma} = 1.5$	0.2322	0.1786	0.1570	0.0512	0.0205	0.0093
λ_4	$\tilde{\sigma} = 0.001$	0.0784	0.0218	0.0078	0.0761	0.0194	0.0086
	$\tilde{\sigma} = 0.5$	0.0646	0.0215	0.0098	0.0836	0.0192	0.0080
	$\tilde{\sigma} = 1$	0.1177	0.0629	0.0670	0.0723	0.0189	0.0095
	$\tilde{\sigma} = 1.5$	0.4233	0.2688	0.3048	0.0679	0.0236	0.0110

Table 5: RMSE for slow time eigenvalues simulated with 128 fast time samples.

		$J = 30$			$J = 250$		
		N = 30	N = 100	N=250	N = 30	N = 100	N=250
λ_{11}	$\tilde{\sigma} = 0.001$	1.8469	0.9701	0.7379	13.5768	13.5665	13.4992
	$\tilde{\sigma} = 0.5$	1.6686	1.0896	0.8517	7.1455	10.3077	11.6714
	$\tilde{\sigma} = 1$	9.4758	1.2103	0.9237	7.1752	2.5418	5.3711
	$\tilde{\sigma} = 1.5$	101.5555	13.2640	1.6906	93.1677	14.1322	2.9325
λ_{12}	$\tilde{\sigma} = 0.001$	3.2060	1.7327	1.6657	13.9818	13.8625	13.8385
	$\tilde{\sigma} = 0.5$	2.5918	1.9239	1.7003	7.2557	10.3525	11.8293
	$\tilde{\sigma} = 1$	4.5779	0.9343	1.3764	6.4043	2.6763	5.0598
	$\tilde{\sigma} = 1.5$	44.1457	7.7889	1.2926	85.9401	10.5846	2.8056
λ_{13}	$\tilde{\sigma} = 0.001$	10.0330	11.1214	11.5982	15.2329	15.4110	15.4244
	$\tilde{\sigma} = 0.5$	7.8444	10.5320	11.1692	11.2733	13.6319	14.4772
	$\tilde{\sigma} = 1$	2.6649	7.1636	9.2670	4.1145	8.6352	10.8284
	$\tilde{\sigma} = 1.5$	9.3609	2.9624	5.8956	12.6877	3.1402	6.9233
λ_{14}	$\tilde{\sigma} = 0.001$	9.4094	8.9353	8.7059	15.0593	14.9770	14.9317
	$\tilde{\sigma} = 0.5$	7.7153	8.3966	8.4274	12.1963	14.0439	14.5229
	$\tilde{\sigma} = 1$	4.0956	6.6006	7.4362	5.3223	9.4466	11.7963
	$\tilde{\sigma} = 1.5$	4.1188	3.6551	5.4942	3.6915	4.5963	7.9345

Table 6: RMSE for slow time eigenvalues simulated with 256 fast time samples.

		$J = 30$			$J = 250$		
		N = 30	N = 100	N=250	N = 30	N = 100	N=250
λ_{11}	$\tilde{\sigma} = 0.001$	21.5886	19.9446	20.2506	1.9667	1.8395	1.8251
	$\tilde{\sigma} = 0.5$	32.0266	21.7969	20.5811	2.1799	0.5359	0.9050
	$\tilde{\sigma} = 1$	166.4864	44.5541	28.2084	112.6337	18.6524	5.8308
	$\tilde{\sigma} = 1.5$	963.2846	195.6753	64.1047	716.2971	161.2565	55.0395
λ_{12}	$\tilde{\sigma} = 0.001$	12.2839	13.4573	12.9574	2.2926	2.0729	2.0667
	$\tilde{\sigma} = 0.5$	20.9164	15.4030	13.5757	2.2290	0.5145	0.8951
	$\tilde{\sigma} = 1$	87.5505	34.9205	19.2723	105.9230	23.6959	6.3190
	$\tilde{\sigma} = 1.5$	324.3976	122.8017	53.0486	540.3350	187.0776	54.6870
λ_{13}	$\tilde{\sigma} = 0.001$	2.1327	0.8575	0.7884	3.2959	3.4773	3.5505
	$\tilde{\sigma} = 0.5$	2.3996	0.5990	0.5763	1.1035	1.8939	2.5808
	$\tilde{\sigma} = 1$	16.6741	1.1821	0.2602	19.0014	1.3945	0.6535
	$\tilde{\sigma} = 1.5$	116.9901	12.9054	1.9962	119.6293	17.7660	3.6761
λ_{14}	$\tilde{\sigma} = 0.001$	1.2111	0.3092	0.1415	3.1895	3.2126	3.1993
	$\tilde{\sigma} = 0.5$	1.3044	0.5372	0.2472	1.0192	2.1258	2.6460
	$\tilde{\sigma} = 1$	8.7832	1.8256	0.7698	7.3652	0.6683	0.8487
	$\tilde{\sigma} = 1.5$	26.2824	7.8890	2.4394	70.0516	8.8245	1.0884

Table 7: RMISE for fast time eigenfunctions simulated with 128 fast time samples.

		$J = 30$			$J = 250$		
		N = 30	N = 100	N=250	N = 30	N = 100	N=250
$\phi_1(t)$	$\tilde{\sigma} = 0.001$	0.5678	0.3718	0.3306	0.5171	0.3814	0.3225
	$\tilde{\sigma} = 0.5$	0.5851	0.4225	0.3365	0.5158	0.3906	0.3234
	$\tilde{\sigma} = 1$	0.6203	0.5049	0.3692	0.5255	0.3964	0.3274
	$\tilde{\sigma} = 1.5$	0.9212	0.5276	0.4298	0.5625	0.4058	0.3402
$\phi_2(t)$	$\tilde{\sigma} = 0.001$	1.5721	1.7621	1.8375	1.5728	1.7596	1.8398
	$\tilde{\sigma} = 0.5$	1.6279	1.7212	1.8012	1.5924	1.7250	1.8239
	$\tilde{\sigma} = 1$	1.6389	1.7478	1.7968	1.5652	1.7281	1.8109
	$\tilde{\sigma} = 1.5$	1.6268	1.7121	1.7605	1.5874	1.7200	1.8135
$\phi_3(t)$	$\tilde{\sigma} = 0.001$	1.0258	0.9265	0.8915	1.0241	0.9220	0.8956
	$\tilde{\sigma} = 0.5$	1.0611	0.9194	0.8943	1.0812	0.9122	0.8885
	$\tilde{\sigma} = 1$	1.1184	0.9522	0.9255	1.0353	0.9374	0.8945
	$\tilde{\sigma} = 1.5$	1.3510	1.0428	0.9701	1.0586	0.9321	0.9019
$\phi_4(t)$	$\tilde{\sigma} = 0.001$	0.3004	0.1539	0.1167	0.2756	0.1467	0.1182
	$\tilde{\sigma} = 0.5$	0.2948	0.1527	0.1179	0.3185	0.1448	0.1170
	$\tilde{\sigma} = 1$	0.4307	0.1752	0.1303	0.3052	0.1520	0.1176
	$\tilde{\sigma} = 1.5$	1.0265	0.2512	0.1365	0.3104	0.1577	0.1172

Table 8: RMISE for fast time eigenfunctions simulated with 256 fast time samples.

		$J = 30$			$J = 250$		
		N = 30	N = 100	N=250	N = 30	N = 100	N=250
$\boldsymbol{\phi}_1(t)$	$\tilde{\sigma} = 0.001$	0.5384	0.4233	0.3221	0.5085	0.3683	0.3268
	$\tilde{\sigma} = 0.5$	0.5742	0.4016	0.3314	0.5265	0.4032	0.3137
	$\tilde{\sigma} = 1$	0.7212	0.4569	0.3356	0.5468	0.3876	0.3233
	$\tilde{\sigma} = 1.5$	0.9560	0.5950	0.4183	0.5673	0.3899	0.3169
$\boldsymbol{\phi}_2(t)$	$\tilde{\sigma} = 0.001$	1.5898	1.7308	1.8302	1.5263	1.7273	1.8072
	$\tilde{\sigma} = 0.5$	1.6042	1.7466	1.7982	1.6240	1.7615	1.8374
	$\tilde{\sigma} = 1$	1.5674	1.7044	1.7780	1.6018	1.7193	1.8047
	$\tilde{\sigma} = 1.5$	1.6424	1.6655	1.7120	1.6216	1.6941	1.8192
$\boldsymbol{\phi}_3(t)$	$\tilde{\sigma} = 0.001$	0.9872	0.9116	0.8865	1.0385	0.9269	0.8874
	$\tilde{\sigma} = 0.5$	1.0423	0.9097	0.8927	1.0665	0.9052	0.8990
	$\tilde{\sigma} = 1$	1.1705	0.9442	0.9218	1.0487	0.9045	0.8882
	$\tilde{\sigma} = 1.5$	1.3437	1.0742	0.9612	1.0669	0.9116	0.9046
$\boldsymbol{\phi}_4(t)$	$\tilde{\sigma} = 0.001$	0.2657	0.1374	0.1160	0.2480	0.1485	0.1178
	$\tilde{\sigma} = 0.5$	0.2868	0.1336	0.1126	0.2822	0.1306	0.1194
	$\tilde{\sigma} = 1$	0.5846	0.1531	0.1102	0.2836	0.1443	0.1112
	$\tilde{\sigma} = 1.5$	1.1807	0.3160	0.1301	0.3031	0.1400	0.1146

Table 9: RMISE for slow time eigenfunctions simulated with 128 fast time samples.

		$J = 30$			$J = 250$		
		N = 30	N = 100	N=250	N = 30	N = 100	N=250
$\psi_{11}(s)$	$\tilde{\sigma} = 0.001$	0.0060	0.0060	0.0060	0.0146	0.0047	0.0079
	$\tilde{\sigma} = 0.5$	0.1945	0.0507	0.0232	0.9610	0.8600	0.5791
	$\tilde{\sigma} = 1$	0.8713	0.4234	0.1682	1.2011	1.0863	1.0585
	$\tilde{\sigma} = 1.5$	1.2228	0.9873	0.6393	1.2059	1.2188	1.1164
$\psi_{12}(s)$	$\tilde{\sigma} = 0.001$	0.0060	0.0060	0.0060	0.0239	0.0107	0.0077
	$\tilde{\sigma} = 0.5$	0.2731	0.0642	0.0276	1.0354	0.8845	0.7421
	$\tilde{\sigma} = 1$	0.9084	0.4830	0.2257	1.1813	1.1610	1.0988
	$\tilde{\sigma} = 1.5$	1.2056	1.0122	0.7225	1.2072	1.1776	1.1016
$\psi_{13}(s)$	$\tilde{\sigma} = 0.001$	0.0062	0.0060	0.0061	0.0902	0.1592	0.2807
	$\tilde{\sigma} = 0.5$	0.4089	0.2069	0.0699	1.1049	1.0026	0.9126
	$\tilde{\sigma} = 1$	0.9164	0.5701	0.3687	1.1388	1.1697	1.1260
	$\tilde{\sigma} = 1.5$	1.1225	0.8600	0.6186	1.1880	1.1503	1.1928
$\psi_{14}(s)$	$\tilde{\sigma} = 0.001$	0.0061	0.0060	0.0060	0.1024	0.1747	0.3256
	$\tilde{\sigma} = 0.5$	0.2399	0.0508	0.0221	1.0187	0.8305	0.6234
	$\tilde{\sigma} = 1$	0.7715	0.2999	0.1110	1.1916	1.0849	1.0899
	$\tilde{\sigma} = 1.5$	1.0946	0.7102	0.3058	1.1532	1.1564	1.1236

Table 10: RMISE for slow time eigenfunctions simulated with 256 fast time samples.

		$J = 30$			$J = 250$		
		N = 30	N = 100	N=250	N = 30	N = 100	N=250
$\psi_{11}(s)$	$\tilde{\sigma} = 0.001$	0.0060	0.0060	0.0060	0.0003	0.0002	0.0002
	$\tilde{\sigma} = 0.5$	0.2154	0.0551	0.0234	1.0569	0.8181	0.5783
	$\tilde{\sigma} = 1$	0.8706	0.4191	0.1562	1.2118	1.0816	1.0866
	$\tilde{\sigma} = 1.5$	1.2322	0.8602	0.6202	1.1920	1.1637	1.1416
$\psi_{12}(s)$	$\tilde{\sigma} = 0.001$	0.0060	0.0060	0.0060	0.0003	0.0002	0.0002
	$\tilde{\sigma} = 0.5$	0.2636	0.0663	0.0291	1.0691	0.9073	0.6803
	$\tilde{\sigma} = 1$	0.9038	0.5198	0.2158	1.1370	1.1251	1.1159
	$\tilde{\sigma} = 1.5$	1.2003	0.9480	0.7476	1.1827	1.1877	1.1303
$\psi_{13}(s)$	$\tilde{\sigma} = 0.001$	0.0060	0.0060	0.0060	0.0079	0.0002	0.0002
	$\tilde{\sigma} = 0.5$	0.4195	0.1702	0.0689	1.0989	1.0333	0.8771
	$\tilde{\sigma} = 1$	0.8499	0.6229	0.3354	1.1913	1.1316	1.1637
	$\tilde{\sigma} = 1.5$	1.1288	0.8867	0.6025	1.1805	1.1718	1.1377
$\psi_{14}(s)$	$\tilde{\sigma} = 0.001$	0.0060	0.0060	0.0060	0.0003	0.0002	0.0002
	$\tilde{\sigma} = 0.5$	0.2670	0.0559	0.0228	1.0054	0.8078	0.6038
	$\tilde{\sigma} = 1$	0.8385	0.2922	0.1124	1.1289	1.0452	1.0081
	$\tilde{\sigma} = 1.5$	1.2323	0.6734	0.3153	1.1856	1.1686	1.1831

Simulation studies show that MMFPCA is effective in estimating the fast time mean functions and the slow time eigenfunctions for many situations. In Tables 1 and 2, the RMISE of $\boldsymbol{\mu}(t)$ is presented for all of the simulated cases. The RMISE in Table 1 is calculated for 128 observations per period, and in Table 2 is calculated for a higher within period sampling rate of 256 samples per period. The RMISE declines as the within period sampling frequency increases. As the number of subjects increases from $N = 30$ to $N = 250$, the RMISE of $\boldsymbol{\mu}(t)$ declines. When the subjects are followed for longer time frames, J increases, the precision of the mean estimates increases. As excepted, the RMISE increase with increasing $\tilde{\sigma}$.

The RMSE of fast time eigenvalues, Tables 3 and 4, has a similar dependence on the simulation parameters to the fast time mean functions. However, the RMSE does not decrease with increasing n. However, MMFPCA has difficulty in estimation of the slow time eigenvalues which have much larger RMSE compared to the fast time eigenvalues, see Tables 5 and 6. As the slow time estimation depends a smoothing bandwidth, these results may be due to bandwidth choice.

The fast time eigenfunctions have two classes of estimation error. In Tables 7 and 8, $\boldsymbol{\phi}_1(t)$ and $\boldsymbol{\phi}_4(t)$ have increasing RMISE when $\tilde{\sigma}$ increases. Also, the RMISE decreases for increasing sample size. However, there is little dependence on either n or J. For $\boldsymbol{\phi}_2(t)$ and $\boldsymbol{\phi}_3(t)$, the RMISE has no dependence on any of the simulation study parameters. Further investigation indicated that one or more of PCA estimation steps was not correctly identifying the sign of

eigenfunction. Attempts to match the sign of the estimates failed to completely correct this problem.

The slow time eigenfunctions show the same patterns in RMISE as the mean functions, see Tables 9, and 10. The RMISE of the slow time eigenfunctions is lower when the fast time density increases from $n = 128$ to $n = 256$. In addition, the RMISE increases as the simulated noise $\tilde{\sigma}$ increases. The precision of the estimates improves with increasing longitudinal follow-up J and number of subject N. In the slow time eigenfunctions, ψ_{qp}, RMISE shows the same dependence on N, n, and $\tilde{\sigma}$. However, the dependence on J shows a no clear pattern, see Tables 9 and 10.

7.0 VENTRICULAR ASSIST DEVICE WAVEFORM ANALYSIS

We analyzed our motivating VAD waveform data using our proposed method, MMFPCA. We added two more types of descriptive plots, heat map and surface plot of smoothed data, to the previously presented Figures 1, 3, 4, and 5. Starting from the point after breaking the data into day long blocks, both heat maps, Figure 9, and surface plots, Figure 10, were used to visualize the circadian cycle and its longitudinal evolution. Before plotting, the data for a single subject was smoothed using a tensor product smoothing splines with ad hoc smoothing parameters (Ramsay and Silverman, 2005). In Figure 9, the heat map shows high PO in orange and yellow with low PO in blue. PO is lower in sleeping hours (2200h - 0700h) than during the waking hours (0800h - 2000h). The heat map presentation highlights the PO level and shows the magnitude changes over time. In Figure 9, the example patient can be seen to be waking up at a later time during the 30 plus days of follow-up because the PO rises at 0700h at day one to 0830h by day thirty. In Figure 10, the same data is presented using a surface plot, which has the same color scheme as the heat map. The surface plot focuses attention on the shape of the circadian cycle instead of the level. Figure 10 shows the same pattern as in Figure 9 of low PO during the hours of 2200h - 0700h with a rise during the day. Both heatmaps and surface plots are used because they visually highlight different aspects of the data.

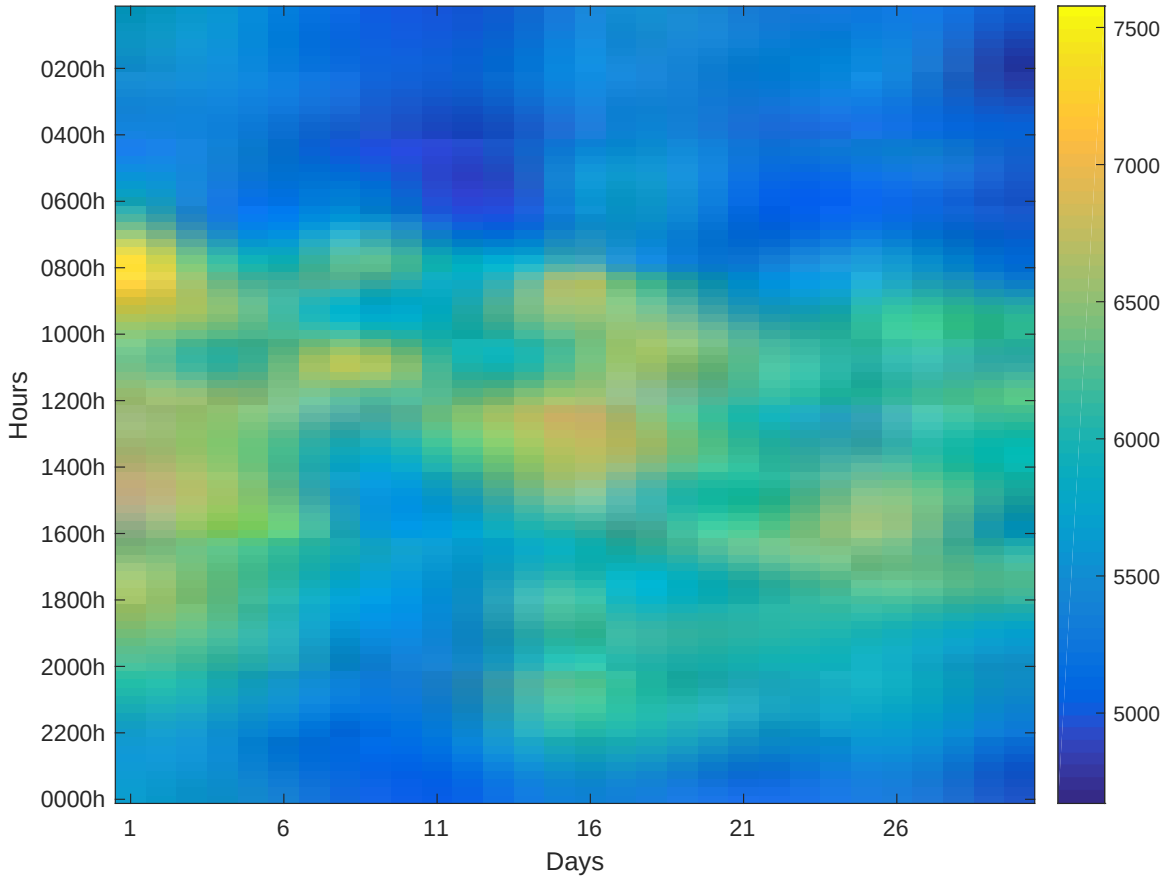

Figure 9: Heat map of pump output surface from one subject.

Thirty days of data plotted showing change of circadian behavior. Surface estimated with a smoothing spline and ad hoc smoothing parameter.

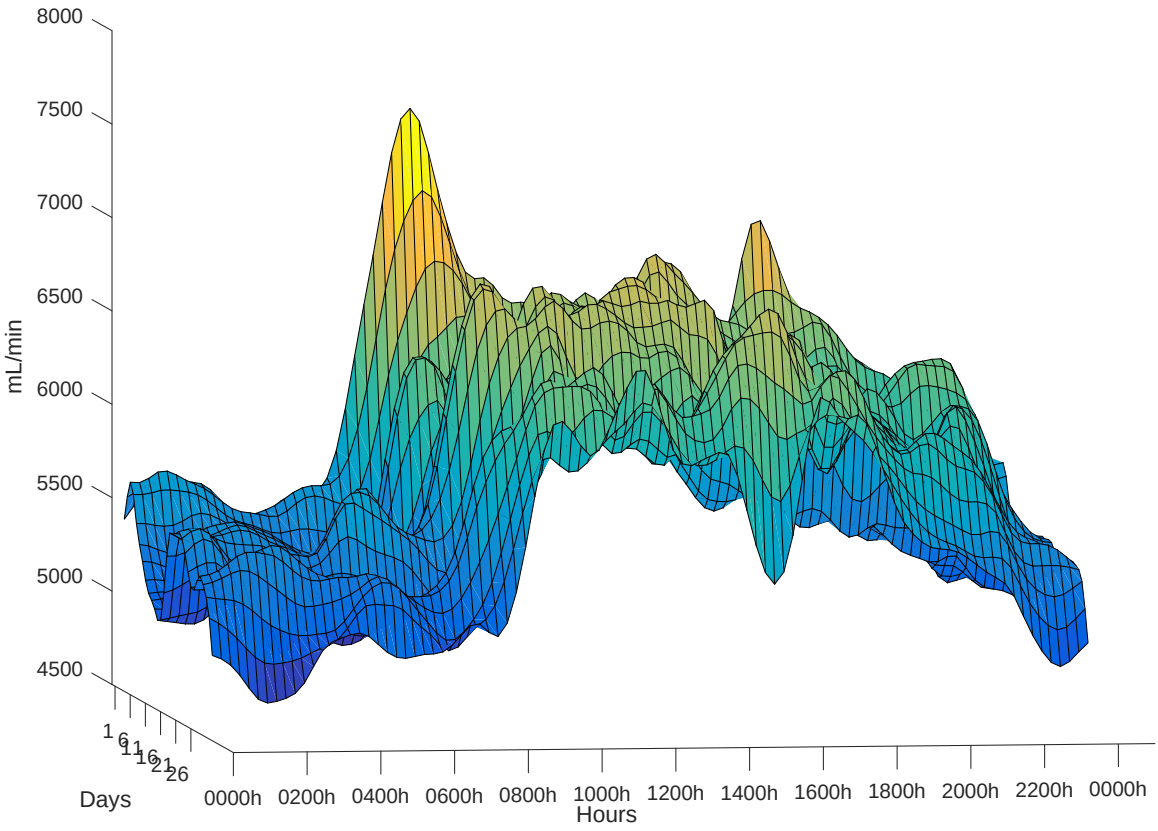

Figure 10: Surface plot of pump output surface from one subject.

Thirty days of data plotted showing change of circadian behavior. Surface estimated with a smoothing spline and ad hoc smoothing parameter.

7.1 APPLICATION OF MMFPCA

Conducting MMFPCA on the pilot cohort, we demonstrated that our method successfully modeled the VAD circadian cycles, fast time scale, and their evolution, slow time scale, at both the population and patient levels. In Figure 11, population average circadian cycles are plotted with a 95% global confidence interval. Neither the PO or PI circadian cycle was statistically significant because of the small sample size. The circadian cycle shows a similar shape to the cycle observed in Figure 10, but differences were observed between PO and PI circadian cycles.

The fast time scale FPCA described important variability modes in the circadian cycles, plotted in Figure 12. Using FVE, we found that one eigenfunction explained 59% of the variability between the patient's average circadian cycle, two eigenfunctions explained 92% of the variability and three eigenfunctions explained 97% of variability. In Figure 12, the top row displays the first three PO eigenfunctions, and the bottom row displays the same three eigenfunctions for PI. The first eigenfunction shows variability in overall shape of the circadian cycle varying from a typical circadian cycle with the highest PO during midday to an atypical circadian variation with the lowest PO during the midday hours. The corresponding variation mode for PI indicates that a strong PI circadian cycle is correlated with the typical PO cycle. A weaker PI circadian cycle is associated with an atypical PO cycle. The second eigenfunction explained variation in average level of the PO which is associated with variation in the shape of the PI circadian cycle. Finally, the third eigenfunction explains variability in the patients' sleep-wake timing. For "morning larks," both PO and PI rise earlier than the cohort average cycle and decline earlier in the day as well. "Night owls" show the opposite pattern with both PO and PI lower in the morning hours and higher in the afternoon to early evening. Focusing on the third eigenfunction, PI appears to increase before PO implying that blood flow variability increases before mean flow level increases. As the physiological mechanism for this is not understood, MMFPCA provides also a method for generating new hypotheses about circadian cycles in VAD patients.

A regular dense subsection of the longitudinal follow-up, 20 days, was analyzed. In Figure 13, the important modes of variation over slow time are seen for the first three fast

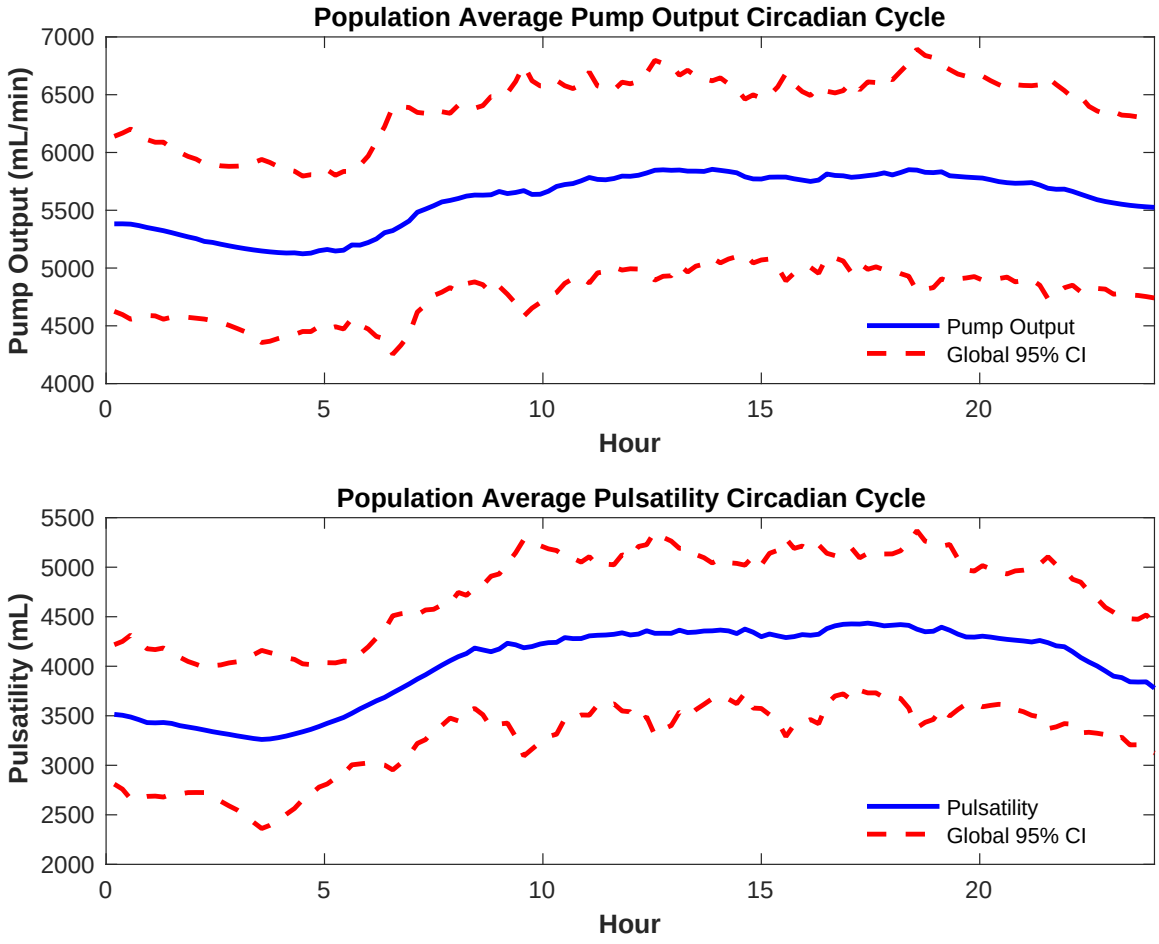

Figure 11: Estimated VAD cohort average pump output and pulsatility.
Top plot presents the population average mean pump output, solid line, with a 95% global confidence band, dashed line. Bottom plot presents the population average mean daily pulsatility, solid line, with a 95% global confidence band, dashed line.

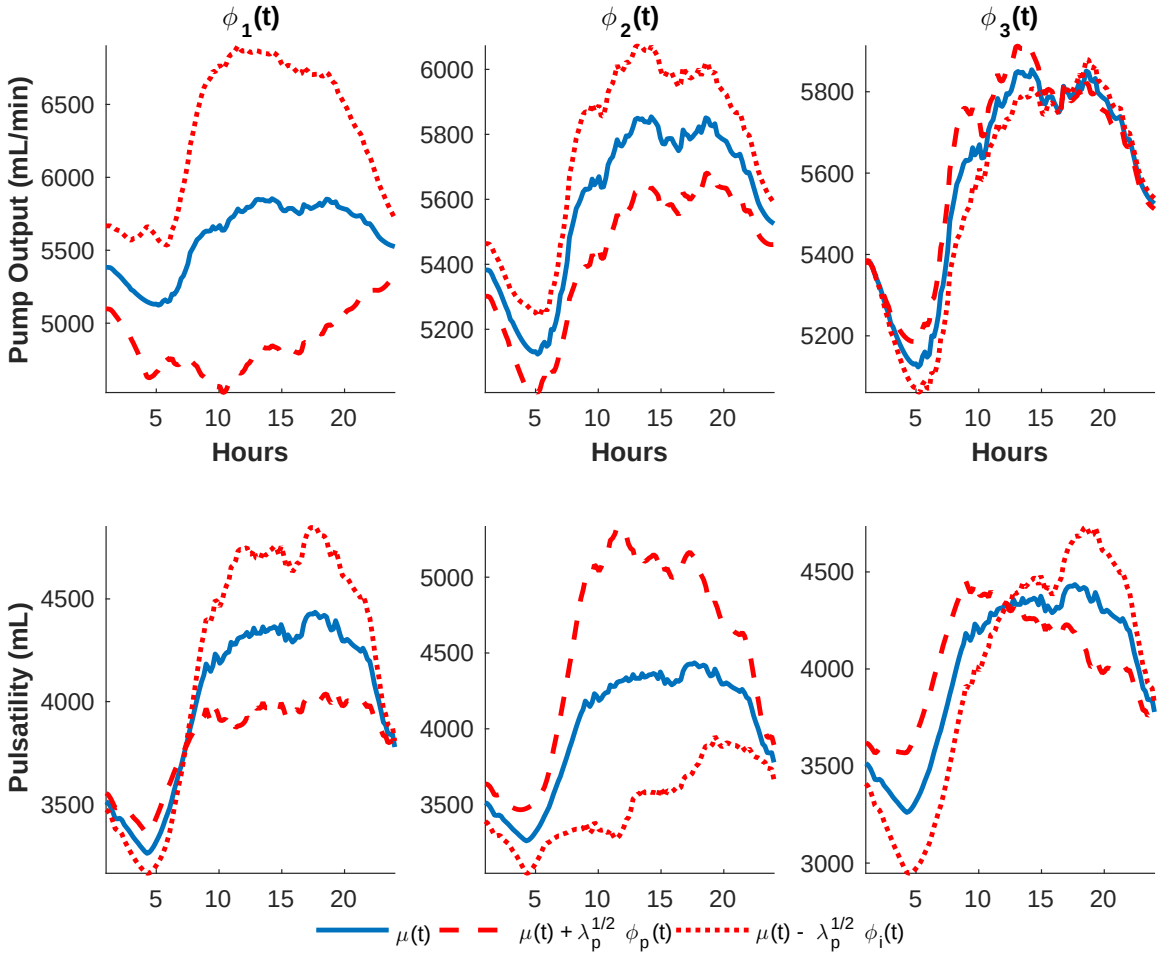

Figure 12: Fast time eigenfunctions for the VAD cohort.

Population average fast time functions are plotted as solid lines for pump output in top row and pulsatility in bottom row. The heavy dashed line plots plus an eigenfunction and the light dashed line plots minus an eigenfunction.

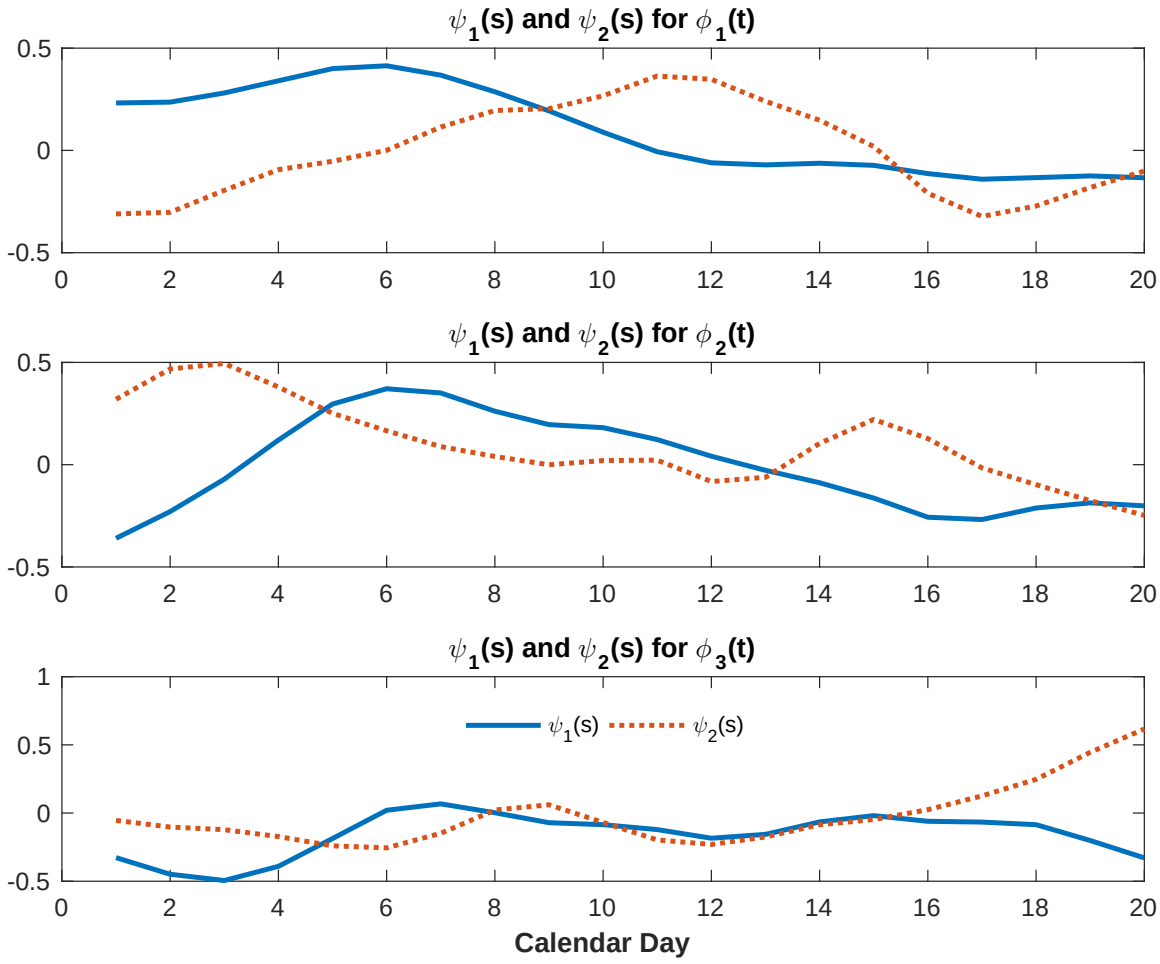

Figure 13: Slow time eigenfunctions for the VAD cohort.

Each figure contains the first two slow time eigenfunctions for the first three fast time eigenfunctions.

time eigenfunctions. The upper two plots show how the mean level and overall shape of the circadian cycles evolve over days. The third eigenfunction, describing wake-up time variability, contains a weak weekly variation. The weekly signal seen in these longitudinal eigenfunctions show that even though these patients are very ill, the social jet lag component seen in healthy circadian cycles shows up in these patients as well but too a smaller extent.

Patient multiple predictions are calculated for the nine analyzed patients on both the fast time scale in slow time scale. On the fast time scale, patient level predicted circadian cycles are shown in Figure 14 and Figure 15. The MMFPCA model predicts the PO and PI circadian cycles in patients who have a minimum in both components at around 6am. The model struggles to predict circadian cycles in patients with earlier or later nadirs in the circadian cycles. Twenty days of the multivariate signal is reconstructed for the nine patients on the slow time scale. In Figure 16, the predicted PO for all patients shows the varied longitudinal evolution of the VAD patient population. For PI, the same signals are presented in Figure 17. The upper left surface plot in both Figures is the same patient as seen in Figure 1, Figure 10, and Figure 9. The decline and subsequent in both PO and PI circadian cycles from days 10-14 is clearly seen in the predicted surfaces while only hinted at in Figure 10. The patient in the bottom right corner is unique as little longitudinal variation in the circadian cycles is seen.

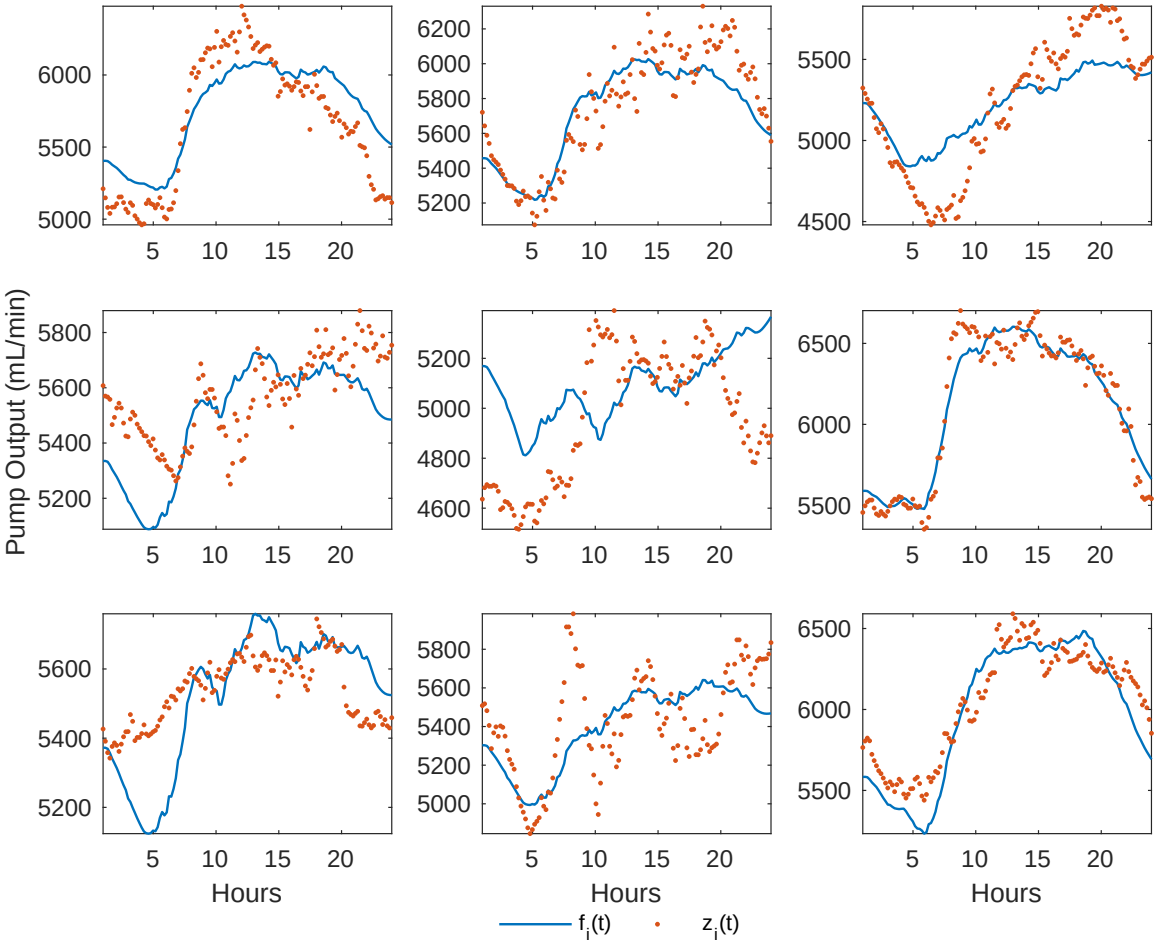

Figure 14: Subject specific predicted pump output circadian cycle.

Predicted pump output for each subject with a circadian cycle. The solid blue lines are estimated subject specific circadian cycles and the red dots are the observed data. Patient 1, upper left, is well predicted; however, several other patients are not as the dots and solid lines do not coincide.

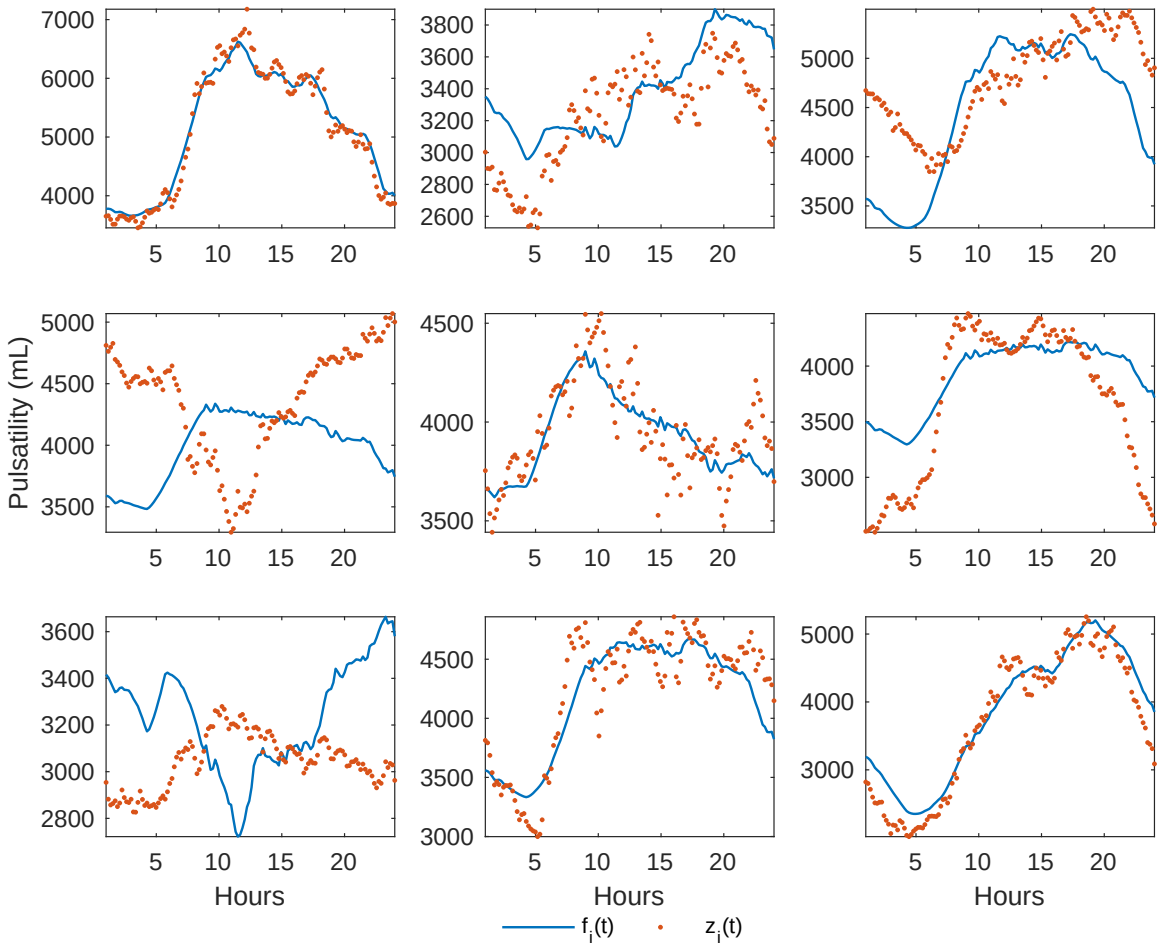

Figure 15: Subject specific predicted pulsatility circadian cycle.

Predicted pulsatility for each subject with a circadian cycle. The solid blue lines are subject specific circadian cycles and the red dots are the observed data. Patient 1, upper left, is well predicted; however, several other patients are not as the dots and solid lines do not coincide.

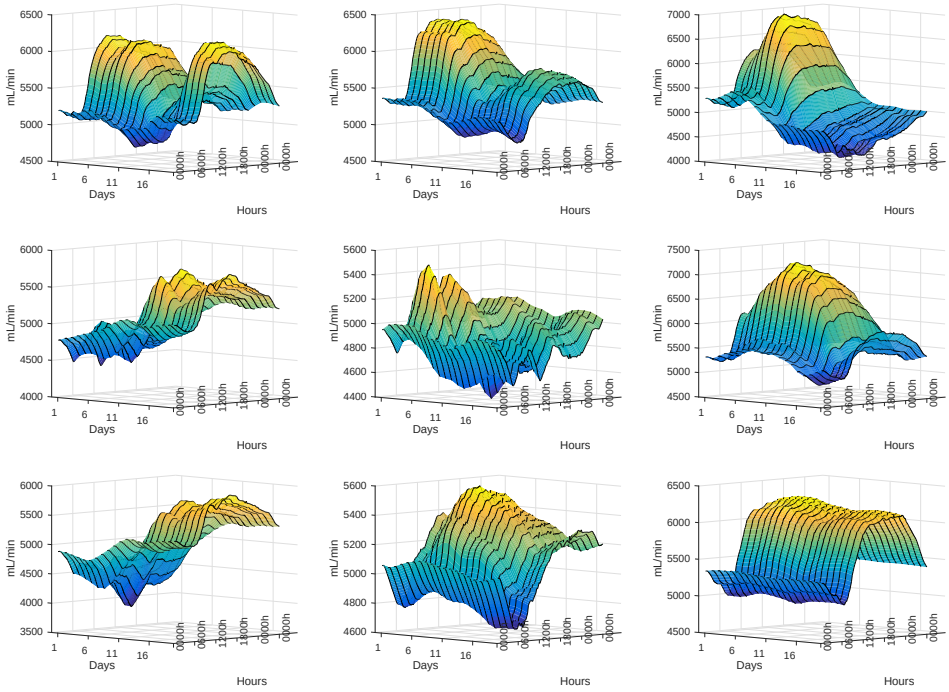

Figure 16: The predicted patient specific pump output surfaces for VAD cohort.

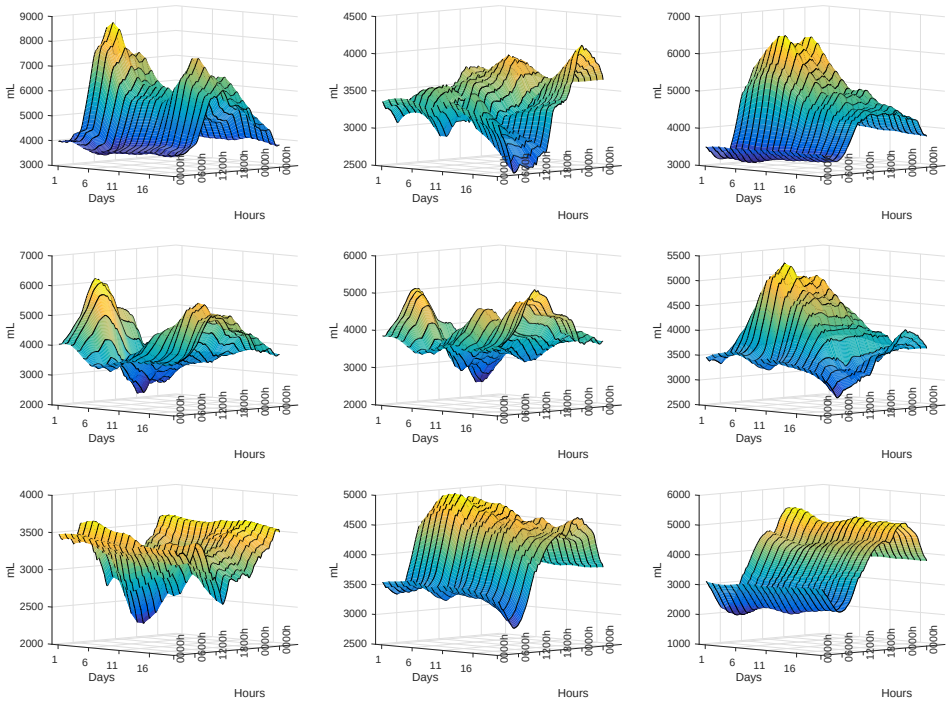

Figure 17: The predicted patient specific pulsatility surfaces for VAD cohort.

8.0 DISCUSSION

We introduce a new approach to the analysis of non-stationary functions motivated by intensively multivariate measured physiological data collected on patients with a VAD. Because the non-stationarity of these data presented a challenge for analysis of the entire time series, we decomposed the original time scale into two new independent time variables (one containing any periodic component, the second containing all changes in the periodic component). This decomposition was motivated by the two-timing solution from non-linear ODEs (Strogatz, 1994). Via the decomposition of the time scales, the estimation for each time scale can be separated into two simpler estimation problems - the complex optimization reduces to a single FPCA analysis of each time scale. The proposed estimation method extends both the RF-FPCA method introduced by Chen and Müller (2012) and the marginal FPCA method introduced in Chen et al. (2016); Park and Staicu (2015) for univariate functions to the multivariate case. The proposed method also differed from RF-FPCA as the covariance $\mathcal{K}(t, t', s, s')$ is decomposed using a marginal formulation instead of the conditional formulation in Chen and Müller (2012); Chen et al. (2016). Also, the proposed method differs from LFPCA is several critical ways: MMFPCA is built to handle multivariate data with dense observations in both directions, a different mean structure is proposed (fast time mean only), and the covariance model has an additional assumption on its structure (an explicit fast time marginal model). Finally, MMFPCA can be shown to converge to the multivariate extension of LFPCA in the sparse longitudinal sampling case.

We found that MMFPCA allowed for robust fast time scale analysis regardless of the specifics of slow time scale model. The proposed method uses a well accepted technique, FPCA, to analyze the between subject variability. While our application involved using bivariate data, the method can extend to any arbitrary dimension. All inference is conducted

using a functional bootstrap with re-sampling conducted on each subject. While this approach to statistical inference is time consuming, it requires minimal assumptions and easily handles the complex dependence on sample sizes seen in these data.

Our development of MMFPCA has several limitations. First, the proposed method currently requires equal longitudinal follow-up in the slow time scale. Also, all functional correlation between outcome components is required to be on the fast time scale. Future work includes extending MMFPCA to incorporate covariate adjustment that can model datasets with patients containing both a circadian cycle and no circadian cycle. In addition, MMFPCA requires that both the fast time functions have all important features aligned in time, in phase. If a patient's circadian cycle is out of phase with the population average circadian cycle, MMFPCA poorly predicts the subjects profile. As well, the population mean function is attenuated in its amplitude variation. A potential future solution to this problem is to integrate function registration into the model. Also, future directions are development of techniques to cluster subjects based on fast time FPCA and slow time FPCA.

APPENDIX A

PLOTS OF VAD PATIENT RAW DATA

In this section of the supplementary materials, we present additional patient data from the cohort of nine patients. All patients are referred to by a study number, 1-9, with 20 days of data for patient 1 presented in main article. The eight patients' data shown in this supplement shows a wider range of behavior in PO and PI compared to patient 1 (see main article). While patient 1 had a strong circadian cycle in both PO and PI that changed slightly from day-to-day, patient 4, Figure 20, does not show any circadian rhythms in the first ten days of follow-up. Around day 12, a circadian cycle starts in the PI waveform but not in the PO waveform. By day eighteen, both PO and PI waveforms show a circadian cycle. Patient 7, Figure 23, has two qualitatively different types of behavior. From day zero until day eight, the patient shows a weak circadian cycle in both PI and PO that is declining in amplitude and ending at day eight. At day eight, both PO and PI suddenly jump levels and all circadian cycles stop. Furthermore, there is a sharp drop in PO and PI in the early morning of day nine. After day nine, patient 8 show some variability in PO and PI, but no circadian cycle is seen by visual inspection. Patient 3, Figure 19, has a circadian cycle throughout the twenty days of follow-up but a drop in both amplitude and mean level is observed during the tenth day. A similar event is seen in patient 5, Figure 21, at during days three and four. Patient 6 shows the loss and restoration of the circadian cycle in the period from day fifteen to eighteen. The clinical significance of these events is currently being investigated. Finally, patients 2 (Figure 18), 8 (Figure 24), and 9 (Figure 25) have a stable circadian for the entire twenty day follow-up.

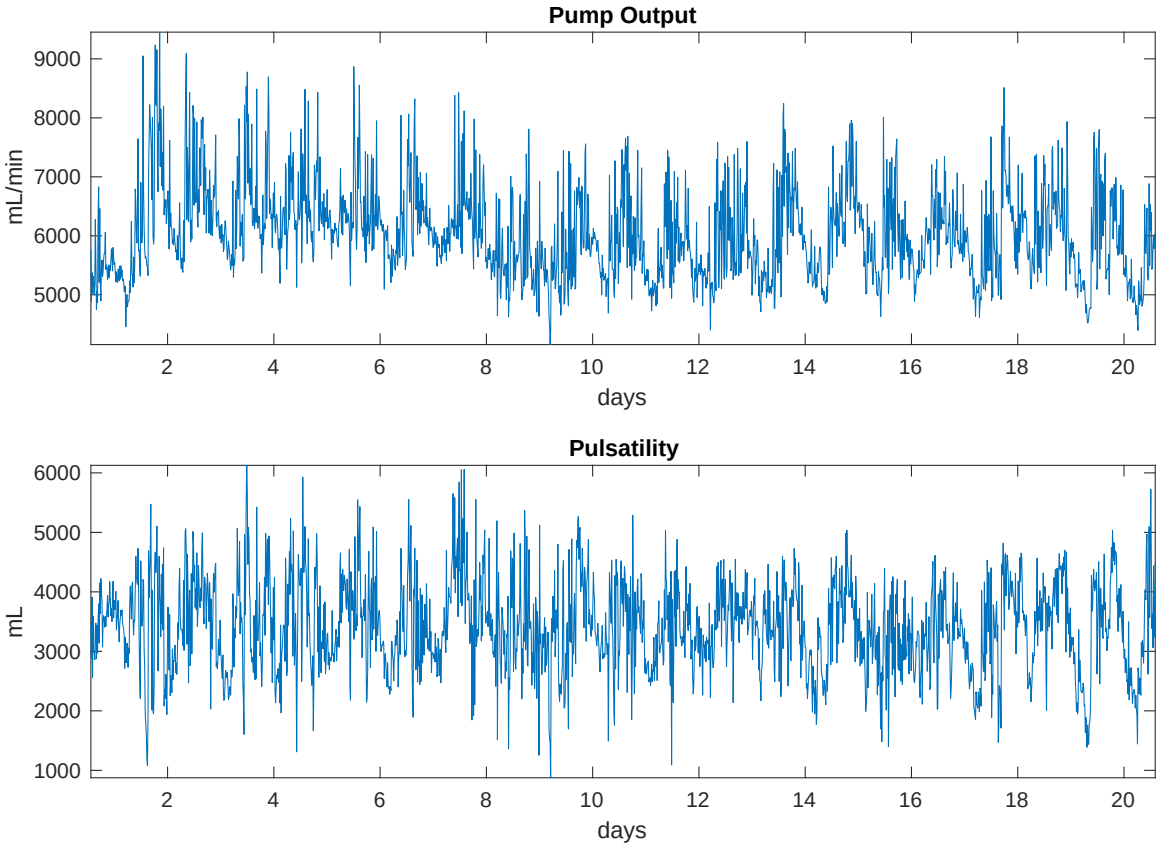

Figure 18: Sample profile for 20 days of pump output and pulsatility for patient 2.

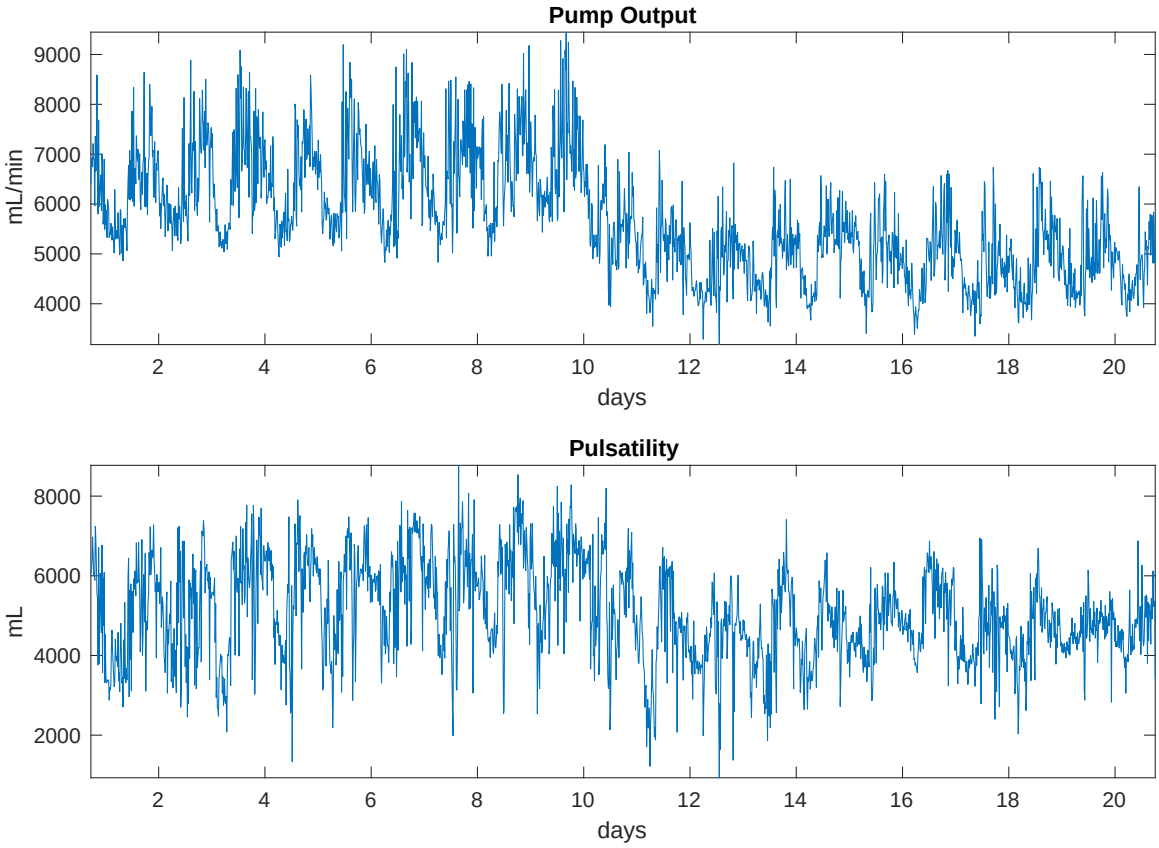

Figure 19: Sample profile for 20 days of pump output and pulsatility for patient 3.

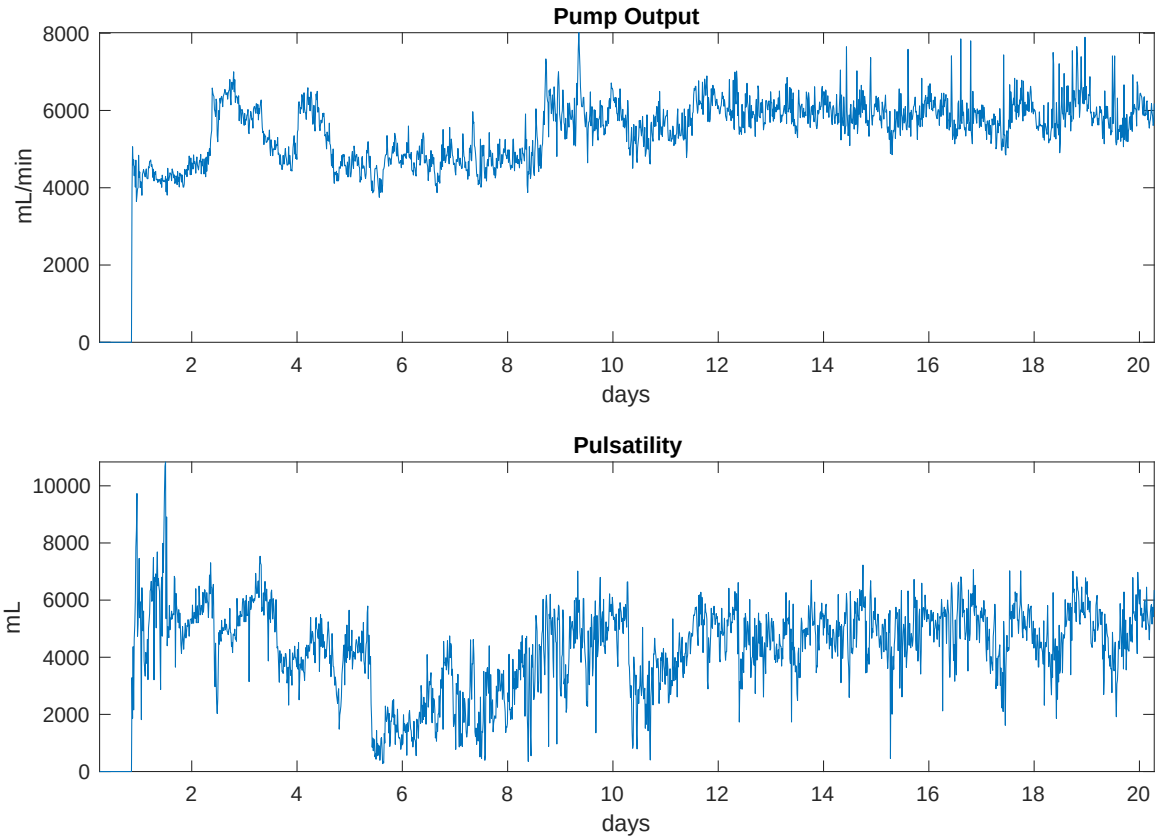

Figure 20: Sample profile for 20 days of pump output and pulsatility for patient 4.

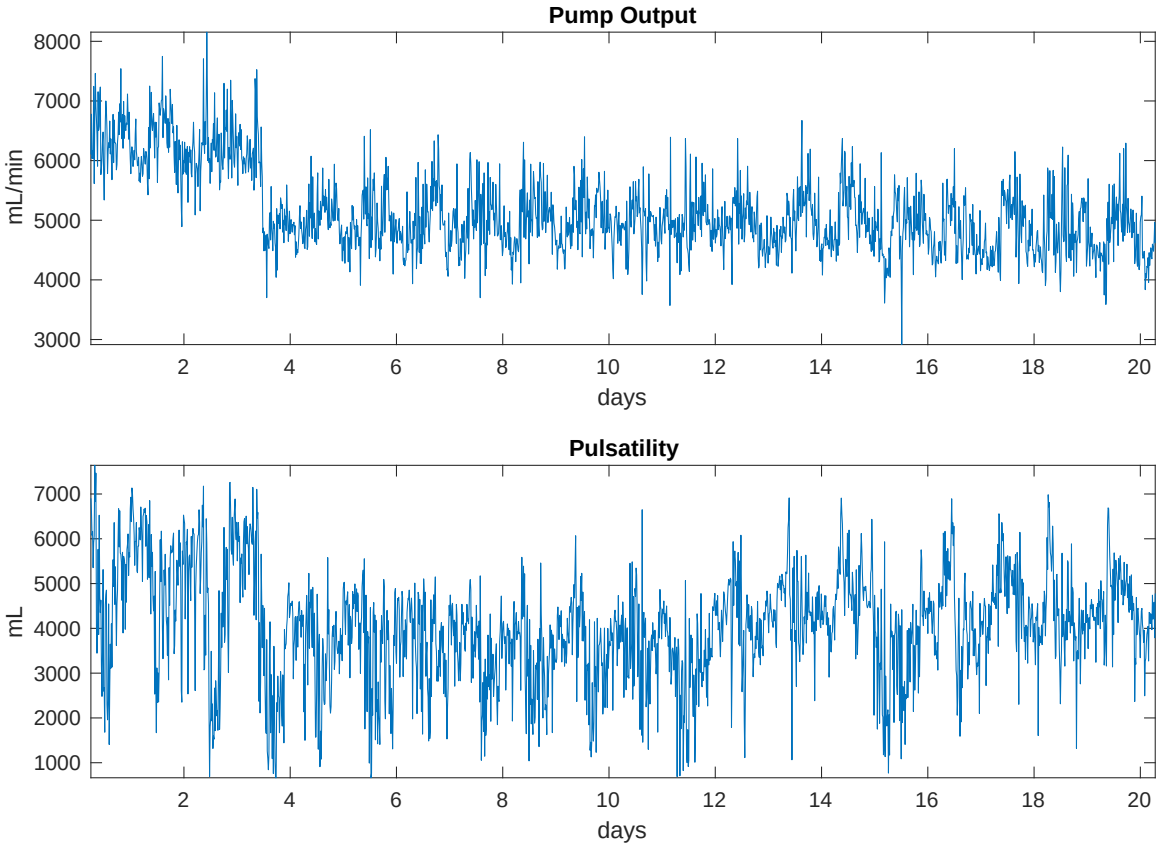

Figure 21: Sample profile for 20 days of pump output and pulsatility for patient 5.

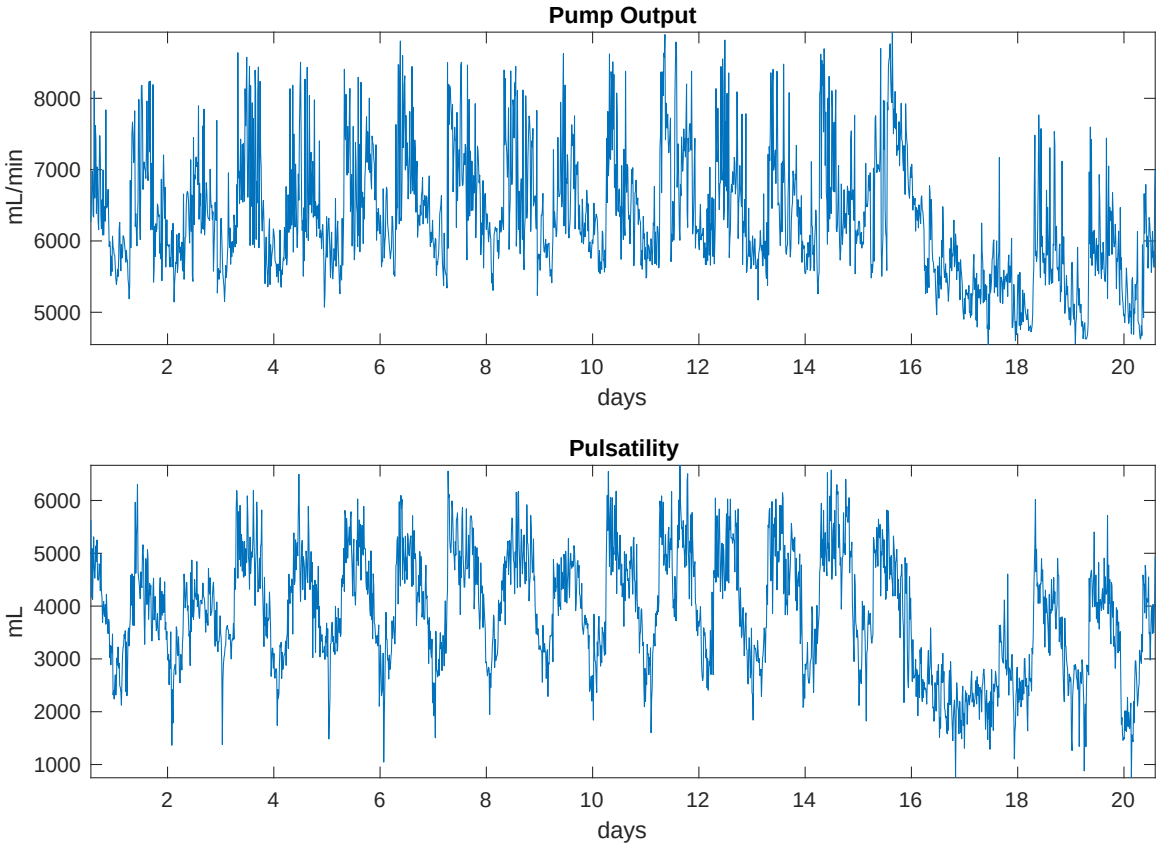

Figure 22: Sample profile for 20 days of pump output and pulsatility for patient 6.

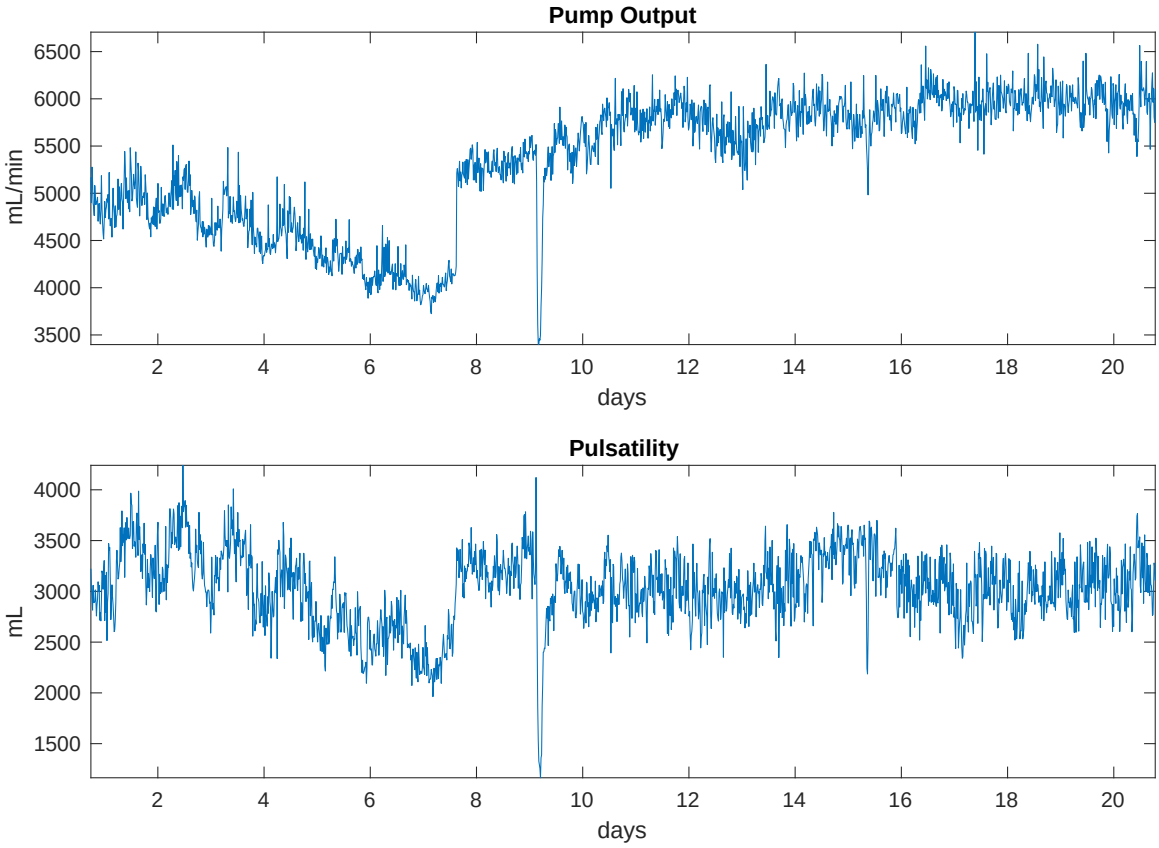

Figure 23: Sample profile for 20 days of pump output and pulsatility for patient 7.

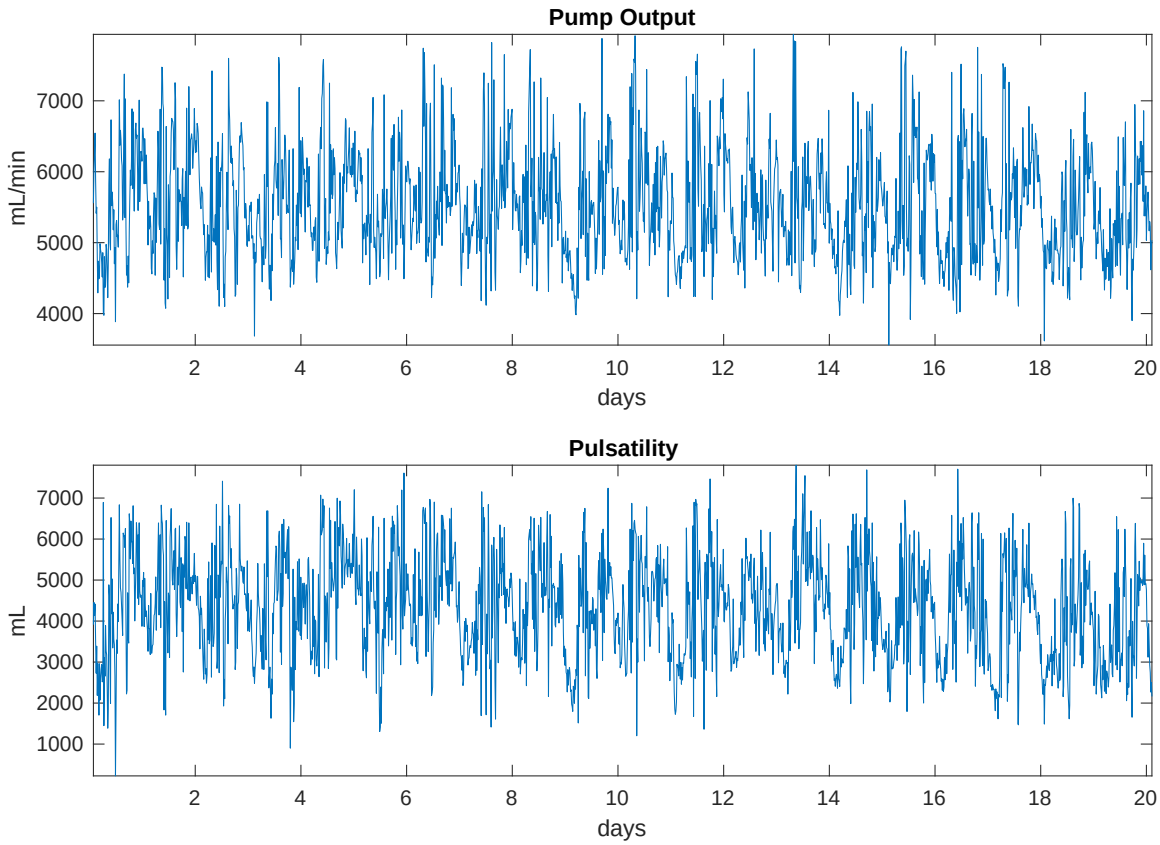

Figure 24: Sample profile for 20 days of pump output and pulsatility for patient 8.

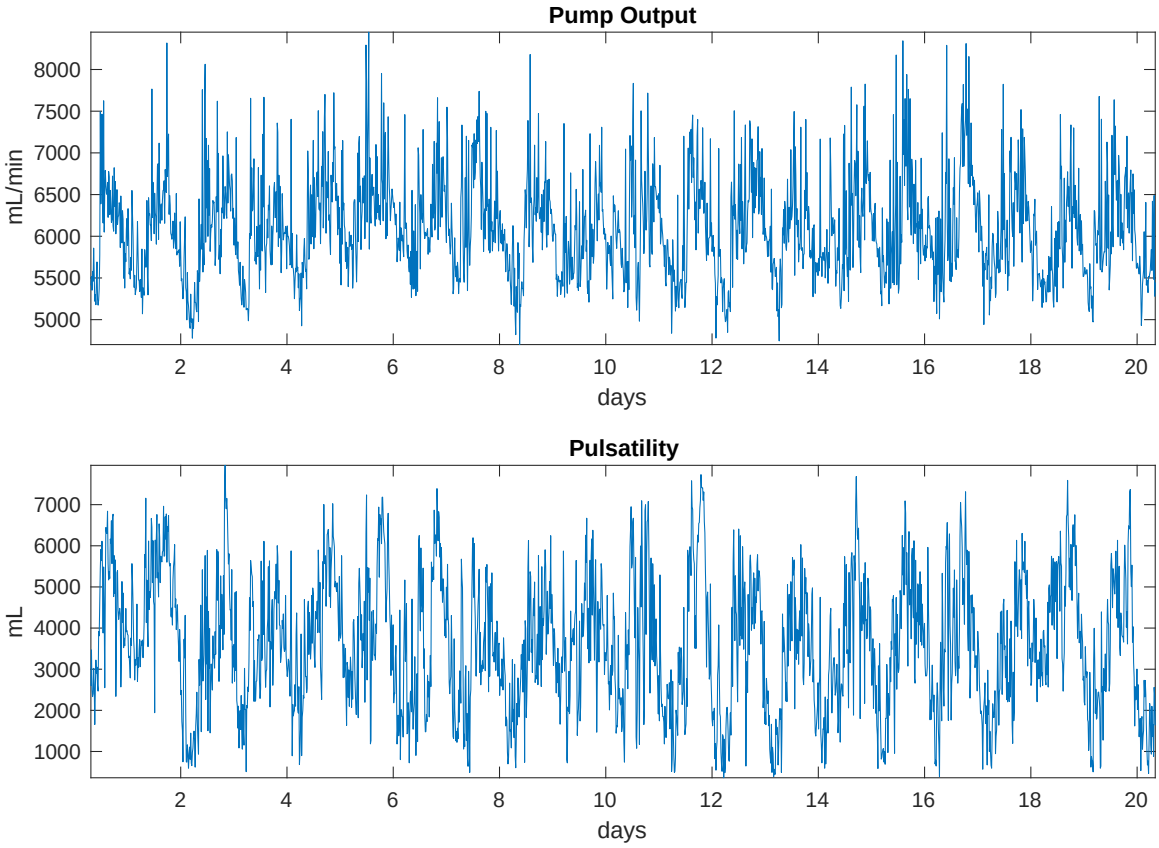

Figure 25: Sample profile for 20 days of pump output and pulsatility for patient 9.

APPENDIX B

ADDITION PERIODOGRAMS FOR VAD COHORT

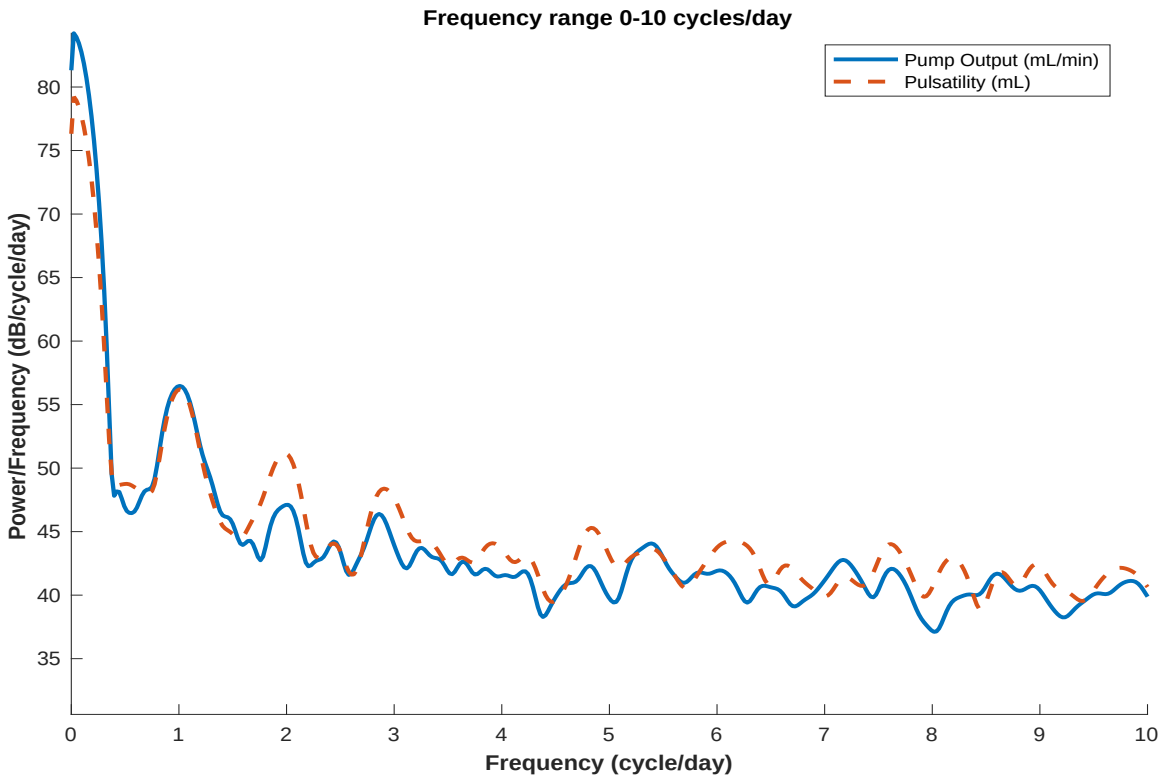

Figure 26: Periodogram from patient 2.

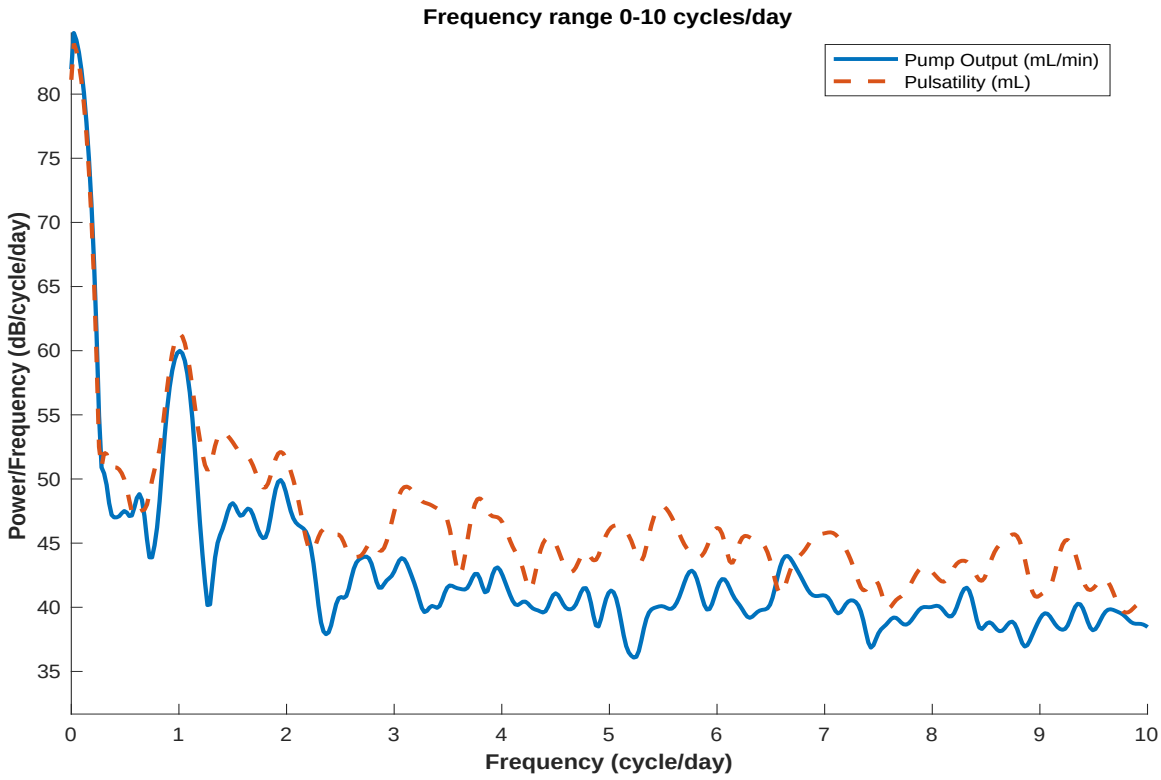

Figure 27: Periodogram from pateint 3.

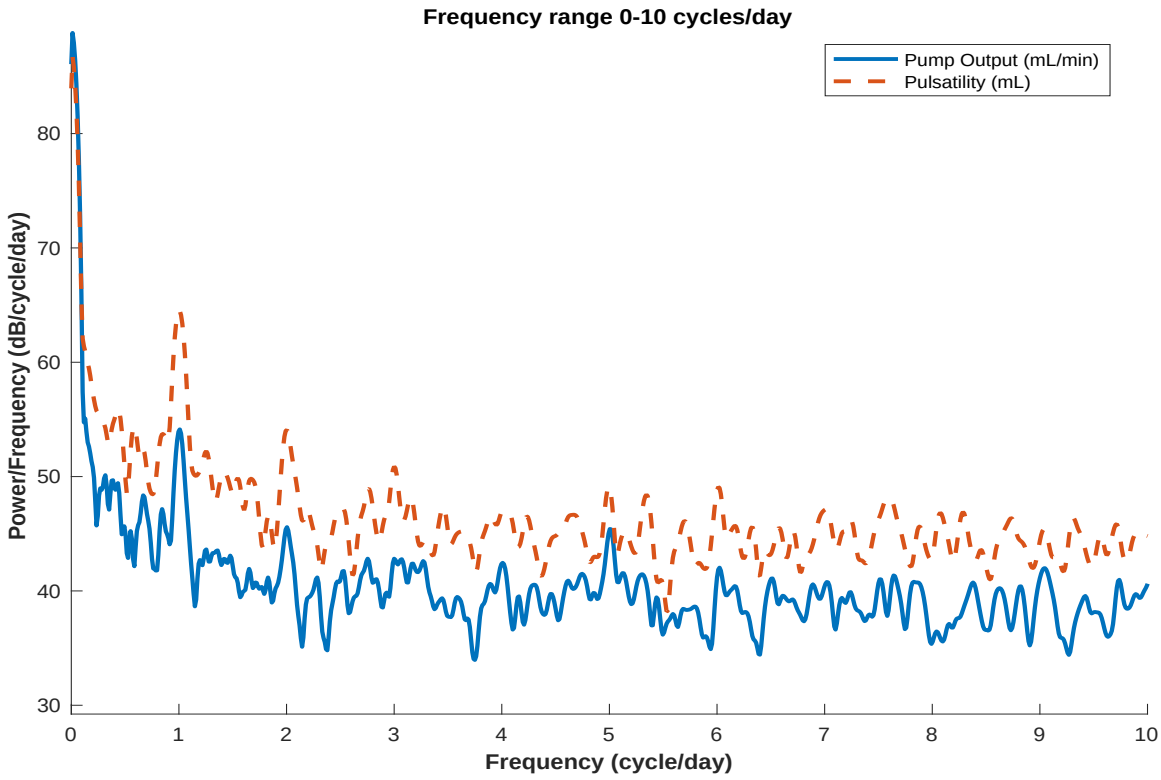

Figure 28: Periodogram from patient 4.

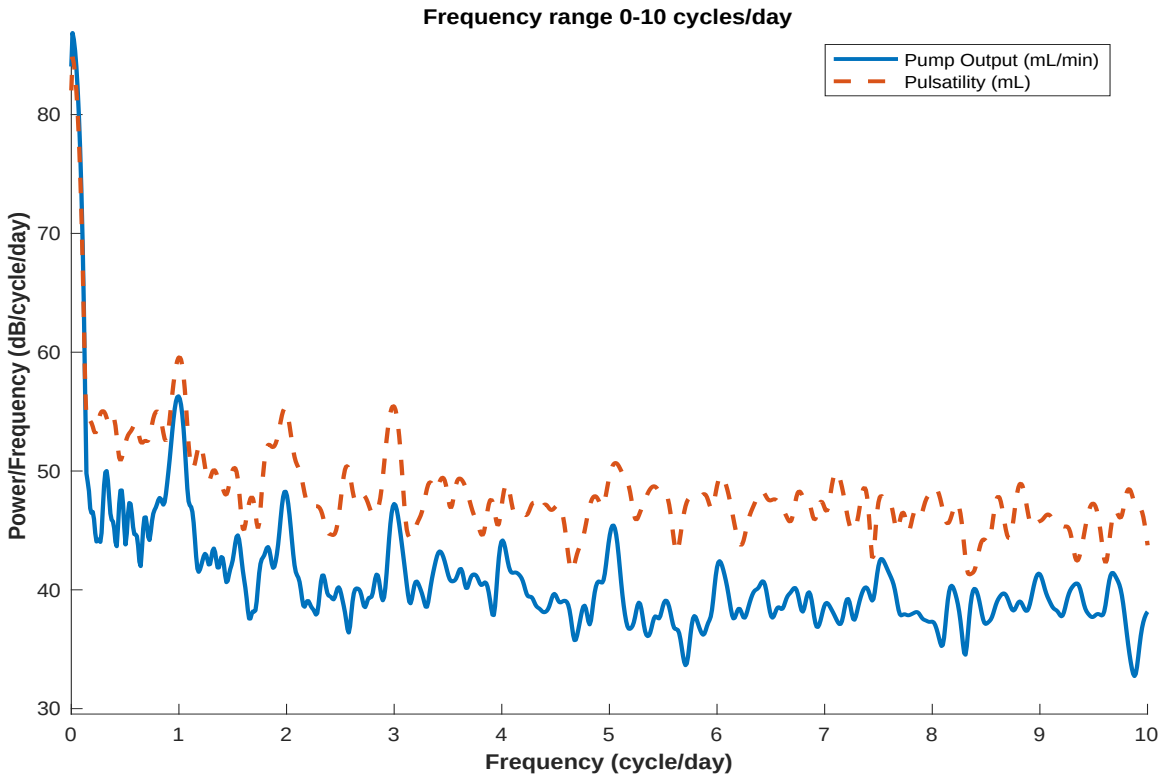

Figure 29: Periodogram from patient 5.

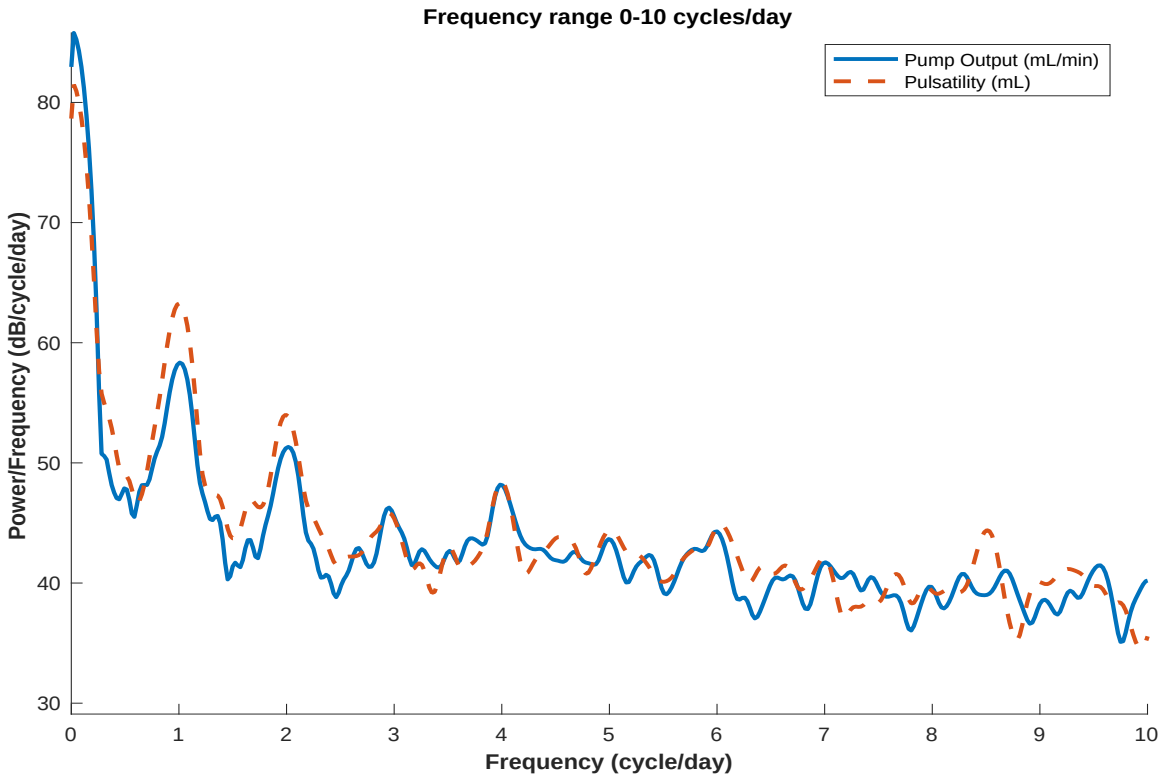

Figure 30: Periodogram from patient 6.

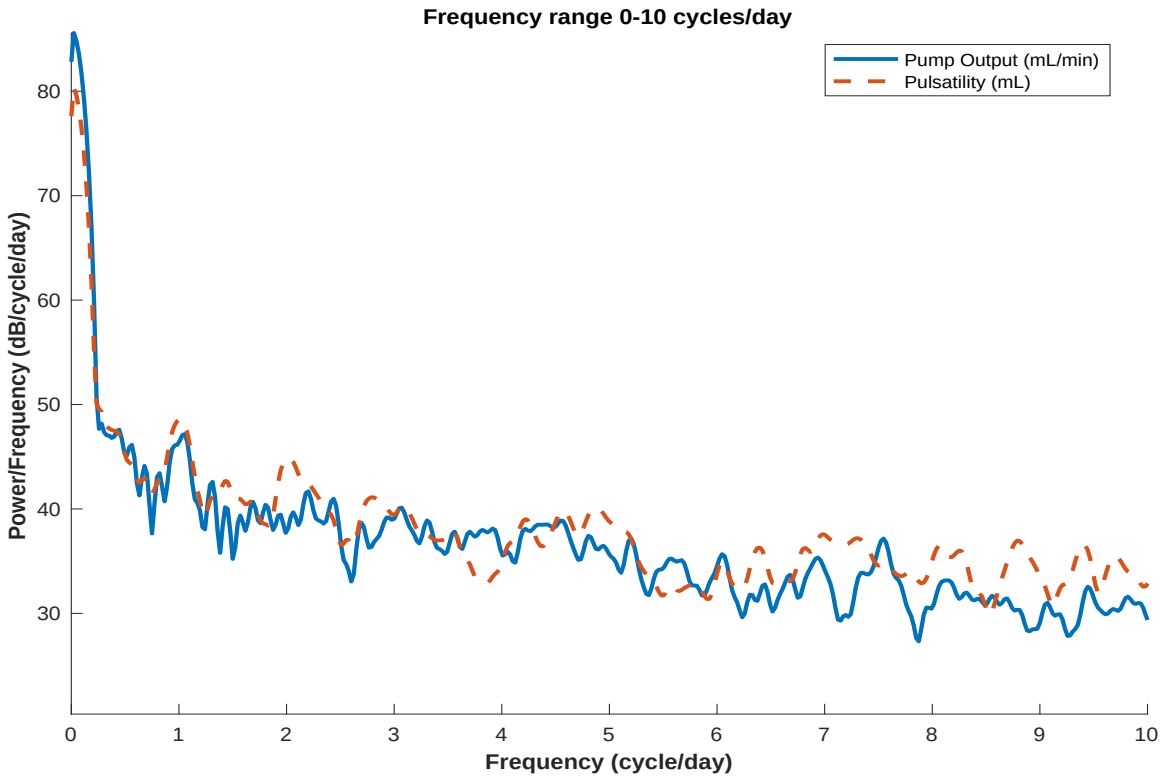

Figure 31: Periodogram from patient 7.

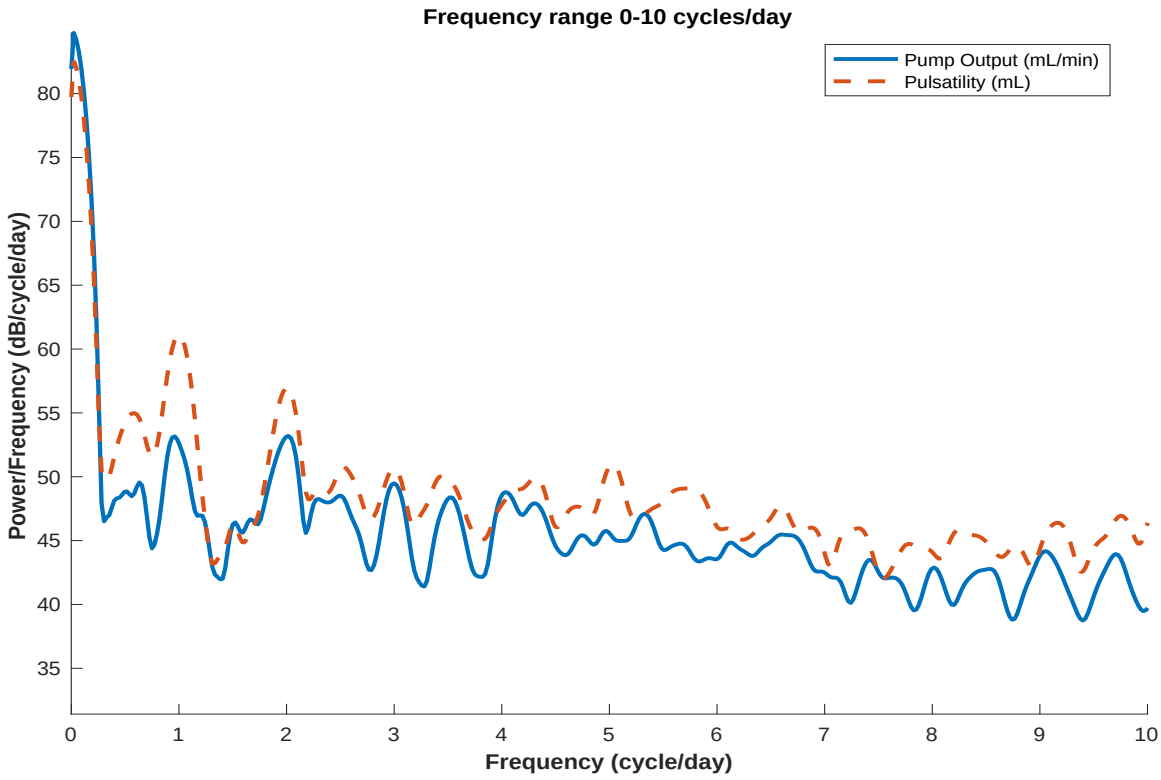

Figure 32: Periodogram from patient 8.

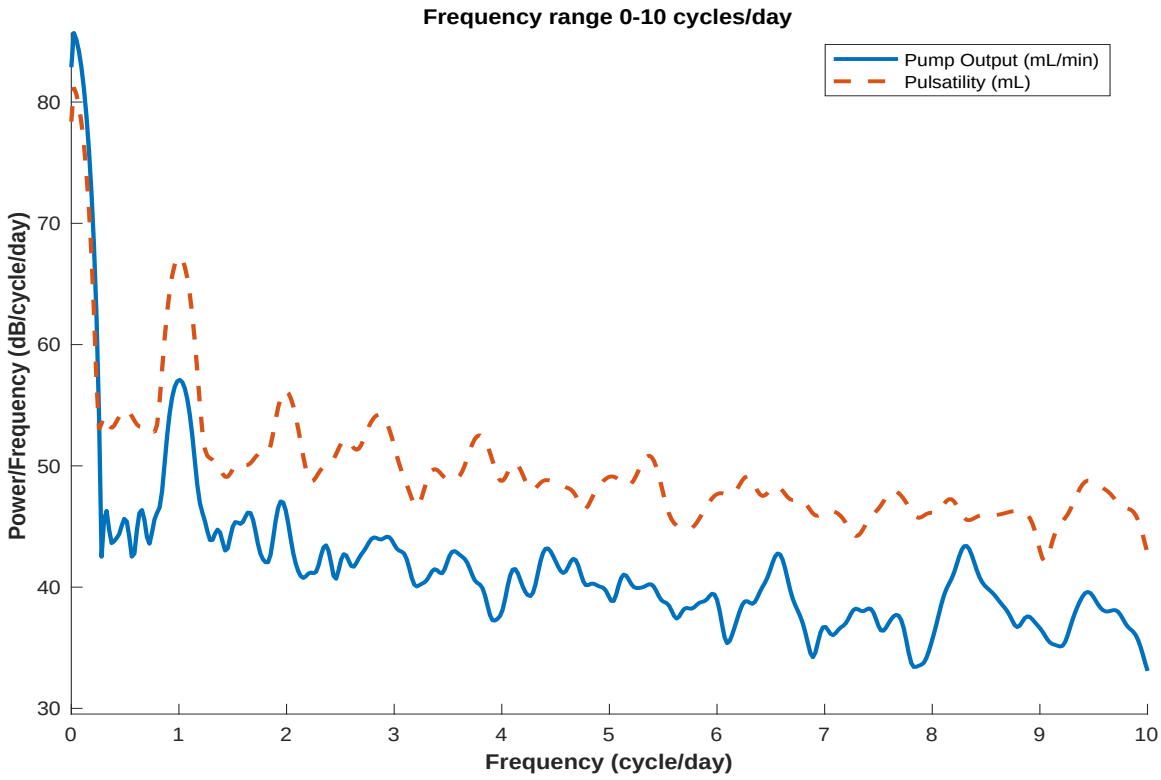

Figure 33: Periodogram from patient 9.

APPENDIX C

MATLAB CODE FOR FITTING MMFPCA

These functions are the primary MATLAB code for running MMFPCA.

```
% Andrew Potter - 9/30/2016
% MMFPCA master function
% 'Y' = observed data matrices (rows: within day, columns: days) organized
% 'nday' = number of slow time observation in each subject
% into a cell array with rows as outcome components and columns as subjects
% 'wnam' = wavelet name for the wavelet domain steps - must be one of the
% wavelets in the matlab wavelet toolbox
% 'thrRule' = wavelet threshold rule for 'wden' - must match matlab names
% 'theType' = hard 'h' or soft 's' threshold
% 'varScal' = 'sln' or 'mln' - noise level model for wden
% 'w' = constant for ASPCA
% 'thrFT_flag' = 1 puts a hard threshold on the fast time eigenfunctions
% 'fveThr' = FVE cutoff for number of eigenfunctions
% 'sSparse' = sparse follow-up in ST, use PACE
% 'sMax' = max slow time follow up if all slow time follow-up is not equal
% 'sBW' = smoothing bandwidth for slow time covariance
% 'sims' = 1 indicates that the data is simulated
% FT denotes fast time
% ST denotes slow time

function [outFT, outST, outPred] = MMFPCA(Y,wnam,thrRule,thrType,varScal,w,thrFT_flag,fveThr
    ,sSparse, sBW, sims)


D = size(Y,1); % number of outcome components
N = size(Y,2); % number of subjects
n_t = size(Y{1,1},1);
n_lev = wmaxlev(n_t,wnam); % number of levels in wavelet decomp
[~,l]  = wavedec(Y{1,1}(:,1),n_lev,wnam); %
n_wav = sum(l(1:(end-1)));

Y_wav = Y;  % wavelet transform of the data
Z = cell(D,N); % subject specfic means
FTmuW_raw = cell(1,D); % raw population average  FT means on wavelet domain
FTmuW = cell(1,D); % thresholded FT means on wavelet domain
FTmu = cell(1,D); % thresholded FT means on time domain
subj_dev = cell(1,D); % subject specific FT deviations
FTvecW = cell(1,D); % FT wavelet domain eigenvector
FTval = cell(1,D); % FT eigenvalues
FTk = cell(1,D); % FT number of eig vecs
FTvec = cell(1,D); % FT time domain eigenvector
FTfve = cell(1,D); % FT FVE
```

```
xbar = cell(1,D); % subject deviation data matrix
uFTfveCut = cell(1,D); % FVE cutoff for each fast time dimension
uFTsc = cell(1,D); % univariate FT scores xi
uFThat = cell(1,D); % univariate FT hat
uFTpred = cell(1,D); % univariate FT prediction
mvFTpred = cell(1,D); % multivariate FT prediction
uFTs = cell(1,D); % univariate observation cut to a regular grid
FTdevS = cell(1,D); % the daily subject specific deviations
uFTscS = cell(1,D); % the daily specific scores
nday = zeros(1,N);

for kk = 1:N
    nday(kk) = size(Y{1,kk},2);
end
if length(unique(nday)) > 1
    sMax = min(nday)-2;
else
    sMax = unique(nday);
end
for jj = 1:D
    for kk = 1:N
        Y_wav{jj,kk} = thr_bys(Y{jj,kk}, thrRule, thrType,varScal,wnam,1); % period by
    period wavelet transform
        Z{jj,kk} = mean(Y_wav{jj,kk},2); % subject specfic means on wavelet domain
    end
    FTmuW_raw{jj} = mean(cell2mat(Z(jj,:)),2); % raw mean
    [FTmu{jj},FTmuW{jj}] = wden(FTmuW_raw{jj},1,thrRule,thrType,varScal,n_lev,wnam); %
    thresholded mean
    subj_dev{jj} = bsxfun(@minus,cell2mat(Z(jj,:)), FTmuW{jj}); % raw subject deviation
    [~, FTvecW{jj},FTval{jj},FTk{jj}] = sparPCA_1d(subj_dev{jj}, w,thrFT_flag); % ASPCA
    FTvec{jj} = zeros(n_t, FTk{jj});
    % time domain eigenfunctions
    for kk = 1:FTk{jj}
        FTvec{jj}(:,kk) = waverec(FTvecW{jj}(:,kk), 1, wnam); % univariate phi on time
    domain
    end

    FTfve{jj} = cumsum(FTval{jj})/sum(FTval{jj});  %FVE criteria
    xbar{jj} = zeros(n_t, N);
    for kk = 1:N
        xbar{jj}(:,kk) = waverec(subj_dev{jj}(:,kk),1,wnam); % time domain fast time dev
    matrix
    end
    uFTfveCut{jj} = find(FTfve{jj} <= fveThr,1,'last'); %number of eigenfunctions
    [uFTsc{jj}, uFThat{jj}] = sparSC(xbar{jj},FTvec{jj},uFTfveCut{jj},1); % s constant FT
    scores
    uFTpred{jj} = bsxfun(@plus,FTmu{jj},uFThat{jj}); % FT subject specific predictions
    uFTs{jj} = zeros(n_t,N,sMax); % empty cell array for observed


    %fix this block of code to handle both equal length and unequal length
    %data.
    % adds zeros to the end of short records.
    for kk = 1:N
        if length(unique(nday)) == 1
            uFTs{jj}(:,kk,:) = Y{jj,kk};
        else
            temp = Y{jj,kk}(:,2:(sMax+1));
            uFTs{jj}(:,kk,:) = temp;
        end
    end
    FTdevS{jj} = bsxfun(@minus,uFTs{jj}, uFTpred{jj}); % calculates the daily residual
    function
    %
    for kk = 1:N
        for ll = 1:sMax
            uFTscS{jj}(kk,ll,:) = sparSC(uFTs{jj}(:,kk,ll),FTvec{jj},uFTfveCut{jj},0);
        end
```

```
    end
end

% at the moment, set the number of eigenfunctions to the minimum across all
% components
neig = min(cell2mat(uFTfveCut));
% cut the univariate score and vector matrices to size for MFPCA

Xi = zeros(N,neig*D);
Phi = zeros(n_t,neig*D);
uniSc = zeros(N,sMax,neig*D);
ii = 1;
for kk = 1:2:2*neig
    for jj = 1:D
        offset = jj-1;
        Xi(:,kk+offset) = uFTsc{jj}(:,ii);
        Phi(:,kk+offset) = FTvec{jj}(:,ii);
        uniSc(:,:,kk+offset) = uFTscS{jj}(:,:,ii);
    end
    ii = ii+1;
end

%% multivariate FPCA

n_eig = neig*ones(D,1);
[mEvec, mEval,uMSc, ~ ,cm] = mFPCA(Xi,n_eig, D, Phi);
%%
mFVE = cumsum(mEval)/sum(mEval);
neM = find(mFVE <= fveThr,1,'last');

%% Fast Time MV Predictions

Temp = zeros(2*n_t, N,neM);

for kk = 1:neM
        for jj = 1:N
            Temp(:,jj,kk) = uMSc(jj,kk) .* mEvec(:,kk);  %% look into here
        end
end
%%
hatMV = sum(Temp,3);
for ii = 1:D
    mvFTpred{ii} = bsxfun(@plus,hatMV(1+n_t*(ii-1):ii*n_t,:), FTmu{ii});
end


%% Daily mv scores

mvSc_s = zeros(N, sMax, neM);
for ii = 1:N
    for rr = 1:neM
        tempSc = bsxfun(@times,permute(uniSc(ii,:,:), [2 3 1]) , cm(:,rr)');
        mvSc_s(ii,:,rr) = sum(tempSc,2);
    end
end
%% ST FPCA
[~,Kv_s,Ke_s,FVE_s,trace_p,FVEcuts,~] = stFPCA(mvSc_s,sBW,[],sMax,[],1,fveThr,0); %modify
    these inputs


%% Complete Signal reconstruction
neS = min(FVEcuts);
hat_stSC = zeros(N,neS, neM);
hat_stpred = zeros(sMax,N,neM);
for pp = 1:neM
    [hat_stSC(:,:,pp), hat_stpred(:,:,pp)] = sparSC(mvSc_s(:,:,pp)', Kv_s{pp},neS,1);
end
%%
```

```
means = cell2mat(FTmu);
meanFT = means(:);
for ii = 1:N
     for kk = 1:sMax
        for jj = 1:neM
            predFS(:,jj) = hat_stpred(kk,ii,jj).*mEvec(:,jj);
        end
        Y_mvpred(:,kk,ii) = meanFT + hatMV(:,ii) + sum(predFS,2);
     end
end


outFT = struct('dimension',D,'meanRaw',FTmuW_raw,'meanThr',FTmu, 'subjDev',subj_dev,...
    'univarFTeigVectorW',FTvecW,'univarFTeigValue',FTval,'univarFTk',FTk,'univarFTeigVector'
    ,FTvec,...
    'univarFTfve',FTfve,'xbar',xbar,'FTfveCut',uFTfveCut,'uFTsc',uFTsc,'uFTscS',uFTscS,...
    'Xi', Xi,'Phi',Phi,'mvFTeigVector',mEvec,'mvFTeigValue',mEval, 'mvFTpred', ...
    mvFTpred, 'mvScoresS',mvSc_s);
outST = struct('STeigVector',Kv_s, 'STeigValue',Ke_s, 'FVE_ST',FVE_s,...
    'STtrace',trace_p,'FVE_ST_coff',FVEcuts);
outPred = Y_mvpred;
end
```

MMFPCA.m

```
% Andrew Potter 4/28/2015
% Multivariate Sparse FPCA
% Implements the method of Happ and Greven only looking at the multivariate
% part of the FPCA decomp.  2-d only

% Inputs: appropriately formatted univariate score matrix (rows -
% observations, columns - scores), vector of number of univariate FPCA
% components, total number of functions, matrix of univariate
% eigenfunctions (rows - time points, columns - eigenfunctions)

% Outputs: multivariate eigenvectors, multivariate eigenvalues, Z_hat,
% multivariate scores

function [mEvec, mEval, mScore, Z_hat,cm] = mFPCA(Xi, n_eig, dim, Phi)
    n = size(Xi,1);
    M_max = size(Xi,2);
    t = size(Phi,1);
    Z_hat = Xi'*Xi/(n-1);
    [Kv, Ke] = eig(Z_hat,'vector');
    [mEval,eI] = sort(Ke,'descend');
    cm = Kv(:,eI);

    % estimate the multivariate eigenfunctions
    mEvec = zeros(2*t, M_max);
    mScore = zeros(n,M_max);
    odd = 1:2:2*n_eig;
    even = 2:2:2*n_eig;
    for ll = 1:M_max
        for jj = 1:dim
            m_j = n_eig(jj);
            if jj == 1
                c_temp = cm(odd,ll);
                u_temp = Phi(:,odd);
            else
                c_temp = cm(even,ll);
                u_temp = Phi(:,even);
            end
            temp = bsxfun(@times, u_temp, c_temp');
            mEvec(((jj-1)*t+1):jj*t,ll) = sum(temp,2);
        end
```

```
        tempSc = bsxfun(@times, Xi, cm(:,ll)');
        mScore(:,ll) = sum(tempSc,2);
    end
return
end
```

mFPCA.m

```
% Andrew Potter - 4/21/2016
% Implimentation of Sparse PCA of Johnstone and Lu 2009

function [S,evec, eval, k] = sparPCA_1d(X,w,thr)
% takes in transform domain data and returns matrix eigenvectors on the
% transformed domain.
% X is the data on the transform domain with mean 0, rows are coefficient and columns are
    subjects.
% w is a constant
% thr is a threshold flag: 0 no threshold on eigenfunctions, 1 threshold on
% eigenfunctions.
p = size(X,1);
n = size(X,2);
S_n = zeros(p,n);

for ii = 1:n
    S_n(:,ii) = X(:,ii).^2;
end
sig_n = mean(S_n,2);  %PA covariance on the transform domain
%S=sig_n;
sig_hat = median(sig_n);

[sig_sort, sig_I] = sort(sig_n, 'descend');

chi_ptile = chi2inv(sig_I./(p+1),n-1);
eta_temp = [sig_sort - sig_hat.*chi_ptile./(n-1) zeros(p)];
eta2 = max(eta_temp,[],2);

eta_max = w*sum(eta2);
eta_tally = 0;
k = 0;

while eta_tally <= eta_max
    k = k+1;
    eta_tally = eta_tally+eta2(k);
end

ind_sm = sig_I(1:k);
X_sm = X(ind_sm,:);
S_sm_temp = zeros(k,k,n);
for ii = 1:n
    S_sm_temp(:,:,ii) = X_sm(:,ii) * X_sm(:,ii)';
end

S = mean(S_sm_temp,3);

% do PCA on S
[Kv, Ke] = eig(S,'vector');

[eval,eI] = sort(Ke,'descend');

Kv = Kv(:,eI);
evec = zeros(p,k);
for ii = 1:k
    evec(sig_I(ii),:) = Kv(ii,:);
end
% hard threshold on K_v to get evec
```

```
if thr == 1
    md = mad(Kv,1);  %MAD for each eigenvector
    thrK = (md./0.6745)*sqrt(2*log(k));
    Kvthr = Kv;
    for ii = 1:k
        sumy = sum(abs(Kv(:,ii))>thrK(ii));
        Kvthr(:,ii) =  Kv(:,ii).*(abs(Kv(:,ii))>thrK(ii));
        if sumy == 0
            normy = 1;
        else
            normy = norm(Kvthr(:,ii));
        end
        Kvthr(:,ii) = Kvthr(:,ii)/normy;
    end

evec = zeros(p,k);
for ii = 1:k
    evec(sig_I(ii),:) = Kvthr(ii,:);
end
end
end
```

sparPCA_1d.m

```
% Andrew Potter 4/28/2015
% SparFPCA scoring function
% Inputs: Univariate observed functions, eigenfunctions, on time domain,
% number of eigenvectors, the observation time vector, and if return Xhat.
% Returns: FPCA scores for each observed subject, FPCA estimate for each
% subject

function [xi, X_hat] = sparSC(X,evec, n_eigen, rXhat)
n = size(X,2); % number of subject
n_s = size(X,1); % vector of observation times
xi = zeros(n,n_eigen); % stores the scores
X_temp = zeros(n_s,n, n_eigen); % temp storage of the weighted eV for each subject
if n_eigen == 1
    evec = evec(:,1);
    prod_temp = bsxfun(@times, X,evec);
    for jj = 1:n
        xi(jj) = trapz(prod_temp(:,jj)); % eigen score for the
        if rXhat == 1
            X_temp(:,jj) = xi(jj) .* evec;  %% look into here
        end
    end
else
    for kk = 1:n_eigen
        prod_temp = bsxfun(@times, X,evec(:,kk));
        for jj = 1:n
            xi(jj,kk) = trapz(prod_temp(:,jj)); % eigen score for the
            if rXhat == 1
                X_temp(:,jj,kk) = xi(jj,kk) .* evec(:,kk);  %% look into here
            end
        end
    end
end

if rXhat == 1
    X_hat = sum(X_temp, 3);
end
end
```

sparSC.m

```
% slow time FPCA
function [Cov_s, Kv_s, Ke_s,FVE_s,trace_p,FVEcuts,Pooled] = stFPCA(mvSc_s,bws,stPCbasis,n_s,
    Q,pooled,FVE_thr,sims)
    P = size(mvSc_s,3);
    Cov_s = cell(P,1);
    Kv_s = cell(P,1);
    Ke_s = cell(P,1);
    FVE_s = zeros(n_s,P);
    trace_p = zeros(1,P);
    FVEcuts = zeros(1,P);
    for ff = 1:P
        cov_temp = cov(mvSc_s(:,:,ff));
        d_temp = .5*diag(cov_temp,-1) +.5*diag(cov_temp,1);
        d_temp2 = interp1(1:n_s-1,d_temp,1:n_s,'linear','extrap');
        cov_temp2 = cov_temp - diag(diag(cov_temp)) + diag(d_temp2);
        Cov_s{ff} = imgaussfilt(cov_temp2,bws);   %need to smooth for higher noise

        [Kv_s{ff},Ke_temp] = eig(Cov_s{ff});
        Ke_s{ff} = diag(Ke_temp);
        [Ke_s{ff},I] = sort(Ke_s{ff},'descend');
        Ke_s{ff} = Ke_s{ff}.*bsxfun(@ge, Ke_s{ff},0);
        Kv_s{ff} = Kv_s{ff}(:,I);
        FVE_s(:,ff) = cumsum(Ke_s{ff})/sum(Ke_s{ff});
        if sims == 1
            FVEcuts(ff) = Q;
        else
            FVEcuts(ff) = find(FVE_s(:,ff) <= FVE_thr,1,'last');%%% have an issue here in
    parfor.  Not sure wy!!!!
        end
        Kv_s{ff} = Kv_s{ff}(:,1:FVEcuts(ff));
        trace_p(ff) = sum(Ke_s{ff}(1:FVEcuts(ff)));
        if ~isempty(Q)
            if FVEcuts(ff) <= Q
                for ii = 1:FVEcuts(ff)
                    Kv_s{ff}(:,ii) = signchk(Kv_s{ff}(:,ii), stPCbasis');
                end
            end
        end
    end
    if pooled == 1
        cov_p = zeros(n_s,n_s);
        for pp = 1:P
            cov_p = cov_p + Cov_s{pp};
        end
        d_temp = .5*diag(cov_p,-1) +.5*diag(cov_p,1);
        d_temp2 = interp1(1:n_s-1,d_temp,1:n_s,'linear','extrap');
        cov_p2 = cov_p - diag(diag(cov_p)) + diag(d_temp2);
        Cov_pooled = imgaussfilt(cov_p2,bws);   %need to smooth for higher noise
        [Kv_pooled,Ke_temp] = eig(Cov_pooled);
        Ke_pooled = diag(Ke_temp);
        [Ke_pooled,I] = sort(Ke_pooled,'descend');
        Ke_pooled = Ke_pooled.*bsxfun(@ge, Ke_pooled,0);
        Kv_pooled = Kv_pooled(:,I);
        FVE_pooled = cumsum(Ke_pooled)/sum(Ke_pooled);
        if sims == 1
            FVEcuts_pooled = Q;
        else
            FVEcuts_pooled = find(FVE_pooled  <= FVE_thr,1,'last');%%% have an issue here in
     parfor.  Not sure wy!!!!
        end
        Kv_pooled = Kv_pooled(:,1:FVEcuts_pooled);
        trace_pooled = sum(Ke_pooled(1:FVEcuts_pooled));
        if ~isempty(Q)
            if FVEcuts_pooled < Q
                for ii = 1:FVEcuts_pooled
                    Kv_pooled(:,ii) = signchk(Kv_pooled(:,ii), stPCbasis');
                end
            end
```

```
        end
        Pooled = {Cov_pooled, Kv_pooled,Ke_pooled,FVE_pooled,FVEcuts_pooled,trace_pooled};
    end
    return
end
```

stFPCA.m

```
function [C_th, l] = thr_bys(Y,th_rule, th_type, th_scal, wnam_t, nodn)
    % a function that does a level dependent threshold de-noising on data
    % for each observed s value.  returns an array of size n_tXn_sXN of
    % de-noised wavelet coefficients and an index vector.

    n_t = size(Y,1);
    n_s = size(Y,2);
    N = size(Y,3);
    n_level = wmaxlev(n_t,wnam_t); % number of levels
    if nodn == 0
        [~,c,l] = wden(Y(:,1,1), th_rule, th_type,th_scal,n_level, wnam_t);
        C_th = zeros(length(c), n_s,N);
        for i = 1:N
            for j = 1:n_s
                [~, C_th(:,j,i)] = wden(Y(:,j,i), th_rule, th_type,th_scal,n_level, wnam_t);
            end
        end
    end
    if nodn == 1
        [c,l] = wavedec(Y(:,1,1),n_level, wnam_t);
        C_th = zeros(length(c), n_s,N);
        for i = 1:N
            for j = 1:n_s
                C_th(:,j,i) = wavedec(Y(:,j,i), n_level, wnam_t);
            end
        end
    end
end
```

thr_bys.m

Code for recreating the simulation studies presented in this dissertation. This code implements MMFPCA as a script instead of a function. We made this decision to prevent errors in FVE in the lowest noise level.

```
%% Andrew Potter 12/11/2016
%
% edit 7/13/16 to fix the normalizations
% simulates MMFPCA to estimate ISE, MISE, MSE for model parameters.


%% The simulation parameters
%
% $n_t = 96$
% $n_s = 50$
% $N = 30$
% $f(t) =-sin(2  \pi t /T)$, $T=24$
%
% a scalar valued function.
%
%clear
```

```
addpath(genpath('/Users/andrewpotter/Documents/MATLAB/'));
boundary='reflection';
reps = 3;%250;   %number of replicates at each level of rho and tau
n_t = 2^7; % number of fast time samples within one period
n_sA = [30 250];  % number of observed periods
bws = [2 15];
P = 2 ; % number of fast time eigenfunctions
Q = 1;       % number of slow time eigenfunctoins
N_all = [30 100 250];    %number of subjects
T = n_t;
t = linspace(0,n_t,n_t);

sig = [0.001 0.5]% 1 1.5];  % noise error

wnam = 'sym8';  % wavelet names
%%
hat_mu1 = cell(size(sig,2),1);  % fast time est 1
hat_mu2 = cell(size(sig,2),1);  % fast time est 2
hat_ke1 = cell(size(sig,2),1);  % fast time eigenvalues 1
hat_ke2 = cell(size(sig,2),1);  % fast time eigenvalues 2
hat_kv1 = cell(size(sig,2),1);  % fast time eigenfuns 1
hat_kv2 = cell(size(sig,2),1);  % fast time eigenfuns 2
hat_fve1 = cell(size(sig,2),1);  % fast time FVE 1
hat_fve2 = cell(size(sig,2),1);  % fast time FVE 2
hat_mv = cell(size(sig,2),1);    % multivar eigenFun
hat_me = cell(size(sig,2),1);    % multivar eigenVal
hat_fve = cell(size(sig,2),1);   % multivar FVE
hat_kvs = cell(size(sig,2),1);   % slow time eigenfuns
hat_kes = cell(size(sig,2),1);   % slow time eigenvals
hat_fves = cell(size(sig,2),1);  % slow time FVE
hat_trace_st = cell(size(sig,2),1);
hat_FVEcuts = cell(size(sig,2),1);
hat_covs = cell(size(sig,2),1);
% the data
% look at multiple values for the random effect.  Need to improve the re
% and eigenfunction to make sure it is from a K-L expansion of a covariance
%Simulation: Continuous functions in s of level 5 detail and approx coefs

%%


%%
X_ft_true1 = -5*sin(2*pi*t/T);
X_ft_true2 = -2.5*sin(2*pi*t/T);


rho = linspace(5, 3, P);
rhoQ = 4;
cor = linspace(2, 0, P);
% ftPCbasis = fourier(t, P, T,0);  % fast time basis
% for pp = 1:P
%     ftPCbasis(:,pp) = ftPCbasis(:,pp)/sqrt(trapz(ftPCbasis(:,pp).^2));
% end

%
ftPCbasis = zeros(n_t,P);

%
for pp = 1:P
    if pp == 1
        for ii = 1:n_t
            if ii >floor(.65*n_t) && ii<floor(.85*n_t)
                ftPCbasis(ii,pp) = -(ii-floor(.75*n_t))^2 + 150;
            else
                ftPCbasis(ii,pp) = 0;
            end
        end
```

```
    elseif pp == 2
        for ii = 1:n_t
            if ii >floor(.15*n_t) && ii<floor(.35*n_t)
                ftPCbasis(ii,pp) = (ii-floor(.25*n_t))^2-150;
            else
                ftPCbasis(ii,pp) = 0;
            end
        end
    end
    ftPCbasis(:,pp) = ftPCbasis(:,pp)/sqrt(trapz(ftPCbasis(:,pp).^2));
end
%%
mu = [0 0];
% sigma_true = zeros(2 ,2, P);
% for ii = 1:P
%     sigma_true(:,:,ii) = [rho(ii) cor(ii); cor(ii) rho(ii)-.5];
% end

sigma_true = [4 1; 1 2];
sigma_true(:,:,2) = [3 .5 ; .5 1];
%sigma_true(:,:,3) = [3 0 ; 0 .5];
ztrue = blkdiag(sigma_true(:,:,1),sigma_true(:,:,2))
%%
ztrue = ztrue([1 3 2 4], [1 3 2 4 ])

[cmTru, nuTru] = eig(ztrue);
[nuTrue, ind] = sort(diag(nuTru), 'descend');
cmTrue = cmTru(:,ind)
%%
mvPCbasis = zeros(2*n_t, 2*P);
mvPCbasis(:,1) = [ftPCbasis(:,1)*cmTrue(1,1); ftPCbasis(:,1)*cmTrue(3,1)];
mvPCbasis(:,2) = [ftPCbasis(:,2)*cmTrue(2,2); ftPCbasis(:,2)*cmTrue(4,2)];
mvPCbasis(:,3) = [ftPCbasis(:,1)*cmTrue(1,3); ftPCbasis(:,1)*cmTrue(3,3)];
mvPCbasis(:,4) = [ftPCbasis(:,2)*cmTrue(2,4); ftPCbasis(:,2)*cmTrue(4,4)];

%% Plot the ft PC basis

subplot(4,2,1)
plot( mvPCbasis(1:n_t,1),'LineWidth',2)
axis([0 128 -.3 .3])
title({'1^{st} Component:'; '\phi_1(t)'},'FontWeight','normal')

subplot(4,2,3)
plot(mvPCbasis(1:n_t,2),'LineWidth',2)
axis([0 128 -.3 .3])
title('\phi_2(t)','FontWeight','normal')
subplot(4,2,5)
plot(mvPCbasis(1:n_t,3),'LineWidth',2)
axis([0 128 -.3 .3])
title('\phi_3(t)','FontWeight','normal')
subplot(4,2,7)
plot(mvPCbasis(1:n_t,4),'LineWidth',2)
axis([0 128 -.3 .3])
title('\phi_4(t)','FontWeight','normal')
% 2nd comp
subplot(4,2,2)
plot( mvPCbasis(n_t+1:end,1),'LineWidth',2)
axis([0 128 -.3 .3])
title({'2^{nd} Component:'; '\phi_1(t)'},'FontWeight','normal')
subplot(4,2,4)
plot(mvPCbasis(n_t+1:end,2),'LineWidth',2)
axis([0 128 -.3 .3])
title('\phi_2(t)','FontWeight','normal')
subplot(4,2,6)
plot(mvPCbasis(n_t+1:end,3),'LineWidth',2)
axis([0 128 -.3 .3])
title('\phi_3(t)','FontWeight','normal')
subplot(4,2,8)
```

```
plot(mvPCbasis(n_t+1:end,4),'LineWidth',2)
axis([0 128 -.3 .3])
title('\phi_4(t)','FontWeight','normal')
%print('ftPC.eps','-depsc')

%% Get DWT by days for all patients.
n_lev = wmaxlev(n_t,wnam);
%W = Get_DWT(wnam, n_t, boundary, 1, n_lev);


%%
parpool('local')
%% Simulation loop

tic
parfor_progress(size(sig, 2));
%g=1
parfor g = 1:size(sig, 2)
    % Report current estimate in the waitbar's message field
    parfor_progress;

    sigl = sig(g);
    hat_mu1_temp = zeros(n_t,reps,size(N_all,2),size(n_sA,2));  % fast time est 1
    hat_mu2_temp = zeros(n_t,reps,size(N_all,2),size(n_sA,2));  % fast time est 2
    hat_ke1_temp = zeros(P,reps,size(N_all,2),size(n_sA,2));  % fast time eigenvalues 1
    hat_ke2_temp = zeros(P,reps,size(N_all,2),size(n_sA,2));  % fast time eigenvalues 2
    hat_kv1_temp = zeros(n_t,P,reps,size(N_all,2),size(n_sA,2)); % fast time eigenfuns 1
    hat_kv2_temp = zeros(n_t,P,reps,size(N_all,2),size(n_sA,2)); % fast time eigenfuns 2
    hat_fve1_temp = zeros(P,reps,size(N_all,2),size(n_sA,2)); % fast time FVE 1
    hat_fve2_temp = zeros(P,reps,size(N_all,2),size(n_sA,2));  % fast time FVE 2
    hat_mv_temp =  zeros(2*n_t,2*P,reps,size(N_all,2),size(n_sA,2));   % multivar eigenFun
    hat_me_temp = zeros(2*P,reps,size(N_all,2),size(n_sA,2));    % multivar eigenVal
    hat_fve_temp = zeros(2*P,reps,size(N_all,2),size(n_sA,2));   % multivar FVE
    hat_kvs_temp = cell(reps,size(N_all,2),size(n_sA,2));   % slow time eigenfuns
    hat_kes_temp = cell(reps,size(N_all,2),size(n_sA,2));   % slow time eigenvals
    hat_fves_temp = cell(reps,size(N_all,2),size(n_sA,2));  % slow time FVE
    hat_trace_temp = cell(reps,size(N_all,2),size(n_sA,2)); %
    hat_FVEcuts_temp = cell(reps,size(N_all,2),size(n_sA,2));
    hat_covs_temp = cell(reps,size(N_all,2),size(n_sA,2));
    for aa = 1:size(N_all,2)
        N = N_all(aa);
        for cc = 1:size(n_sA,2)
            n_s = n_sA(cc);
            for bb = 1:reps

                % Slow Time FPCA
                s = linspace(0,n_s-1,n_s);
                stPCbasis = -s/n_s + .5; % slow time basis
                stPCbasis = stPCbasis/trapz(stPCbasis.^2);
%                 stPCbasis = fourier(s, Q, n_s,0); % slow time basis
%                 for qq = 1:Q
%                     stPCbasis(:,qq) = stPCbasis(:,qq)/sqrt(trapz(stPCbasis(:,qq).^2));
%                 end

                X = zeros(n_t,n_s,N);
                X_sim1 = zeros(n_t,n_s,N);
                X_sim2 = zeros(n_t,n_s,N);

                for i = 1:N
                    for j = 1:n_s
                       X_sim1(:,j,i) = X_ft_true1;
                       X_sim2(:,j,i) = X_ft_true2;
                    end
                end
                % Fast Time FPCA

                ftSC = zeros(N,2,P);
                for jj = 1:P
```

```
222                     ftSC(:,:,jj) = mvnrnd(mu,sigma_true(:,:,jj),N);
                    end
224

226                 stSC = bsxfun(@times,randn(N,Q,P),rhoQ);

228                 % The simmed data

230                 tempF1 = zeros(n_t,P);
                    tempF2 = zeros(n_t,P);
232                 ftDev1 = zeros(n_t,N);

234                 stDev = zeros(n_s,N,P);
                    Y_sim1 = X_sim1;
236                 ftDev2 = zeros(n_t,N);

238                 Y_sim2 = X_sim2;
                    tempS = zeros(n_s,Q);
240                 tempFS = zeros(n_t,P);
                    for ii = 1:N
242                     for jj = 1:P

244                         tempS = stSC(ii,1,jj).* stPCbasis;

246                         stDev(:,ii, jj) = tempS;
                            tempF1(:,jj) = ftSC(ii,1,jj).*ftPCbasis(:,jj);
248                         tempF2(:,jj) = ftSC(ii,2,jj).*ftPCbasis(:,jj);
                        end
250                     ftDev1(:,ii) = sum(tempF1,2);
                        ftDev2(:,ii) = sum(tempF2,2);
252                     for kk = 1:n_s
                            for jj = 1:P
254                             tempFS(:,jj) = stDev(kk,ii,jj).*ftPCbasis(:,jj);
                            end
256                         Y_sim1(:,kk,ii) = X_sim1(:,kk,ii) + ftDev1(:,ii) + sum(tempFS,2);
                            Y_sim2(:,kk,ii) = X_sim2(:,kk,ii) + ftDev2(:,ii) + sum(tempFS,2);
258                     end
                    end
260

262                 % add noise
                    sz = size(Y_sim1);
264                 noise1 = sigl.* Y_sim1.*randn(sz);
                    noise2 = sigl.* Y_sim2.*randn(sz);
266                 Y1 = Y_sim1 + noise1;
                    Y2 = Y_sim2 + noise2;
268

                    [~,l]  = wavedec(Y1(:,1,1),n_lev,wnam);
270                 n_wav = sum(l(1:(end-1)));
                    %
272

                    z1 = zeros(n_wav,n_s,N);
274                 z2 = zeros(n_wav,n_s,N);
                    zbar1 = zeros(n_wav,N);
276                 zbar2 = zeros(n_wav,N);
                    for k = 1:N
278                     z1(:,:,k) = thr_bys(Y1(:,:,k), 'rigrsure','s','sln',wnam,1);
                        z2(:,:,k) = thr_bys(Y2(:,:,k), 'rigrsure','s','sln',wnam,1);
280                     zbar1(:,k) = mean(z1(:,:,k),2);
                        zbar2(:,k) = mean(z2(:,:,k),2);
282                 end

284                 % unsmoothed MME for mean
                    alpha_hm1 = mean(zbar1,2);
286                 alpha_hm2 = mean(zbar2,2);

288                 % thresholded means
```

```
            [hat_mu1_temp(:,bb,aa,cc),hat_alpha1] = wden(alpha_hm1, l,'rigrsure','s','
mln',n_lev,wnam);
290         [hat_mu2_temp(:,bb,aa,cc),hat_alpha2] = wden(alpha_hm2, l,'rigrsure','s','
mln',n_lev,wnam);

292
            % covariance matrix
294         ss_dev1 = bsxfun(@minus, zbar1, hat_alpha1);
            ss_dev2 = bsxfun(@minus, zbar2, hat_alpha2);
296
            % Adaptive Sparse PCA univariate
298
            [~, Kv1,ke1,k1] = sparPCA_1d(ss_dev1, 0.999,1);
300         [~, Kv2,ke2,k2] = sparPCA_1d(ss_dev2, 0.999,1);
            hat_ke1_temp(:,bb,aa) = ke1(1:P);
302         hat_ke2_temp(:,bb,aa) = ke2(1:P);
            Kv1_ft = zeros(n_t, k1);
304         Kv2_ft = zeros(n_t, k2);

306         for jj = 1:min(k1,k2)
                Kv1_ft(:,jj) = waverec(Kv1(:,jj), l, wnam);
308             Kv2_ft(:,jj) = waverec(Kv2(:,jj), l, wnam);
            end
310
            FVE1 = cumsum(hat_ke1_temp(:,bb,aa))/sum(hat_ke1_temp(:,bb,aa));
312         hat_fve1_temp(:,bb,aa) = FVE1(1:P);
            FVE2 = cumsum(hat_ke2_temp(:,bb,aa))/sum(hat_ke2_temp(:,bb,aa));
314         hat_fve2_temp(:,bb,aa) = FVE2(1:P);
            % get sign agreement between sim and est phi
316         ne = P;
            for jj = 1:ne;
318             hat_kv1_temp(:,jj,bb,aa,cc) = signchk(Kv1_ft(:,jj), ftPCbasis(:,jj));
                hat_kv2_temp(:,jj,bb,aa,cc) = signchk(Kv2_ft(:,jj), ftPCbasis(:,jj));
320         end

322         % time domain fast time data matrix
            ybar1 = zeros(n_t, N);
324         ybar2 = zeros(n_t, N);

326         for jj = 1:N
                ybar1(:,jj) = waverec(ss_dev1(:,jj),l,wnam);
328             ybar2(:,jj) = waverec(ss_dev2(:,jj),l,wnam);
            end
330

332         % univariate FPCA scores

334         [sc1, hat1] = sparSC(ybar1,hat_kv1_temp(:,:,bb,aa,cc),ne,1);
            [sc2, hat2] = sparSC(ybar2,hat_kv2_temp(:,:,bb,aa,cc),ne,1);
336
            Xi = [sc1 sc2];
338
            Phi = [hat_kv1_temp(:,:,bb,aa,cc) hat_kv2_temp(:,:,bb,aa,cc)];
340
            pred1 = bsxfun(@plus,hat1,  hat_mu1_temp(:,bb,aa));
342         pred2 = bsxfun(@plus,hat2,  hat_mu2_temp(:,bb,aa));


344

346             % multivariate FPCA

348             n_eig = [ne,ne];
                [mv, me,mSc, Z_hat,cm] = mFPCA(Xi,n_eig, 2, Phi);  %%% add a signchk
step to the mFPCA function.
350             hat_mv_temp(:,:,bb,aa,cc) = mv(:,1:2*P);
                hat_me_temp(:,bb,aa,cc) = me(1:2*P);
352             %
                mFVE = cumsum(hat_me_temp(:,bb,aa,cc))/sum(hat_me_temp(:,bb,aa,cc));
```

```
354                     hat_fve_temp(:,bb,aa,cc) = mFVE(1:2*P);

356                     %
                        v1 = bsxfun(@minus,permute(Y1,[1 3 2]), pred1);
358                     v2 = bsxfun(@minus,permute(Y2,[1 3 2]), pred2);

360                     % get the daily FPCA scores

362                     sc1_s = zeros(N, n_s, ne); %subject, day, eigen vector
                        sc2_s = zeros(N, n_s, ne); %subject, day, eigen vector
364                     for ii = 1:N
                            for jj = 1:n_s
366                             sc1_s(ii,jj,:) = sparSC(v1(:,ii,jj),Kv1_ft,ne,0);
                                sc2_s(ii,jj,:) = sparSC(v2(:,ii,jj),Kv2_ft,ne,0);
368                         end
                        end
370                     uniSc = cat(3,sc1_s, sc2_s);
                        % Daily mv scores
372     %
                        mvSc_s = zeros(N, n_s, 2*P);
374                     for ii = 1:N
                            for ll = 1:2*P
376                             tempSc = bsxfun(@times,permute(uniSc(ii,:,:), [2 3 1]) , cm(:,ll
    )');
                                mvSc_s(ii,:,ll) = sum(tempSc,2);
378                         end
                        end
380     % Slow time FPCA
```

MSAsims_jasaapps.m

BIBLIOGRAPHY

Anderson, T. W. (2003). *An Introduction to Multivariate Statistical Analysis*. Wiley, 3rd edition.

Chen, K., Delicado, P., and Müller, H.-G. (2016). Modeling functional-valued stochastic processes, with applications to fertility dynamics. To appear: *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, 2016.

Chen, K. and Müller, H.-G. (2012). Modeling repeated functional observations. *Journal of the American Statistical Association*, 107(500):1599–1609.

Crainiceanu, C. M., Staicu, A.-M., Ray, S., and Punjabi, N. (2012). Bootstrap-based inference on the difference in the means of two correlated functional processes. *Statistics in Medicine*, 31(26):3223–3240.

Craven, P. and Wahba, G. (1979). Smoothing noisy data with spline functions. *Numerische Mathematik*, 403:377–403.

de Boor, C. (1972). On calculating with B-splines. *Journal of Approximation Theory*, 6(1):50–62.

Donoho, D. L. and Johnstone, I. M. (1994). Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455.

Donoho, D. L. and Johnstone, I. M. (1995). Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90(432):1200–1224.

Eilers, P. H. C. and Marx, B. D. (1996). Flexible smoothing with B-splines and penalties. *Statistical Science*, 11(2):89–121.

Eilers, P. H. C. and Marx, B. D. (2003). Multivariate calibration with temperature interaction using two-dimensional penalized signal regression. *Chemometrics and Intelligent Laboratory Systems*, 66(2):159–174.

Fink, J. P., Hall, W. S., and Hausrath, A. R. (1974). A convergent two-time method for periodic differential equations. *Journal of Differential Equations*, 15(3):459 – 498.

Greven, S., Crainiceanu, C. M., Caffo, B., and Reich, D. (2010). Longitudinal functional principal component analysis. *Electronic Journal of Statistics*, 4:1022–1054.

Gu, C. (2013). *Smoothing Spline ANOVA Models*. Springer, 2nd edition.

Happ, C. and Greven, S. (2015). Multivariate functional principal component analysis for data observed on different (dimensional) domains. *arXiv Preprint arXiv1509.02029v1*.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 2nd edition.

Johnstone, I. M. (2013). *Gaussian estimation: Sequence and wavelet models*. Draft.

Johnstone, I. M. and Lu, A. Y. (2009). On consistency and sparsity for principal components analysis in high dimensions. *Journal of the American Statistical Association*, 104(486):682–693.

Johnstone, I. M. and Silverman, B. W. (1997). Wavelet threshold estimators for data with correlated noise. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 59(2):319–351.

Krafty, R. T. and Collinge, W. O. (2013). Penalized multivariate Whittle likelihood for power spectrum estimation. *Biometrika*, 100(2):447–458.

Lombardi, F., Sandrone, G., Mortara, A., La Rovere, M., Colombo, E., Guzzetti, S., and Malliani, A. (1992). Circadian variation of spectral indices of heart rate variability after myocardial infarction. *American Heart Journal*, 123(6):1521–1529.

Mallat, S. (1999). *A wavelet tour of signal processing*. Academic press.

Millar-Craig, M., Bishop, C., and Raftery, E. (1978). Circadian variation of blood-pressure. *The Lancet*, 311(8068):795–797.

Nievergelt, Y. (2001). *Wavelets Made Easy*. Birkhäuser, 2nd edition.

Park, S. Y. and Staicu, A.-M. (2015). Longitudinal functional data analysis. *Stat*, 4(1):212–226. sta4.89.

Pini, A. and Vantini, S. (2016). The interval testing procedure: A general framework for inference in functional data analysis. *Biometrics*, 72(3):835–845.

Ramsay, J. O. and Silverman, B. W. (2005). *Functional Data Analysis*. Springer.

Refinetti, R., Cornélissen, G., and Halberg, F. (2007). Procedures for numerical analysis of circadian rhythms. *Biological Rhythm Research*, 38(4):275–325. PMID: 23710111.

Schumaker, L. L. (2007). *Spline Functions: Basic Theory*. Cambridge University Press, 3rd edition.

Shumway, R. H. and Stoffer, D. S. (2011). *Time Series Analysis and Its Applications*. Springer.

Silverman, B. W. (1985). Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 47(1):1–52.

Slaughter, M. S., Ising, M. S., Tamez, D., O'Driscoll, G., Voskoboynikov, N., Bartoli, C. R., Koenig, S. C., and Giridharan, G. A. (2010). Increase in circadian variation after continuous-flow ventricular assist device implantation. *The Journal of Heart and Lung Transplantation*, 29(6):695 – 697.

Strogatz, S. H. (1994). *Nonlinear Dynamics and Chaos*. Westview Press, 1st edition.

Suzuki, K., Nishinaka, T., Miyamoto, T., Ichihara, Y., and Yamazaki, K. (2014). Circadian variation of motor current observed in fixed rotation speed continuous-flow left ventricular assist device support. *Journal of Artificial Organs*, 17(2):157–161.

Takeda, N. and Maemura, K. (2011). Circadian clock and cardiovascular disease. *Journal of Cardiology*, 57(3):249 – 256.

van de Borne, P., Leeman, M., Primo, G., and Degaute, J.-P. (1992). Reappearance of a normal circadian rhythm of blood pressure after cardiac transplantation. *The American Journal of Cardiology*, 69(8):794–801.

Welch, P. D. (1967). The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *Audio and Electroacoustics, IEEE Transactions on*, 15(2):70–73.

Xiao, L., Yingxing, L., and Ruppert, D. (2013). Fast bivariate P-splines: the sandwich smoother. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, 75(Part 3):577–599.

Yao, F., Müller, H.-G., and Wang, J.-L. (2005). Functional data analysis for sparse longitudinal data. *Journal of the American Statistical Association*, 100(470):577–590.