Distribution Agreement

In presenting this thesis as a partial fulfillment of the requirements for a degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis in whole or in part in all forms of media, now or hereafter now, including display on the World Wide Web. I understand that I may select some access restrictions as part of the online submission of this thesis. I retain all ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles or books) all or part of this thesis.

Scott Masterson April 7, 2024

Modified Functional Principal Component Regression

by

Scott Masterson

Bree Ettinger
Adviser

Applied Math and Statistics

Bree Ettinger
Adviser

Julianne Chung

Committee Member

Talea Mayo

Committee Member

Shomu Banerjee

Committee Member

2024

Modified Functional Principal Component Regression

By

Scott Masterson

Bree Ettinger

Adviser

An abstract of
a thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Science with Honors

Applied Math and Statistics

2024

Abstract

Modified Functional Principal Component Regression

By Scott Masterson

Using simulated temperature data as a proof of concept, we present a novel approach to prediction via the development and application of Modified Functional Principal Component Regression (MFPCR). By employing bivariate splines over triangulations in a functional linear regression model, this study offers a proof of concept for the modified approach that include additional data to improve the Function Principal Component Regression (FPCR). The mock temperature prediction problem has few complexities and clear influencing factors. Our comparative analysis of MFPCR and FPCR reveals insights into predictive accuracy, uncertainty quantification, and the spatial distribution of functional data, setting the stage for more precise neighborhood-level forecasting in future investigations.

Modified Functional Principal Component Regression

By

Scott Masterson

Bree Ettinger

Adviser

A thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Science with Honors

Applied Math and Statistics

2024

Acknowledgments

I thank my family and friends for always believing in my talents and abilities. Even when I have doubts, they remind me that my ambitions are attainable.

Contents

1 Abstract

Using simulated temperature data as a proof of concept, we present a novel approach to prediction via the development and application of Modified Functional Principal Component Regression (MFPCR). By employing bivariate splines over triangulations in a functional linear regression model, this study offers a proof of concept for the modified approach that include additional data to improve the Function Principal Component Regression (FPCR). The mock temperature prediction problem has few complexities and clear influencing factors. Our comparative analysis of MFPCR and FPCR reveals insights into predictive accuracy, uncertainty quantification, and the spatial distribution of functional data, setting the stage for more precise neighborhood-level forecasting in future investigations.

2 Introduction and Background

Ground-level ozone poses a significant threat as a harmful air pollutant, leading to health issues such as coughing, chest pain, and the exacerbation of chronic conditions. This issue disproportionately affects low-income communities situated close to pollution sources. Unlike direct emissions, ground-level ozone forms through the chemical interactions between nitrogen oxides (NOx) and volatile organic compounds (VOC) under sunlight, complicating the modeling and prediction of ground-level ozone [1].

To predict ground-level ozone concentrations, this thesis explores the efficacy of a modified functional linear regression model (MFPCR), particularly those employing bivariate splines over triangulations. Due to the aforementioned complexities of ozone prediction, we develop a modeling application in a more controlled environment in this paper. We construct a temperature example that has fewer nuances: the cyclical nature makes temperature more predictable than ground-level ozone, and there are more clear factors that influence the current temperature.

Previous work has used functional principal component analysis (FPCR) [1] and random forest machine learning methods to estimate ground-level ozone [5]. These approaches to forecasting ozone levels predominantly rely on analyzing time series data from individual monitoring stations. A significant limitation of traditional approaches is their inability to account for spatial interdependencies—specifically, how ozone concentrations at nearby locations may influence or be indicative of future concentrations at the target location.

A further complexity in predicting ozone levels arises from the spatial distribution of Environmental Protection Agency (EPA) monitoring stations across the United States. This distribution is characterized by its heterogeneity; certain areas are densely populated with stations, and other regions are sparsely covered, leading to data scarcity. This uneven distribution poses challenges for achieving a comprehensive and balanced analysis of ozone levels nationwide.

Moreover, the dataset pertaining to ozone levels is both sparse and irregular.

The variability in data points, combined with the volatile chemical-nature of ground-level ozone fluctuations, distinguishes the challenge of predicting ozone concentrations. The behavior of ground-level ozone is notably more erratic, exhibiting sharp fluctuations that are less predictable and more sensitive to a variety of influencing factors [1].

To visualize the disparity in monitoring coverage, consider the distribution map of EPA stations above. This map underscores the spatial challenges inherent in the dataset, highlighting areas of both data richness and scarcity, and underscoring the need for advanced predictive models that can effectively interpolate and extrapolate ozone levels across the varied landscape of the United States. One method that addresses these concerns is the Functional Principle Component Approach in [1].

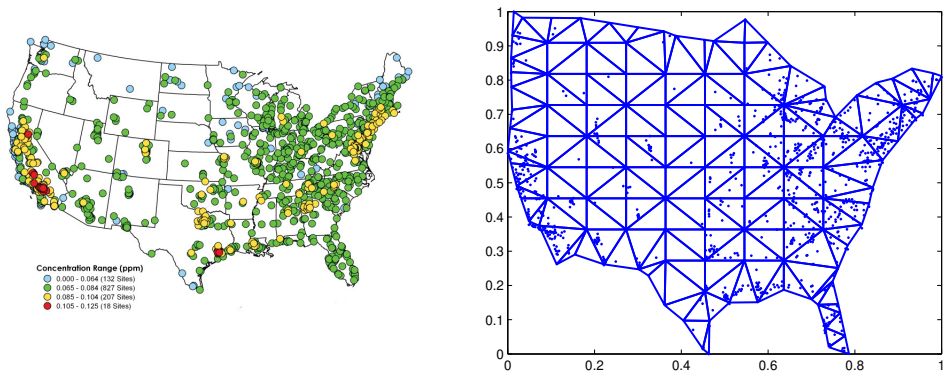
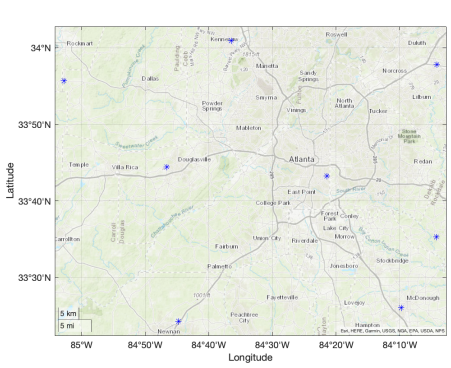
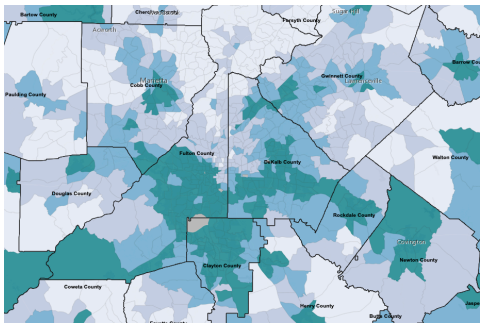


Figure 1: Left: EPA stations measuring Ozone, Right: Triangulation of the USA and the locations of EPA Stations

Despite the EPA station density we observe at the national level, when we zoom into Georgia's Greater Atlanta Area we see that the locations of EPA stations are sparse (see Figure 2a). The proposed MPFCR method addresses a way to include additional spatially distributed data like the environmental justice index data which could help predict ground-level ozone concentrations across our domain of interest. See in Figure 2b.



(a) EPA Stations in the Atlanta Area



(b) The Environmental Justice Index of the Greater Atlanta Area

Figure 2: EPA Stations and Environmental Justice Index in Atlanta

In this paper we implement a novel Modified Functional Principal Component Regression (MFPCR) model. Functional Principal Component Regression (FPCR) is a functional data analysis technique that uses bivariate splines over triangulations [1]. This modeling method has found use when data has temporal or spatial correlation. For example, the current temperature is related to what the temperature was ten minutes ago. Applications of FPCR include environmental science [1], financial modeling [3] and econometrics [4]. The prediction technique combines the ideas of functional data analysis (FDA) and principal component regression (PCR):

- Functional data analysis: Traditional statistical methods are designed to handle scalars, whereas FDA treats an entire function as a basic unit of observation. In the following we will use bivariate splines over triangulations to represent these functions [2].
- Principal component regression: PCR relies on principal component analysis (PCA). PCA is a dimensionality reduction technique used to transform the predictors into a set of linearly uncorrelated variables called principal components. In this investigation, PCA is undergone by finding the covariance matrix corresponding to our data. Then, we compute the eigenvalues and eigenvectors of the covariance matrix to obtain the principal components. The components found through PCA are then used as predictors in a linear regression model.

In this investigation, we use a simulated temperature example to demonstrate the efficacy of the MFPCR approach.

3 Modified Functional Principal Component Regression

In the following we will describe the MFPCR approach. We refer to the motivating problem of predicting ground-level ozone concentrations in our examples. First we set up FPCR [1]. We will develop the following definitions and assumptions:

- **Locations**: Data points where we want to know the variable of interest.
- **Stations**: Data points where we have measurements of the variable of interest.
- The ozone concentration over $\mathcal{D}$ is a random surface, X, with known values at N locations.
- We model the ground-level ozone concentration at a specific location for each hour of each day as a function, denoted by $f(X)$, of the previous day's ozone concentration surface, which we represent by X, across the domain $\mathcal{D}$.
- To estimate X, we employ a bivariate spline space, $S_d^r(\triangle)$, characterized by smoothness r (with $r > 0$) and degree d (where $d > r$), over a triangulation $\triangle$ of $\mathcal{D}$. This method uses measurements from N stations to approximate the ozone concentration surface.

To fit the surface to the data, we will use the Penalized Least Squares (PLS) Method. For PLS, we let $\{(x_i, y_i, f(x_i, y_i)),\ i = 1, \ldots, N\}$ be a scattered data set where *N is a relatively large integer*. Then the penalized least squares method is to find $s_f \in \mathcal{S}$ such that for a positive weight $\rho > 0$:

$$P_\rho(s_f) := \min_{s \in \mathcal{S}} P_\rho(s)$$

where

$$P_\rho(s) := \sum_{i=1}^{N} |s(x_i, y_i) - f(x_i, y_i)|^2 + \rho E(s)$$

and the penalty term $E(s)$ is given to be the following:

$$E(s) := \sum_{T \in \triangle} \int_T \left(s_{xx}^2 + 2s_{xy}^2 + s_{yy}^2\right) dxdy.$$

$E(s)$ is known as the energy functional. We see that $E(s)$ is composed of the Surface Laplacian $s_{xx}^2 + 2s_{xy}^2 + s_{yy}^2$, and is the sum of the areas of all triangles in our domain. We include the energy functional to avoid over-fitting our noisy data as its inclusion penalizes curvature. We think of ρ as a value we can adjust to increase or decrease the amount of noise, that is, how closely we want our data to interpolate our observed values. A larger value for ρ would allow for

our model to fit the data more closely by increasing $E(s)$, thus reducing noise. Contrarily, a low value for ρ would decrease the value of $E(s)$ and our model would fit the noisy data more loosely.

We will use the ozone surface splines as the input to a Functional Linear Model. This is similar to a how we think of a linear model except instead of data points, we are inputting surfaces into the model. Let Y be a real-valued random variable, which is a functional of random surface X, then:

$$Y = f(X) + \epsilon = \langle \alpha, X \rangle + \epsilon \tag{1}$$

where $\mathbb{E}(\epsilon) = 0$. Let α be the solution of the following:

$$\alpha = \arg\min_{g \in H} \mathbb{E}\left[(f(X) + \epsilon - \langle g, X \rangle)^2\right] \tag{2}$$

where $H = L_2(\mathcal{D})$ is a standard Hilbert space of all square integrable functions over $\mathcal{D}$. See the definition of a square integrable function below:

$$\int_{-\infty}^{\infty} |f(x)|^2 \, dx < \infty.$$

We use the infinite-dimensional Hilbert space due to the assurance of functions having desirable properties for numerical methods – the completeness of the Hilbert space, and the space's separability. Additionally, let $\langle f, g \rangle$ denote the standard inner product of f, g on H. The inner product of two functions f and g in the standard Hilbert space is given as follows:

$$\langle f, g \rangle = \int_{-\infty}^{\infty} f(x)g(x) \, dx.$$

Where the previous property allows for the inner-product to be finite. Thus, the functional Y has the following mapping:

$$Y : H = L_2(\mathcal{D}) \to \mathbb{R}$$

Furthermore, the Hilbert space is a normed space. Let $\|f\|$ be the norm of f, (X, Y) be a pair of random variables defined on the same probability space Ω, with X valued in H and $Y = f(X)$ valued in $\mathbf{R}$, and $\mathcal{X} \subset H$ be a given set of random surface distributions.

3.1 Autoregressive Process

To find a solution to the minimization problem (2) we write the equation in terms of the Covariance and Cross Covariance functions.

Covariance:

$$\Gamma g(s) = \int_{s \in \mathcal{D}} \mathbb{E}[X(s)X(t)]g(s)ds, \quad \forall g \in H.$$

Cross Covariance:

$$\Delta f = \int_{s \in \mathcal{D}} \mathbb{E}[X(t)Y]f(t)dt, \quad \forall f \in H.$$

Now, we seek to write our functional Y in terms of the continuous form of the covariance and cross-covariance functions. We use properties of the Hilbert space to do so. Hence we have,

$$\begin{aligned} Y &= \langle \alpha, X \rangle + \epsilon \\ \langle X, x \rangle Y &= \langle X, x \rangle \langle \alpha, X \rangle + \langle X, x \rangle \epsilon \\ \mathbb{E}\left[\langle X, x \rangle Y\right] &= \mathbb{E}\left[\langle X, x \rangle \langle \alpha, X \rangle + \langle X, x \rangle \epsilon\right] \\ \langle \mathbb{E}[XY], x \rangle &= \langle \alpha, \mathbb{E}[\langle X, x \rangle X] \rangle + 0 \\ \Delta(x) &= \langle \alpha, \Gamma(x) \rangle. \end{aligned}$$

In practice, we do not observe the continuous random surface X_i but we only observe the random surface at design points $s_k \in \mathcal{D},\ k = 1, \ldots, N$:

$$\{z_{i,k},\ k = 1, \ldots, N\}.$$

Since we cannot observe the full continuous surface X_i we choose to discretize our covariance and cross-covariance functions to use later in PCA. We approximate X_i by a smooth, fifth-degree bivariate spline denoted S_{X_i} ([2]). Thus, the empirical estimators can be approximated by

$$\widetilde{\Gamma_n}(x) = \frac{1}{n} \sum_{i=1}^{n} \langle S_{X_i}, x \rangle S_{X_i} = \sum_{j=1}^{m} \widetilde{\lambda_j} \langle \widetilde{v_j}, x \rangle \widetilde{v_j}$$

$$\widetilde{\Delta_n}(x) = \frac{1}{n} \sum_{i=1}^{n} \langle S_{X_i}, x \rangle Y_i,$$

where $\widetilde{\lambda_j}$ and $\widetilde{v_j}$ are a pair of eigenvalue and eigenvector of $\widetilde{\Gamma_n}$ and m is the dimension of the spline space $S_d^r(\triangle)$. It then follows that

$$\widetilde{\Delta_n}(x) = \langle \alpha_n, \widetilde{\Gamma_n} x \rangle$$

for some $\alpha_n \in H$.

Assume that the first k_n largest eigenvalues $\widetilde{\lambda_j}, j = 1, \cdots, k_n$ are nonzero. Then the principal component regression estimator of α_n is

$$\widetilde{\alpha}_{PCR} = \sum_{j=1}^{k_n} \frac{\Delta_n(\widetilde{v_j})}{\widetilde{\lambda_j}} \widetilde{v_j}. \tag{3}$$

This method as described so far is the FPCR method from [1].

Now we suggest the modification the the FPCR method described above. The idea is to use co-varying spatially distributed data that can augment the prediction with more precise local information. For example, in ground-level ozone prediction problem, we could use another surface such as the EJI (Environmental Justice Index) over the same domain to augment the predictions. We will call this additional surface W, with known values at M locations. We

use the same bivariate spline space $S_d^r(\triangle)$ to approximate W using the given measurement values at the M locations. Next we can include this surface as an additional principal component to shape the prediction function by first fitting the surface W with a PLS spline S_W. Then using S_W to augment the prediction function as a penalty term $\hat{\lambda}$ with

$$\widetilde{\alpha}_{MPCR} = \sum_{j=1}^{k_n} \frac{\Delta_n(\widetilde{v_j})}{\widetilde{\lambda_j}} \widetilde{v_j} + \hat{\lambda} S_W = \widetilde{\alpha}_{PCR} + \hat{\lambda} S_W. \tag{4}$$

Here we capitalize using bivariate splines over triangulations as both $\widetilde{\alpha}_{PCR}$ and S_W are splines over the same space and thus have the same dimension.

The $\widetilde{\alpha}$ in equations (3) and (4) gives a prediction function for one station in the domain. To create a prediction for each location in the domain, we run the model for each station and then fit the PLS spline through the predictions to get a predicted surface for a single day and time. We can evaluate the predicted surface at any point in the domain to obtain a prediction for that location.

3.2 Computational Method Outline:

For each station in the domain, we want to solve the system which will yield a coefficient vector for a spline S_α such that

$$f(X) \approx \langle S_\alpha, S_X \rangle. \tag{5}$$

The following steps outline the process:

1. Fit each X_i with a PLS spline S_{X_i}.
2. Use S_{X_i} to calculate $\widetilde{\Gamma_n}$ and $\widetilde{\Delta_n}$.
3. Compute the SVD of $\widetilde{\Gamma_n}$.
4. Select appropriate number of non-zero eigenvalues k_n.
5. Compute $\widetilde{\alpha}_{PCR}$ in (3).
6. Fit the surface W with a PLS spline S_W.
7. Compute the augmented the prediction function $\widetilde{\alpha}_{MPCR} = \widetilde{\alpha}_{PCR} + \lambda S_W$ as in (4).

   Note here that (3) returns the coefficients for the prediction spline using FPCR and (4) yields the coefficients for the prediction spline from the MFPCR approach. Once we have the prediction spline we can evaluate it at the last known surface to get the prediction for the next values. For example, to generate a 24 hour prediction for ground-level ozone you would collect the low-level ozone surfaces in $\widetilde{\Gamma_n}$ and the value at the location of interest 24 hours later in $\widetilde{\Delta_n}$. Then solve for the prediction function using

FPCA or MFPCA following the method above. We make the prediction by evaluating (5), i.e.,

$$\underbrace{f(X)}_{\text{The predicted ozone value}} \approx \langle \quad \underbrace{S_\alpha}_{\text{The prediction function}} \quad , \quad \underbrace{S_X}_{\text{The last known ozone value}} \quad \rangle.$$

8. To obtain the the prediction at a given station for a given day and time we take the last know surface X and compute

$$f(X) \approx \langle S_{\alpha_{MPCR}}, S_X \rangle$$

9. Once you have a prediction for each station, fit the PLS spline through the predictions to get a predicted surface that we can evaluate at any location.

4 Simulated Temperature Data

As a proof of concept, we construct a temperature data set to run simulations. We start with a triangulated domain on the unit square, $\mathcal{D}$ (see Figure 3). We simulate data at 1000 locations where we would like to predict the temperature measurements then randomly choose 30 of those locations to be measurement stations. Recall *stations* are data points where we collect the variable of interest while *locations* are data points where we would like to know the variable of interest. Here, the number of points within the square and the amount of triangulation have been preset, and will not be altered during this investigation.

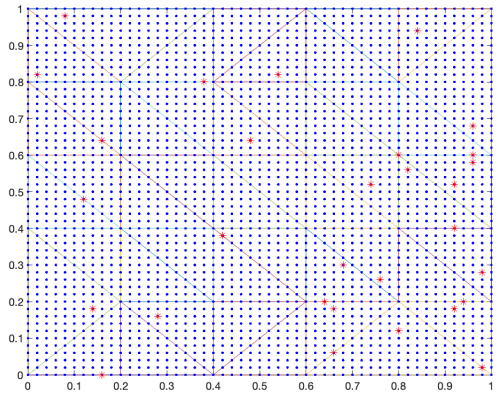


Figure 3: The triangulation of the domain. Each point represents one of the 1000 locations where we want to know the temperature and each of the 30 red asterisks represents a station where temperature is measured.

4.1 Constructing a Temperature Data Set

We seek to generate realistic temperature data set with properties that we can access later. The data used in this investigation is simulated through a combination of "input functions" that mimic attributes that may affect temperature. For example, topological features, latitude (north-south effect), the time of day (hourly variation), and seasonality all influence the present temperature. To model these natural phenomena, we consider the following functions that may influence temperature and include the graphical representation of these functions at the end of this section.

4.2 Topological Features

Temperature is affected by elevation: it may be cooler in the mountains than at sea level. To simulate this in our data creation, consider the following function:

$$f_1(x, y) = \sin\left(2\pi(x^2 + y^2)\right) \tag{6}$$

where x, y denote a coordinate on the triangulated domain. See Figure 4a
Later in this investigation, we develop an experiment concerning principal components by adjusting the topological feature input function to a function less gradual than $f_1(x, y)$:

$$f_2(x) = \begin{cases} 1 & \text{for } x \geq \frac{1}{2}, \\ 0 & \text{for } x < \frac{1}{2}. \end{cases} \tag{7}$$

We implement the function into our model using the following MATLAB code:

```
function y = stepFunction(x)
    y = x >= 1/2;
end
```

4.3 North-South Effect

Assuming the x-axis is the equator, moving north of the x-axis would result on cooler temperatures. We have modeled this phenomena through the following simple function:

$$f(x, y) = y$$

where x, y denote a coordinate on the triangulated domain. See Figure 4b.

4.4 Hourly Variation

When the sun is present in the sky, temperatures tend to be higher. Typically the highest temperature occurs between 3:00pm and 6:00pm, and the lowest

occurs late into the night. To reflect this, consider the following function that leverages the cyclical nature of cosine:

$$f(h) = -\cos\left(\frac{2\pi h}{23}\right)$$

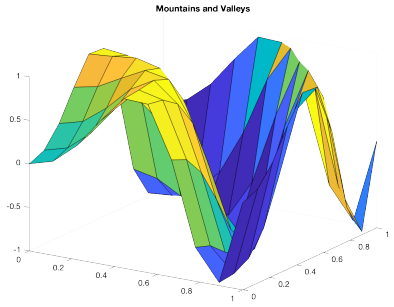
where h denotes the hour of the day, that is, $h \in [0, 24]$. See Figure 4c.

4.5 Seasonality

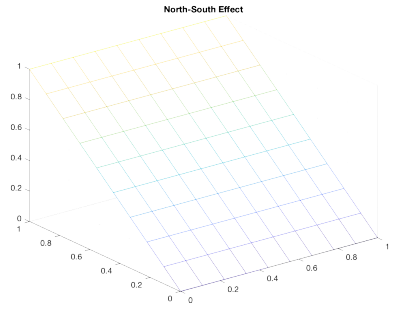
The month of the year may affect the present temperature. The following function simulates this effect for use in generating the data:

$$f(d) = -\cos\left(\frac{2\pi d}{364}\right)$$

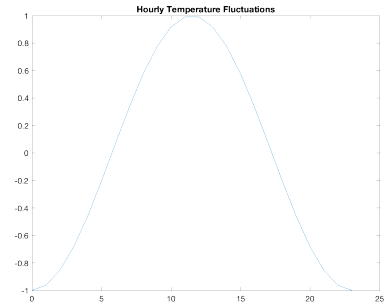
where d denotes the specific day in a full year, that is, $d \in [1, 365]$. See Figure 4d.



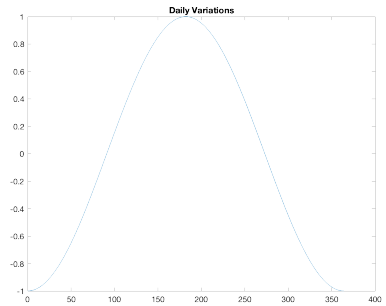
(a) An undulating surface that represents mountains and valleys



(b) Temperature variance from North to South



(c) Temperature Fluctuation by Hour of the day



(d) Temperature Fluctuation Throughout a Year

Figure 4: Composite figure illustrating the various factors affecting temperature: topological features, north-south variance, hourly fluctuations, and seasonality.

4.6 Adding Noise to the Signal

The model in (1) we have a noise term ϵ. The noise ϵ is a real random variable that satisfies $E(\epsilon) = 0$ and $E(X(s)\epsilon) = 0$ for all $s \in \mathcal{D}$ The goal of adding noise is to add a sense of "realism" to the data; perfect smoothness in a function is hardly observed in a natural environment, so the addition of noise logically replicates what we perceive in nature.

There are several types of noise. Generally, they can be classified as systematic or random noise. In this investigation, we implement Gaussian noise through the use of MATLAB's built-in `normrnd` function. The Gaussian Distribution has X normally distributed with mean μ and standard deviation σ:

$$X \sim \mathcal{N}(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

The we add Gaussian noise into the signal with $\mu = 0$ and $\sigma = 1$.

4.7 Combining the Features to Generate the Data

We have combined the previous features to simulate temperature data over a defined amount of days. The following for loop from MATLAB demonstrates this combination to generate data for 121 days of the year:

```
data_hour = zeros(num_locations, 8);
data =[];
for d = 1:121
       for h = 0:23
            data_hour(:,1) = xx(:); % x locations
            data_hour(:,2) = yy(:); % y locations
            data_hour(:,8) = mv(:) + ns(:)
            + -d_const*cos(d*(2*pi/364))-h_const*cos(h*(2*pi/23))+location_ave;
            data_hour(:,4) = d*ones(num_locations,1);
            data_hour(:,5) = h*ones(num_locations,1);
            data_hour(:,6) = [1:num_locations];
            data_hour(:,7) = normrnd(0,1,size( data_hour(:,1)));
            data_hour(:,3) = data_hour(:,8)+data_hour(:,7);
            data =[data; data_hour];
            data_hour = zeros(num_locations, 6);
       end

end
```

The purpose of displaying the code is to show how the various input functions have been incorporated into data generation. In the code, "mv" relays the mountain-value effect, "ns" the north-south effect, and the cosine functions defined earlier are implemented as well. The data generated from the for loop and input functions has the following structure:

X Pos	Y Pos	Temp.	Day	Hour	Label	Noise	Temp. (Noise)
0	0	40.5406	1	0	1	0.5377	40.0030
0	0.1	41.9997	1	0	2	1.8339	40.1658
0	0.2	38.1928	1	0	3	-2.2588	40.4517
0	0.3	41.7010	1	0	4	0.8622	40.8388
0	0.4	41.5661	1	0	5	0.3188	41.2473
0	0.5	40.1953	1	0	6	-1.3077	41.5030
0	0.6	40.9399	1	0	7	-0.4336	41.3735
0	0.7	41.1084	1	0	8	0.3426	40.7658
0	0.8	43.6109	1	0	9	3.5784	40.0325
0	0.9	42.7426	1	0	10	2.7694	39.9732
0	1	39.6531	1	0	11	-1.3499	41.0030
0.1	0	43.1007	1	0	12	3.049	40.0658

Table 1: Sample of Generated Temperature Data with Noise

An example of the generated date is in Table 1. The data generated gives the (x, y) location in the domain, the simulated temperature generated by combining features described in this section, the day, the time, a station label, the noise, and then finally the temperature with the noise. The temperature with noise is what will be used for our simulations in the next section.

5 Simulated Temperature Experiments

This section outlines several experiments using the simulated data in Section 4. The objective of these experiments is to assess the robustness of the MFPCR approach by changing features of the problem. We will analyze the MFPCR approach give some insight into the value of incorporating the additional term in when calculating the prediction function.

5.1 Fixed and Adjusted Attributes of the Simulations

Some features of the analysis will remain unaffected, while others are altered. For example, we will fix the triangulation and use the spline basis with $d = 5$ for optional approximation order [2]. View more information regarding these attributes in the table below:

Fix	Adjust
Number of prediction days	Mountain-Valley Input Function
Spline Basis	Gaussian Noise Level
Grid	Number of Eigenvalues
Domain	Weight

Table 2: Fixed and Adjusted Attributes

5.2 Principal Component Analysis

Since PCA must be performed to develop our MFPCR model, we first take a moment to examine how altering input functions in our data will affect the principal components used in our predictive model.

First, we assess how many principal components we wish to introduce in the model. A common method for determining the number of principal components to be retained can be performed visually using a "Scree" plot and a "Cumulative Variance Explained" plot.

A scree plot is a simple line segment plot that shows the eigenvalues for each individual principal component. A scree plot shows the eigenvalues on the y-axis and the index of the eigenvalue on the x-axis. The magnitude of the eigenvalues are arranged in descending order within the plot, a by-product of orthogonal diagonalization, and thus scree plots always display a downward curve. Most scree plots look similar in shape, starting high on the left, falling quickly, and then flattening out as the number of factors grow.

Below, we outline the generic process to obtain the cumulative variance explained plot in this investigation:

1. **Obtain Eigenvalues:** There are many ways to retrieve the eigenvalues of data. For our purposes, let Σ be the covariance matrix derived from our temperature data. The eigenvalue decomposition of Σ is given by:

   $$\Sigma = Q\Lambda Q^{-1}$$

   where Λ is a diagonal matrix with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ on the diagonal, and Q is an orthogonal matrix of the eigenvectors which corresponds to the eigenvalues. These eigenvalues are sorted in descending order of magnitude, as mentioned previously.

2. **Variance Explained by Each Component:** Each eigenvalue λ_i represents the variance explained by the i-th principal component. The total variance explained by all components is:

   $$V_{\text{total}} = \sum_{i=1}^{n} \lambda_i$$

3. **Cumulative Sum of Variance:** The cumulative variance explained by the first k components is the sum of the first k eigenvalues:

   $$V_{\text{cumulative}}(k) = \sum_{i=1}^{k} \lambda_i$$

4. **Normalization:** The fraction of the total variance explained by the first k components is obtained by dividing the cumulative variance by the total variance:

   $$P(k) = \frac{V_{\text{cumulative}}(k)}{V_{\text{total}}}$$

5. **Cumulative Variance Table:** A table can be constructed to display the number of components k, the cumulative variance $V_{\text{cumulative}}(k)$, and the proportion $P(k)$ for each k, generally ranging from 1 to n, where n is the total number of components.

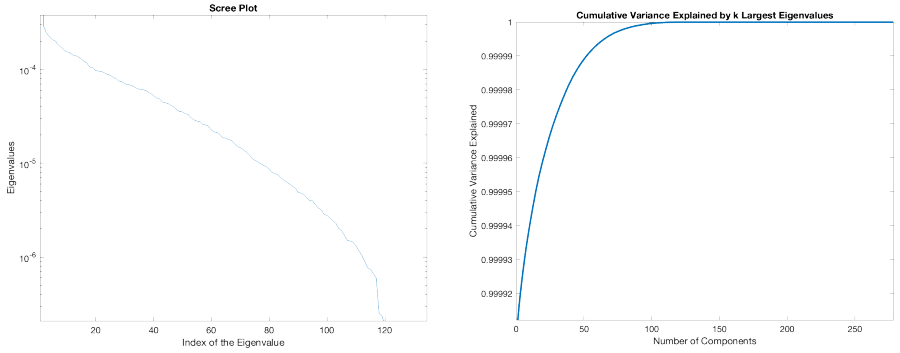
   The table format is as follows:

Number of Components (k)	Cumulative Variance	Percentage of Variance Explained
1	$V_{\text{cumulative}}(1)$	$P(1) \times 100\%$
2	$V_{\text{cumulative}}(2)$	$P(2) \times 100\%$
$\vdots$	$\vdots$	$\vdots$
n	$V_{\text{cumulative}}(n)$	$P(n) \times 100\%$

Table 3: Cumulative variance explained by the principal components.

   This table elucidates the incremental variance each principal component contributes and the total variance captured up to that point. It assists in determining the optimal number of principal components to retain, based on the cumulative variance explained. Using the previous table to guide the creation of our plot, the number of components are plotted on the x-axis and percentage of variance explained is plotted on the y-axis.

The code with the mountain-valley input function gives the following scree plot and cumulative variance explained plot:



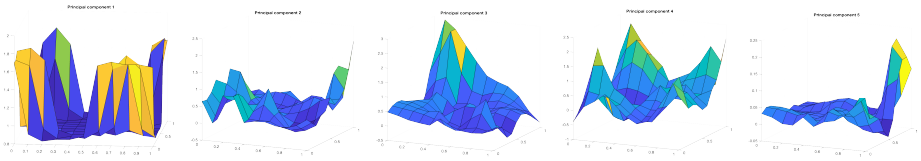
(a) Scree Plot (b) Cumulative Variance Explained

Figure 5: Scree plot and cumulative variance explained by principal components with the inclusion of the mountain-valley function $f_1(x, y)$.

The plot shows two sharp "elbows." One occurs around 5 eigenvalues, and the other around 120 eigenvalues. We can see that in Figure 5b, almost 99% of the data is explained within the first few eigenvalues, so we will use the first 5 principal components to use in our PCA.

Recall that the default mountain-value function in (6) (see Figure 4a).

The five principal components have the following appearance in the model with data generated using $f_1(x, y)$. We can see some of the effects of using f_1 to create the data in the fourth principle component. See Figure 6d.



(a) PC1 (b) PC2 (c) PC3 (d) PC4 (e) PC5

Figure 6: Principal Components in the Mountain Valley Model

Next, keeping all else constant, we alter the topological input function $f_1(x, y)$ into something that we expect to be more pronounced in the plots of the principal components. Next, we use the step function f_2 (7) as the topological input function, i.e. we consider a sheer cliff with a steep drop-off seen in Figure 7.

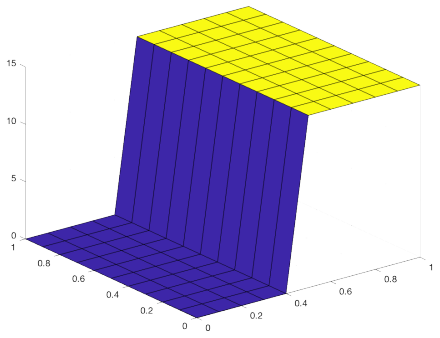
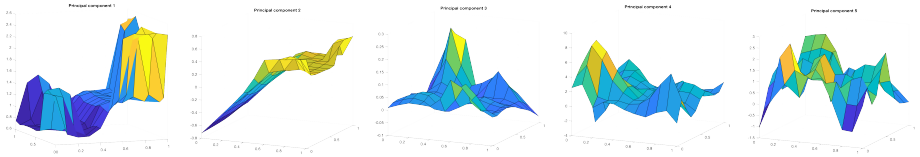


Figure 7: Step Function

When implemented into the code, we choose to multiply the step function by 50 to ensure clear results visually. Figure 8 shows the plotted principal components generated by using step function in (7) instead of the mountain-valley function in (6).

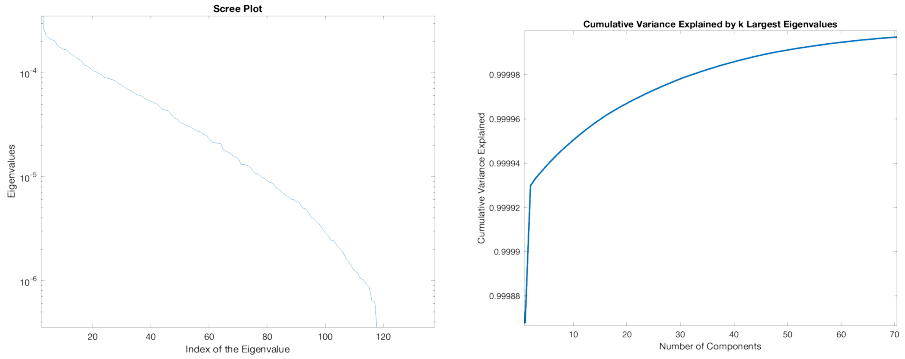


(a) Step PC1 (b) Step PC2 (c) Step PC3 (d) Step PC4 (e) Step PC5

Figure 8: Stepwise Principal Components

Here, we notice that the steepness of the new input function $f_2(x, y)$ has impacted the principal components visually. Specifically in PC1 and PC2, we see a sharp increase in the middle of the plots.

Lastly, we include a scree plot and cumulative variance plot for the data generated with the inclusion of the step function:



(a) Scree Plot (b) Cumulative Variance Explained

Figure 9: Scree plot and cumulative variance explained by principal components with the inclusion of the step function $f_2(x, y)$.

Here we see that the scree plot above (9a) shows a steeper drop in eigenvalues, as opposed to (5a). Additionally, there is now a sharp kink in the cumulative variance explained plot (9b), which contrasts the smoothness of the original cumulative variance plot (5b).

5.3 Assessing the Prediction Function through MSE Comparison Amongst Adjusted Attributes

The purpose of this section is to determine how 'robust' our prediction function is when we change features of the problem. We want our model's predictive capability to remain constant through several alterations. We use mean squared error (MSE) as our metric for comparison. MSE is generally defined as the

following:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Calculating MSE requires information about the true value of our data Y_i and our predicted values, $\hat{Y}_i$. Squaring the difference between the two terms penalizes greater differences, and is a common method of analyzing the efficacy of predictive models. Here, our predicted $\hat{Y}_i$ is found using our MFPCR model. To evaluate the predictive efficacy of the MFPCR model, we will compare the MSE across one week when adjusting some attributes of the model. The same seven days are used for comparison, that is, April 10th to April 17th.

5.3.1 Base Case: Mountain Valley

We start with the Mountain Valley input function (4a) as our base case. We will list some of the features of this base case below before making changes later in the experiment:

Feature	**Base Case Setting**
Input Function	$f_1(x, y) = \sin(2\pi(x^2 + y^2))$
Weight	S(Eig+1,Eig+1)
Noise	Gaussian Noise
Number of Eigenvalues	5

Table 4: Base Case Features of the Model

The base case 7-day simulation data is given below:

Day of Week	MSE
1	0.0093
2	0.0090
3	0.0097
4	0.0105
5	0.0090
6	0.0088
7	0.0088

Table 5: MSE Table for the Base Case Simulation

The following subsections will contain similar tables which can be referenced for comparison. While tables will be shown throughout this section for each altered attribute, in Table 11 all of the tables have been combined for ease of comparison.

5.3.2 Altered Input Function

A previous section (4) demonstrated how the inclusion of a step function will affect the principal components. We now seek to understand how changing the mountain-valley input function to the step function will affect the MSE of our prediction function. The following table demonstrates this change in the MSEs:

Day of Week	MSE
1	0.0054
2	0.0046
3	0.0051
4	0.0062
5	0.0052
6	0.0056
7	0.0060

Table 6: MSEs for the Step Function Simulation

Here we see that the altered input function lead to lower MSEs for each day when compared to the base case simulation.

5.3.3 Weight Adjustments

The weight, ρ, will be set to 10^{-3} for this experiment. The following table shows the 7 day prediction with the new weight:

Day of Week	MSE
1	0.0093
2	0.0086
3	0.0100
4	0.0097
5	0.0083
6	0.0087
7	0.0085

Table 7: MSEs for the Simulation with Altered Weight

The previous table shows MSEs that range from 0.0083 to 0.0100, similar to the range of the base case simulation.

5.3.4 Noise

The noise we implemented earlier in this investigation will now be scaled by a factor of 5. We expect that the data with greater amount of noise will result in a higher MSE than our default. Within the data generation for loop, the new code has the following form:

```
data_hour(:,7) = 5 * normrnd(0,1,size( data_hour(:,1)));
```

The corresponding MSEs for the 7-day period are as follows:

Day of Week	MSE
1	0.0076
2	0.0128
3	0.0071
4	0.0177
5	0.0117
6	0.0078
7	0.0121

Table 8: MSEs for the Increased Noise Simulation

The MSE for each day varies the most when increasing the amount of noise. In day four of this simulation, we obtained our largest MSE (0.0177) out of the data simulations so far.

5.3.5 Number of Eigenvalues

In this experiment, we have set the number of eigenvalues to 2 instead of 5. In theory, this should increase our MSE since fewer principal components will be implemented into the model:

Day of Week	MSE
1	0.0098
2	0.0096
3	0.0081
4	0.0094
5	0.0106
6	0.0077
7	0.0106

Table 9: MSEs for the Simulation with Fewer Eigenvalues

Reducing the number of eigenvalues did not lead large deviations from our base case scenario. Recall the cumulative variance explained plot (5b), where only selecting a couple of eigenvalues explained about 99.99% of variance of our simulated temperature data.

5.3.6 Overall MSE Comparison

We include a table which shows the altered settings of our model for reference:

Feature	**Base Case Setting**	**Alteration**
Input Function	$f_1(x,y) = \sin(2\pi(x^2+y^2))$	Step Function
Weight	S(Eig+1,Eig+1)	10^{-3}
Noise	Gaussian Noise	$5 *$ Gaussian Noise
Number of Eigenvalues	5	2

Table 10: Base Case Features of the Model and Alterations

The table below shows each of the previous alterations to the problem that have been made and the corresponding MSEs across the 7 day span:

Day of week	Base Case	Step Function	Weight	Eigenvalues	Noise
1	0.0093	0.0054	0.0093	0.0098	0.0076
2	0.0090	0.0046	0.0086	0.0096	0.0128
3	0.0097	0.0051	0.0100	0.0081	0.0071
4	0.0105	0.0062	0.0097	0.0094	0.0177
5	0.0090	0.0052	0.0083	0.0106	0.0117
6	0.0088	0.0056	0.0087	0.0077	0.0078
7	0.0088	0.0060	0.0085	0.0106	0.0121

Table 11: Comparison of MSE across different Simulations

Next, this table shows the average MSEs for each day of the week:

	Base Case	Step Function	Weight	Eigenvalues	Noise
Average	0.009300	0.005443	0.009014	0.009400	0.011000

Table 12: Average MSE Values for Different Simulation Parameters

Using the information in the previous two tables, we see that the step function MSE per day is almost half that of the base case MSE. This highlights the simplicity of the step function and the graphical complexity of the trigonometric mountain-valley function. Additionally, decreasing the amount of eigenvalues used in the model did not lead to extreme deviation from the base-case scenario. Lastly, increasing the amount of Gaussian noise led to a greater MSE on average as expected.

6 Comparing the Predictions

In this section, we compare MFPCR and FPCR. Also, we update the model to predict at more locations and use the information to generate a more accurate prediction to the true temperature.

6.1 MFPCR vs. FPCR

We choose to predict one week's temperature using MFPCR and FPCR. We will compare the prediction functions' respective MSEs. Using April 17th, the same prediction location, and input data will allow for the greater comparability. We consider the MSEs for 7 days of simulation outlined in the table below:

Day of Week	MSE for MFPCR	MSE for FPCR
1	0.0093	0.0146
2	0.0090	0.0131
3	0.0097	0.0132
4	0.0105	0.0062
5	0.0090	0.0041
6	0.0088	0.0059
7	0.0088	0.0101
Average	0.0093	0.0096

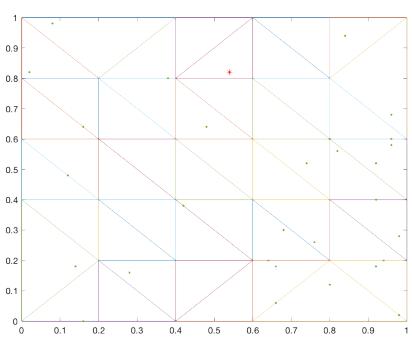
Table 13: MSE Model Comparison

We see that the two models are comparable in MSEs overall; however, the MSE for the FPCR model tends to vary more across each day than the MSE for MFPCR.

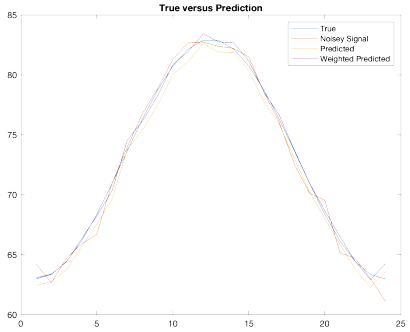
6.2 Comparing Prediction Functions with Updated Station Data

In the final section we plot the prediction functions with the true temperature values and adjust the code to predict at multiple locations. In our temperature example, we leverage the fact that we have "ground truth." Since we have generated the true temperature data at all locations, we can visually juxtapose our prediction with the true values to assess the accuracy of our fit. Additionally, we have made updates to the code to perform prediction at more stations which allows fine-tuned predictions at more locations.

Our new code outputs the prediction location and a graph which compares a few features of the model in a one-dimensional format. The following images are an example of an output when our model uses information from 30 locations:



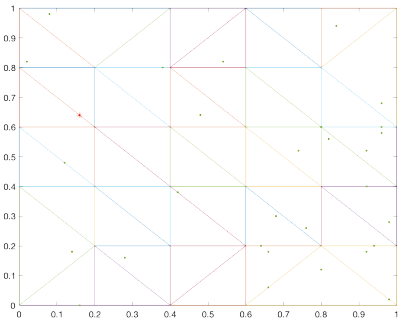
(a) Triangulation and sample points



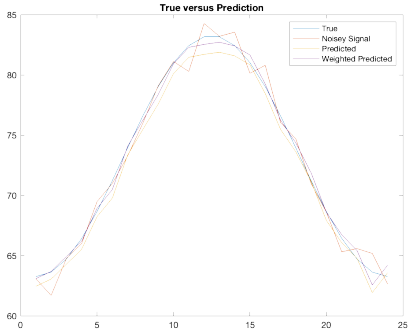
(b) True vs. Predicted Values

Figure 10: Visual comparison of the triangulated domain and the true versus predicted temperature values at prediction location number 12

The code uses other locations for the prediction. Consider location 23:



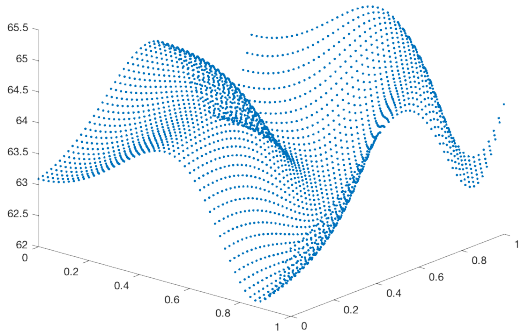
(a) Triangulation and sample points



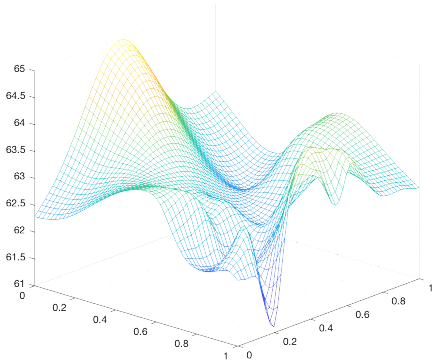
(b) True vs. Predicted Values

Figure 11: Visual comparison of the triangulated domain and the true versus predicted temperature values at prediction location number 23

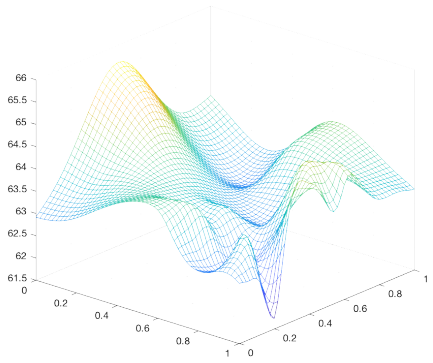
Using 30 locations, our predictive surface more closely interpolates the true data. We made adjustments to the model when we noticed that the prediction was continually under-estimating the true temperature. See the following array of plots which demonstrates the improvement made in prediction:



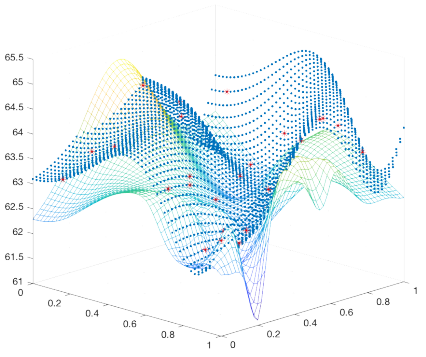
(a) True temperature



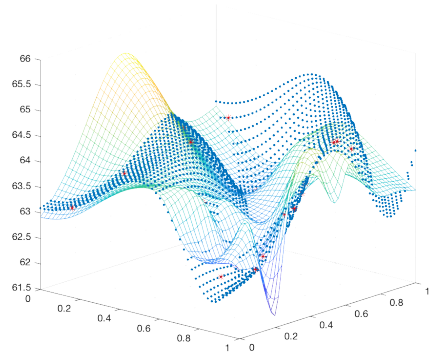
(b) Surface of predicted values using FPCR



(c) Surface of predicted values using MFPCR



(d) FPCR: Surface of predicted values compared with true temperature values



(e) MFPCR: Surface of predicted values compared with true temperature values

Figure 12: Visual comparison of predictive models with true temperature values

7 Conclusion and Extension to the Ground-Level Ozone Application

The temperature example provides a controlled environment for experimenting with MFPCR and FPCR. We simulated the data to bypass any abnormalities that we may have encountered in real ground-level ozone data, and we constructed an ideal domain. We use this toy problem as a way to test a novel MFPCR method with the inclusion of a penalty term. The implementation of the step function highlights the explanatory capabilities of PCA, and the effectiveness of including additional surfaces in our model in the overall prediction. We assessed the robustness of the model by altering several key features of the problem. When the weight, the amount of eigenvalues, and the amount of noise were altered, the model still showed similar error in the base case scenario.

The findings show that the MFPCR method could be used to model ground-level ozone. We propose using EJI data as the additional input into the new method to improve accuracy in ground-level ozone estimation at a more local level. These findings are promising, and highlight the efficacy of modifying FPCR to MFPCR to improve prediction.

References

[1] B. Ettinger, S. Guillas, M. J. Lai, Bivariate Splines for Functional Regression Models with Application to Ozone Forecasting,*Environmetrics*, 23 (2012) pp. 317-328. `http://alpha.math.uga.edu/~mjlai/papers/EGL12.pdf`

[2] Lai, Ming-Jun and L. L. Schumaker, *Spline Functions over Triangulation*, Cambridge University Press, Cambridge, U.K., 2007.

[3] Wang, Zhiliang; Sun, Yalin; and Li, Peng, *Functional Principal Components Analysis of Shanghai Stock Exchange 50 Index*, Discrete Dynamics in Nature and Society, Hindawi Publishing Corporation, 2014.

[4] Frončková, Kateřina and Pražák, Pavel, *Functional Data Analysis in Econometrics*, Ph.D. dissertation, University of Hradec Králové, Hradec Králové, Czech Republic, 2021.

[5] Wang, Wenhao; Liu, Xiong; Bi, Jianzhao; and Liu, Yang, *A Machine Learning Model to Estimate Ground-Level Ozone Concentrations in California Using TROPOMI Data and High-Resolution Meteorology*, Environment International, Elsevier, 2022.