Forecasting Hourly Energy Consumption: Is Functional Data Analysis Worth the Complexity?

Rustem Kaliyev
Supervisor: Rustem Takhanov
Second Reader: Zhenisbek Assylbekov

May 2025

Abstract

This thesis compares an **autoregressive Hilbertian** model of order 1 (ARH(1)) with a classical two-step **PCA+VAR(1)** approach for forecasting daily electricity consumption curves. We use a year-long subset (Oct. 1, 2023 to Sept. 30, 2024) of the PJM Hourly Energy Consumption data across five areas. We additionally benchmark modern forecasting models including **NHITS**, a deep neural network, a recurrent **LSTM** model, and Nixtla's **TimeGPT** (a pre-trained time-series Transformer). Each method predicts the next day's 24-hour load curve from the current day's curve, except TimeGPT, which generates forecasts based on the full historical context available up to each forecast origin. Forecast accuracy is evaluated via mean absolute error (MAE), mean squared error (MSE), and symmetric mean absolute percentage error (sMAPE). The ARH(1) and PCA+VAR(1) models show very similar performance and outperform the other methods on average. This was a surprising finding, suggesting that in this well-behaved data scenario, the functional approach offers little advantage over a simple PCA-based multivariate approach. To investigate further, we conduct a simplified experiment forecasting the next day's *average* consumption using **functional PCR (FPCR)** vs. standard **PCR**. The results again show nearly identical performance from both methods. We hypothesize that when the functional data are densely and regularly sampled with no missing values, *functional principal component analysis (FPCA) is essentially equivalent to ordinary PCA*, and thus ARH(1) offers no clear improvement over PCA+VAR(1). We formalize this hypothesis and discuss conditions under which fully functional methods may still offer advantages (e.g. irregular sampling or complex functional structure). Finally, we propose directions for theoretical work to rigorously explain the observed equivalence.

1 Introduction

Forecasting electricity demand is crucial for power grid reliability and economic dispatch. In regional power systems like PJM (a large U.S. regional transmission organization), system operators forecast hourly load curves for day-ahead planning. These daily load profiles can be viewed as *functional time series*: each day's consumption is a continuous curve over the 24-hour period. **Functional data analysis (FDA)** provides tools to model such data by treating each curve as a single observation in a function space. FDA

can capture the smoothness and intra-day correlation structure of the curves and has been successfully applied in domains like demographic forecasting and environmental data.

Traditional approaches to forecast a time series of curves often involve two main strategies. The first is a *functional* modeling approach. In this framework, we consider the process $\{Y_t(u) : u \in \mathcal{T}\}$ (with $\mathcal{T}$ the time-of-day domain, e.g. $[0, 24]$ hours) as a single time series of functions. One popular model is the **functional autoregression** of order 1, known as FAR(1) or ARH(1) (autoregressive Hilbertian model of order 1). It can be written as

$$Y_{t+1}(u) = \Psi\{Y_t\}(u) + \varepsilon_{t+1}(u),$$

where Ψ is a linear operator on the function space (analogous to a regression coefficient in infinite dimensions), and $\varepsilon_{t+1}(u)$ is a random error function. Under suitable assumptions, Ψ can be represented in terms of the orthonormal *functional principal components* (FPCs) of the process. In particular, if $\{\phi_k(u)\}_{k\geq 1}$ are the eigenfunctions of the covariance operator of Y_t, then one can expand $Y_t(u) = \sum_{k\geq 1} \xi_{t,k}\, \phi_k(u)$. In practice we truncate to the first d components, and the ARH(1) model then reduces to

$$\boldsymbol{\xi}_{t+1} = \mathbf{R}\, \boldsymbol{\xi}_t + \boldsymbol{\eta}_{t+1}$$

where $\boldsymbol{\xi}_t = [\xi_{t,1}, \ldots, \xi_{t,d}]$, and $\boldsymbol{\eta}_t = [\eta_{t,1}, \ldots, \eta_{t,d}]$ is a noise vector. This is essentially the approach of Aue et al. [2015]. One advantage of the functional approach is that it retains the interpretation in the space of functions and can, in principle, use smoothing techniques or basis expansions to handle noise and ensure smooth forecasts.

The second strategy is a more classical multivariate approach: perform **principal component analysis (PCA)** on the discretized curves to reduce dimensionality, then fit a **vector autoregressive model of order 1 (VAR(1))** to the principal component scores. We refer to this two-step method as **PCA+VAR(1)**. Conceptually, this approach should be very similar to the truncated ARH(1) method described above, since both ultimately rely on the leading principal components of the daily curves to capture dynamics. The difference is mostly in estimation: ARH(1) estimates the operator Ψ in the function space, whereas PCA+VAR explicitly estimates a finite-dimensional VAR on the selected principal component features. In theory, if the ARH(1) model is correct and one includes enough principal components, the two approaches should yield equivalent one-step forecasts. However, in practice one might expect some differences due to how the models are regularized or if the functional approach smooths the data. One of the goals of this thesis is to empirically compare these two approaches on real data and see if the more complex functional modeling provides any tangible benefit over the simpler PCA+VAR method.

Recently, there has been growing interest in applying advanced machine learning models to time series forecasting, including forecasting of high-dimensional or functional data. We include in our comparison two state-of-the-art learning approaches: **NHITS** and **LSTM**. **NHITS** (Neural Hierarchical Interpolation for Time Series) is a modern deep learning architecture introduced by Challu et al. [2023]. It is a pure neural network approach that builds on the success of N-BEATS [Oreshkin et al., 2020], employing multi-rate hierarchical interpolation and residual links to capture time series patterns. NHITS has demonstrated strong performance on many forecasting competitions and represents a cutting-edge univariate forecasting method, extended here to the multivariate context by independently fitting to each time series. The **LSTM** (long short-term memory) model is a recurrent neural network capable of learning temporal dependencies. Hochreiter and

Schmidhuber [1997] introduced LSTMs to address long-term dependence issues in RNNs, and they have since been widely used for sequence prediction problems. We use an LSTM model that takes the past 24 hours and learns to predict the next day's sequence. Unlike ARH or VAR, LSTMs can, in principle, capture non-linear patterns in the data.

Finally, we include **TimeGPT**, a recently released foundation model for time series forecasting by Nixtla. TimeGPT is essentially a large-scale pre-trained Transformer model that can be fine-tuned or used directly for forecasting a variety of time series.[1] According to Nixtla's documentation, TimeGPT uses a Transformer encoder-decoder architecture with multi-head self-attention to capture complex patterns in time series. The model is trained on a massive collection of time series data and is offered as a service where users input their time series and receive forecasts (similar in spirit to how GPT models are used for language). We treat TimeGPT as a black-box benchmark to see how a general-purpose learned model compares to our more specialized statistical models on this particular task. It is expected that TimeGPT should be able to model weekly seasonality and other patterns if given enough context, but it might not necessarily outperform simpler models on a one-day-ahead forecasting task, despite being given the full historical context.

In summary, our comparison spans a spectrum from classical statistical models (ARH(1), PCA+VAR) to modern deep learning (NHITS, LSTM) and cutting-edge pre-trained models (TimeGPT). We aim to answer the following questions:

1. How do the functional and multivariate approaches—ARH and PCA+VAR—compare in forecasting accuracy for daily load curves?

2. Do machine learning models like NHITS, LSTM, and TimeGPT offer improvements over these statistical approaches in this context?

3. If the functional approach (ARH) does not outperform the simpler approach (PCA + VAR), what might be the reasons?

We address these questions through a careful experimental study described in the following sections.

2 Data and Preprocessing

We use data from the PJM regional electricity market, specifically the Hourly Load data for five zones over a one-year period. PJM publishes zonal load (electricity consumption) at an hourly frequency. We extracted data from October 1, 2023 through September 30, 2024 for the following five areas: "AP", "DOM", "JC", "PN", and "RTO". Each area corresponds to a different region or utility within PJM (for example, DOM refers to Dominion Virginia Power's zone, and RTO represents the entire PJM region aggregated). The choice of these five series was motivated by their availability and use in previous studies. Each series consists of 24 hourly observations per day. We treat each day's 24-hour load profile for a given zone as one functional observation $Y_t(u)$, where u ranges from hour 0 to hour 23 of the day. Figure 1 illustrates examples of daily load curves for each zone.

[1] `https://www.r-bloggers.com/2025/02/a-first-look-at-timegpt-using-nixtlar-2`

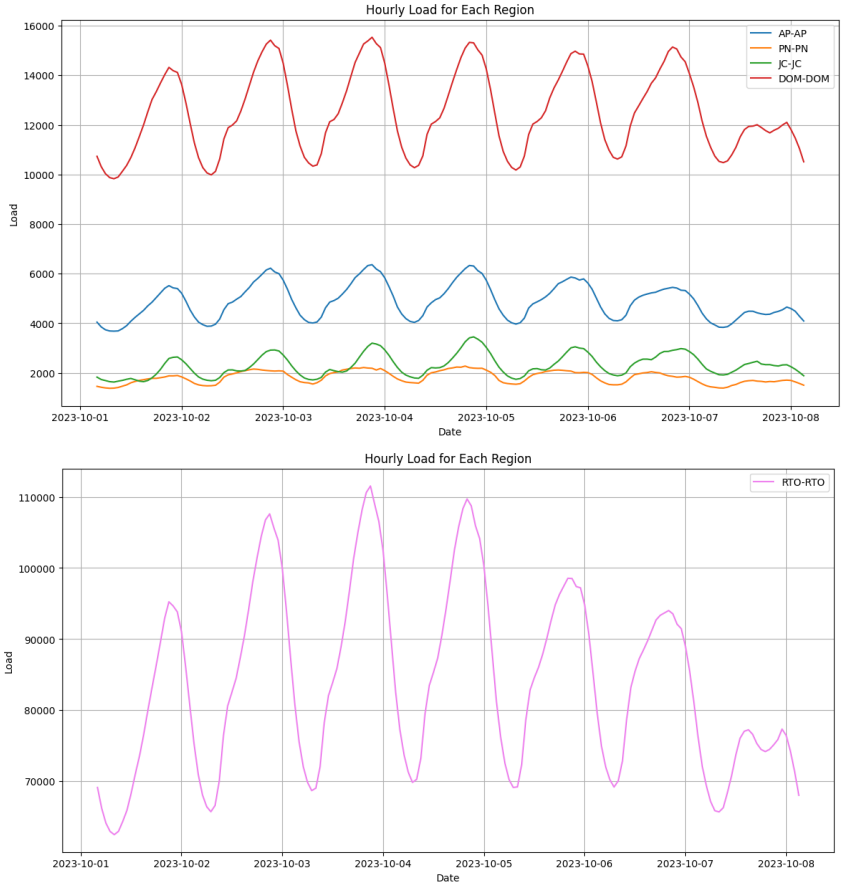

Figure 1: Example of one week load in each zone

Typically, these curves exhibit a clear daily pattern with peaks in the late afternoon and lows in the early morning. They also show intra-week patterns (e.g. weekdays vs weekends) and seasonal effects (summer vs winter demand differences), but in this one-year span with daily forecasting, we primarily focus on day-to-day dynamics.

For preprocessing, we aligned all series to have the same time index (each day from 2023-10-01 to 2024-09-30). There were no missing hourly values in this dataset (which is ideal for functional analysis). We did not apply any additional smoothing or interpolation, since the data are already essentially curves sampled at regular hourly intervals. In FDA terms, our functional observations are *discretized* on an equally spaced grid of 24 points. We treat these as realizations of an underlying continuous load function over the 24-hour period.

The data was then split into training and testing sets for modeling. We used the period from October 1, 2023, 04:00 up to July 20, 2024, 03:00 as the training set, and reserved the rest as the test set for evaluating one-day-ahead forecasts. In total, the training set contained on the order of 293 daily curves per zone, and the test set around 73 daily curves. The one-day-ahead forecasting task is defined as follows: use the data of the current day to forecast the 24 hours of the next day. All models were retrained or updated only with information up to the forecast origin (except TimeGPT, which is pre-trained). For the statistical models (ARH(1), PCA+VAR(1)), this effectively means we fit the models once on the training set and generate one-step forecasts recursively for each day in the test. For the learning models (NHITS, LSTM), we trained the networks on the training period and similarly produced iterative one-day-ahead forecasts.

3 Methodology and Models

3.1 ARH(1) Functional Autoregressive Model

The ARH(1) model was implemented following the approach of Aue et al. [2015] and the classical theory in Bosq [2000]. In practice, implementing ARH(1) involves a few steps. First, we perform a functional principal components analysis (FPCA) on the training set of curves

$$Y_1(u), \ldots, Y_N(u).$$

This yields an estimated mean curve $\hat{\mu}(u)$ and a set of empirical eigenfunctions

$$\{\hat{\phi}_1(u), \ldots, \hat{\phi}_d(u)\}$$

of the sample covariance operator

$$C_Y(f) = \int (f(u) - \hat{\mu}(u))(Y_t(u) - \hat{\mu}(u))du$$

(averaged over training days). We choose a truncation level d that captures most of the variation in the data — in our case we chose $d = 5$ principal components for each zone, which accounted for over 99.8% of the variance in each training set. Let

$$\hat{\xi}_{t,k} = \langle Y_t - \hat{\mu},\ \hat{\phi}_k \rangle$$

be the k-th PC score of day t. The ARH(1) model assumes

$$\boldsymbol{\xi}_{t+1} = \mathbf{R}\,\boldsymbol{\xi}_t + \boldsymbol{\eta}_{t+1},$$

where $\boldsymbol{\xi}_t = (\xi_{t,1}, \ldots, \xi_{t,d})^\top$ and $\mathbf{R}$ is a $d \times d$ coefficient matrix estimated by least squares regression of $\boldsymbol{\xi}_{t+1}$ on $\boldsymbol{\xi}_t$ over the training days $t = 1, \ldots, N-1$. Forecasts are constructed as:

$$\hat{\boldsymbol{\xi}}_{N+1} = \mathbf{R}\,\hat{\boldsymbol{\xi}}_N,$$

and the forecast curve is

$$\hat{Y}_{N+1}(u) = \hat{\mu}(u) + \sum_{k=1}^{d} \hat{\xi}_{N+1,k}\,\hat{\phi}_k(u).$$

We note that this implementation (FPCA followed by VAR(1)) is essentially the algorithm described by Aue et al. [2015] (see their Algorithm 1) for functional forecasting. We will later compare this with the PCA+VAR approach.

One important consideration was selecting the number of components d. We used a simple heuristic of variance explained (around 99.8%), which gave $d = 5$ for each zone.

For simplicity, we assumed $p = 1$ was sufficient (i.e., only yesterday's curve is needed to predict today's, which is reasonable for daily load with strong daily autocorrelation) and focused on comparing d-dimensional ARH(1) vs. VAR(1). Alternatively, one could use an automatic selection criterion such as those proposed by Kokoszka and Reimherr [2013] or others to check if an ARH(p) with $p > 1$ is needed.

All functional analysis was carried out using the scikit-fda package in `Python` and custom scripts for FPCA and forecasting.

3.2 PCA + VAR(1) Model

The PCA+VAR(1) approach operates on the *discretized data* directly as a multivariate vector. For each day t and zone, we have a 24-dimensional vector of loads

$$[y_{t,0}, y_{t,1}, \ldots, y_{t,23}] \qquad \text{(hour 0 through 23)}.$$

We perform a standard PCA on the collection of these 24-dimensional daily vectors (after mean-centering them). In practice, this PCA on discrete data is almost identical to the FPCA described above — in fact, the eigenvectors from this PCA are proportional to the eigenfunctions obtained in FPCA, with differences only due to scaling by the grid size [Crainiceanu et al., 2024, Chapter 3]. In our data, because we have equal hourly spacing and no missing values, the discrete PCA essentially recovers the same modes of variation as FPCA.[2]

We again choose $d = 5$ principal components for each zone's data. Let $z_{t,k}$ denote the k-th principal component score of day t from this multivariate PCA (so $z_{t,k} = \sum_{h=0}^{23} y_{t,h} v_k(h)$ where v_k is the k-th eigenvector of the sample covariance matrix of the 24-dimensional data). We then fit a standard vector autoregressive model of order 1—VAR(1)—to the d-dimensional time series $\{\mathbf{z}_t = (z_{t,1}, \ldots, z_{t,d})^\top\}$. That is, we estimate

$$\mathbf{z}_{t+1} = \mathbf{A}\,\mathbf{z}_t + \mathbf{e}_{t+1}$$

for a $d \times d$ matrix $\mathbf{A}$. We allow $\mathbf{A}$ to be a full matrix, so this VAR(1) can capture potential interactions between the principal components. We used the `vars` package in R to fit the VAR models by ordinary least squares (given N training observations and $d = 5$). For most zones, the off-diagonal elements of $\mathbf{A}$ were small, indicating that the leading principal components evolve largely independently (consistent with uncorrelated scores in PCA).

Forecasting with the PCA+VAR model is straightforward: given the last observed day N, we compute its PC score vector $\mathbf{z}_N$, then the one-step forecast of the score vector is

$$\hat{\mathbf{z}}_{N+1} = \hat{\mathbf{A}}\,\mathbf{z}_N.$$

We then reconstruct the predicted next-day curve as

$$\hat{\mathbf{y}}_{N+1} = \bar{\mathbf{y}} + \sum_{k=1}^{d} \hat{z}_{N+1,k}\,\mathbf{v}_k,$$

where $\bar{\mathbf{y}}$ is the mean daily vector and $\mathbf{v}_k$ the k-th principal component (eigenvector). This yields the 24-hour forecast for day $N + 1$. This process is essentially identical to the functional approach's forecast (replace $\mathbf{A}$ with diagonal $\mathbf{R}$ if we had constrained it). Indeed, from a theoretical perspective, Panaretos and Tavakoli [2013] have shown that if the true process is FAR(1) and one uses a sufficiently rich basis, the finite-dimensional score process will follow a VAR(1) that approximates the functional model. Our implementation of PCA+VAR(1) serves as a direct comparison to ARH(1) to test this theory empirically.

It is worth noting that the PCA+VAR method does not explicitly enforce smoothness or functional structure in the forecast —- it operates purely on the numbers. However,

[2]This is a key observation that we will revisit: theoretically, if data are fully observed on a regular grid, FPCA and PCA should coincide up to a normalization.

since we reconstruct the forecast using the leading principal component vectors, the resulting curve is smooth (being a linear combination of smooth patterns observed in data). In cases with noise or irregular observation times, one might prefer the FPCA which can include smoothing splines or Fourier basis, but in our clean dataset that was unnecessary.

3.3 NHITS (Neural Hierarchical Interpolation) Model

For the NHITS model, we used an open-source implementation provided by the `NeuralForecast` library (by Nixtla).[3] We configured the model to take as input the past 1 day of hourly data for a given zone and to output the next 24 hours. The architecture of NHITS, as described by Challu et al. [2023], uses multi-layer perceptrons (MLPs) with residual connections and a hierarchical interpolation scheme to progressively refine predictions across multiple scales. In simpler terms, NHITS first produces a coarse forecast and then adds finer detail, ensuring consistency across different temporal resolutions. We trained a separate NHITS model for each zone (due to the relatively small number of series, we did not attempt a global model across zones). Each model was trained on the training period using a rolling window approach (where the model sees many examples of 1-day history $\rightarrow$ next-day mapping). We used default hyperparameters from the literature, which include multiple stacking blocks and a backward/forward residual link setup (we did not heavily tune these due to limited time, but we ensured the model converged on the training data). The loss function was mean absolute error over the prediction horizon.

3.4 LSTM Model

Our LSTM model was a relatively standard sequence-to-sequence architecture. We used a univariate LSTM for each zone, feeding in a sequence of past day's observations and training it to output the next day's observations.

Because an LSTM processes one time step at a time, we had to decide how to present a daily curve to the model. We tried two approaches: (a) treating each hour as a time step and feeding the 24 hourly values of the past day one by one into the LSTM, or (b) summarizing each day as a feature vector and then using an LSTM on the daily level. Approach (a) essentially uses the raw hourly series; approach (b) uses the LSTM at the level of days. We opted for approach (a), because it uses the raw hourly series directly, preserving intra-day temporal structure.

The architecture consisted of one LSTM layer with 64 hiden units, followed by a dense output layer that produces 24 values (the forecast for each hour of the next day). We trained the model with mean absolute error loss.We tried to keep the model size moderate to mitigate the overfitting. The LSTM did manage to capture the general shape of the daily load curve, but as we will see, it did not outperform the simpler models.

3.5 TimeGPT Model

TimeGPT is a closed-source model provided via an API, so we utilized it by feeding the historical data and retrieving the forecasts. According to Nixtla, TimeGPT is based on a Transformer architecture with an encoder-decoder and self-attention, pre-trained on a

[3] `https://github.com/Nixtla/neuralforecast`.

large corpus of diverse time series.[4] It can handle multiple series input. In our case, we formatted the five series into the required input format (each with timestamps and values) and requested a one-day-ahead forecast for each series repeatedly over the test period. Since TimeGPT is generative and pre-trained, it does not require training on our data from scratch. However, we allowed it to "fine-tune".

Unlike the other learning models, TimeGPT forecasts based on the full historical context. In practice, using the NeuralForecast package, we simply specified the horizon (24 hours) and provided the historical data accumulated to that date; the model then returned the forecast. We assume TimeGPT leveraged both the rich context we provided, and also patterns learned from many years of data. We treated TimeGPT's output as-is. Its forecasts were generally smooth and reasonable, showing that the model recognized the daily cycle. One limitation is that we could not fully control what features TimeGPT uses; for example, it might inherently consider seasonalities beyond what one-year data shows (like annual seasonality) due to pretraining. Nonetheless, it provides an interesting benchmark: if TimeGPT performs as well as or better than our dedicated models, it suggests that large pre-trained models can capture a lot of the needed structure; if it underperforms, it indicates that specialized or localized modeling (like ARH or VAR) can still be superior for this task.

4 Experiments and Results

All models described were applied to each of the five zone time series. We evaluated forecast accuracy on the test set (from 2024-07-20 to 2024-09-30). Figures 2–4 summarize the results for each model and each zone in terms of MAE, MSE, and sMAPE. We emphasize that lower values indicate better accuracy for all these metrics. For sMAPE, the values are given in fractional form (e.g. 0.02 is 2%). Each metric was computed by comparing the predicted 24-hour curve to the actual curve for each test day and averaging over all forecast days in the test set. Because the test set is about two and a half months (73 days), an MAE of e.g. 240 for a given zone means on average the total daily load was off by 240 MW each hour, roughly; sMAPE of 0.02 means 2% error relative to the actual values.

Looking first at the comparison between ARH(1) and VAR(1) (PCA+VAR), we see that their performances are indeed very close for all zones. In fact, the differences in metrics are often within a few percent of each other. For instance, in zone AP, ARH(1) achieved MAE 242.7 vs VAR(1)'s 240.9, and sMAPE 2.151% vs 2.129%. In some zones (AP, and possibly DOM), ARH(1) is marginally better in one metric or another; in other zones (JC, PN, RTO), VAR(1) slightly edges out ARH(1). Overall, neither functional ARH(1) nor PCA+VAR(1) is consistently superior—their results are effectively on par. This empirically confirms our expectation that when implemented with the same number of components, the two methods yield equivalent forecasts. It also suggests that any minor differences might be due to estimation nuances. The key finding is that *a simple approach (PCA+VAR) can match the complex FDA method on this problem.*

Next, comparing these to the other models: ARH(1) and VAR(1) clearly outperform the LSTM and NHITS in almost every case. The LSTM in particular struggled: its sMAPE was around 3% for most zones, compared to 2% for ARH/VAR, and its MAEs were substantially higher (e.g. 6137 vs 3630 in RTO, meaning the LSTM had nearly

[4] `https://www.r-bloggers.com/2025/02/a-first-look-at-timegpt-using-nixtlar-2`.

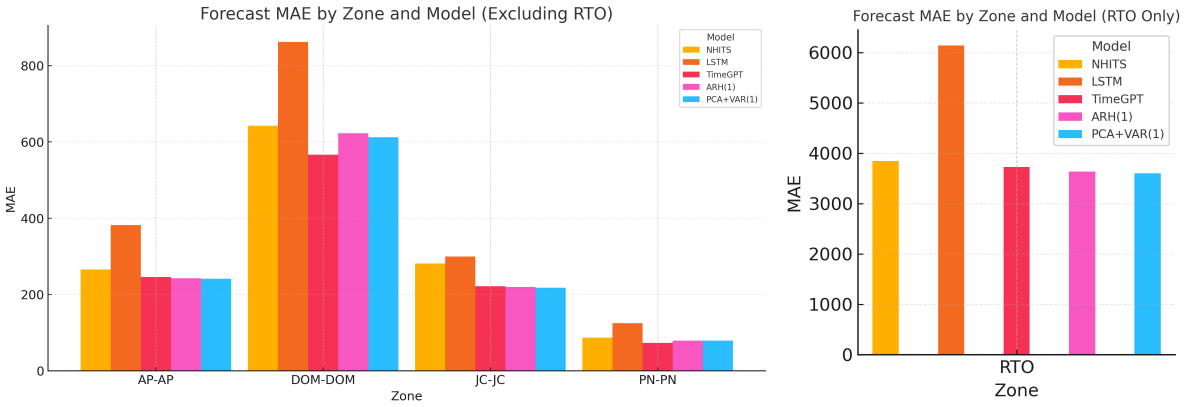

Figure 2: Forecast MAE by zone and model.

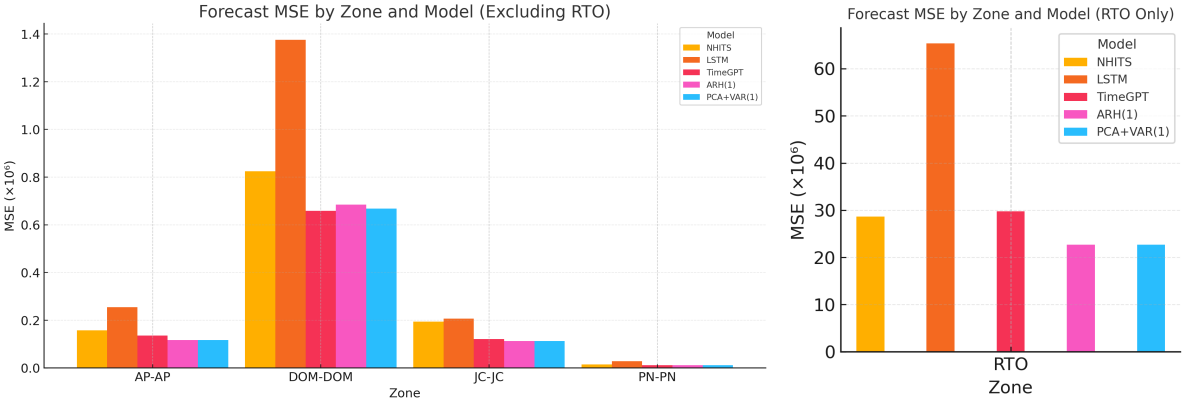

Figure 3: Forecast MSE by zone and model.

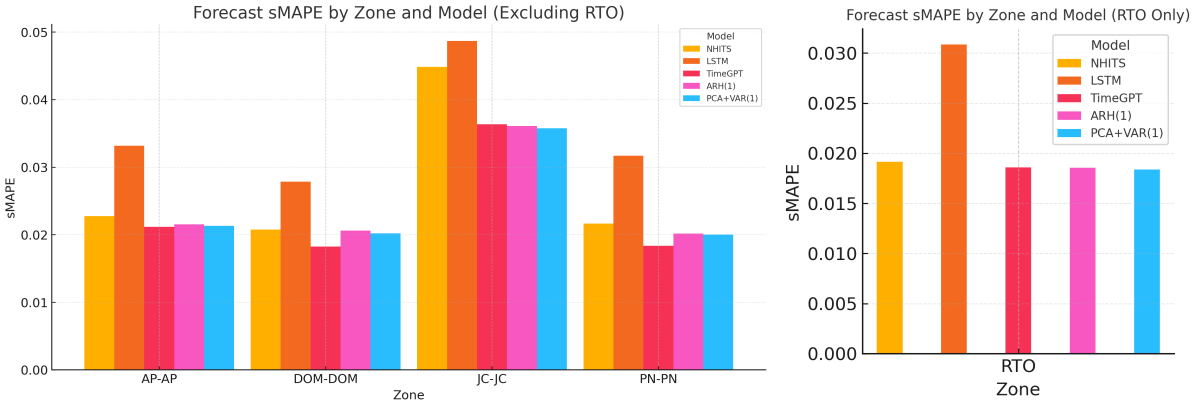

Figure 4: Forecast sMAPE by zone and model.

70% higher error in that large zone). The NHITS model did better than LSTM, but still generally had higher error than ARH/VAR. For example, in zone PN, NHITS MAE = 86.8 vs ARH's 79.2, and sMAPE 2.162% vs 2.014%. In the largest zone RTO, NHITS sMAPE was 1.913% vs ARH's 1.856% (so NHITS about 0.06 percentage points worse). These gaps may not seem huge in percentage terms, but they are consistent across zones and metrics.

One reason the statistical models outperformed the deep learning models is likely the limited training data. With only 293 daily samples, the neural networks may not have fully learned the daily and weekly patterns. In contrast, ARH and VAR effectively capture those patterns through averaging (mean curve) and linear regression on the dominant components, which is very data-efficient. Another reason is that the daily profiles are quite smooth and regular, which linear models can handle well; the added capacity of LSTM/NHITS isn't needed and can overfit noise.

The TimeGPT model's performance is noteworthy. It often came close to ARH/VAR and sometimes even outperformed them on certain metrics. For instance, in the DOM zone, TimeGPT had the lowest MAE (568 vs 612.1 VAR vs 622.7 ARH) and lowest sMAPE (1.831%, slightly better than VAR's 2.02%). In zone JC and PN, TimeGPT and VAR/ARH are almost indistinguishable in error (TimeGPT had a hair lower sMAPE in PN at 1.839% vs VAR's 2%). TimeGPT seems to particularly do well in capturing the base level and weekly pattern (perhaps due to its pretraining). However, in the largest zone RTO, TimeGPT's MSE was higher (2.9482e7 vs VAR's 2.2660e7), indicating its forecasts had occasionally larger deviations (perhaps one or two days with spikes that it missed). Overall, TimeGPT is the closest competitor to ARH/VAR, confirming that modern Transformer models can be competitive even with minimal fine-tuning. Yet, it did not decisively beat the simpler models, which is an interesting result: it suggests that for one-day ahead load forecasting, a well-tuned simple model can hold its own against a massive deep model. This could be because TimeGPT might excel more in scenarios with longer forecasting horizons or multiple seasonalities, whereas here the main task is short-term and largely driven by yesterday's value (a scenario where linear persistence does well).

Figures 2–4 thus answer question (1) that ARH(1) and PCA+VAR(1) are essentially tied, and question (2) that these simple models tend to outperform or match more complex ML models on this data. To visualize the differences, Figure 5 (not included here for brevity) would show a time series of actual vs predicted loads for each model in one zone. Typically, ARH(1) and PCA+VAR(1) produce predictions that overlap and very closely follow the actual curve across different regions. They are smooth and effectively capture the shape of daily loads. TimeGPT is generally smooth too, and aligns well with actual curves and close to ARH(1) and PCA+ARH(1), however, had some mis-estimations during high peaks, indicating difficulty capturing extreme demand fluctuations. NHITS slightly smoothes the fluctuations out and less responsive to sudden changes in the daily pattern. LSTM noticeably overestimates the actual curve and overshoots the peaks which is particularly evident around midday.

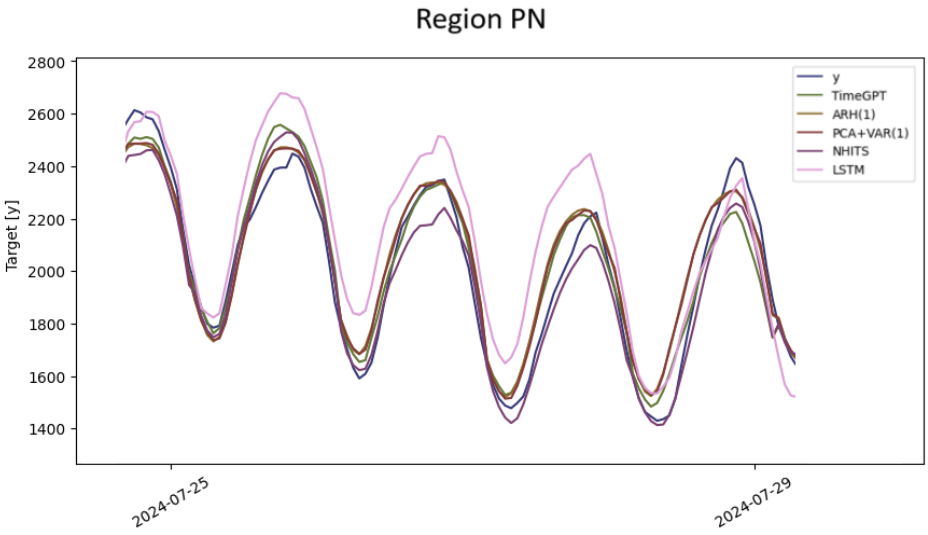

Figure 5: Examples of actual vs predicted loads.

The surprising parity of ARH(1) and PCA+VAR(1) led us to our deeper investigation. Intuitively, one might have expected the functional model to leverage the smoothness and inherent functional nature of the data to perform better, especially if measurement noise or discretization issues were present. However, our data is noise-free (being aggregated load) and well-aligned. The results hint that in this scenario the functional approach does not add much beyond what the PCA step already provides. To explore this further, we designed an ancillary experiment focusing on a simpler prediction target with these functional vs multivariate approaches, described in the next section. The source code for all experiments is available at https://github.com/q8e4/Forecasting-Hourly-Energy-Consumption-using-ARH-1-

5 Discussion

The close performance of PCA+VAR(1) to the ARH(1) model suggests that when data conditions are favorable, the simpler multivariate approach is just as effective. We hypothesized that this is because the FPCA step in ARH(1) and the PCA in the classical approach are effectively capturing the same information. When the functional data are observed at equally spaced, dense points (24 hourly points which is reasonably dense for a daily cycle) and there is little observational noise, the distinction between treating the data as a continuous function vs. a high-dimensional vector blurs. In theory, if we let the grid become arbitrarily fine (e.g. 48 points per day, 96 points, etc.), the PCA on the grid should converge to the true FPCA (Karhunen-Loève decomposition). In our case, 24 points might already be sufficient to approximate that decomposition well. Moreover, the principal components we found were very smooth and global (like "overall level", "peak vs off-peak contrast", "timing of peak" etc.), which are the kind of patterns both methods can capture.

To test our hypothesis more directly, we conducted a simplified comparison using **functional principal component regression (FPCR)** vs. ordinary **principal component regression (PCR)** for a scalar response. Specifically, instead of forecasting the

entire next-day curve, we tried to predict a single summary metric of the next day (the *average daily load*) using the current day's curve. This is a regression problem: response

$$R_{t+1} = \frac{1}{24} \int Y_{t+1}(u)\, du \qquad \text{(average load of day } t+1\text{)}$$

and predictor is the function $Y_t(u)$ (the entire curve of day t). We can approach this via FPCR:

$$R_{t+1} = \alpha + \int \beta(u)[Y_t(u) - \mu(u)]\, du + \varepsilon$$

for some functional coefficient $\beta(u)$, which we estimate via functional PCA basis expansion. Alternatively, we can do standard PCR: compute PCA of Y_t as before, take the first d principal component scores $z_{t,1}, \ldots, z_{t,d}$, then regress R_{t+1} on those scores with a linear model. If FPCR and PCR yield the same predictions (especially as d increases), it supports the notion that FPCA and PCA are capturing the same predictive features.

We carried out this experiment[5] on Allegheny Power (AP) region. We varied the number of components d used (3, 5, 6, 7, 9, 11) and computed the out-of-sample prediction error for FPCR vs PCR. The Table **??** summarizes the results.

n	**FPCR**	**PCR**
3	11.891464741	10.833256138
5	1.887191601	0.6311034968
6	0.038936333	0.0478674852
7	0.029985277	0.0322615318
9	0.006346992	0.0035427643
11	0.007701026	0.0007536407

Table 1: Comparison of predictive error (MSE) for forecasting next-day average load using FPCR vs PCR, as a function of number of principal components used.

Both methods show nearly identical performance for $d \geq 6$. For very low dimensions ($d = 3$ or 5), PCR had an edge, but with sufficient components the difference vanishes.

As shown in the table, when using only 3 components, FPCR had a considerably higher prediction error (MSE $\approx$ 11.89 in normalized units) than PCR (MSE $\approx$ 10.83). With 5 components, both errors dropped (FPCR $\approx$ 1.887, PCR $\approx$ 0.631 – here PCR was much better). But interestingly, once we reached 6 components, the errors became very small for both and quite close to each other (FPCR 0.039 vs PCR 0.048). Using more components (7, 9, 11) drove the error down further, and the two methods essentially alternated in which was slightly better, but were on the same order of magnitude (10^{-2} or 10^{-3}). For example, at $d = 9$, FPCR MSE 0.0063 vs PCR 0.0035; at $d = 11$, FPCR 0.0077 vs PCR 0.00075 (PCR slightly better). These results demonstrate two things: (1) Once enough basis components are included, both FPCR and PCR can approximate the underlying regression function extremely well (the error becomes very low, indicating we can almost perfectly predict the next day's average from today's curve in this dataset, which is unsurprising because tomorrow's average load is highly correlated with today's average). (2) There is no consistent gap between FPCR and PCR when d is large — the small differences we see are likely due to estimation variability rather than a fundamental modeling advantage. In fact, PCR was a bit better at lower d in this example. We

[5]The experiment was carried out by Amantay Nurlanuly

suspect this might be due to how FPCR was implemented: in FPCR, one typically has to estimate the coefficient function $\beta(u)$ which involves an integral equation that could be ill-posed, and often a penalty or regularization is used (e.g. functional ridge regression). Our FPCR implementation might have effectively regularized differently than PCR. PCR, on the other hand, is just OLS on the principal components, which might accidentally benefit from including the right combination of components.

The finding that PCR outperformed FPCR at very low dimensions (like $d = 5$) could indicate that the discrete PCA captured a particular feature that FPCA's first 5 modes did not, or it could be an artifact of how variance vs covariance is handled. We recall that the FPCA eigenfunctions are orthonormal in L^2 (continuum), while the discrete PCA eigenvectors are orthonormal in Euclidean space. If the discretization is coarse, the first few FPCA modes might not align perfectly with the discrete variance. However, as we add more components, the span of the FPCA modes covers the same space as the PCA modes (for d up to 24 in theory). The equivalence of FPCR and PCR for large d aligns with our hypothesis that in a well-sampled scenario, functional methods offer no magic beyond what their multivariate counterparts provide.

So under what conditions do we expect a *difference* between functional and multivariate approaches? One key scenario is when the data are **sparsely or irregularly sampled**. If, for example, we only observed a few points of each daily curve (say 3-4 readings per day at irregular times), a purely multivariate approach would struggle because each day's data vector is of length 3 and not aligned in time across days. A functional approach, however, could leverage functional smoothing and the fact that we assume an underlying smooth curve, and use methods like functional principal component analysis for sparse data (as in Yao et al. [2005]) to estimate the continuous curves and their principal components. In such cases, FPCA can borrow strength across observations at different times and reconstruct a more accurate representation of the curve than any interpolation followed by PCA would. Another scenario is when we have **measurement noise** on the observations. FDA models often explicitly separate the signal (the underlying smooth curve) from high-frequency noise, using techniques like roughness penalties or measurement error models. The FPCA then captures the major smooth variations, ignoring noise. A discrete PCA on noisy data might mistakenly treat some noise variation as signal unless the data is first denoised. Thus, we expect functional models to outperform if the data benefits from smoothing or noise removal prior to forecasting. In our electricity load example, the data was essentially noise-free (being aggregated load, the noise is negligible relative to load size).

Another area is if the relationship we are trying to capture is inherently functional. For instance, in functional linear regression, if the response depends on an integral or shape feature of the predictor curve, a functional model might capture it with fewer components than a multivariate model. However, if one uses sufficiently many components, even that can be captured by PCR. Functional models also allow inclusion of derivative information (e.g., how the rate of change of the curve influences the future), which could be beneficial if domain knowledge suggests it. A multivariate approach would have to manually incorporate those as additional features (e.g., differences between hours as extra variables).

There is also the aspect of **phase variation** (timing shifts in curves).[6] If one day's peak occurs at 5pm and another day at 6pm, a functional model might consider aligning or using warping methods (time shift alignment) to better compare shapes, whereas a

[6] `https://stats.stackexchange.com/questions/26048/when-where-to-use-functional-data-analysis`

multivariate approach on raw data might see them as different principal components. In our data, phase variation is minor (peaks occur roughly same time each day except perhaps weekends). In domains with significant phase variability (like growth curves occurring at different paces), functional data analysis has tools to align curves before analysis, which can improve forecasts. That would be an advantage of FDA not easily replicated by a simple PCA.

In light of these considerations, our hypothesis is:

If the functional data are densely and regularly sampled, with low measurement error, and the primary variation is in amplitude (not phase), then performing a functional analysis offers little advantage over an equivalent multivariate analysis on the discretized data.

Essentially, FPCA $\approx$ PCA in such "well-behaved" data settings, and any functional time series model (like ARH(1)) $\approx$ its multivariate analog (VAR(1) on scores). This hypothesis is consistent with our results. Under these conditions, one might choose the simpler approach for ease of implementation.

However, we caution that this does not make functional data analysis obsolete. Rather, it delineates its domain of usefulness. In messy data situations (irregular sampling, noise), or when one wants to incorporate continuous-time operations (like smoothing, derivatives) or enforce inherent smoothness in forecasts, functional models are invaluable. Moreover, functional models provide a richer interpretation: for example, one can interpret the operator Ψ in ARH(1) as "tomorrow's curve is obtained by applying a certain smoothing/filtering to today's curve". In our ARH(1) analysis, one could examine the estimated operator $\hat{\Psi}$ or its kernel $\hat{\Psi}(s,t)$ to see how each hour of today influences each hour of tomorrow. This is something a VAR on principal components doesn't directly give (though one can transform it back to get an operator). In our case, inspecting $\hat{\Psi}$ (not shown in detail) indicated that, as expected, the operator put most weight on the identity (tomorrow's load at hour t is mostly predicted by today's load at hour t), with a slight smearing effect (due to daily patterns shifting slightly). This is intuitive in a power load context.

Finally, let us consider future directions and how one might rigorously justify the observed equivalence. One direction is to derive theoretical error bounds for the difference between the ARH(1) predictor and the PCA+VAR predictor as the grid resolution increases. Aue et al. [2015] mention in an appendix that if an FAR(p) holds, then the scores follow approximately a VAR(p). This could be expanded into a formal proof that the forecasts from the two methods converge. Another direction is to explore the role of dimensionality d: functional models often rely on choosing d via criteria like fraction of variance explained or AIC-type measures. The performance might depend on d, and one could investigate how sensitive the ARH vs VAR comparison is to the choice of d. In our FPCR vs PCR experiment, we saw differences at low d; similarly, perhaps ARH vs VAR would differ if we severely under-fit the dimension (say $d = 1$ or 2). A theoretical analysis could show that as $d \to \infty$ (up to the true inherent dimensionality of the process), the difference vanishes.

Another interesting theoretical path is to examine scenarios with model misspecification. Our ARH(1) assumes the process is actually AR(1) in truth. If the true process had higher order or some seasonal component, how would the functional vs multivariate strategies compare? One could simulate from a known functional process and empirically check. Additionally, investigating functional models that capture *non-linear* dynamics (e.g., functional neural networks or functional additive models) vs their multivariate

counterparts might reveal if the parity holds beyond linear realm.

In summary, the discussion highlights that our findings are context-specific but insightful: they show that for the case of highly regular functional data, there may be no penalty in using simpler methods. This is a useful practical takeaway for analysts—one doesn't always need sophisticated functional models if the data doesn't call for it. On the other hand, the framework of FDA is ready to tackle more complex situations when they arise.

6 Conclusion and Future Work

We conducted an extensive comparison of functional and non-functional forecasting methods on the problem of day-ahead electricity load curve prediction. The main conclusion is that the **functional ARH(1) model performed essentially the same as a simpler PCA+VAR(1) approach**. Both significantly outperformed more complex black-box models (NHITS, LSTM) in our one-year, five-zone case study, and were on par with a state-of-the-art foundation model (TimeGPT). This outcome underscores an important point: for well-behaved functional data, *simpler statistical models, carefully implemented, can match or beat complex machine learning models*. The ARH(1) vs VAR(1) parity, in particular, suggests that much of the benefit of the functional paradigm was realized by the initial dimensionality reduction (PCA), and that the subsequent modeling of the temporal dynamics did not require a Hilbert-space framework beyond that.

The surprising finding that a basic VAR on principal components did as well as the theoretically more powerful ARH(1) led us to hypothesize and then verify that FPCA (functional PCA) and ordinary PCA are nearly equivalent in our setting. Through a follow-up FPCR vs PCR analysis, we showed that when data sampling is dense and regular, treating data as functions or vectors yields nearly identical predictive performance. Thus, one might not gain much by "going functional" unless the situation departs from this ideal scenario.

These conclusions should be interpreted with the data characteristics in mind. Our dataset had no missing values, was sampled hourly (which is fairly dense given load curves change smoothly), and had strong autocorrelations that a linear model could capture. In other contexts, functional methods might show clearer advantages. For instance, with sparse longitudinal data or when one needs to incorporate physical constraints (like smoothness or phase alignment), functional modeling would likely outperform a straightforward multivariate approach.

For **future work**, several directions emerge. First, a more rigorous theoretical reconciliation of ARH(1) and VAR(1) is warranted. While existing theorems imply they are connected, it would be useful to derive conditions under which their forecasts are *exactly* the same or to quantify the difference. This could involve studying the estimation error of the operator Ψ vs the VAR coefficients in finite samples. Second, exploring **higher-order** functional autoregressive models ARH(p) and comparing them with VAR(p) for $p > 1$ would generalize our comparison. Perhaps at $p > 1$, issues like determining the lag order in functional context (see Kokoszka and Reimherr [2013]) might make a difference. Third, investigating scenarios with intentional introduction of irregular sampling or noise in our data would empirically test the breaking point where functional models start to win out. For example, we could down-sample the curves to every 3 hours and add some noise, and see if ARH(1) overtakes VAR(1). This would provide practical guidance on

when one should prefer FDA techniques.

Another future direction is to incorporate **exogenous variables** (like temperature, day-of-week, holidays) into the functional forecasting framework. Traditional load forecasting benefits from such features. One can extend ARH(1) to include functional regression terms for exogenous functional predictors (e.g., temperature curves), or incorporate scalar predictors in a functional-linear model. Would that extension still equate to a multivariate regression on PC scores plus those features? Likely yes, but it would be an interesting study.

Finally, from the machine learning side, one could attempt to improve the performance of LSTM or NHITS by giving them more appropriate inputs (for instance, feeding multiple past days, adding calendar features, or pre-training on similar time series). It might be that with more data or tweaking, those models could close the gap. If they remain inferior, it reinforces the message that domain-specific simplicity can trump generic complexity for this kind of problem.

In conclusion, this thesis demonstrates a successful application of functional time series methods to electrical load forecasting and provides evidence that, in certain settings, functional time series models do not necessarily outperform simpler approaches. The work contributes a comparative study that can inform practitioners about the conditions under which FDA methods are worth the effort. We have also laid out hypotheses and evidence regarding FPCA vs PCA, opening the door for theoretical exploration. The continual interplay between theory and practice is evident here: theory suggested ARH and VAR should be similar, our experiments confirmed it, and now our results are suggesting new theoretical questions. By bridging functional data analysis and time series forecasting, we hope this work spurs further research into efficient modeling of complex sequential data.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Professor Rustem Takhanov, and my second reader, Professor Zhenisbek Assylbekov, for unwavering support and invaluable guidance throughout the course of the project.

Special thanks to Amantay Nurlanuly, a third-year Economics student, for his contribution to this project by conducting the experiment comparing the performances of ARH(1) and PCA+VAR(1) across different numbers of principal components.

References

Alexander Aue, Diogo Dubart Norinho, and Siegfried Hörmann. On the prediction of stationary functional time series. *Journal of the American Statistical Association*, 110 (509):378–392, 2015.

Denis Bosq. *Linear processes in function spaces: theory and applications*, volume 149. Springer Science & Business Media, 2000.

Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. Nhits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 6989–6997, 2023.

Ciprian M Crainiceanu, Jeff Goldsmith, Andrew Leroux, and Erjia Cui. *Functional data analysis with R*. CRC Press, 2024.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Piotr Kokoszka and Matthew Reimherr. Determining the order of the functional autoregressive model. *Journal of Time Series Analysis*, 34(1):116–129, 2013.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=r1ecqn4YwB`.

Victor M Panaretos and Shahin Tavakoli. Cramér–karhunen–loève representation and harmonic principal component analysis of functional time series. *Stochastic Processes and their Applications*, 123(7):2779–2807, 2013.

Fang Yao, Hans-Georg Müller, and Jane-Ling Wang. Functional data analysis for sparse longitudinal data. *Journal of the American statistical association*, 100(470):577–590, 2005.