A Functional Data Analysis Approach to Looking at Handwriting Data

The University of Michigan, Department of Statistics
Honors College, Class of 2016

Student: MA Tian-wen **Advisor:** Dr. EDWARD Rothman

[Abstract] Handwriting is a complicated and individual-oriented movement that involves fingers, wrist and forearm; handwriting recognition has played an important role in text recognition, writer identification, and forgery detection. Two major methods including shape analysis and movement analysis were developed to handle such problem. This thesis applies the latter method by reviewing the principal differential model developed by James Ramsay and uses new data to justify the model. Proper curve registration technique has been applied to new data before performing the principal differential method. The model captures the writing features well and yields satisfactory categorizing results.

[Key Words] Writer identification; Functional data analysis; Curve registration; Principal differential models

Table of Contents

1. Problem and Background Information

1.1. What is the problem?

Handwriting is a very complicated movement that involves fingers, wrist, and forearm. Handwriting results display variations not only among individuals, but also within individuals (Ramasy, 2000). Since handwriting recognition plays an important role in improvement of text recognition systems, PC-personalization, writer identification, and forgery detection, it has gained increasing attention in research and application (Elarian, Abdel-Aal, Ahmad, Parvez, & Zidouri, 2014). This thesis aims to review the principal differential model developed by James Ramsay and apply it to the new data set to understand the classification mechanism. Proper curve registration technique has been applied to new data before performing the principal differential method.

1.2. Shape Analysis and Movement Analysis

Elarian et al. (2014) concluded that there are two major directions of handwriting analysis. One direction looks at the outcome of the words, and characterizes words by their shapes and thickness of strokes. This bottom-up approach is called handwriting shape analysis. The way we handle the handwriting data is similar to signing your signature in the bank, and bank tellers verify your identity by your current signature and past records. However, studying the difference of shapes can sometimes be unreliable because shapes are static images, which means the method only considers the trace after someone has finished the writing. Moreover, it is highly possible for some well-trained criminals who are good at shape imitation to make a forgery. Therefore, shape analysis method may fail to detect the minor difference between the true and the forgery and yield unsatisfactory results.

The other direction of handling handwriting by recording the whole writing process can provide plausible solutions to the above problem. This top-down approach is called handwriting movement analysis. In fact, the motivation of the project is to improve the accuracy of signature verification in the bank. Since most signatures are processed via electronic devices such as POS machines and tablets, what if the machine is able to record the formation of a signature and compare it with past records? The method takes more than the static shape into account. Recording the whole process gives us the instantaneous feedback when the subject is writing, and it enables us to calculate the velocity and acceleration. Ramsay and Silverman (2002) proposed that these measurements can reflect the changes of their mental activities. For example,

although a person can show similar handwriting images under different mental circumstances, the velocity or acceleration plots can show drastic differences. This would inspire researchers to develop new methods to characterize individuals under different mental conditions by handwriting, and perhaps the application will make handwriting forgery harder than it is now.

2. Introduction to Functional Data Analysis

This chapter is a brief summary of the functional data analysis developed by Ramsay (2009). He created a new functional data object (I will call it FDA object for simplicity) in R for each dataset by using the combination of basis functions and proper choice of roughness penalties. Since handwriting is a very complicated movement, the coordinate plots for each data file display significant variations. On one hand, the FDA objects that are used to estimate the coordinate curves should capture features as much as possible. On the other hand, the number of parameters in the FDA objects should be as few as possible to reduce computations and avoid over-fitting.

2.1. The Spline Basis Functions

In order to fit successful FDA objects, spline basis functions are introduced. Splines are constructed by dividing the interval of observation into sub-intervals. The points at boundaries are called breaks. Splines can be regarded as piecewise polynomials that have fixed degrees over any sub-intervals. The degree is the highest power in the polynomial, and the order is defined as degree plus one. Splines can also be defined in terms of knots, which are related to breaks in the sense that each knot should have the same value as the break point, but the values may be different in the boundaries. In this setting, splines capture the complicated features of handwriting locally.

One particular requirement of splines is that neighboring polynomials are constrained to have a certain number of matching derivatives. The number depends on the choice of the degree (or the order). This is important because it cannot only capture the local features with different polynomials, but also guarantees the smoothness of the estimated curves to some degree. In this case, the estimated curves are globally constructed and their derivatives can be extracted directly from the curves.

2.2. Building Functional Data Object

After specifying the basis system, the next step is to define a functional data object by setting the coefficients $c_k, k = 1, \cdots, K$ to the basis system. The coefficients are not the outcome of

interest, but the linear combination of basis functions with coefficients are. If one function is defined, the coefficients are a vector of length K. If one functional data object contains multiple functions such as handwriting samples for one subject, the coefficients are a matrix of K by N. Since the handwriting experiment uses multivariate functions for one trial of one subject, the coefficient matrix will be generalized to $K \times N \times M$, where $M = 3$. It will give me results immediately after I have plugged in the raw data.

2.3. Smoothing Curves from Raw Data

The fitting curves can be very messy because the process simply interpolates these points with lines. One of the solutions is to build a functional parameter data object with proper roughness penalties, aiming to compromise between capturing important writing features and reduce computations and over-fitting.

The roughness penalty is to minimize the following mathematical expression:

$$F(\mathbf{c}) = \sum_j [y_j - x(t_j)]^2 + \lambda \int L^2 dt$$

, where the first part on the right side is the ordinary sum of squared errors of residuals (SSE) under the error model

$$y_j = x(t_j) + \epsilon_j = \mathbf{c}^{'}\phi(t) + \epsilon_j = \phi^{'}(t_j)\mathbf{c} + \epsilon_j$$

The true errors or residuals ϵ_j are statistically independent and are assumed to follow a Gaussian distribution with mean zero and constant variance. The second part is the measure of roughness, and L can be derivative functions or differential operators. The smoothing parameter λ specifies the emphasis on the L relative to the goodness of fit in the SSE. λ takes values in any non-negative real numbers. As λ approaches positive infinity, curves become less rough and converge to a straight line. On the contrary, as λ goes to zero, roughness penalties have less influence on the curvature, and the curves display more significant fluctuations. In the extreme case, the curves simply interpolate each data observations. In practice, the λ is changed on a logarithmic scale so that it is convenient to implement and interpret.

One of the quantatitive methods to determine λ is to calculate the generalized cross-validation (GCV) with respect to the λ. The criterion has a mathematical expression that

$$GCV(\lambda) = (\frac{n}{n - df(\lambda)})(\frac{SSE}{n - df(\lambda)})$$

The minimum of the GCV function exists and is unique once λ is given (controlling for other

factors), and the proper λ is chosen to minimize the GCV function. In particular, when subjects are writing the words, their hands are also having involuntary physiological hand tremor due to muscle contraction and relaxation (Marshall & Geoffery, 1956). The machine with high working frequency record such variations, leading to the increasing roughness of the curve. Therefore, it is practice to use significantly large λ such as 10^8 to reduce the effect of the random noise.

3. Theory of Principal Differential Analysis

3.1. Template-based Feature Correspondence

The Template-based Feature Correspondence (Wang, Wu, Xu, Shum, & Ji, 2002) provides theoretical support for the curve registration in Chapter 4. This writing style model assumes that subjects usually generate their handwriting samples based on their inherent styles, known as templates. If one subject is required to write the same word many times, the handwriting samples will display similar features. The model is described in the following mathematical expression. Suppose W_i is the word to be written (The index i refers to the i^{th} handwriting sample), S is the template that the subject follows and F_i is the writing process to generate the word W. Furthermore, each sample has its unique global style-independent parameters such as scale, location and slant. Therefore, I have

$$W_i = F_i(S) = F(S, N_i, A_i)$$

, where N_i is random noise, and A_i represents affine transform parameters. This process suggests that for the same word written by the same subject, the differences result from two parts. One is the random noise from the hand tremors and the machine errors that are inevitable. The other one is the affine transform, which include translation, expansion and contraction. Curve registration reverses the above process by identifying S for each subject with W_i. In particular, random noise can be modified by applying the roughness penalties to raw data, and the unique global style-independent parameters are synthesized by data truncation and normalization. Therefore, curve registration technique integrates affine transform to determine the final template S for each subject.

The concept of template can be generalized to coordinate, velocity and acceleration curves. They are the dynamic criteria in categorizing handwriting across subjects. Since the acceleration is associated with forces by muscle contraction, which determines the dynamic templates the most, Ramsay (2000) argued that the magnitude of the acceleration was considered as the fundamental criterion. It is the magnitude of the 3 dimensional vector defined as

$$\sqrt{D^2X + D^2Y + D^2Z}$$

Although coordinate curves are studied separately by decomposing the movement into three dimensions, three-dimensional curves are actually subjected to the external forces at the same time. Therefore, registering the magnitude of the acceleration vector is plausible to present the idea.

3.2. Principal Differential Model

The FDA objects allow us to extract derivative curves, and the template theory suggests that these derivatives serve as criteria in categorizing writing patterns of subjects. Based on Newton's Second Law ($F = ma$), Ramsay and Silverman (1997) started with a second-order linear differential model

$$D^2X(t) = -\beta_0 X(t) - \beta_1 D_1 X(t) + \alpha g(t)$$

which integrates three types of forces in one formula. The model is then generalized with time-varying coefficients. Thus, given repeated measurements of sample processes, the linear differential operator for each subject is defined as

$$LX_i(t) = \omega_{0i}(t)X_i(t) + \omega_{1i}(t)D_1X_i(t) + \omega_{2i}(t)D_2X_i(t) + \cdots D_mX_i(t)$$

, where m is the highest order of derivative, and ω_{ji} is the corresponding weight function of order j for the subject i. The model has the following generic equation:

$$LX_i(t) = f_i(t)$$

, where $f_i(t)$ on the right is the forcing function. The forcing functions normally contain two parts. One is external force $\mu_i(t)$, which represents the external force that cannot be explained by the main differential equation itself. The other is the error function $\epsilon_i(t)$. The above model with two forcing functions is to minimize the objective function with respect to coefficient functions ω:

$$\text{PDASSE}(\omega) = \sum_{i=1}^{N} \int [L_\omega X_i(t) - \alpha_i(t)\mu_i(t)]^2 dt$$

In the handwriting setting, the target functions are of dimension three. Therefore, Ramsay (2000) modified the linear differential equation

$$LX_{qi}(t) = D_mX_{qi}(t) + \sum_{k=0}^{m-1}\sum_{j=1}^{d} \omega_{ijk}(t)D_kX_{qj}(t)$$

, where q is the observation index of a multivariate functional data. Here, I only take X, Y coordinates into account because Z coordinate does not yield satisfactory registration result.

3.3. Idea of Subject Categorization

The coefficient functions ω based on a group of writing samples from the same subject should be sensitive to samples written by other subjects. The idea of differentiating handwriting is as follows: for each subject, I fit the above model for X and Y coordinates, and extract the residual functions, separately. These are the residual functions generated by their own weight functions. Then I apply data of one subject to the model of the other one and extract the residual functions again. These are the residual functions generated by alternate weight functions. The former residual functions are expected to display variation with white noise characteristics, indicating that the differential operator succeeds in capturing most handwriting features, and the handwriting samples match the subject. Meanwhile, the latter residual functions are expected to display variations with drastic patterns, indicating that the differential operator fails to capture the most handwriting features, and the handwriting samples do not match the subject (Ramsay & Silverman, 2002).

There are several methods for model evaluations. One visual method examines whether the model captures the variation well by co-plotting the mean residual function $\epsilon_i(t)$ and the average third derivative function $D^3X_i(t)$. If the magnitude of $\epsilon_i(t)$ is smaller than $D^3X_i(t)$, then the model fits well. Otherwise, residual functions may contain mixed components of higher-order derivatives, and further separation work needs finishing. A squared multiple correlation measure of fit R^2 defined by

$$\frac{\sum_{k=1}^{3}\sum_{i=1}^{10}\int_0^T (D^m X_{ik})^2 - f_{ik}^2(t)dt}{\sum_{k=1}^{3}\sum_{i=1}^{10}\int_0^T (D^m X_{ik})^2(t)dt}$$

can also demonstrate the goodness of fit for the model (Ramsay & Silverman, 2002). The R^2 is similar to the R^2 in linear regression, and takes values from $[0, 1]$. The closer the R^2 is to 1, the better the goodness of fit is.

4. Curve Registration

Although I can obtain the derivative curves from smoothed curves directly, the variation of curves within the same subject is too drastic to perform differential analysis. However, the template-based feature correspondence suggests that samples within the same subject do share certain characteristics. Ramsay and Silverman (1997) developed the curve registration method to integrate affine transforms and retrieve the final template S. Therefore, proper registration methods have been applied before I perform the principal differential analysis.

4.1. Amplitude and Phase Variation

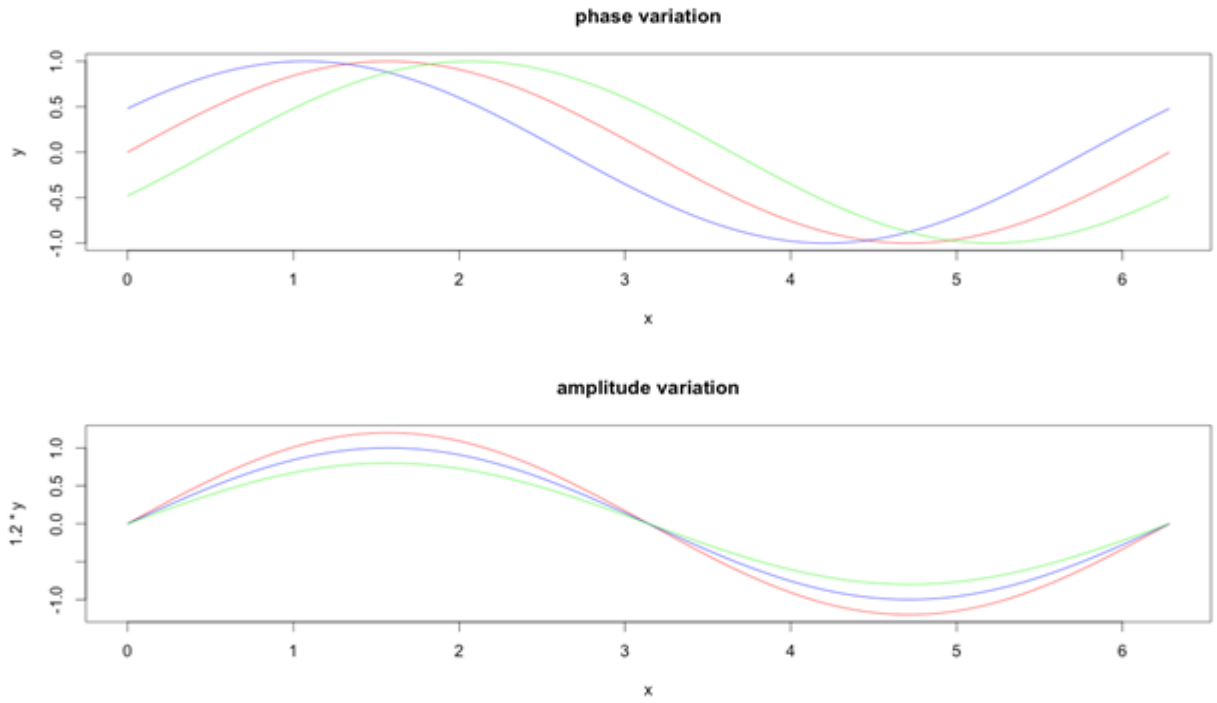

Figure 1: Curves witih only phase and amplitude variation

Phase variation and amplitude variation are two representations of differences from the affine transforms. The top panel in Figure 1 displays curves with mere phase variation, where the red one is the curve $y = \sin(x), 0 \leq x \leq 2\pi$. Three curves achieve their maxima and minima at different x's, but with the same values. Green and blue curves can be generated by right shifting red curve with 1 and -1 unit. The bottom panel in Figure 1 displays curves with mere amplitude variation. Although three curves have distinct maxima and minima, but they arrive at the values at the same x. The red and green curves can be generated by multiplying the blue curve by 1.2 and 0.8.

4.2. Landmark Registration (LM) and Continuous Registration (CR)

The simplest curve alignment procedure is called landmark registration. A landmark is a significant feature at a certain time location that is common to all curves. Landmarks may be the location of minima, maxima or zero points. The alignment is achieved by transforming t for each curve so that each of them arrives at their landmark locations at the same time t. Landmark registration method can only yield satisfactory results when raw curves display drastic and consistent changing patterns.

Landmark registration is usually a good start to remove phase variation, but a more sophisticated

method is needed to modify the existing curves if landmarks are not obvious to detect. James and Silverman (1997) illustrated the method of continuous registration that can perform registration automatically and reduce the magnitude of amplitude variation. The continuous registration method assumes that the dominant difference between pre-registered curves and post-registered curves is the amplitude variation, and their coordinate values will be almost proportional to each other across the whole period. If the registered curve is plotted against template one, it is ideal to see a straight line tending to pass through the origin. If this is true, each curve can be further registered towards the template curve by minimizing the smallest eigenvalue of the cross-product matrix.

$$\begin{bmatrix} \int \{x_0(t)\}^2 dt & \int x_0(t)x[h(t)]dt \\ \int x_0(t)x[h(t)]dt & \int \{x[h(t)]\}^2 \end{bmatrix}$$

If the curves are multivariate such as the handwriting example, then it is the sum of the smallest eigenvalue across dimensions that are minimized.

One method that evaluates the registration method is to compute the mean squared error (MSE) for amplitude and phase variation before and after the CR method. The phase MSE after landmark registration is also computed if the CR method is done based on the LM result. If the difference between two phase MSEs yields negative numbers, then the CR method shows poor alignment. If the LM method is applied to raw curves first, the ratio of phase MSE by total MSE, known as RSQR, also reflects how well the LM method has done to reduce phase variations. The smaller percentage it achieves; the better result it has after applying the CR method.

4.3. Time-warping Functions

The time-warping function is a monotonically increasing function that maps the template writing time to the actual writing time for each sample. In addition, the time-warping function should be smooth enough to calculate derivatives. If the curves are observed over a common interval $[0, T]$, then the time-warping functions must satisfy the constraints $h(0) = 0$ and $h(T) = T$. The registered coordinate functions are $x_i^*(t) = x_i[h_i^{-1}(t)]$, where the aligning functions $h^{-1}(t)$ and $h(t)$ are inverse functions.

Here, the time-warping functions are defined over the interval $[0, 2400]$ since all the samples are finished around 6 seconds and the machine frequency is set as 400 Hz. Take the small-break time between “Ann” and “Arbor” as an example. Each subject stops at different time points across

samples relative to clock time. In terms of stopping time in the template, all samples should arrive at their peaks at the same time. The time-warping function serves as a connection that ensures all coordinate curves display certain characteristic (such as achieving peaks and crossing zeros at the same time), so that the coordinate curves after transformation are comparable. Time-warping functions can also tell whether the subject is writing faster than normal by calculating the time lag $h(t) - t$. However, it is not the outcome of interest because the template S is achieved by minimizing the individual differences.

4.4. Cross-sectional Mean and Registered Mean

After obtaining registered curves from raw curves, the mean curve is extracted by taking the average of coordinates at each time. Such registered mean curves are the writing template S. Compared to cross-sectional mean curves, which only take the average of coordinates of the raw curves, registered mean curves avoid obscuring the sharp acceleration peaks and troughs due to the messy timing variation. The registered mean curves can reduce the unusual effect of certain sample. Therefore, it is better to use registered mean curves than cross-sectional mean curves to denote the template S. Figure 7 in Chapter 6 shows the mean word plot "Ann Arbor" with LM and CR methods.

5. Handwriting Data Collection and Data Pre-processing

5.1. Data Collection

The new data set is obtained from the handwriting experiment in the Biomechanic Research Lab, University of Michigan. With the assistance of Sasha Kapshai, I set up the new coordinate system, and five subjects (including myself) were required to write “Ann Arbor” for twenty times, respectively. Among twenty writing samples, each subject was required to write in a non-cursive way for the first ten times as opposed to last ten times in a cursive way. The data were recorded by the Optotrak Certus machine, and subjects were holding a specially made pen with six sensors connected to the computer. The machine recorded the handwriting based on movements of six sensors and convert them to the movement of pen tip. The frequency of the machine was set to 400 Hz and the time length was set to 7 seconds. The basic unit of measurement for distance was millimeter.

Among five subjects, all writing samples were finished around 6 seconds, and it took five subjects approximate 3.6 seconds to write “Arbor”. The computer produced one data file after

subjects finished one writing sample, and each data file contained 7 columns and 2800 rows. There should be 100 data files in total. The first three columns were the radian differences between the new coordinate system and its intrinsic coordinate system. The next three columns were the values of three-dimensional coordinates under the new coordinate system, and the last column was the pen error. Only “pen.x”, “pen.y” and “pen.z” were used to perform future analysis.

pen Rz	pen Ry	pen Rx	pen x	pen y	pen z	pen error
0.3342025	0.3141596	-0.0929014	-86.900222	729.42852	119.91264	0.1340895

Table 1: The column names and sample data of subject 0 and 1

5.2. Data Selection and Normalization

Writing samples by the subject 3 and 4 were discarded due to the large amounts of missing data. (Two subjects held the pen in the way that their hands blocked the sensors from being detected by the machine, leading to lots of missing data in the raw data set.) The cursive samples of all subjects were also discarded due to poor registration results.

Among “pen.x”, “pen.y”, and “pen.z”, I removed the minus sign of “pen.x” and switch “pen.y” and “pen.z” columns in the data file to plot the proper word image. Since all the non-cursive writing samples were finished around 6 seconds, each data file needed truncating to 2400 rows with the help of the Z coordinate. When people are writing, they have to exert force on the pen to leave the trace. In that case, Z coordinate will remain constant. When people finish writing one part of the word, such as “Ann” of “Ann Arbor”, they will take a small break by raising their hands. When they finish writing the whole word or phrases, the gesture of raising their hands will be much more significant. Therefore, the extra Z coordinate plays an important role in determining the starting point and ending point for each data file. Although my eyes cannot detect them, the machine amplifies the variation by displaying significant jumps in the Z dimension while the remaining points form a constant horizontal line. Normally, where the Z coordinates have the first jump is the starting point, and where the Z coordinates have last (and usually the greatest) jump is the ending point.

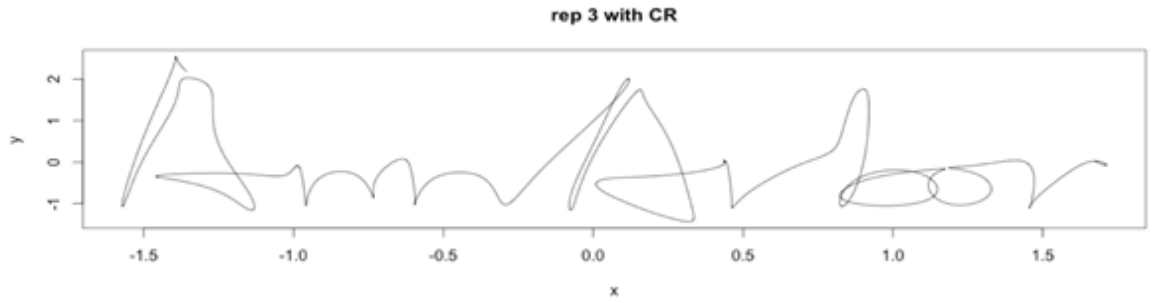

Figure 2: Distortion influence on individual sample due to consecutive missing data

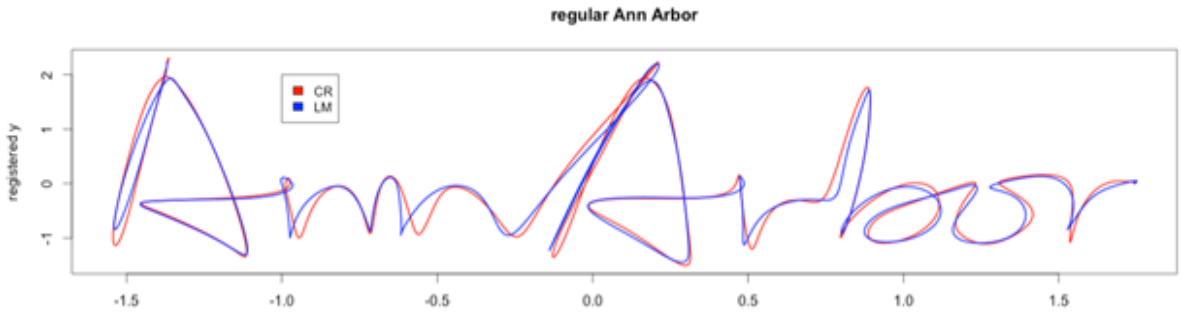

Figure 3: Mean "Ann Arbor" plot

For curve registration part, I randomly picked 1000 rows from the total 2400 rows. It compromised between capturing necessary writing features and avoiding over-fitting. For the principal differential analysis part, to simplify the model, I kept the word "Arbor" by randomly selecting 600 observations so that it corresponded to the fact that 60% time was spent in writing "Arbor". After truncating the data file to 1000 rows, normalization was performed to reduce the effect of magnitude. The mean and standard deviance for each column in all data file were calculated by the following formula:

$$X^{'} = (X - \bar{X})/\mathrm{sd}(X)$$

Missing data were universally set to 0 after scaling the data to minimize the difference between the true shape and actual shape. The magnitude of scaled data ranges from -1 to 3. Although missing data have an influence on each writing sample, the overall shape is relatively unchanged. For example, the first "A" of "Ann Arbor" in Figure 2 has flatter top than the second "A" due to the consecutive missing data in the raw data file. However, the average word plot in Figure 3 doesn't reflect this unusual local feature because taking the average minimizes the differences across samples.

6. Results and Discussions

6.1. Registering the Coordinate Curves

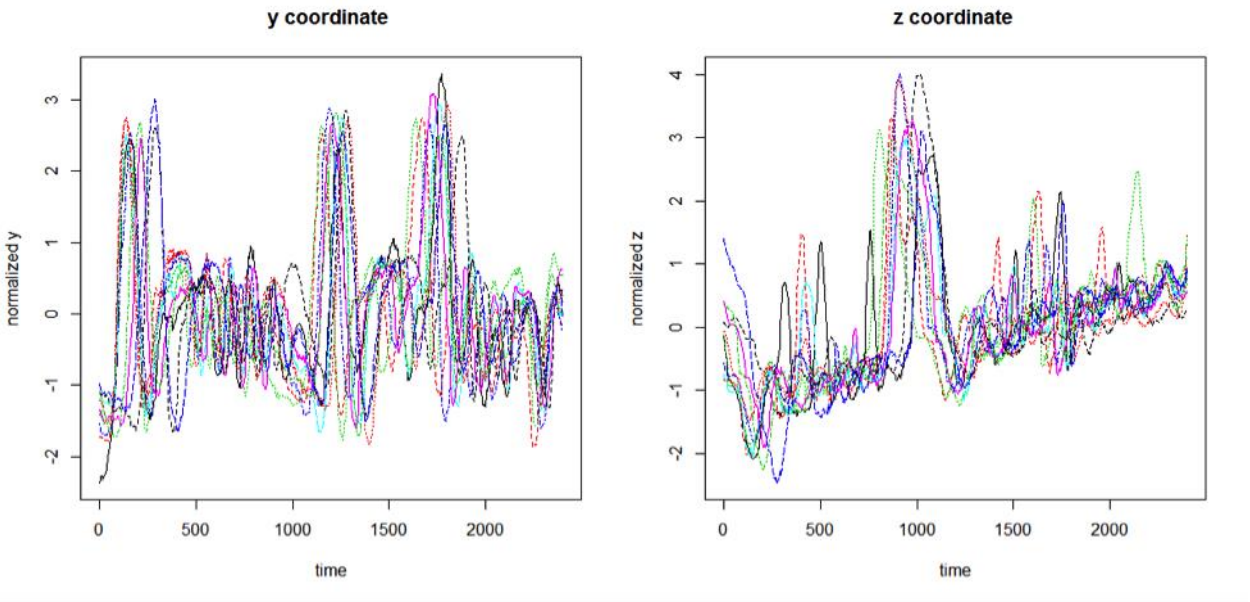

Figure 4: Y and Z coordinate matplot of subject 0

Figure 4 shows the matplot of smoothing Y and Z coordinates for subject 0 prior to registration. The Z coordinate plot on the right displays significant and consistent jumps around 2.5 second (or 1000 in the plot), which is the small break between “Ann” and “Arbor”. Those ten samples arrive at their peaks at different time points and different values, which demonstrates the effects of phase variation and amplitude variation.

I choose local maximas of Y coordinates as landmarks by hand because Y coordinates show more consistency than X and Z coordinates. I set one landmark when the curve reaches its local maxima. Figure 5 shows landmarks of the first two samples by subject 0. There are 15 landmarks for each sample of subject 0. The number of landmarks differ across subjects, but are generally greater than 10. However, there are a few points that are not located at the peaks. For example, the curves after the first peak as well as after four consecutive small peaks experience quite flat sections. They happen at around 1 second and 3 second, respectively. Therefore, I fit two landmarks at the boundaries of these sections to keep the flatness in the registration result.

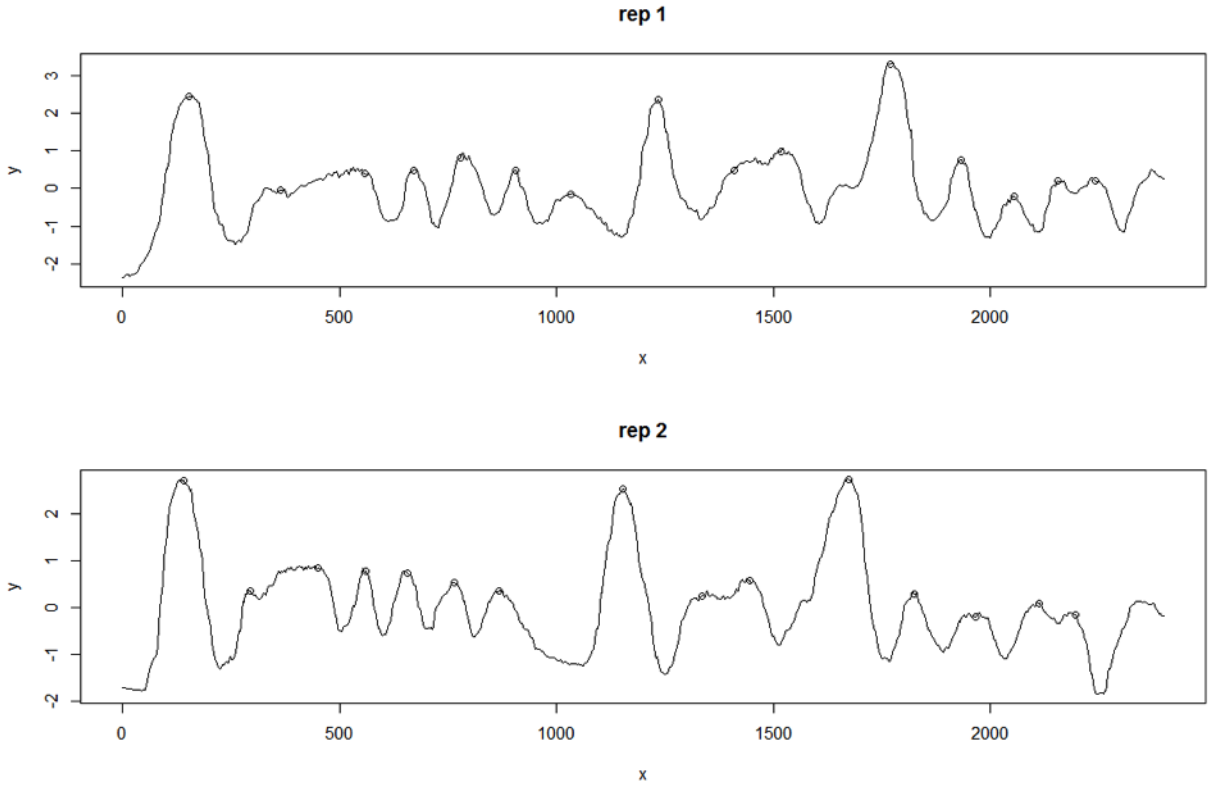

Figure 5: Landmark selection of the first two Y coordinate plots of subject 0

In order to use landmarks to represent the curves accurately, they should themselves display sharp but consistent curvatures. Fortunately, the handwriting curves fit the category well because when each subject is required to write “Ann Arbor”, their intrinsic styles force subjects to control their muscle to produce necessary variations so that people can identify the word correctly, which leads to significant but consistent curvatures.

I apply the LM method first, and then apply the CR method to LM registered curves. I will use subject 0 as an example to illustrate how the CR method improves the alignment of coordinate curves. The MSEs of X and Y dimensions after the LM registration are 10.4, 4.5 and 93.7, 561, with the first one for amplitude variation. The variation of X is quite satisfactory and only 30% variation is produced by phase. But the variation of Y is much bigger than expected, and 85.7% variation is produced by phase. The large phase variation for the Y coordinate implies that it is necessary to perform the CR method on Y dimension to further reduce the variation. The MSEs of X and Y coordinate curves after the CR registration are 5.45, 12.8 and 36.2, 59.1. Although the variations for X increase, the CR method reduces the variation of Y coordinates significantly. Therefore, in this case, the CR method does improve the alignment of LM registered curves. I

also perform the CR method directly to raw curves, but the registered curves display poor alignment, compared to the former method. This is mainly because handwriting curves are too complicated to handle automatically. The raw data contain both phase variation and amplitude variation, which violates the assumption for performing the CR method. Either manual splitting into sub-intervals or applying the LM method first should be done to substantially reduce phase variation as much as possible before performing the CR method.

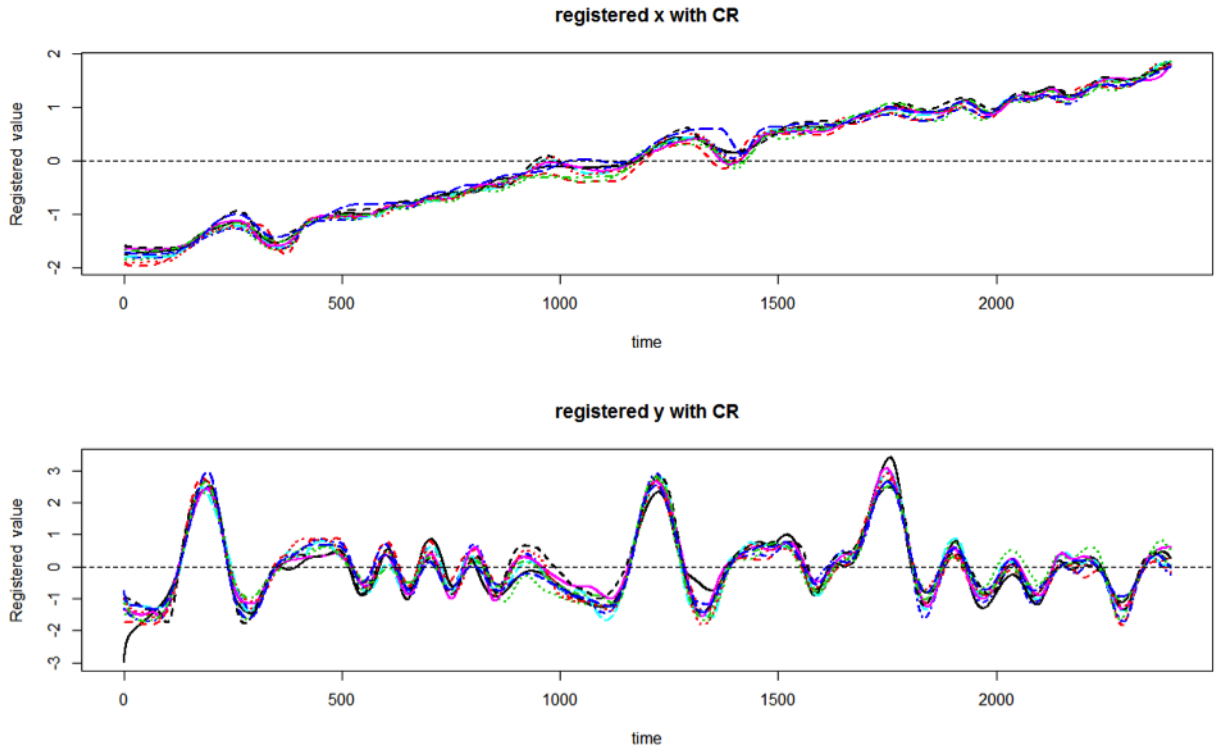

Figure 6: Registered X and Y coordinate plots with CR method of subject 0

Figure 6 shows the X and Y coordinate curves of subject 0 after the CR registration. X coordinate curves for all three subjects display monotonically increasing trends with fewer fluctuations, while Y coordinate curves display sharper fluctuations. This is because subjects are used to writing words horizontally. During the writing process, they keep their elbows and wrists as pivots. Subjects try to keep their forearms still to allow only hands to move from the left side to the right side because the fixation of arms also prevents the paper from moving around. Since the horizontal movement mainly involves continuous rotation, small amount of force is needed to finish the movement. On the contrary, for Y coordinate, subjects need to rotate their wrists to let the pen tip interact with the paper to write from the upper part to the lower part. Moreover, when subjects finish one up-to-down cycle, they have to jump to the upper place on the right to start a new one, making the movement have the zigzag shape. Therefore, extra forces are exerted

to finish the movement, which accounts for the sharp fluctuation of Y dimension.

6.2. Obtaining the Word Plot

The three word plots in Figure 7 are generated using templates of subject 0, 1, and 2. Results produced by two methods are marked with different colors. (Note that CR results are based on LM registered curves.) Differences across subjects are quite easy to tell, and for each subject, two methods produce almost the same word plot, suggesting that either method works fine for generating the shape template.

Figure 7: Template word plots for three subjects with LM and CR methods

6.3. Registering the Derivative Curves

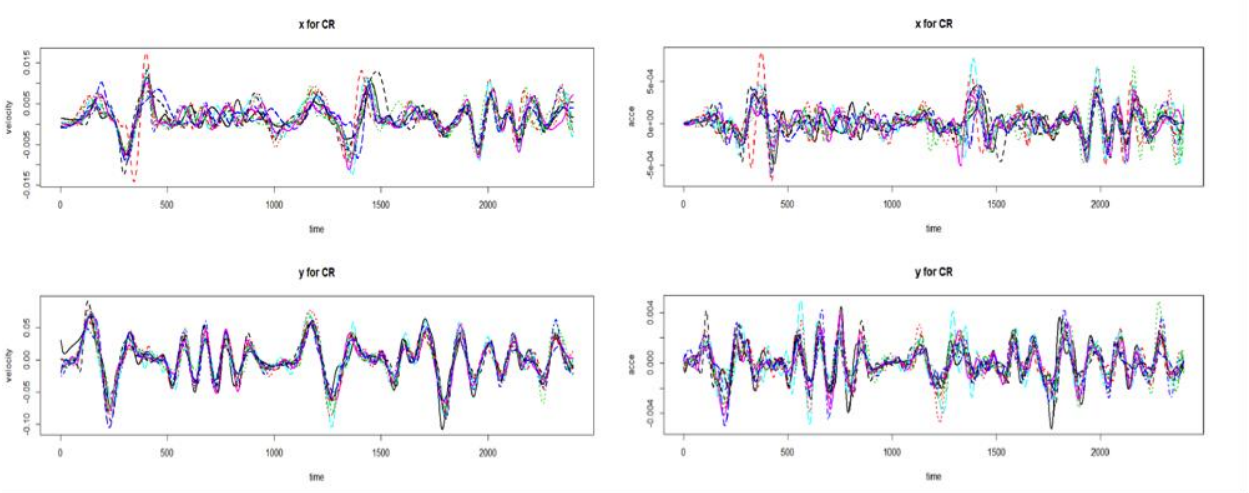

Figure 8: Registered velocity and acceleration plots for subject 0

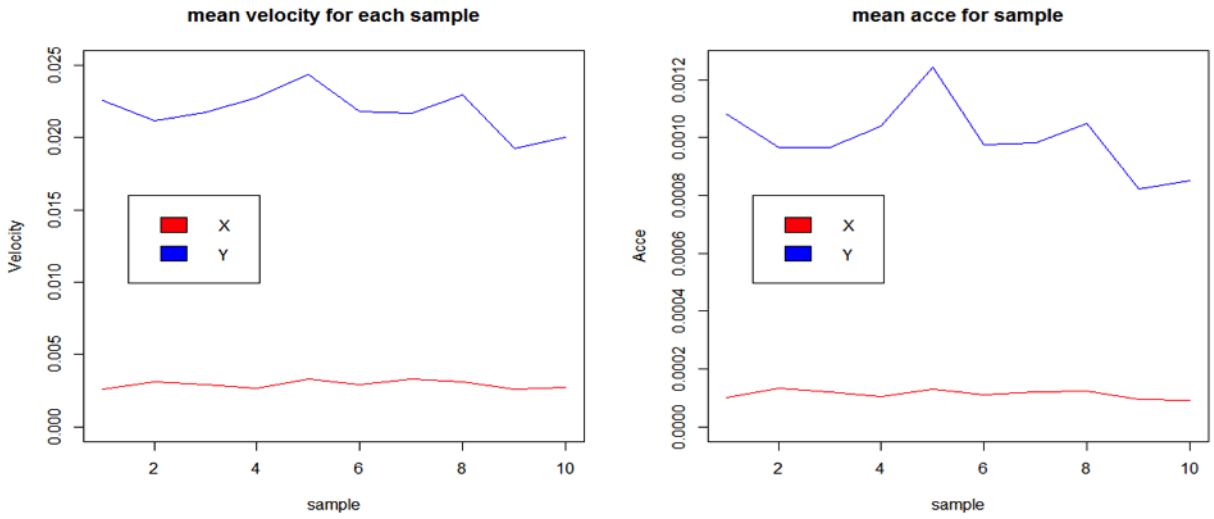

Figure 9: Mean velocity and accelerations of ten samples of subject 0

However, it is not adequate to analyze the template word plot because they only tell the shape difference. The "fda" package enables me to extract the aligned velocity and acceleration curves directly. The four plots in Figure 8 are the two-dimensional registered velocity and acceleration curves of subject 0 using the time-warping function from registered coordinate curves. It is shown that Y dimension yields more desirable results while X dimension plots have messy ones, possibly because the CR method contributes to alignment of Y coordinate curves, and I select the landmarks of the Y coordinate curves. The scale of curves is also different between two dimensions. Figure 9 shows the mean velocity and acceleration curves against each sample. The magnitude of Y dimension is 8 times as large as X dimension for velocity and acceleration, suggesting that large force is applied on Y dimension.

Although the order of acceleration is 10^{-4} from the plot, the real value can be as large as 10^2. It is very counter-intuitive that handwriting movement requires very large amount of force. Two reasons explain the enormous distinctions. First, scaling procedure is applied to raw data to remove magnitude effect on registration, making the range of new data less than 1/10 of raw data. Second, the time range is not represented in second, but second multiplied by 400 because of the working frequency of the machine. The stretching effect on time also reduces the magnitude of acceleration when the derivative is calculated.

It is also worth mentioning that the peaks of velocity and acceleration curves correspond to different locations in word plots. When velocity arrives at local peaks, it is usually located in points where the nearby strokes display small curvature, such as the straight-line portion of both letter “A” in “Ann Arbor”. Subjects simply write along a straight line without external interference, thus leading to increasing velocity at first; if there is a turn in the stroke, their writing velocity decreases to change the direction later. When acceleration arrives at local peaks, it is usually located in the points where the nearby strokes show sharp curvature such as the connection between ”b” and “o” because external force has to be applied in order to have abrupt direction changes. The pattern also explains why the number of peaks for velocity and acceleration are the same, but they appear alternately.

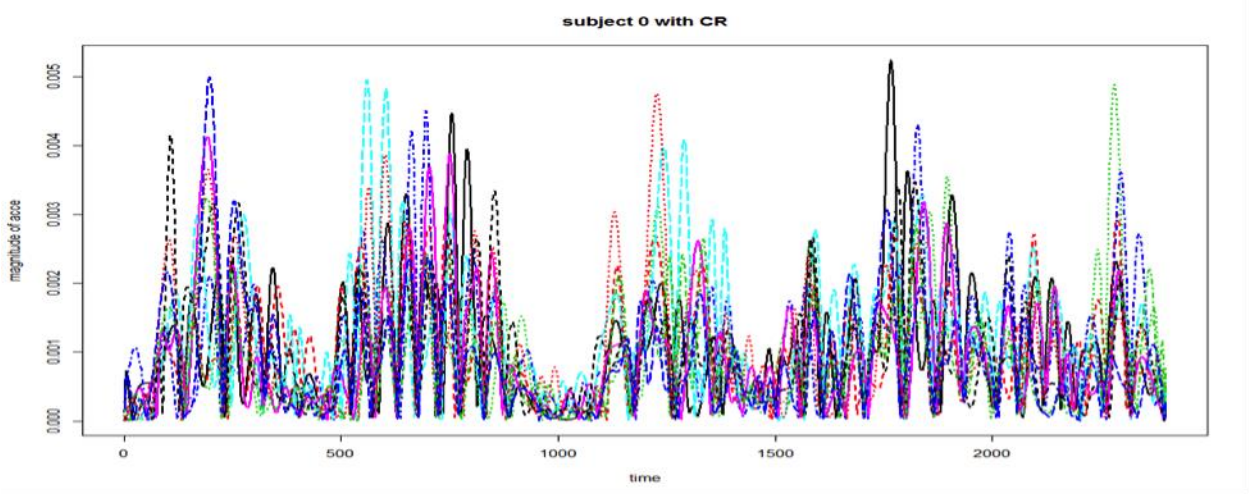

Figure 10: Registered acceleration plots for subject 0

If I look at the magnitude of acceleration $\sqrt{(D^2X^2 + D^2Y^2)}$, there will be more fluctuations due to the superimposition of X and Y dimensions. I exclude the Z coordinate because they yield terrible registration results. The plot itself is too complicated to interpret. Nevertheless,

throughout the total 6 seconds, it is clear that there are more than 30 jumps in the plot. It demonstrates handwriting is a movement of high frequency, which agrees with the earlier argument that handwriting movement requires large amount of force. In other words, handwriting is, to some extent, compromising between recognition and smoothness in a dynamic sense. Since scripts are limited to space, they usually contain certain breaks and turns to distinguish themselves from others. Certain amounts of forces have to be exerted on pens to yield the desirable result; otherwise, it will be very difficult for people to recognize the scripts.

Notice that acceleration values are approaching zero around 400 (1s), 1000 (2.5s), and 1400 (3.5s). The second one lasts the longest because it is when subject 0 takes a small break between “Ann” and “Arbor”. The only movement subject 0 makes is to lift his hand up, therefore, nothing significant changes in X and Y dimensions make the magnitude of acceleration value around zero. The other two reflect the flatness in the Y coordinate curves where I select two landmarks at each boundary.

6.4. Extracting Forcing Functions

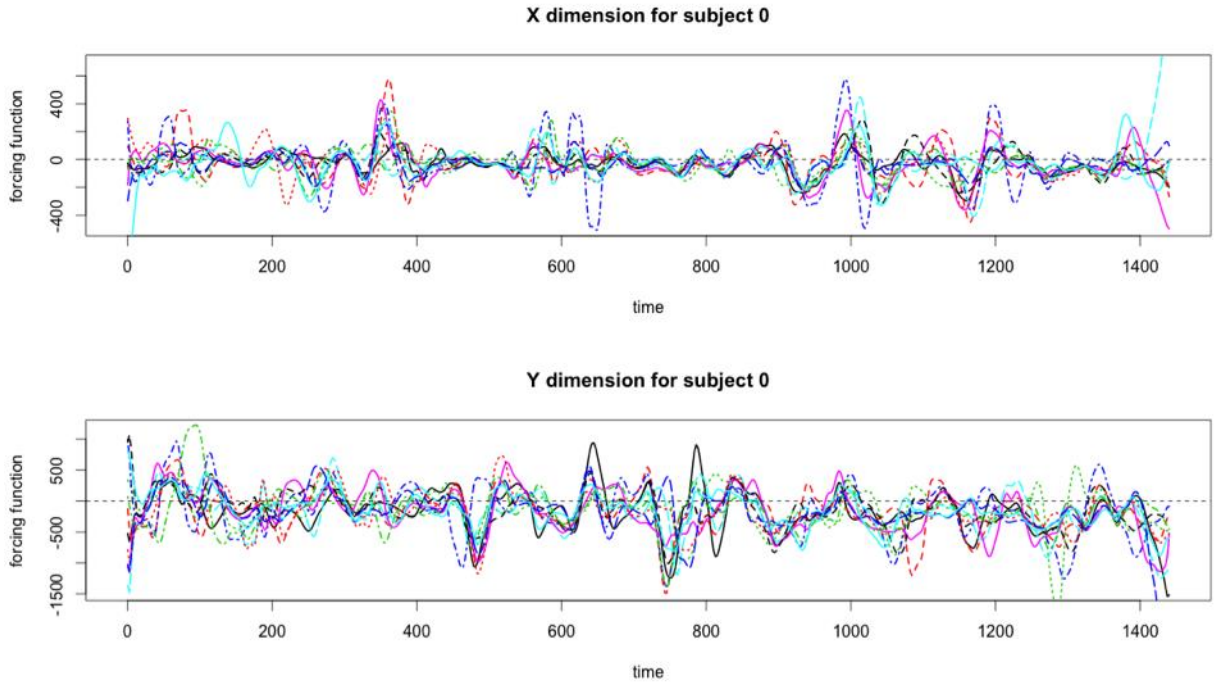

Figure 11: Forcing functions of X, Y dimensions for subject 0

Figure 11 shows forcing functions of X and Y dimensions for subject 0. To simplify the model, I only use "Arbor" section, and the time ranges from 2.4 to 6 seconds (0 to 1440 in the plot). In the linear differential model, forcing functions represent the external forces that cannot be explained

by the main part of the differential equation. For X dimension, forcing functions have significant fluctuations around 360 (3.3s), 1000 (4.9s), and 1200 (5.4s). They correspond to the connection between “A” and “n”, “b” and “o”, and “o” and “r”. These locations happen to have large curvatures, making external force available to capture the writing pattern. Similarly, for Y dimension, forcing functions have significant fluctuations around 480 (3.6s), 640-800 (4-4.4s), and 1000-1200 (4.9-5.4s). They correspond to the lower part of first “r”, the connection between “r” and “b”, and “b” and “r”. Figure 12 displays the third derivative and residual function of the first two subjects. The linear differential model up to the third derivative successfully captures the majority of variation, as the magnitude of residual function is much smaller than the third derivative. Therefore, I have successfully obtained the coefficient function for subject 0 and subject 1.

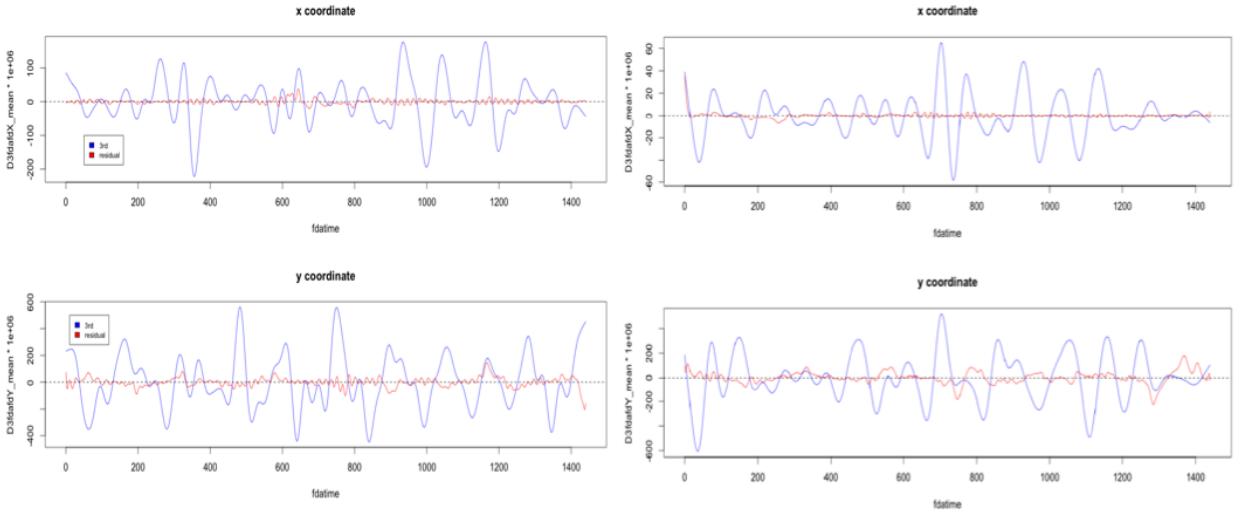

Figure 12: Third derivatives and residual functions of X, Y dimensions

6.5. Performing Principal Differential Analysis

After I apply the first two subjects' data to their models, residual functions are extracted for X and Y dimensions. The upper-left and lower-right plots in Figure 13 and 14 are situations where the models are applied to their own handwriting samples, while the other two are situations where the models are applied to the alternate handwriting samples. The upper-left and lower-right plots show small and messy residual functions. The remaining two plots show obvious and large patterns, meaning that the coefficient functions do not match the identity of input data. Applying the data of subject 1 to the model of subject 0 does not yield satisfactory result of X dimension, possibly because the CR method reduces the alignment of registered curves, obscuring the difference between two plots in the first row. Therefore, this linear differential

model with forcing functions captures the majority of writing features, and successfully categorizes the samples of two subjects.

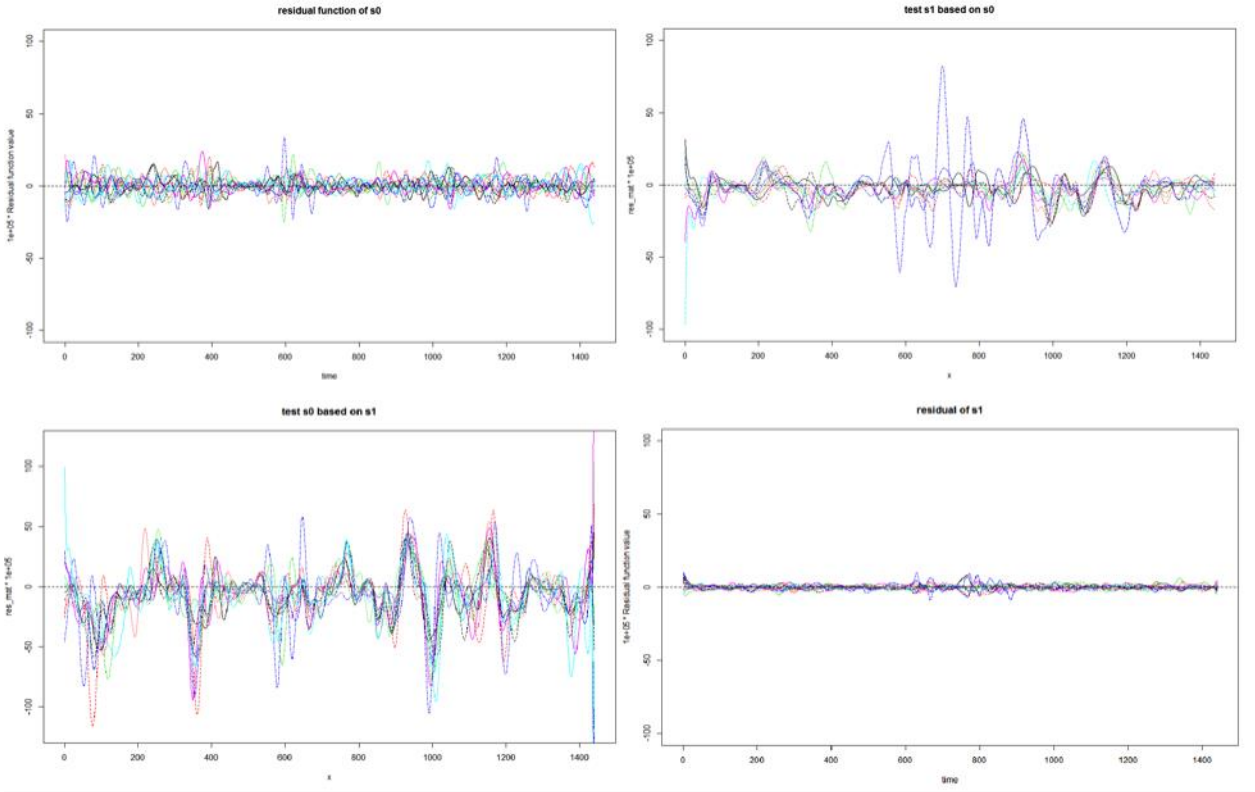

Figure 13: Residual plots of X dimensions between subject 0 and 1

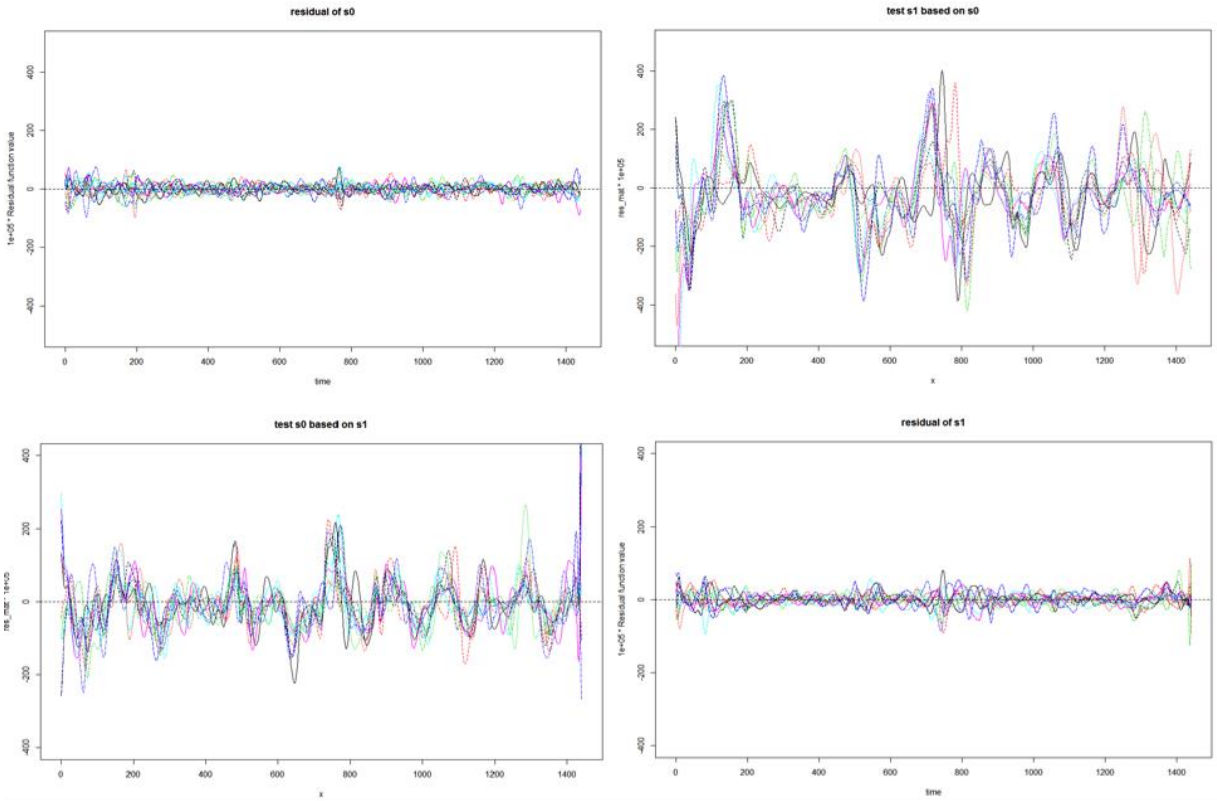

Figure 14: Residual plots of Y dimensions between subject 0 and 1

7. Challenges with Analysis

The first drawback of this analysis is the smoothness of raw curves. Since I cannot attach sensors to the ordinary pen, subjects have to replace it with the long stick. Unaccustomed to the writing object and unable to see the actual writing outcome, subjects may produce slightly different writing samples than usual, increasing the roughness of raw data. Such difference may become more obvious when I perform principal differential analysis.

Secondly, the normalization method in the data pre-processing part is simply to let data points minus their column means, and then divided by their standard deviations. Since the patterns of cursive samples after the normalization are too insignificant for me to pick out the landmarks by hand, all the cursive samples are discarded. If there is an advanced technique to normalize the raw data and amplify the patterns, I can apply the previous analysis to cursive handwriting and categorize cursive handwriting samples or to look at the difference between non-cursive and cursive handwriting samples within the same subject.

Thirdly, since subjects may inevitably block the sensors from being detected by the machine, missing data exist in the raw data. The way I handle the missing data is simply to assign them to zero after normalization. If there exists a consecutive part of missing data, simply regarding them as zero will distort the individual word plot, which may affect the registration process and formation of template. However, one of the refinements is to fit a local linear regression or kernel regression to fill in the gap so that the shape after fitting FDA object is more authentic.

Finally, manually selected landmarks may contain artificial errors in alignment, especially when the local peaks have small curvatures. It is then very difficult for me to decide the number of landmarks, let alone picking them out correctly. In addition, since I apply landmarks from Y dimension to X dimension, the landmarks show poor fit with X coordinate curves, especially on the boundaries. It will be very effective that certain technique is developed to evaluate the location of landmarks, which can adjust their locations automatically to perform an optimal fit.

8. Bibliography

Elarian, Y., Abdel-Aal, R., Ahmad, I., Parvez, M., \& Zidouri, A. (2014). Handwriting synthesis: Classifications and techniques. *IJDAR International Journal on Document Analysis and Recognition (IJDAR)*,17(4), 455-469.

Marshall, J., \& Geoffery, W. (1956). Physiological Tremor. *Journal of Neurology, Neurosurgery, and Psychiatry*, 19(4), 260-267. doi:10.1136/jnnp.19.4.260

Ramsay, J. (2000). Functional Components of Variation in Handwriting. *Journal of the American Statistical Association*, 95(449), 9-15. doi:10.1080/01621459.2000.10473894

Ramsay, J. O., & Silverman, B. W. (1997). *Functional data analysis*. New York: Springer.

Ramsay, J., \& Silverman, B. (2002). *The Dynamics of Handwriting Printed Characters. In Applied functional data analysis methods and case studies*. New York, NY: Springer.

Ramsay, J. (2009). *Functional data analysis with R and MATLAB*. New York, NY: Springer-Verlag New York.

Wang, J., Wu, C., Xu, Y., Shum, H., \& Ji, L. (2002). Learning-based cursive handwriting synthesis. *Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition*, 157-162. doi:10.1109/IWFHR.2002.1030902

9. Acknowledgements

I would like to thank Dr. Edward Rothman for giving me the opportunity to explore this handwriting project and providing me with valuable suggestions when I reach an impasse. I would like to thank Dr. James Ashton-Miller and his research assistant Aliaksandra Sasha Kapshai for allowing me to use the Optotrak Certus machine and helping me set up the experiment. Finally, I would like to thank four volunteers for participating in the experiment and record their handwriting.

10. Code Index

```
#sub0.R library(fda) #import dataset
#setwd("/Users/mtianwen/Downloads")
#s0r1<-read.csv("/Tianwen himself/6_12_15_2015_06_12_094112_003_6d.csv", header=T)
#s0r2<-read.csv("/Tianwen himself/6_12_15_2015_06_12_094112_004_6d.csv", header=T)
#s0r3<-read.csv("/Tianwen himself/6_12_15_2015_06_12_094112_005_6d.csv", header=T)
#s0r4<-read.csv("/Tianwen himself/6_12_15_2015_06_12_094112_006_6d.csv", header=T)
#s0r5<-read.csv("/Tianwen himself/6_12_15_2015_06_12_094112_007_6d.csv", header=T)
#s0r6<-read.csv("/Tianwen himself/6_12_15_2015_06_12_094112_008_6d.csv", header=T)
#s0r7<-read.csv("/Tianwen himself/6_12_15_2015_06_12_094112_009_6d.csv", header=T)
#s0r8<-read.csv("/Tianwen himself/6_12_15_2015_06_12_094112_010_6d.csv", header=T)
#s0r9<-read.csv("/Tianwen himself/6_12_15_2015_06_12_094112_011_6d.csv", header=T)
#s0r10<-read.csv("/Tianwen himself/6_12_15_2015_06_12_094112_012_6d.csv", header=T)
#s0r11<-read.csv("/Tianwen himself/6_12_15_2015_06_12_094112_013_6d.csv", header=T)

#x0<-data.frame(s0r1$pen.x, s0r2$pen.x, s0r3$pen.x, s0r4$pen.x, s0r5$pen.x, s0r6$pen.x,
s0r7$pen.x, s0r8$pen.x, s0r9$pen.x, s0r10$pen.x, s0r11$pen.x, s0r12$pen.x, s0r13$pen.x,
s0r14$pen.x, s0r15$pen.x, s0r16$pen.x, s0r17$pen.x, s0r18$pen.x, s0r19$pen.x, s0r20$pen.x,
s0r21$pen.x, s0r22$pen.x)
#x0<-(-x0)
#colnames(x0)<-c("rep1", "rep2", "rep3", "rep4", "rep5", "rep6", "rep7", "rep8", "rep9", "rep10",
#"rep11", "rep12", "rep13", "rep14", "rep15", "rep16", "rep17", "rep18", "rep19",
#                    "rep20", "rep21", "rep22")
#y0<-data.frame(s0r1$pen.z, s0r2$pen.z, s0r3$pen.z, s0r4$pen.z, s0r5$pen.z, s0r6$pen.z,
s0r7$pen.z, s0r8$pen.z, s0r9$pen.z, s0r10$pen.z, s0r11$pen.z, s0r12$pen.z,
s0r13$pen.z, s0r14$pen.z, s0r15$pen.z, s0r16$pen.z, s0r17$pen.z, s0r18$pen.z,
s0r19$pen.z, s0r20$pen.z, s0r21$pen.z, s0r22$pen.z)
#colnames(y0)<-colnames(x0)
#z0<-data.frame(s0r1$pen.y, s0r2$pen.y, s0r3$pen.y, s0r4$pen.y, s0r5$pen.y, s0r6$pen.y,
s0r7$pen.y, s0r8$pen.y, s0r9$pen.y, s0r10$pen.y, s0r11$pen.y, s0r12$pen.y,
s0r13$pen.y, s0r14$pen.y, s0r15$pen.y, s0r16$pen.y, s0r17$pen.y, s0r18$pen.y,
s0r19$pen.y, s0r20$pen.y, s0r21$pen.y, s0r22$pen.y)
#colnames(z0)<-colnames(z0)
#for(i in 1:11) {
#   plot(x0[,i], y0[,i], type="l", main=paste("rep ", i, sep=""))
#}
```

```
#s0_eng_start<-c(30, 270, 330, 370, 500, 70, 300, 300, 380, 310, 200,   #3rd 330
#                        300, 250, 310, 260, 360, 200, 270, 210, 500, 370, 440)
#s0_eng_end<-c(1900, 2350, 2320, 2350, 2340, 1880, 2000, 2150, 2180, 1950, 1850, #3rd 2320
#                      1600, 1620, 1800, 1560, 1650, 1580, 1640, 1720, 2030, 1820, 1960)
#par(mfrow=c(2,1))
#for(i in 1:11) {
#   plot(x0[s0_eng_start[i]:s0_eng_end[i],i], y0[s0_eng_start[i]:s0_eng_end[i],i], type="l",
main=paste("rep ", i, sep=""))
#   plot(rep(s0_eng_start[i]: s0_eng_end[i]), z0[s0_eng_start[i]:s0_eng_end[i],i], type="l",
main=paste("z of rep ", i, sep=""))
#}
Temp<-NULL; k<-1000
#for(i in 1:11) {
#   index<-sort(sample(s0_eng_start[i]:s0_eng_end[i], k, replace=F))
#   Temp<-cbind(Temp, x0[index,i], y0[index,i], z0[index,i])
#}
#colnames(Temp)<-rep(colnames(x0)[1:11], each=3)
#Temp<-scale(Temp, center = T, scale=T)
#Temp[is.na(Temp)]<-0
#write.csv(Temp, "/Users/mtianwen/Downloads/s0_eng_1.csv", row.names=F)
#import dataset
s0_eng_1<-read.csv("s0_eng_1.csv", header = T)
s0_eng_1<-s0_eng_1[,-c(7:9)] #3rd sample writes something wrong... ignore this part data
colnames(s0_eng_1)<-rep(c("rep1", "rep2", "rep3", "rep4", "rep5",
                                  "rep6", "rep7", "rep8", "rep9", "rep10"), each=3)
s0_eng_1<-as.matrix(s0_eng_1)

s0_eng_basis_1<-create.bspline.basis(rangeval=c(0,k), nbasis=floor(k/3), norder=6,
#dropind=NULL, quadvals=NULL,
#values=NULL, basisvalue=NULL, names="subject0_english bspline")
fdatime<-seq(0, 2400, length.out = 1000)
par(mfrow=c(3,1))
matplot(fdatime, s0_eng_1[,seq(1,30,3)], xlab="time", type="l",
main="x coordinate ", ylab="normalized x")
matplot(fdatime, s0_eng_1[,seq(2,30,3)], xlab="time", type="l",
main="y coordinate ", ylab="normalized y")
matplot(fdatime, s0_eng_1[,seq(3,30,3)], xlab="time", type="l",
```

```
main="z coordinate ", ylab="normalized z")
s0_eng_1_x<-Data2fd(argvals=fdatime, y=s0_eng_1[,seq(1,30,3)], basisobj=s0_eng_basis_1)
s0_eng_1_x_eval<-eval.fd(fdatime, s0_eng_1_x)
s0_eng_1_y<-Data2fd(argvals=fdatime, y=s0_eng_1[,seq(2,30,3)], basisobj=s0_eng_basis_1)
s0_eng_1_y_eval<-eval.fd(fdatime, s0_eng_1_y)
s0_eng_1_z<-Data2fd(argvals=fdatime, y=s0_eng_1[,seq(3,30,3)], basisobj=s0_eng_basis_1)
plot(s0_eng_1_x, main="x coordinate against time for first 10 rep")
plot(s0_eng_1_y, main="y coordinate against time for first 10 rep")
plot(s0_eng_1_z, main="z coordinate against time for first 10 rep")
par(mfrow=c(2,1))
for(i in 1:10) {
  plot(s0_eng_1_x_eval[,i], s0_eng_1_y_eval[,i], type="l", xlab="x", ylab="y",
  main=paste("Ann Arbor of rep ", i, sep=""))
  #plot(s0_eng_1_y[i], main=paste("z of rep ", i, sep=""))
}
################################################################################
#########
########      perform landmark registration for the first ten sample trials       ########
i<-2
x<-fdatime; y<-s0_eng_1_y_eval[,i]
plot(x, y, type="l", main=paste("rep ", i, sep=""))
identify(x, y, labels=rep(1:k))
points(x[ximarks[i,]], y[ximarks[i,]], pty=1)
mark_data<-c(65, 152, 233, 280, 325, 378, 431, 515, 588, 633, 737, 805, 856, 898, 934,
              60, 124, 188, 234, 274, 319, 362, 481, 557, 603, 697, 760, 819, 880, 915,
              #61, 130, 186, 342, 389, 440, 480, 571, 641, 673, 754, 811, 870, 918, 938,
              64, 132, 174, 221, 255, 286, 336, 482, 549, 585, 682, 745, 804, 852, 908,
              70, 141, 188, 234, 281, 326, 375, 497, 567, 614, 714, 775, 828, 883, 923,
              59, 151, 184, 239, 286, 327, 388, 525, 597, 633, 734, 805, 857, 904, 933,
              92, 157, 200, 248, 291, 336, 387, 505, 588, 628, 721, 791, 853, 898, 930,
             123, 197, 243, 295, 326, 365, 420, 534, 617, 674, 783, 834, 881, 920, 943,
              59, 154, 181, 233, 280, 322, 378, 539, 628, 662, 752, 806, 854, 901, 929,
              88, 168, 189, 233, 269, 314, 353, 511, 588, 634, 736, 800, 863, 898, 920,
             121, 225, 257, 306, 343, 385, 432, 526, 613, 656, 750, 818, 873, 919, 940)
#mark_data<-mark_data/200
ximarks<-matrix(mark_data, 10, 15, byrow=T)
PGSctrmean=colMeans(ximarks)
```

```
wbasisLM<-create.bspline.basis(c(0,k), 18, 3, c(0, PGSctrmean, k))
WfdLM<-fd(matrix(0,18,1), wbasisLM)
WfdParLM<-fdPar(WfdLM, 1, 1e-5)
par(mfrow=c(2,1))
regListLM_z<-landmarkreg(fdobj=s0_eng_1_z, ximarks=ximarks, x0marks=PGSctrmean,
WfdPar=WfdParLM, monwrd = T)
plot(regListLM_z$regfd, main="registered z with LM", lwd=2)
plot(regListLM_z$warpfd, main="warp function for z", lwd=2)
regListLM_x<-landmarkreg(fdobj=s0_eng_1_x, ximarks=ximarks, x0marks=PGSctrmean,
WfdPar=WfdParLM, monwrd = T)
plot(regListLM_x$regfd, main="registered x with LM", lwd=2)
plot(regListLM_x$warpfd, main="warp function for x", lwd=2)
regListLM_y<-landmarkreg(fdobj=s0_eng_1_y, ximarks=ximarks, x0marks=PGSctrmean,
WfdPar=WfdParLM, monwrd = T)
plot(regListLM_y$regfd, main="registered y with LM", lwd=2)
plot(regListLM_y$warpfd, main="warp function for y", lwd=2)
for(i in 1:10) {
  #plot(regListLM_x$regfd[i], main=paste("x of rep ", i, sep=""))
  plot(regListLM_y$regfd[i], main=paste("y of rep ", i, sep=""))
  #plot(regListLM_z$regfd[i], main=paste("z of rep ", i, sep=""))
}
#warp deformation plots
par(mfrow=c(1,1))
matplot(rep(1:k), (eval.fd(rep(1:k), regListLM_y$warpfd)-matrix(rep(1:k), k, 10, byrow=F)),
ylab="h_i(t)-t", type="l", lty=1:2, col=1:9, lwd=2, main="coordinate warp deformation")

AmPhasList_1_x<-AmpPhaseDecomp(s0_eng_1_x, regListLM_x$regfd, regListLM_x$warpfd)
#10.4, 4.5, 0.302, 0.996;
AmPhasList_1_y<-AmpPhaseDecomp(s0_eng_1_y, regListLM_y$regfd, regListLM_y$warpfd)
#93.7, 561, 0.857, 0.995;
AmPhasList_1_z<-AmpPhaseDecomp(s0_eng_1_z, regListLM_z$regfd, regListLM_z$warpfd)
#107, 301, 0.737, 1.01;
reg_x_LM_eval<-eval.fd(rep(1:k), regListLM_x$regfd)
reg_y_LM_eval<-eval.fd(rep(1:k), regListLM_y$regfd)
#write.csv(reg_y_LM_eval, file="/Users/mtianwen/Downloads/s0_reg_y_LM_eval.csv",
row.names=F)
#plot each "Ann Arbor"
```

```
par(mfrow=c(2,1))
for(i in 1:10) {
  plot(reg_x_LM_eval[,i], reg_y_LM_eval[,i], type="l", xlab="x", ylab="y",
       main=paste("rep ", i, " with LM", sep=""))
}
x_mean_LM<-rowMeans(reg_x_LM_eval)
plot(rep(1:k), x_mean_LM, xlab="time", ylab="registered x", main="mean x with LM versus time",
type="l", lwd=2)
y_mean_LM<-rowMeans(reg_y_LM_eval)
plot(rep(1:k), y_mean_LM, xlab="time", ylab="registered y", main="mean y with LM versus time",
type="l", lwd=2)
plot(x_mean_LM, y_mean_LM, type="l", xlab="registered x", ylab="registered y",
main="Ann Arbor with LM registration", lwd=2)
plot.new()
#remove trend factor and retrieve residuals
s0_eng_1_x_residuals<-NULL
for(i in 1:10) {
  model<-lm(reg_x_LM_eval[,i] ~ rep(1:k))
  s0_eng_1_x_residuals<-cbind(s0_eng_1_x_residuals, model$residuals)
}
matplot(s0_eng_1_x_residuals, type="l", lwd=2, xlab="time", ylab="residuals of x",
main="residuals of registered x with LM")
abline(h=0, lty=2, lwd=2)
plot(regListLM_x$regfd, main="registered x with LM", lwd=2)
#write.csv(s0_eng_1_x_residuals, file="/s0_eng_1_x_residuals.csv", row.names=F)
#try continuous registration with Function register.fd based on the landmarker result.
wbasisCR<-create.bspline.basis(c(0,k), 20, 4)
Wfd0CR<-fd(matrix(0,20,10), wbasisCR)
WfdParCR<-fdPar(Wfd0CR, 2, 0.1)
regListCR_x<-register.fd(mean(regListLM_x$regfd), regListLM_x$regfd, WfdParCR)
plot(regListCR_x$regfd, main="registered x with CR", lwd=2)
regListCR_y<-register.fd(mean(regListLM_y$regfd), regListLM_y$regfd, WfdParCR)
plot(regListCR_y$regfd, main="registered y with CR", lwd=2)
regListCR_z<-register.fd(mean(regListLM_z$regfd), regListLM_z$regfd, WfdParCR)
plot(regListCR_z$regfd, main="registered z with CR", lwd=2)
```

```
AmPhasList_1_x_cf<-AmpPhaseDecomp(regListLM_x$regfd, regListCR_x$regfd,
regListCR_x$warpfd)
#5.45, 12.8, 0.702, 1;
AmPhasList_1_y_cf<-AmpPhaseDecomp(regListLM_y$regfd, regListCR_y$regfd,
regListCR_y$warpfd)
#36.2, 59.1, 0.62, 1; significant progress
AmPhasList_1_z_cf<-AmpPhaseDecomp(regListLM_z$regfd, regListCR_z$regfd,
regListCR_z$warpfd)
#72.3, 56.6, 0.439, 1.01;
#warp deformation plots
par(mfrow=c(1,1))
matplot(rep(1:k), (eval.fd(rep(1:k), regListCR_x$warpfd)-matrix(rep(1:k), k, 10, byrow=F)),
ylab="h_i(t)-t", type="l", lty=1:2, col=1:9, lwd=2, main="x coordinate warp deformation")
matplot(rep(1:k), (eval.fd(rep(1:k), regListCR_y$warpfd)-matrix(rep(1:k), k, 10, byrow=F)),
ylab="h_i(t)-t", type="l", lty=1:2, col=1:9, lwd=2, main="y coordinate warp deformation")
reg_x_CR_eval<-eval.fd(rep(1:k), regListCR_x$regfd)
reg_y_CR_eval<-eval.fd(rep(1:k), regListCR_y$regfd)
#plot each "Ann Arbor"
par(mfrow=c(2,1))
for(i in 1:10) {
  plot(reg_x_CR_eval[,i], reg_y_CR_eval[,i], type="l", lwd=2, xlab="x", ylab="y",
       main=paste("rep ", i, " with CR", sep=""))
}
x_mean_CR<-rowMeans(reg_x_CR_eval)
plot(rep(1:k), x_mean_CR, xlab="time", ylab="registered x", main="mean x with CR versus
time",
type="l", lwd=2)
y_mean_CR<-rowMeans(reg_y_CR_eval)
plot(rep(1:k), y_mean_CR, xlab="time", ylab="registered x", main="mean y with CR versus
time",
type="l", lwd=2)
plot(x_mean_CR, y_mean_CR, type="l", xlab="registered x", ylab="registered y",
main="Ann Arbor with CR registration", lwd=2)
#superimpose two plots together
plot(x_mean_CR, y_mean_CR, type="l", lwd=2, xlab="registered x",
ylab="registered y", main="regular Ann Arbor", col="red")
lines(x_mean_LM, y_mean_LM, col="blue", lwd=2)
legend(-1, 2.5, legend=c("CR", "LM"), fill=c("red", "blue"), cex=0.8)
```

```
#remove the trend factor, and retrieve the residuals
s0_eng_1_x_residuals<-NULL
for(i in 1:10) {
  model<-lm(reg_x_CR_eval[,i] ~ rep(1:k))
  s0_eng_1_x_residuals<-cbind(s0_eng_1_x_residuals, model$residuals)
}
matplot(s0_eng_1_x_residuals, type="l", xlab="time", ylab="residuals of x",
main="residuals of registered x with CR", lwd=2)
abline(h=0, lty=2)
plot(regListCR_x$regfd, main="registered x with CR", lwd=2)
#register velocity curves with LM method
D1_x_regfdLM<-register.newfd(deriv.fd(s0_eng_1_x, 1), regListLM_x$warpfd, type='direct')
D1_y_regfdLM<-register.newfd(deriv.fd(s0_eng_1_y, 1), regListLM_y$warpfd, type='direct')
matplot(eval.fd(rep(1:k), D1_x_regfdLM), xlab="time", ylab="velocity", type="l",
lwd=2, main=paste("x for LM"))
matplot(eval.fd(rep(1:k), D1_y_regfdLM), xlab="time", ylab="velocity", type="l",
lwd=2, main=paste("y for LM"))
#apply fdPar to smooth the velocity curves
lambda<-1e+1
D1_x_regfdPar<-fdPar(D1_x_regfdLM, 3, lambda)
D1_x_smoothList<-smooth.basis(rep(1:k), eval.fd(rep(1:k), D1_x_regfdLM), D1_x_regfdPar)
D1_y_regfdPar<-fdPar(D1_y_regfdLM, 3, lambda)
D1_y_smoothList<-smooth.basis(rep(1:k), eval.fd(rep(1:k), D1_y_regfdLM), D1_y_regfdPar)
#plotfit.fd(eval.fd(rep(1:k), D1_x_regfdLM), rep(1:k), D1_x_smoothList$fd, type="l")
matplot(eval.fd(rep(1:k), D1_x_regfdLM), xlab="time", ylab="velocity", type="l",
lwd=2, main=paste("x for LM"))
matplot(eval.fd(rep(1:k), D1_x_smoothList$fd), xlab="time", ylab="velocity", type="l",
lwd=2, main=paste("x for LM with lambda ", lambda, sep=""))
#plotfit.fd(eval.fd(rep(1:k), D1_y_regfdLM), rep(1:k), D1_y_smoothList$fd, type="l")
matplot(eval.fd(rep(1:k), D1_y_regfdLM), xlab="time", ylab="velocity", type="l",
lwd=2, main=paste("y for LM"))
matplot(eval.fd(rep(1:k), D1_y_smoothList$fd), xlab="time", ylab="velocity", type="l",
lwd=2, main=paste("y for LM with lambda ", lambda, sep=""))
#lambda<-seq(-8, -6, length.out=41)
#n<-length(lambda); gcv<-matrix(0, 2, n); sse<-matrix(0, 2, n)
#for(i in 1:n) {
#   if(!(i%%5)) print(i)
```

```
#   D1_x_regfdPar<-fdPar(D1_x_regfdLM, 3, exp(lambda[i]))
#   D1_x_smoothList<-smooth.basis(rep(1:k), eval.fd(rep(1:k), D1_x_regfdLM),
D1_x_regfdPar)
#   gcv[1,i]<-sum(D1_x_smoothList$gcv); sse[1,i]<-D1_x_smoothList$SSE
#   D1_y_regfdPar<-fdPar(D1_y_regfdLM, 3, exp(lambda[i]))
#   D1_y_smoothList<-smooth.basis(rep(1:k), eval.fd(rep(1:k), D1_y_regfdLM),
D1_y_regfdPar)
#   gcv[2,i]<-sum(D1_y_smoothList$gcv); sse[2,i]<-D1_y_smoothList$SSE
#}
#plot(lambda, gcv[1,], type="b")
#plot(lambda, gcv[2,], type="b")
#plot(lambda, sse[1,], type="b")
#plot(lambda, sse[2,], type="b")
#register acceleration curves with LM method
D2_x_regfdLM<-register.newfd(deriv.fd(s0_eng_1_x, 2), regListLM_x$warpfd, type='direct')
D2_y_regfdLM<-register.newfd(deriv.fd(s0_eng_1_y, 2), regListLM_y$warpfd, type='direct')
matplot(eval.fd(rep(1:k), D2_x_regfdLM), xlab="time", ylab="acce", type="l",
lwd=2, main="x for LM", ylim=c(-0.025, 0.025))
matplot(eval.fd(rep(1:k), D2_y_regfdLM), xlab="time", ylab="acce", type="l",
lwd=2, main="y for LM", ylim=c(-0.2, 0.2))
#apply fdPar to smooth the acceleration curves
lambda<- 5e+0
D2_x_regfdPar<-fdPar(D2_x_regfdLM, 2, lambda)
D2_x_smoothList<-smooth.basis(rep(1:k), eval.fd(rep(1:k), D2_x_regfdLM), D2_x_regfdPar)
D2_y_regfdPar<-fdPar(D2_y_regfdLM, 2, lambda)
D2_y_smoothList<-smooth.basis(rep(1:k), eval.fd(rep(1:k), D2_y_regfdLM), D2_y_regfdPar)
#plotfit.fd(eval.fd(rep(1:k), D2_x_regfdLM), rep(1:k), D2_x_smoothList$fd, type="l")
matplot(eval.fd(rep(1:k), D2_x_regfdLM), xlab="time", ylab="acce", type="l",
lwd=2, main=paste("x for LM"), ylim=c(-0.02, 0.02))
matplot(eval.fd(rep(1:k), D2_x_smoothList$fd), xlab="time", ylab="acce", type="l",
lwd=2, main=paste("x for LM with lambda ", lambda, sep=""), ylim=c(-0.01, 0.01))
#plotfit.fd(eval.fd(rep(1:k), D2_y_regfdLM), rep(1:k), D2_y_smoothList$fd, type="l")
matplot(eval.fd(rep(1:k), D2_y_regfdLM), xlab="time", ylab="acce", type="l",
lwd=2, main=paste("y for LM"), ylim=c(-0.16, 0.16))
matplot(eval.fd(rep(1:k), D2_y_smoothList$fd), xlab="time", ylab="acce", type="l",
lwd=2, main=paste("y for LM with lambda ", lambda, sep=""), ylim=c(-0.1, 0.1))
#calculate the third derivative known as "jerk"
#register tangent accleration curves
```

```
D2mag_s0<-sqrt(eval.fd(rep(1:k), D2_x_smoothList$fd)^2 + eval.fd(rep(1:k),
D2_y_smoothList$fd)^2)
matplot(D2mag_s0, type="l", xlab="time", ylab="acce", lwd=2,
ylim=c(0, 0.1), main="Tangent acceleration")
D2mag_mean_s0<-apply(D2mag_s0, 1, mean)
plot(rep(1:k), D2mag_mean_s0, type="l", xlab="time", ylab="tangent acce", ylim=c(0, 0.03),
lwd=2, main="Mean tangent acceleration")
##############################################################################
########
#CR method sucks...
#register velocity curves with CR method
D1_x_regfdCR<-register.newfd(deriv.fd(fdafdX, 1), regListCR_x$Wfd, type='monotone')
D1_y_regfdCR<-register.newfd(deriv.fd(fdafdY, 1), regListCR_y$Wfd, type='monotone')
matplot(eval.fd(rep(1:k), D1_x_regfdCR), xlab="time", ylab="velocity", type="l",
lwd=2, main=paste("x for CR"))
matplot(eval.fd(rep(1:k), D1_y_regfdCR), xlab="time", ylab="velocity", type="l",
lwd=2, main=paste("y for CR"))
#register acceleration curves with CR method
D2_x_regfdCR<-register.newfd(deriv.fd(fdafdX, 2), regListCR_x$Wfd, type='monotone')
D2_y_regfdCR<-register.newfd(deriv.fd(fdafdY, 2), regListCR_y$Wfd, type='monotone')
matplot(eval.fd(rep(1:k), D2_x_regfdCR), xlab="time", ylab="acce", type="l",
lwd=2, main="x for CR")
matplot(eval.fd(rep(1:k), D2_y_regfdCR), xlab="time", ylab="acce", type="l",
lwd=2, main="y for CR")
mag_acce<-sqrt(eval.fd(rep(1:k), D2_x_regfdCR)^2 + eval.fd(rep(1:k), D2_y_regfdCR)^2)
matplot(rep(1:k), mag_acce, type="l", lwd=2, xlab="time", ylab="magnitude of acce",
main="subject 0 with CR")
par(mfrow=c(1,2))
v1<-abs(eval.fd(rep(1:k), D1_x_regfdCR))
v2<-abs(eval.fd(rep(1:k), D1_y_regfdCR))
plot(c(1,10), c(0, 0.025), type="n", xlab="sample", ylab="Velocity",
main="mean velocity for each sample")
lines(colMeans(v1), col="red")
lines(colMeans(v2), col="blue")
legend(1.5, 0.016, legend=c("X", "Y"), fill=c("red", "blue"))

a1<-abs(eval.fd(rep(1:k), D2_x_regfdCR))
a2<-abs(eval.fd(rep(1:k), D2_y_regfdCR))
```

```
plot(c(1,10), c(0, 0.00125), type="n", xlab="sample", ylab="Acce",
main="mean acce for sample")
lines(colMeans(a1), col="red")
lines(colMeans(a2), col="blue")
legend(1.5, 0.0008, legend=c("X", "Y"), fill=c("red", "blue"))
sub1.R
library(fda)
#import 40 observations of subject 1 and create array "ann arbor"
s1r1<-read.csv("/Subject1/Subject1_2015_06_12_132558_001_6d.csv", header=T)
s1r2<-read.csv("/Subject1/Subject1_2015_06_12_132558_002_6d.csv", header=T)
s1r3<-read.csv("/Subject1/Subject1_2015_06_12_132558_003_6d.csv", header=T)
s1r4<-read.csv("/Subject1/Subject1_2015_06_12_132558_004_6d.csv", header=T)
s1r5<-read.csv("/Subject1/Subject1_2015_06_12_132558_005_6d.csv", header=T)
s1r6<-read.csv("/Subject1/Subject1_2015_06_12_132558_006_6d.csv", header=T)
s1r7<-read.csv("/Subject1/Subject1_2015_06_12_132558_007_6d.csv", header=T)
s1r8<-read.csv("/Subject1/Subject1_2015_06_12_132558_008_6d.csv", header=T)
s1r9<-read.csv("/Subject1/Subject1_2015_06_12_132558_009_6d.csv", header=T)
s1r10<-read.csv("/Subject1/Subject1_2015_06_12_132558_010_6d.csv", header=T)
s1r11<-read.csv("/Subject1/Subject1_2015_06_12_132558_011_6d.csv", header=T)
s1r12<-read.csv("/Subject1/Subject1_2015_06_12_132558_012_6d.csv", header=T)
s1r13<-read.csv("/Subject1/Subject1_2015_06_12_132558_013_6d.csv", header=T)
s1r14<-read.csv("/Subject1/Subject1_2015_06_12_132558_014_6d.csv", header=T)
s1r15<-read.csv("/Subject1/Subject1_2015_06_12_132558_015_6d.csv", header=T)
s1r16<-read.csv("/Subject1/Subject1_2015_06_12_132558_016_6d.csv", header=T)
s1r17<-read.csv("/Subject1/Subject1_2015_06_12_132558_017_6d.csv", header=T)
s1r18<-read.csv("/Subject1/Subject1_2015_06_12_132558_018_6d.csv", header=T)
s1r19<-read.csv("/Subject1/Subject1_2015_06_12_132558_019_6d.csv", header=T)
s1r20<-read.csv("/Subject1/Subject1_2015_06_12_132558_020_6d.csv", header=T)
x1<-data.frame(s1r1$pen.x, s1r2$pen.x, s1r3$pen.x, s1r4$pen.x, s1r5$pen.x, s1r6$pen.x,
s1r7$pen.x,
s1r8$pen.x, s1r9$pen.x, s1r10$pen.x, s1r11$pen.x, s1r12$pen.x, s1r13$pen.x, s1r14$pen.x,
s1r15$pen.x, s1r16$pen.x, s1r17$pen.x, s1r18$pen.x, s1r19$pen.x, s1r20$pen.x)
x1<-(-x1)
#x1<-scale(x1, scale=T)
#x1[is.na(x1)]<-0
colnames(x1)<-c("rep1", "rep2", "rep3", "rep4", "rep5", "rep6", "rep7", "rep8", "rep9", "rep10",
"rep11", "rep12", "rep13", "rep14", "rep15", "rep16", "rep17", "rep18", "rep19", "rep20")
```

```
y1<-data.frame(s1r1$pen.z, s1r2$pen.z, s1r3$pen.z, s1r4$pen.z, s1r5$pen.z, s1r6$pen.z,
s1r7$pen.z,
s1r8$pen.z, s1r9$pen.z, s1r10$pen.z, s1r11$pen.z, s1r12$pen.z, s1r13$pen.z, s1r14$pen.z,
s1r15$pen.z, s1r16$pen.z, s1r17$pen.z, s1r18$pen.z, s1r19$pen.z, s1r20$pen.z)
colnames(y1)<-colnames(x1)
#y1<-scale(y1, scale=T)
#y1[is.na(y1)]<-0
z1<-data.frame(s1r1$pen.y, s1r2$pen.y, s1r3$pen.y, s1r4$pen.y, s1r5$pen.y, s1r6$pen.y,
s1r7$pen.y,
s1r8$pen.y, s1r9$pen.y, s1r10$pen.y, s1r11$pen.y, s1r12$pen.y, s1r13$pen.y, s1r14$pen.y,
s1r15$pen.y, s1r16$pen.y, s1r17$pen.y, s1r18$pen.y, s1r19$pen.y, s1r20$pen.y)
colnames(z1)<-colnames(z1)
#z1<-scale(z1, scale=T)
#z1[is.na(z1)]<-0
#find the beginning of "Arbor"
s1_eng_start<-c(1150, 1300, 1320, 890, 840, 930, 1300, 910, 900, 840,
                640, 940, 1150, 1000, 800, 960, 770, 840, 900, 900)
s1_eng_end<-c(2000, 2250, 2070, 1800, 1700, 1900, 2270, 1980, 2070, 1900,
              1500, 2130, 2140, 1960, 1650, 1800, 1600, 1850, 1830, 1780)
par(mfrow=c(2,1), ask=F)
for(i in 1:20) {
  plot(x1[s1_eng_start[i]:s1_eng_end[i],i], y1[s1_eng_start[i]:s1_eng_end[i],i], lwd=1,
  main=paste("rep ", i, sep=""))
  abline(h=0, lty=2)
  plot(rep(s1_eng_start[i]:s1_eng_end[i]), z1[s1_eng_start[i]:s1_eng_end[i], i],
  main=paste("z of rep ", i, sep=""))
}
Temp<-NULL; k<-600
for(i in 1:20) {
  index<-sort(sample(s1_eng_start[i]:s1_eng_end[i], k, replace=F))
  Temp<-cbind(Temp, x1[index, i], y1[index, i], z1[index, i])
}
colnames(Temp)<-rep(colnames(x1)[1:20], each=3)
Temp<-scale(Temp, center=T, scale=T)
Temp[is.na(Temp)]<-0
#output the data to .csv file and store it
write.csv(Temp, "/Users/mtianwen/Downloads/s1_eng_arbor.csv", row.names = F)
#import .csv data to R
```

```
s1_eng_arbor<-read.csv("/Users/mtianwen/Downloads/s1_eng_arbor.csv", header=T)
#extract subset of 11 to 20 repetitions
#s1_eng_2_arbor<-subset(s1_eng_arbor, select=c(1:30))
column_name<-c("rep1", "rep2", "rep3", "rep4", "rep5", "rep6", "rep7", "rep8", "rep9", "rep10",
"rep11", "rep12", "rep13", "rep14", "rep15", "rep16", "rep17", "rep18", "rep19", "rep20")
colnames(s1_eng_arbor)<-rep(column_name, each=3)
s1_eng_arbor<-as.matrix(s1_eng_arbor)
N<-600
par(mfrow=c(3,1), ask=F)
matplot(rep(1:N), s1_eng_arbor[,seq(1,60,3)], type="l", main="x coordinate plot",
ylab="normalized x")
matplot(rep(1:N), s1_eng_arbor[,seq(2,60,3)], type="l", main="y coordinate plot",
ylab="normalized y")
matplot(rep(1:N), s1_eng_arbor[,seq(3,60,3)], type="l", main="z coordinate plot",
ylab="normalized z")
##############################################################################
#######
s1_reg_x_LM_eval<-read.csv("/Users/mtianwen/Downloads/s1_reg_x_LM_eval.csv", header =
T)
s1_reg_y_LM_eval<-read.csv("/Users/mtianwen/Downloads/s1_reg_y_LM_eval.csv", header =
T)
s1_reg_x_LM_eval<-as.matrix(s1_reg_x_LM_eval)
s1_reg_y_LM_eval<-as.matrix(s1_reg_y_LM_eval)
fdaarray<-array(0, dim=c(1000, 10, 2),
dimnames=list(rep(1:k),c("rep1", "rep2", "rep3", "rep4", "rep5",
"rep6", "rep7", "rep8", "rep9", "rep10"),
c("X", "Y")))
fdaarray[,,1]<-s1_reg_x_LM_eval
fdaarray[,,2]<-s1_reg_y_LM_eval

k<-2400
fdatime<-seq(0, 2400, len=1000)
fdarange <- c(0, k)
nbasis<-1005
norder<-7
fdabasis = create.bspline.basis(fdarange,nbasis, norder)
#   parameter object for coordinates
fdafd   <- fd(array(0, c(nbasis,10,2)), fdabasis)
```

```
lambda <- 1e8
fdaPar <- fdPar(fdafd, 5, lambda)
fdafdX <- smooth.basis(fdatime, fdaarray[,,1], fdaPar)$fd
plot(fdafdX)
fdafdY <- smooth.basis(fdatime, fdaarray[,,2], fdaPar)$fd
plot(fdafdY)
s1_eng_1_x_eval<-eval.fd(fdatime, fdafdX)
s1_eng_1_y_eval<-eval.fd(fdatime, fdafdY)

s1_eng_start<-fdatime[c(490, 490, 495, 500, 510, 510, 500, 500, 500, 500)]
for(i in 1:10) {
  plot(s1_eng_1_x_eval[c(s1_eng_start[i]:1000),i], s1_eng_1_y_eval[c(s1_eng_start[i]:1000),i],
  type="l")
}
Temp<-matrix(0, 600, 20); k<-600
for(i in 1:10) {
  Index<-seq(s1_eng_start[i], 2400, len=k)
  Temp[,2*i-1]<-eval.fd(Index, fdafdX[i])
  Temp[,2*i]<-eval.fd(Index, fdafdY[i])
}

#output the data to .csv file and store it
write.csv(Temp, "/Users/mtianwen/Downloads/s1_eng_arbor.csv", row.names = F)
write.csv(Temp[,seq(1,20,2)], "/Users/mtianwen/Downloads/s1_reg_x_LM_arbor.csv",
row.names = F)
write.csv(Temp[,seq(2,20,2)], "/Users/mtianwen/Downloads/s1_reg_y_LM_arbor.csv",
row.names = F)
#import .csv data to R
s1_reg_x_LM_arbor<-read.csv("/Users/mtianwen/Downloads/s1_reg_x_LM_arbor.csv",
header=T)
s1_reg_y_LM_arbor<-read.csv("/Users/mtianwen/Downloads/s1_reg_y_LM_arbor.csv",
header=T)
matplot(s1_eng_arbor, type="l", main="matplot of arbor for subject 1")
########   The below code is applide for k=1000, try landmark with first ten repetitions
########
i<-10
x<-rep(1:k); y<-s1_eng_1_z_eval[,i]
plot(x, y, type="l", main=paste("rep ", i, sep=""))
```

```
identify(x, y, labels=rep(1:k))
points(x[ximarks[i,]], y[ximarks[i,]], pty=1)
mark_data<-c(80, 138, 282, 330, 408, 455, 515, 561, 584, 651, 731, 794, 855, 902, 930,
             97, 134, 247, 292, 390, 438, 516, 562, 574, 661, 756, 804, 850, 889, 921,
             97, 143, 235, 285, 350, 411, 525, 608, 625, 712, 786, 841, 890, 937, 957,
             68, 125, 217, 275, 354, 410, 464, 525, 542, 637, 713, 777, 834, 880, 915,
             54, 110, 198, 259, 333, 386, 449, 521, 539, 640, 745, 816, 882, 940, 965,
             86, 132, 223, 281, 352, 404, 470, 523, 540, 652, 742, 810, 877, 919, 940,
             65, 106, 208, 266, 329, 372, 434, 492, 512, 673, 758, 816, 873, 923, 950,
             33,  72, 191, 247, 317, 370, 433, 499, 515, 617, 690, 743, 797, 840, 875,
             85, 124, 201, 253, 306, 356, 419, 472, 483, 622, 702, 752, 792, 832, 858,
             12,  51, 160, 218, 294, 345, 408, 468, 486, 597, 697, 757, 838, 889, 930)
ximarks<-matrix(mark_data, 10, 15, byrow=T)
PGSctrmean=colMeans(ximarks)
wbasisLM<-create.bspline.basis(c(0,k), 18, 3, c(0, PGSctrmean, k))
WfdLM<-fd(matrix(0,18,1), wbasisLM)
WfdParLM<-fdPar(WfdLM, 1, 1e-5)
##############################################################################
#######
par(mfrow=c(2,1))
regListLM_z<-landmarkreg(fdobj=fdafdZ, ximarks=ximarks, x0marks=PGSctrmean,
WfdPar=WfdParLM, monwrd = T)
plot(regListLM_z$regfd, main="registered z with LM", lwd=2)
plot(regListLM_z$warpfd, main="warp function for z", lwd=2)
regListLM_x<-landmarkreg(fdobj=fdafdX, ximarks=ximarks, x0marks=PGSctrmean,
WfdPar=WfdParLM, monwrd = T)
plot(regListLM_x$regfd, main="registered x with LM", lwd=2)
plot(regListLM_x$warpfd, main="warp function for x", lwd=2)
regListLM_y<-landmarkreg(fdobj=fdafdY, ximarks=ximarks, x0marks=PGSctrmean,
WfdPar=WfdParLM, monwrd = T)
plot(regListLM_y$regfd, main="registered y with LM", lwd=2)
plot(regListLM_y$warpfd, main="warp function for y", lwd=2)
for(i in 1:10) {
  plot(regListLM_y$regfd[i], main=paste("rep ", i, sep=""))
}
#warp deformation plots
par(mfrow=c(1,1))
```

```
matplot(rep(1:k), (eval.fd(rep(1:k), regListLM_x$warpfd)-matrix(rep(1:k), k, 10, byrow=F)),
ylab="h_i(t)-t", type="l", lty=1:2, col=1:9, lwd=2, main="coordinate warp deformation")
#independent of x,y,z
AmPhasList_1_x<-AmpPhaseDecomp(s1_eng_1_x, regListLM_x$regfd, regListLM_x$warpfd)
#11.9, -4.4, -0.59, 0.99
AmPhasList_1_y<-AmpPhaseDecomp(s1_eng_1_y, regListLM_y$regfd, regListLM_y$warpfd)
#277, 553, 0.666, 0.954
AmPhasList_1_z<-AmpPhaseDecomp(s1_eng_1_z, regListLM_z$regfd, regListLM_z$warpfd)
#222, 310, 0.583, 0.971
reg_x_LM_eval<-eval.fd(rep(1:k), regListLM_x$regfd)
reg_y_LM_eval<-eval.fd(rep(1:k), regListLM_y$regfd)
par(mfrow=c(2,1))
for(i in 1:10) {
  plot(reg_x_LM_eval[,i], reg_y_LM_eval[,i], type="l", xlab="x", ylab="y",
  main=paste("rep ", i, " with LM", sep=""))
}
x_mean_LM<-rowMeans(reg_x_LM_eval)
plot(rep(1:k), x_mean_LM, lwd=2, xlab="time", ylab="registered x",
main="registered x with LM versus time", type="l")
y_mean_LM<-rowMeans(reg_y_LM_eval)
plot(rep(1:k), y_mean_LM, lwd=2, xlab="time", ylab="registered y",
main="registered y with LM versus time", type="l")
plot(x_mean_LM, y_mean_LM, lwd=2, type="l", xlab="registered x",
ylab="registered y", main="Ann Arbor with LM registration")
plot.new()
#remove trend factor and retrieve residuals
s1_eng_1_x_residuals<-NULL
for(i in 1:10) {
  model<-lm(reg_x_LM_eval[,i] ~ rep(1:k))
  s1_eng_1_x_residuals<-cbind(s1_eng_1_x_residuals, model$residuals)
}
matplot(s1_eng_1_x_residuals, type="l", lwd=2, xlab="time", ylab="residuals of x",
main="residuals of registered x with LM")
abline(h=0, lty=2, lwd=2)
plot(regListLM_x$regfd, main="registered x with LM", lwd=2)
#Notice that maybe it is necessary to fit a functional data object to residual data.
#try continuous registration with Function register.fd based on the landmarker result.
```

```
wbasisCR<-create.bspline.basis(c(0,k), 20, 4)
Wfd0CR<-fd(matrix(0,20,10), wbasisCR)
WfdParCR<-fdPar(Wfd0CR, 2, 0.1)
regListCR_x<-register.fd(mean(regListLM_x$regfd), regListLM_x$regfd, WfdParCR)
plot(regListCR_x$regfd, main="registered x with CR", lwd=2)
regListCR_y<-register.fd(mean(regListLM_y$regfd), regListLM_y$regfd, WfdParCR)
plot(regListCR_y$regfd, main="registered y with CR", lwd=2)
AmPhasList_1_x_cf<-AmpPhaseDecomp(regListLM_x$regfd, regListCR_x$regfd,
regListCR_x$warpfd)
#6.9, 6.62, 0.49, 1.01
AmPhasList_1_y_cf<-AmpPhaseDecomp(regListLM_y$regfd, regListCR_y$regfd,
regListCR_y$warpfd)
#220, 76, 0.257, 1.03 significant progress!
reg_x_CR_eval<-eval.fd(rep(1:k), regListCR_x$regfd)
reg_y_CR_eval<-eval.fd(rep(1:k), regListCR_y$regfd)
#plot each "Ann Arbor"
for(i in 1:10) {
  plot(reg_x_CR_eval[,i], reg_y_CR_eval[,i], type="l", xlab="x", ylab="y",
       main=paste("rep ", i, " with CR", sep=""))
}
x_mean_CR<-rowMeans(reg_x_CR_eval)
plot(rep(1:k), x_mean_CR, xlab="time", ylab="registered x", main="registered x with CR versus
time",
type="l", lwd=2)
y_mean_CR<-rowMeans(reg_y_CR_eval)
plot(rep(1:k), y_mean_CR, xlab="time", ylab="registered x", main="registered y with CR versus
time",
type="l", lwd=2)
plot(x_mean_CR, y_mean_CR, type="l", xlab="registered x", ylab="registered y",
main="Ann Arbor with CR registration", lwd=2)
#superimpose two plots together
plot(x_mean_CR, y_mean_CR, type="l", lwd=2, xlab="registered x", ylab="registered y",
main="regular Ann Arbor", col="red")
lines(x_mean_LM, y_mean_LM, col="blue", lwd=2)
legend(-1, 2, legend=c("CR", "LM"), fill=c("red", "blue"), cex=0.8)
#remove the trend factor, and retrieve the residuals
s1_eng_1_x_residuals<-NULL
for(i in 1:10) {
```

```
  model<-lm(reg_x_CR_eval[,i] ~ rep(1:k))
  s1_eng_1_x_residuals<-cbind(s1_eng_1_x_residuals, model$residuals)
}
matplot(s1_eng_1_x_residuals, type="l", xlab="time", ylab="residuals of x",
main="residuals of registered x with CR", lwd=2)
abline(h=0, lty=2, lwd=2)
#register velocity curves with LM method
D1_x_regfdLM<-register.newfd(deriv.fd(s1_eng_1_x, 1), regListLM_x$warpfd, type='direct')
D1_y_regfdLM<-register.newfd(deriv.fd(s1_eng_1_y, 1), regListLM_y$warpfd, type='direct')
matplot(eval.fd(rep(1:k), D1_x_regfdLM), xlab="time", ylab="velocity", type="l",
lwd=2, main=paste("x for LM"), ylim=c(-0.05, 0.05))
matplot(eval.fd(rep(1:k), D1_y_regfdLM), xlab="time", ylab="velocity", type="l",
lwd=2, main=paste("y for LM"), ylim=c(-0.5, 0.5))
#apply fdPar to smooth the velocity curves
lambda<-1e+2
D1_x_regfdPar<-fdPar(D1_x_regfdLM, 2, lambda)
D1_x_smoothList<-smooth.basis(rep(1:k), eval.fd(rep(1:k), D1_x_regfdLM), D1_x_regfdPar)
D1_y_regfdPar<-fdPar(D1_y_regfdLM, 2, lambda)
D1_y_smoothList<-smooth.basis(rep(1:k), eval.fd(rep(1:k), D1_y_regfdLM), D1_y_regfdPar)
#plotfit.fd(eval.fd(rep(1:k), D1_x_regfdLM), rep(1:k), D1_x_smoothList$fd, type="l")
matplot(eval.fd(rep(1:k), D1_x_regfdLM), xlab="time", ylab="velocity", type="l",
lwd=2, main=paste("x for LM"), ylim=c(-0.05, 0.05))
matplot(eval.fd(rep(1:k), D1_x_smoothList$fd), xlab="time", ylab="velocity", type="l",
lwd=2, main=paste("x for LM with lambda ", lambda, sep=""), ylim=c(-0.05, 0.05))
#plotfit.fd(eval.fd(rep(1:k), D1_y_regfdLM), rep(1:k), D1_y_smoothList$fd, type="l")
matplot(eval.fd(rep(1:k), D1_y_regfdLM), xlab="time", ylab="velocity", type="l",
lwd=2, main=paste("y for LM"), ylim=c(-0.5, 0.5))
matplot(eval.fd(rep(1:k), D1_y_smoothList$fd), xlab="time", ylab="velocity", type="l",
lwd=2, main=paste("y for LM with lambda ", lambda, sep=""), ylim=c(-0.4, 0.4))
#register acceleration curves with LM method
D2_x_regfdLM<-register.newfd(deriv.fd(s1_eng_1_x, 2), regListLM_x$warpfd, type='direct')
D2_y_regfdLM<-register.newfd(deriv.fd(s1_eng_1_y, 2), regListLM_y$warpfd, type='direct')
matplot(eval.fd(rep(1:k), D2_x_regfdLM), xlab="time", ylab="acce", type="l",
lwd=2, main="x for LM", ylim=c(-0.025, 0.025))
matplot(eval.fd(rep(1:k), D2_y_regfdLM), xlab="time", ylab="acce", type="l",
lwd=2, main="y for LM", ylim=c(-0.2, 0.2))
#apply fdPar to smooth the acceleration curves
```

```
lambda<- 1e+2
D2_x_regfdPar<-fdPar(D2_x_regfdLM, 2, lambda)
D2_x_smoothList<-smooth.basis(rep(1:k), eval.fd(rep(1:k), D2_x_regfdLM), D2_x_regfdPar)
D2_y_regfdPar<-fdPar(D2_y_regfdLM, 2, lambda)
D2_y_smoothList<-smooth.basis(rep(1:k), eval.fd(rep(1:k), D2_y_regfdLM), D2_y_regfdPar)
#plotfit.fd(eval.fd(rep(1:k), D2_x_regfdLM), rep(1:k), D2_x_smoothList$fd, type="l")
matplot(eval.fd(rep(1:k), D2_x_regfdLM), xlab="time", ylab="acce", type="l",
lwd=2, main=paste("x for LM"), ylim=c(-0.015, 0.015))
matplot(eval.fd(rep(1:k), D2_x_smoothList$fd), xlab="time", ylab="acce", type="l",
lwd=2, main=paste("x for LM with lambda ", lambda, sep=""), ylim=c(-0.006, 0.006))
#plotfit.fd(eval.fd(rep(1:k), D2_y_regfdLM), rep(1:k), D2_y_smoothList$fd, type="l")
matplot(eval.fd(rep(1:k), D2_y_regfdLM), xlab="time", ylab="acce", type="l",
lwd=2, main=paste("y for LM"), ylim=c(-0.2, 0.2))
matplot(eval.fd(rep(1:k), D2_y_smoothList$fd), xlab="time", ylab="acce", type="l",
lwd=2, main=paste("y for LM with lambda ", lambda, sep=""), ylim=c(-0.05, 0.05))
#register tangent accleration curves
D2mag_s0<-sqrt(eval.fd(rep(1:k), D2_x_regfdLM)^2 + eval.fd(rep(1:k), D2_x_regfdLM)^2)
matplot(D2mag_s0, type="l", xlab="time", ylab="acce",
lwd=2, ylim=c(0, 0.05), main="Tangent acceleration for LM")
D2mag_mean_s0<-apply(D2mag_s0, 1, mean)
plot(rep(1:k), D2mag_mean_s0, type="l", xlab="time", ylab="tangent acce", ylim=c(0, 0.03),
lwd=2, main="Mean tangent acceleration for LM")
D2mag_s0_lambda<-sqrt(eval.fd(rep(1:k), D2_x_smoothList$fd)^2 + eval.fd(rep(1:k),
D2_y_smoothList$fd)^2)
matplot(D2mag_s0_lambda, type="l", xlab="time", ylab="acce", lwd=2,
ylim=c(0, 0.05), main=paste("Tangent acceleration for LM ", lambda, sep=""))
D2mag_mean_s0_lambda<-apply(D2mag_s0_lambda, 1, mean)
plot(rep(1:k), D2mag_mean_s0_lambda, type="l", xlab="time", ylab="tangent acce",
ylim=c(0, 0.03), lwd=2, main=paste("Mean tangent acceleration for LM ", lambda, sep=""))

##############################################################################
############
#register velocity curves with CR method
D1_x_regfdCR<-register.newfd(deriv.fd(regListLM_x$regfd, 1), regListCR_x$Wfd,
type='periodic')
D1_y_regfdCR<-register.newfd(deriv.fd(regListLM_y$regfd, 1), regListCR_y$Wfd,
type='periodic')
matplot(eval.fd(rep(1:k), D1_x_regfdCR), xlab="time", ylab="velocity", type="l",
```

```
lwd=2, main=paste("x for CR"))
matplot(eval.fd(rep(1:k), D1_y_regfdCR), xlab="time", ylab="velocity", type="l",
lwd=2, main=paste("y for CR"))
#apply fdPar to smooth the velocity curves
lambda<-1e+2
D1_x_regfdPar<-fdPar(D1_x_regfdCR, 3, lambda)
D1_x_smoothList<-smooth.basis(rep(1:k), eval.fd(rep(1:k), D1_x_regfdCR), D1_x_regfdPar)
D1_y_regfdPar<-fdPar(D1_y_regfdCR, 3, lambda)
D1_y_smoothList<-smooth.basis(rep(1:k), eval.fd(rep(1:k), D1_y_regfdCR), D1_y_regfdPar)
#plotfit.fd(eval.fd(rep(1:k), D1_x_regfdCR), rep(1:k), D1_x_smoothList$fd, type="l")
matplot(eval.fd(rep(1:k), D1_x_regfdCR), xlab="time", ylab="velocity", type="l",
lwd=2, main=paste("x for CR"), ylim=c(-0.05, 0.05))
matplot(eval.fd(rep(1:k), D1_x_smoothList$fd), type="l",
lwd=2, main=paste("x for CR with lambda ", lambda, sep=""), ylim=c(-0.05, 0.05))
#plotfit.fd(eval.fd(rep(1:k), D1_y_regfdCR), rep(1:k), D1_y_smoothList$fd, type="l")
matplot(eval.fd(rep(1:k), D1_y_regfdCR), xlab="time", ylab="velocity",
type="l", lwd=2, main=paste("y for CR"), ylim=c(-0.5, 0.5))
matplot(eval.fd(rep(1:k), D1_y_smoothList$fd), type="l",
lwd=2, main=paste("y for CR with lambda ", lambda, sep=""), ylim=c(-0.4, 0.4))
#register acceleration curves with CR method
D2_x_regfdCR<-register.newfd(deriv.fd(s1_eng_1_x, 2), regListCR_x$Wfd, type='monotone')
D2_y_regfdCR<-register.newfd(deriv.fd(s1_eng_1_y, 2), regListCR_y$Wfd, type='monotone')
matplot(eval.fd(rep(1:k), D2_x_regfdCR), xlab="time", ylab="acce", type="l", lwd=2,
main="x for CR", ylim=c(-0.02, 0.02))
matplot(eval.fd(rep(1:k), D2_y_regfdCR), xlab="time", ylab="acce", type="l", lwd=2,
main="y for CR", ylim=c(-0.15, 0.15))
#apply fdPar to smooth the acceleration curves
lambda<- 1e+1
D2_x_regfdPar<-fdPar(D2_x_regfdCR, 2, lambda)
D2_x_smoothList<-smooth.basis(rep(1:k), eval.fd(rep(1:k), D2_x_regfdCR), D2_x_regfdPar)
D2_y_regfdPar<-fdPar(D2_y_regfdCR, 2, lambda)
D2_y_smoothList<-smooth.basis(rep(1:k), eval.fd(rep(1:k), D2_y_regfdCR), D2_y_regfdPar)
#plotfit.fd(eval.fd(rep(1:k), D2_x_regfdLM), rep(1:k), D2_x_smoothList$fd, type="l")
matplot(eval.fd(rep(1:k), D2_x_regfdCR), xlab="time", ylab="acce", type="l", lwd=2,
main=paste("x for CR"), ylim=c(-0.02, 0.02))
matplot(eval.fd(rep(1:k), D2_x_smoothList$fd), xlab="time", ylab="acce", type="l", lwd=2,
main=paste("x for CR with lambda ", lambda, sep=""), ylim=c(-0.005, 0.005))
```

```
#plotfit.fd(eval.fd(rep(1:k), D2_y_regfdLM), rep(1:k), D2_y_smoothList$fd, type="l")
matplot(eval.fd(rep(1:k), D2_y_regfdCR), xlab="time", ylab="acce", type="l", lwd=2,
main=paste("y for CR"), ylim=c(-0.16, 0.16))
matplot(eval.fd(rep(1:k), D2_y_smoothList$fd), xlab="time", ylab="acce", type="l", lwd=2,
main=paste("y for CR with lambda ", lambda, sep=""), ylim=c(-0.05, 0.05))
classification to others.R
library(fda)
#import data
reg_x_CR_eval<-read.csv("/Users/mtianwen/Downloads/s0_reg_x_CR_eval_arbor.csv", header
= T)
reg_y_CR_eval<-read.csv("/Users/mtianwen/Downloads/s0_reg_y_CR_eval_arbor.csv", header
= T)
reg_x_CR_eval<-as.matrix(reg_x_CR_eval)*5
reg_y_CR_eval<-as.matrix(reg_y_CR_eval)*5
l<-11
fdaarray<-array(0, dim=c(600, l, 2),
dimnames=list(rep(1:600),c("rep1", "rep2", "rep3", "rep4", "rep5",
"rep6", "rep7", "rep8", "rep9", "rep10", "rep11"),
c("X", "Y")))

fdaarray[,,1]<-reg_x_CR_eval
fdaarray[,,2]<-reg_y_CR_eval
#set up the fda object for forcing function and its weight function
s1_reg_x_LM_arbor<-read.csv("/Users/mtianwen/Downloads/s1_reg_x_LM_arbor.csv",
header=T)
s1_reg_y_LM_arbor<-read.csv("/Users/mtianwen/Downloads/s1_reg_y_LM_arbor.csv",
header=T)
reg_x_LM_eval<-as.matrix(s1_reg_x_LM_arbor)*5
reg_y_LM_eval<-as.matrix(s1_reg_y_LM_arbor)*5
l<-10
fdaarray<-array(0, dim=c(600, l, 2),
dimnames=list(rep(1:k),c("rep1", "rep2", "rep3", "rep4", "rep5",
"rep6", "rep7", "rep8", "rep9", "rep10"),
c("X", "Y")))
fdaarray[,,1]<-reg_x_LM_eval
fdaarray[,,2]<-reg_y_LM_eval
#The below two lines are valid for psuedo data
fdaarray[,,1]<-X
```

```
fdaarray[,,2]<-Y
#Perfome pda.fd without forcing function
k<-1440
fdatime<-seq(0, k, len=600)
fdarange <- c(0, k)
nbasis<-605
norder<-7
fdabasis = create.bspline.basis(fdarange,nbasis, norder)
#   parameter object for coordinates
fdafd   <- fd(array(0, c(nbasis,l,2)), fdabasis)
lambda <- 1e8
fdaPar <- fdPar(fdafd, 5, lambda)

#   set up the two forcing functions
ufdlist <- vector("list", 2)
#   constant forcing
constbasis <- create.constant.basis(fdarange)
constfd      <- fd(matrix(1,1,l), constbasis)
ufdlist[[1]] <- constfd
# time forcing
linbasis     <- create.monomial.basis(fdarange, 2)
lincoef      <- matrix(0,2,l)
lincoef[2,] <- 1
ufdlist[[2]] <- fd(lincoef, linbasis)
awtlist      <- vector("list", 2)
constfd      <- fd(1, constbasis)
constfdPar <- fdPar(constfd)
awtlist[[1]] <- constfdPar
awtlist[[2]] <- constfdPar
wbasis125 <- create.bspline.basis(fdarange, 125)
#pdf("092815_2.pdf")
#First apply the equation to the same writer
fdafdX <- smooth.basis(fdatime, fdaarray[,,1], fdaPar)$fd
xfdlist<-vector("list", 1)
xfdlist[[1]]<-fdafdX
plot(fdafdX)
#set the number of basis functions to 125
```

```
bfd       <- fd(matrix(0,125,1), wbasis125)
bfdPar   <- fdPar(bfd, 1, 0)
bwtlist <- vector("list", 2)
bwtlist[[1]] <- bfdPar
bwtlist[[2]] <- bfdPar
bwtlist[[3]] <- bfdPar
#carry out principal differential analysis
pdaList <- pda.fd(xfdlist, bwtlist, awtlist, ufdlist)
bestwtlist <- pdaList$bwtlist
aestwtlist <- pdaList$awtlist
resfdlist   <- pdaList$resfdlist
par(mfrow=c(2,1), ask=F)
#evaluate forcing functions
resfdX   <- resfdlist[[1]]
plot(resfdX, main="Forcing functions for subject 0 with 125 bspline basis")
resmeanfdX<-mean(resfdX)
resmeanfdX_eval<-eval.fd(fdatime, resmeanfdX)
#extract and plot the weight functions
beta_X<-matrix(0, 600, 3)
#par(mfrow=c(1,1), ask=F)
for(j in 1:3) {
  betafdPar<-bestwtlist[[j]]
  betafd<-betafdPar$fd
  betafd_eval<-eval.fd(fdatime, betafd)
  beta_X[,j]<-betafd_eval
  plot(fdatime, betafd_eval, type="l", main=paste("weight function for ", j, sep=""))
}
#extract and evaluate weight for forcing functions
w1_X<-aestwtlist[[1]]$fd$coefs
w2_X<-aestwtlist[[2]]$fd$coefs
fdafdY <- smooth.basis(fdatime, fdaarray[,,2], fdaPar)$fd
fdlist<-vector("list", 1)
xfdlist[[1]]<-fdafdY
plot(fdafdY)
#set the number of basis functions to 125
bfd       <- fd(matrix(0,125,1), wbasis125)
bfdPar   <- fdPar(bfd, 1, 0)
```

```
bwtlist <- vector("list", 2)
bwtlist[[1]] <- bfdPar
bwtlist[[2]] <- bfdPar
bwtlist[[3]] <- bfdPar
#carry out principal differential analysis
pdaList <- pda.fd(xfdlist, bwtlist, awtlist, ufdlist)
bestwtlist <- pdaList$bwtlist
aestwtlist <- pdaList$awtlist
resfdlist   <- pdaList$resfdlist
par(mfrow=c(2,1))
#evaluate forcing functions
resfdY    <- resfdlist[[1]]
plot(resfdY, main="Forcing functions for subject 0 with 125 bspline basis")
resmeanfdY<-mean(resfdY)
resmeanfdY_eval<-eval.fd(fdatime, resmeanfdY)
#extract and plot the weight functions
beta_Y<-matrix(0, 600, 3)
#par(mfrow=c(1,1), ask=F)
for(j in 1:3) {
  betafdPar<-bestwtlist[[j]]
  betafd<-betafdPar$fd
  betafd_eval<-eval.fd(fdatime, betafd)
  beta_Y[,j]<-betafd_eval
  plot(fdatime, betafd_eval, type="l", main=paste("weight function for ", j, sep=""))
}

#extract and evaluate weight for forcing functions
w1_Y<-aestwtlist[[1]]$fd$coefs
w2_Y<-aestwtlist[[2]]$fd$coefs
par(mfrow=c(2,1))
plot(resfdX)
plot(resfdY)
D3fdafdX<-eval.fd(fdatime, fdafdX, 3)
D3fdafdY<-eval.fd(fdatime, fdafdY, 3)
D3fdafdX_mean<-apply(D3fdafdX, 1, mean)
D3fdafdY_mean<-apply(D3fdafdY, 1, mean)
plot(fdatime, D3fdafdX_mean*1000000, type="l", col="blue", main="x coordinate")
```

```
lines(fdatime, resmeanfdX_eval*1000000, type="l", col="red")
abline(h=0, lty=2)
legend(50, -100, legend=c("3rd", "residual"), fill=c("blue", "red"), cex=0.7)
plot(fdatime, D3fdafdY_mean*1000000, type="l", col="blue", main="y coordinate")
lines(fdatime, resmeanfdY_eval*1000000, type="l", col="red")
abline(h=0, lty=2)
legend(10, 500, legend=c("3rd", "residual"), fill=c("blue", "red"), cex=0.7)
par(mfrow=c(1,2))
#calculate forcing function, which is the L(X(t)) and L(Y(t)), the linear differential operator
LX<-matrix(0, 600, 11)
D1_X<-eval.fd(fdatime, fdafdX, 1)
D2_X<-eval.fd(fdatime, fdafdX, 2)
LX<-beta_X[,1]*fdaarray[,,1] + beta_X[,2]*D1_X + beta_X[,3]*D2_X + matrix(-w1_X, 600,
11)
matplot(fdatime, LX*10^5, type="l", ylim=c(-50, 70), xlab="time", ylab="forcing function",
main="X dimension for subject 0")
abline(h=0, lty=2)
LY<-matrix(0, 600, 11)
D1_Y<-eval.fd(fdatime, fdafdY, 1)
D2_Y<-eval.fd(fdatime, fdafdY, 2)
LY<-beta_Y[,1]*fdaarray[,,2] + beta_Y[,2]*D1_Y + beta_Y[,3]*D2_Y + matrix(-w1_Y, 600,
11)
matplot(fdatime, LY*10^5, type="l", ylim=c(-150, 120), xlab="time", ylab="forcing function",
main="Y dimension for subject 0")
abline(h=0, lty=2)
classification to itself.R
library(fda)
#import data
reg_x_CR_eval<-read.csv("/Users/mtianwen/Downloads/s0_reg_x_CR_eval_arbor.csv", header
= T)
reg_y_CR_eval<-read.csv("/Users/mtianwen/Downloads/s0_reg_y_CR_eval_arbor.csv", header
= T)
reg_x_CR_eval<-as.matrix(reg_x_CR_eval)*5
reg_y_CR_eval<-as.matrix(reg_y_CR_eval)*5
l<-11
fdaarray<-array(0, dim=c(600, l, 2),
dimnames=list(rep(1:600),c("rep1", "rep2", "rep3", "rep4", "rep5",
"rep6", "rep7", "rep8", "rep9", "rep10", "rep11"),
```

```
c("X", "Y")))
fdaarray[,,1]<-reg_x_CR_eval
fdaarray[,,2]<-reg_y_CR_eval
#set up the fda object for forcing function and its weight function
s1_reg_x_LM_arbor<-read.csv("/Users/mtianwen/Downloads/s1_reg_x_LM_arbor.csv",
header=T)
s1_reg_y_LM_arbor<-read.csv("/Users/mtianwen/Downloads/s1_reg_y_LM_arbor.csv",
header=T)
reg_x_LM_eval<-as.matrix(s1_reg_x_LM_arbor)*5
reg_y_LM_eval<-as.matrix(s1_reg_y_LM_arbor)*5
l<-10
fdaarray<-array(0, dim=c(600, l, 2),
dimnames=list(rep(1:k),c("rep1", "rep2", "rep3", "rep4", "rep5",
"rep6", "rep7", "rep8", "rep9", "rep10"),
c("X", "Y")))
fdaarray[,,1]<-reg_x_LM_eval
fdaarray[,,2]<-reg_y_LM_eval
#The below two lines are valid for psuedo data
fdaarray[,,1]<-X
fdaarray[,,2]<-Y
#Perfome pda.fd without forcing function
k<-1440
fdatime<-seq(0, k, len=600)
fdarange <- c(0, k)
nbasis<-605
norder<-7
fdabasis = create.bspline.basis(fdarange,nbasis, norder)
#   parameter object for coordinates
fdafd   <- fd(array(0, c(nbasis,l,2)), fdabasis)
lambda <- 1e8
fdaPar <- fdPar(fdafd, 5, lambda)
#   set up the two forcing functions
ufdlist <- vector("list", 2)
#   constant forcing
constbasis <- create.constant.basis(fdarange)
constfd      <- fd(matrix(1,1,l), constbasis)
ufdlist[[1]] <- constfd
# time forcing
```

```
linbasis    <- create.monomial.basis(fdarange, 2)
lincoef     <- matrix(0,2,l)
lincoef[2,] <- 1
ufdlist[[2]] <- fd(lincoef, linbasis)
awtlist     <- vector("list", 2)
constfd     <- fd(1, constbasis)
constfdPar <- fdPar(constfd)
awtlist[[1]] <- constfdPar
awtlist[[2]] <- constfdPar
wbasis125 <- create.bspline.basis(fdarange, 125)
#pdf("092815_2.pdf")
#First apply the equation to the same writer
fdafdX <- smooth.basis(fdatime, fdaarray[,,1], fdaPar)$fd
xfdlist<-vector("list", 1)
xfdlist[[1]]<-fdafdX
plot(fdafdX)
#set the number of basis functions to 125
bfd       <- fd(matrix(0,125,1), wbasis125)
bfdPar   <- fdPar(bfd, 1, 0)
bwtlist <- vector("list", 2)
bwtlist[[1]] <- bfdPar
bwtlist[[2]] <- bfdPar
bwtlist[[3]] <- bfdPar
#carry out principal differential analysis
pdaList <- pda.fd(xfdlist, bwtlist, awtlist, ufdlist)
bestwtlist <- pdaList$bwtlist
aestwtlist <- pdaList$awtlist
resfdlist   <- pdaList$resfdlist
par(mfrow=c(2,1), ask=F)
#evaluate forcing functions
resfdX   <- resfdlist[[1]]
plot(resfdX, main="Forcing functions for subject 0 with 125 bspline basis")
resmeanfdX<-mean(resfdX)
resmeanfdX_eval<-eval.fd(fdatime, resmeanfdX)
#extract and plot the weight functions
beta_X<-matrix(0, 600, 3)
#par(mfrow=c(1,1), ask=F)
```

```
for(j in 1:3) {
  betafdPar<-bestwtlist[[j]]
  betafd<-betafdPar$fd
  betafd_eval<-eval.fd(fdatime, betafd)
  beta_X[,j]<-betafd_eval
  plot(fdatime, betafd_eval, type="l", main=paste("weight function for ", j, sep=""))
}
#extract and evaluate weight for forcing functions
w1_X<-aestwtlist[[1]]$fd$coefs
w2_X<-aestwtlist[[2]]$fd$coefs
fdafdY <- smooth.basis(fdatime, fdaarray[,,2], fdaPar)$fd
fdlist<-vector("list", 1)
xfdlist[[1]]<-fdafdY
plot(fdafdY)
#set the number of basis functions to 125
bfd       <- fd(matrix(0,125,1), wbasis125)
bfdPar   <- fdPar(bfd, 1, 0)
bwtlist <- vector("list", 2)
bwtlist[[1]] <- bfdPar
bwtlist[[2]] <- bfdPar
bwtlist[[3]] <- bfdPar
#carry out principal differential analysis
pdaList <- pda.fd(xfdlist, bwtlist, awtlist, ufdlist)
bestwtlist <- pdaList$bwtlist
aestwtlist <- pdaList$awtlist
resfdlist   <- pdaList$resfdlist
par(mfrow=c(2,1))
#evaluate forcing functions
resfdY   <- resfdlist[[1]]
plot(resfdY, main="Forcing functions for subject 0 with 125 bspline basis")
resmeanfdY<-mean(resfdY)
resmeanfdY_eval<-eval.fd(fdatime, resmeanfdY)
#extract and plot the weight functions
beta_Y<-matrix(0, 600, 3)
#par(mfrow=c(1,1), ask=F)
for(j in 1:3) {
  betafdPar<-bestwtlist[[j]]
```

```
  betafd<-betafdPar$fd
  betafd_eval<-eval.fd(fdatime, betafd)
  beta_Y[,j]<-betafd_eval
  plot(fdatime, betafd_eval, type="l", main=paste("weight function for ", j, sep=""))
}
  #extract and evaluate weight for forcing functions
w1_Y<-aestwtlist[[1]]$fd$coefs
w2_Y<-aestwtlist[[2]]$fd$coefs
par(mfrow=c(2,1))
plot(resfdX)
plot(resfdY)
D3fdafdX<-eval.fd(fdatime, fdafdX, 3)
D3fdafdY<-eval.fd(fdatime, fdafdY, 3)
D3fdafdX_mean<-apply(D3fdafdX, 1, mean)
D3fdafdY_mean<-apply(D3fdafdY, 1, mean)
plot(fdatime, D3fdafdX_mean*1000000, type="l", col="blue", main="x coordinate")
lines(fdatime, resmeanfdX_eval*1000000, type="l", col="red")
abline(h=0, lty=2)
legend(50, -100, legend=c("3rd", "residual"), fill=c("blue", "red"), cex=0.7)
plot(fdatime, D3fdafdY_mean*1000000, type="l", col="blue", main="y coordinate")
lines(fdatime, resmeanfdY_eval*1000000, type="l", col="red")
abline(h=0, lty=2)
legend(10, 500, legend=c("3rd", "residual"), fill=c("blue", "red"), cex=0.7)
par(mfrow=c(1,2))
#calculate forcing function, which is the L(X(t)) and L(Y(t)), the linear differential operator
LX<-matrix(0, 600, 11)
D1_X<-eval.fd(fdatime, fdafdX, 1)
D2_X<-eval.fd(fdatime, fdafdX, 2)
LX<-beta_X[,1]*fdaarray[,,1] + beta_X[,2]*D1_X + beta_X[,3]*D2_X + matrix(-w1_X, 600,
11)
matplot(fdatime, LX*10^5, type="l", ylim=c(-50, 70), xlab="time", ylab="forcing function",
main="X dimension for subject 0")
abline(h=0, lty=2)
LY<-matrix(0, 600, 11)
D1_Y<-eval.fd(fdatime, fdafdY, 1)
D2_Y<-eval.fd(fdatime, fdafdY, 2)
LY<-beta_Y[,1]*fdaarray[,,2] + beta_Y[,2]*D1_Y + beta_Y[,3]*D2_Y + matrix(-w1_Y, 600,
11)
```

```
matplot(fdatime, LY*10^5, type="l", ylim=c(-150, 120), xlab="time", ylab="forcing function",
main="Y dimension for subject 0")
abline(h=0, lty=2)
```