Forecasting time series with constraints

Nathan Doumèche NATHAN.DOUMECHE@SORBONNE-UNIVERSITE.FR
Sorbonne University, EDF R&D

Francis Bach FRANCIS.BACH@INRIA.FR
Inria, ENS, PSL Research University

Eloi Bedek ELOI.BEDEK@EDF.FR
EDF R&D

Gérard Biau GERARD.BIAU@SORBONNE-UNIVERSITE.FR
Sorbonne University, IUF

Claire Boyer CLAIRE.BOYER@UNIVERSITE-PARIS-SACLAY.FR
Université Paris-Saclay, IUF

Yannig Goude YANNIG.GOUDE@EDF.FR
EDF R&D, Paris-Saclay University

Abstract

Time series forecasting presents unique challenges that limit the effectiveness of traditional machine learning algorithms. To address these limitations, various approaches have incorporated linear constraints into learning algorithms, such as generalized additive models and hierarchical forecasting. In this paper, we propose a unified framework for integrating and combining linear constraints in time series forecasting. Within this framework, we show that the exact minimizer of the constrained empirical risk can be computed efficiently using linear algebra alone. This approach allows for highly scalable implementations optimized for GPUs. We validate the proposed methodology through extensive benchmarking on real-world tasks, including electricity demand forecasting and tourism forecasting, achieving state-of-the-art performance.

Keywords: Physics-informed machine learning, constraints, hierarchical forecasting, transfer learning, load forecasting, tourism forecasting

1 Introduction

Time series forecasting. Time series data are used extensively in many contemporary applications, such as forecasting supply and demand, pricing, macroeconomic indicators, weather, air quality, traffic, migration, and epidemic trends (Petropoulos et al., 2022). However, regardless of the application domain, forecasting time series presents unique challenges due to inherent data characteristics such as observation correlations, non-stationarity, irregular sampling intervals, and missing values. These challenges limit the availability of relevant data and make it difficult for complex black-box or overparameterized learning architectures to perform effectively, even with rich historical data (Lim and Zohren, 2021).

Constraints in time series. In this context, many modern frameworks incorporate physical constraints to improve the performance and interpretability of forecasting models. The strongest form of such constraints are typically derived from fundamental physical properties of the time series data and are represented by systems of differential equations. For example, weather forecasting often relies on solutions to the Navier-Stokes equations (Schultz et al., 2021). In addition to defining physical relationships, differential constraints can also serve as regularization mechanisms. For example, spatiotemporal regression on graphs can involve penalizing the spatial Laplacian of the regression function to enforce smoothness across spatial dimensions (Jin et al., 2024).

However, time series rarely satisfy strict differential constraints, often adhering instead to more relaxed forms of constraints (Coletta et al., 2023). Perhaps the most successful example of such weak constraints are the generalized additive models (GAMs, Hastie and Tibshirani, 1986), which have been applied to time series forecasting in epidemiology (Wood et al., 2017), earth sciences (Augustin et al., 2009), and energy forecasting (Fasiolo et al., 2021). GAMs model the target time series (or some parameters of its distribution) as a sum of nonlinear effects of the features, thereby constraining the shape of the regression function. Another example of weak constraint appears in the context of spatiotemporal time series with hierarchical forecasting. Here, the goal is to combine regional forecasts into a global forecast by enforcing that the global forecast must be equal to the sum of the regional forecasts (Wickramasuriya et al., 2019). Although this may seem like a simple constraint, hierarchical forecasting is challenging because of a trade-off: using more granular regional data increases the available information, but also introduces more noise as compared to the aggregated total. Another common and powerful constraint in time series forecasting arises when combining multiple forecasts (Gaillard et al., 2014). This is done by creating a final forecast by weighting each of the initial forecasts, with the constraint that the sum of the weights must equal one.

PIML and time series. Although weak constraints have been studied individually and applied to real-world data, a unified and efficient approach is still lacking. It is important here to mention physics-informed machine learning (PIML), which offers a promising way to integrate constraints into machine learning models. Based on the foundational work of Raissi et al. (2019), PIML exploits the idea that constraints can be applied with neural networks and optimized by backpropagation, leading to the development of physics-informed neural networks (PINNs). PINNs have been successfully used to predict time series governed by partial differential equations (PDEs) in areas such as weather modeling (Kashinath et al., 2021), and stiff chemical reactions (Ji et al., 2021). Weak constraints on the shape of the regression function have also been modeled with PINNs (Daw et al., 2022). However, PINNs often suffer from optimization instabilities and overfitting (Doumèche et al., 2024b). As a result, alternative methods have been developed for certain differential constraints that offer improved optimization properties over PINNs. For example, data assimilation techniques in weather forecasting have been shown to be consistent with the Navier-Stokes equations (Nickl and Titi, 2024). Moreover, Doumèche et al. (2024a) showed that forecasting with linear differential constraints can be formulated as a kernel method, yielding closed-form solutions to compute the exact empirical risk minimum. An additional advantage of this kernel modeling is that the learning algorithm can be executed on GPUs, leading to significant speedups compared to the gradient-descent-based optimization of PINNs (Doumèche et al., 2024).

Contributions. In this paper, we present a principled approach to effectively integrate constraints into time series forecasting. Each constrained problem is reformulated as the minimization of an empirical risk consisting of two key components: a data-driven term and a regularization term that enforces the smoothness of the function and the desired physical constraints. For nonlinear regression tasks, we rely on a Fourier expansion. Our framework allows for efficient computation of the exact minimizer of the empirical risk, which is easily optimized on GPUs for scalability and performance.

In Section 2, we introduce a unified mathematical framework that connects empirical risks constrained by various forms of physical information. Notably, we highlight the importance of distinguishing between two categories of constraints: shape constraints, which limit the set of admissible functions, and learning constraints, which introduce an initial bias during parameter optimization. In Section 3, we explore shape constraints and illustrate their relevance using the example of electricity demand forecasting. In Section 4, we define learning constraints and show how they can be applied to tourism forecasting. This common modeling framework for shape and learning constraints allows for efficient integration of multiple constraints, as illustrated by the WeaKL-T in Section 4, which combines hierarchical forecasting with additive models and transfer learning. Each empirical risk can then be minimized on a GPU using linear algebra, ensuring scalability and computational efficiency. This direct computation guarantees that the proposed estimator exactly minimizes the empirical risk, preventing convergence to potential local minima—a common limitation of modern iterative and gradient descent methods used in PINNs. Our method achieves significant performance improvements over state-of-the-art approaches. The code for the numerical experiments and implementation is publicly available at `https://github.com/NathanDoumeche/WeaKL`.

2 Incorporating constraints in time series forecasting

Throughout the paper, we assume that n observations $(X_{t_1}, Y_{t_1}), \dots, (X_{t_n}, Y_{t_n})$ are drawn on $\mathbb{R}^{d_1} \times \mathbb{R}^{d_2}$. The indices $t_1, \dots, t_n \in T$ correspond to the times at which an unknown stochastic process $(X, Y) := (X_t, Y_t)_{t \in T}$ is sampled. Note that, all along the paper, the time steps need not be regularly sampled on the index set $T \subseteq \mathbb{R}$. We focus on supervised learning tasks that aim to estimate an unknown function $f^\star : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$, under the assumption that $Y_t = f^\star(X_t) + \varepsilon_t$, where ε is a random noise term. Without loss of generality, upon rescaling, we assume that $X_t := (X_{1,t}, \dots, X_{d_1,t}) \in [-\pi, \pi]^{d_1}$ and $-\pi \le t_1 \le \dots \le t_{n+1} \le \pi$. The goal is to construct an estimator $\hat{f}$ for $f^\star$.

A simple example to to keep in mind is when Y is a stationary, regularly sampled time series with $t_j = j/n$, and the lagged value $X_j = Y_{t_{j-1}}$ serves as the only feature. In this specific case, where $d_1 = d_2$, the model simplifies to $Y_t = f^\star(Y_{t-1/n}) + \varepsilon_t$. Thus, the regression setting reduces to an autoregressive model. Of course, we will consider more complex models that go beyond this simple case.

Model parameterization. We consider parameterized models of the form

$$f_\theta(X_t) = (f_\theta^1(X_t), \dots, f_\theta^{d_2}(X_t)) = (\langle \phi_1(X_t), \theta_1 \rangle, \dots, \langle \phi_{d_2}(X_t), \theta_{d_2} \rangle), \tag{1}$$

where each component $f_\theta^\ell(X_t)$ is computed as the inner product of a feature map $\phi_\ell(X_t) \in \mathbb{C}^{D_\ell}$, with $D_\ell \in \mathbb{N}^\star$, and a vector $\theta_\ell \in \mathbb{C}^{D_\ell}$. The parameter vector $\theta \in \mathbb{C}^{D_1 + \dots + D_{d_2}}$ of the model is defined as the concatenation of $\theta_1, \dots, \theta_{d_2}$. Note that f_θ is uniquely determined by θ and the maps ϕ_ℓ. To simplify the notation, we write $\dim(\theta) = D_1 + \dots + D_{d_2}$.

Our goal is to learn a parameter $\hat{\theta} \in \mathbb{C}^{\dim(\theta)}$ such that $\hat{Y}_t = f_{\hat{\theta}}(X_t)$ is an estimator of the target Y_t. Equivalently, $f_{\hat{\theta}}$ is an estimator of the target function $f^\star$. To this end, the core principle of our approach is to consider $\hat{\theta}$ to be a minimizer over $\mathbb{C}^{\dim(\theta)}$ of an empirical risk of the form

$$L(\theta) = \frac{1}{n} \sum_{j=1}^{n} \|\Lambda(f_\theta(X_{t_j}) - Y_{t_j})\|_2^2 + \|M\theta\|_2^2, \tag{2}$$

where Λ and M are complex-valued matrices with problem-dependent dimensions, which are not necessarily square. The matrix M encodes a regularization penalty, which may include hyperparameters to be tuned through validation, as we will see in several examples.

Explicit formula for the empirical risk minimizer: WeaKL. The following proposition shows how to compute the exact minimizer of (2). (Throughout the document, $*$ denotes the conjugate transpose operation.)

Proposition 2.1 (Empirical risk minimizer.) *Suppose both M and Λ are injective. Then, there is a unique minimizer to* (2)*, which takes the form*

$$\hat{\theta} = \Big(\Big(\sum_{j=1}^{n} \Phi_{t_j}^* \Lambda^* \Lambda \Phi_{t_j}\Big) + nM^*M\Big)^{-1} \sum_{j=1}^{n} \Phi_{t_j}^* \Lambda^* \Lambda Y_{t_j}, \tag{3}$$

where Φ_t is the $d_2 \times \dim(\theta)$ block-wise diagonal feature matrix at time t, defined by

$$\Phi_t = \begin{pmatrix} \phi_1(X_t)^* & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \phi_{d_2}(X_t)^* \end{pmatrix}. \tag{4}$$

This result, proven in Appendix A.1, generalizes well-known results on kernel ridge regression (see, e.g., Mohri et al., 2012, Equation 10.17). In the rest of the paper, we refer to the estimator $\hat{\theta}$ as the weak kernel learner (WeaKL). The strength of WeaKL lies in its exact computation via (3). Unlike current implementations of GAMs and PINNs, WeaKL is free from optimization errors. Furthermore, since WeaKL relies solely on linear algebra, it can take advantage of GPU programming to accelerate the learning process. This efficiency enables effective hyperparameter optimization, as demonstrated in Section 3.2 through applications to electricity demand forecasting.

Algorithmic complexity. The formula (3) used in this article to minimize the empirical risk (2) can be implemented with a complexity of $O(\dim(\theta)^3 + \dim(\theta)^2 n)$. Note that the dimensions d_1 and d_2 of the problem only impact the complexity of WeaKL through $\dim(\theta) = D_1 + \cdots + D_{d_2}$. By construction, $\dim(\theta) \geq d_2$, but the influence of d_1 is more subtle and depends on the chosen dimension D_ℓ of the maps $\phi_j : [-\pi, \pi]^{d_1} \to \mathbb{C}^{D_j}$. In particular, if all the maps have the same dimension, i.e., $D_\ell = D$, then $\dim(\theta) = Dd_2$.

Notably, this implementation runs in less than ten seconds on a standard GPU (e.g., an NVIDIA $L4$ with 24 GB of RAM) when $\dim(\theta) \leq 10^3$ and $n \leq 10^5$. We believe that this framework is particularly well suited for time series, where data sampling is often costly, thus limiting both n and d_2. Moreover, in many cases, the distribution of the target time series changes significantly over time, making only the most recent observations relevant for forecasting. This further limits the size of n. For example, in the Monash time series forecasting archive (Godahewa et al., 2021), 19 out of 30 time series have $d_2 \leq 10^3$ and $n \leq 10^5$. However, there are relevant time series where either the dimension d_2 or the number of data points n is large. In such cases, finding an exact minimizer of the empirical risk (2) becomes very computationally expensive. Efficient techniques have been developed to approximate the minimizer of (2) in these regimes (see, e.g., Meanti et al., 2020), but a detailed discussion of these methods is beyond the scope of this paper.

Some important examples. Let us illustrate the mechanism with two fundamental examples. Of course, the case where $\phi_\ell(x) = x$ and where Λ and M are identity matrices corresponds to the well-known ridge linear regression. On the other hand, a powerful example of a non-parametric regression map is the Fourier map, defined as $\phi_\ell(x) = (\exp(i\langle x, k\rangle/2))^\top_{\|k\|_\infty \le m} = (\exp(i\langle x, k\rangle/2))^\top_{-m\le k_1,\dots,k_{d_1}\le m}$, where the Fourier frequencies are truncated at $m \ge 0$. This map leverages the expressiveness of the Fourier basis to capture complex patterns in the data. Thus, for the ℓ-th component of f_θ, we consider the Fourier decomposition

$$f_\theta^\ell(x) = \sum_{\|k\|_\infty \le m} \theta_{\ell,k} \exp(-i\langle x, k\rangle/2),$$

which can approximate any function in $L^2([-\pi,\pi]^{d_1}, \mathbb{R})$ as $m \to \infty$. In this example, we have $\theta_\ell = (\theta_{\ell,k})^\top_{\|k\|_\infty \le m} \in \mathbb{C}^{(2m+1)^d}$. Next, for $s \in \mathbb{N}^\star$, let M be the $(2m+1)^{d_1} \times (2m+1)^{d_1}$ positive diagonal matrix such that

$$\|M\theta_\ell\|_2^2 = \lambda \sum_{\|k\|_\infty \le m} \theta_{\ell,k}^2 (1 + \|k\|_2^{2s}),$$

where $\lambda > 0$ is an hyperparameter. Then, $\|M\theta_\ell\|_2$ is a Sobolev norm on the derivatives up to order s of f_{θ_ℓ}. When $\lambda = 1$, we will denote this norm by $\|f_\theta^\ell\|_{H^s}$. This approach regularizes the smoothness of $f_{\hat\theta}^\ell$, encouraging the recovery of smooth solutions. Moreover, choosing Λ as the identity matrix and $\lambda = n^{-2s/(2s+d_1)}$ achieves the Sobolev minimax rate $\mathbb{E}(\|f_{\hat\theta}^\ell(X) - Y_\ell\|_2^2) = O(n^{-2s/(2s+d_1)})$ (Blanchard and Mücke, 2020). This result justifies why the Fourier decomposition serves as an effective nonparametric mapping.

These fundamental examples illustrate the richness of the approach, making it possible to incorporate constraints into models of chosen complexity, from very light models like linear regression, up to nonparametric models such as Fourier maps.

Classification of the constraints. In order to clarify our discussion as much as possible, we find it helpful, after a thorough analysis of the existing literature, to consider two families of constraints. This distinction arises from the need to address two fundamentally different aspects of the forecasting problem.

1. **Shape constraints**, described in Section 3, include additive models, online adaption after a break, and forecast combinations (detailed in Appendix B.1). In these models, prior information is incorporated by selecting custom maps ϕ_ℓ. The set of admissible models f_θ is thus restricted by shaping the structure of the function space through this choice of maps. Here, the matrix M serves only as a regularization term, while Λ is the identity matrix.

2. **Learning constraints**, described in Section 4, include transfer learning, hierarchical forecasting, and differential constraints (detailed in Appendix B.2). In these models, prior information or constraints are incorporated through the matrices M and Λ. The goal is to increase the efficiency of parameter learning by introducing additional regularization.

It is worth noting, however, that certain specific shape constraints cannot be penalized by a kernel norm, such as those in isotonic regression. In the conclusion, we discuss possible extensions to account for such constraints.

3 Shape constraints

3.1 Mathematical formulation

In this section, we introduce relevant feature maps ϕ that incorporate prior knowledge about the shape of the function $f^\star : [-\pi, \pi]^{d_1} \to \mathbb{C}^{d_2}$. To simplify the notation, we focus on the one-dimensional case where $d_2 = 1$ and $\Lambda = 1$. This simplification comes without loss of generality, since the feature maps developed in this section can be applied directly to (1).

As a result, the model reduces to $f_\theta(X_t) = \langle \phi_1(X_t), \theta_1 \rangle$, and (3) simplifies to

$$\hat{\theta} = (\Phi^*\Phi + nM^*M)^{-1}\Phi^*\mathbb{Y}, \tag{5}$$

where $\mathbb{Y} = (Y_{t_1}, \dots, Y_{t_n})^\top \in \mathbb{R}^n$ and the $n \times \dim(\theta)$ matrix Φ takes the form

$$\Phi = (\phi_1(X_{t_1}) \mid \cdots \mid \phi_1(X_{t_n}))^*.$$

Note that Φ is the classical feature matrix, and that it is related to the matrix Φ_t of (4) by $\Phi^*\Phi = \sum_{j=1}^n \Phi_{t_j}^* \Phi_{t_j} = \sum_{j=1}^n \phi_1(X_{t_j})\phi_1(X_{t_j})^*$.

Additive model: Additive WeaKL. The additive model constraint assumes that $f^\star(x_1, \dots, x_{d_1}) = \sum_{\ell=1}^{d_1} g_\ell^\star(x_\ell)$, where $g_\ell^\star : \mathbb{R} \to \mathbb{R}$ are univariate functions. This constraint is widely used in data science, both in classical statistical models (Hastie and Tibshirani, 1986) and in modern neural network architectures (Agarwal et al., 2021). Indeed, additive models are interpretable because the effect of each feature x_ℓ is captured by its corresponding function $g_\ell^\star$. In addition, univariate effects are easier to estimate than multivariate effects (Ravikumar et al., 2009). These properties allow the development of efficient variable selection methods (see, for example, Marra and Wood, 2011), similar to those used in linear regression.

In our framework, the additivity constraint directly translates into the model as

$$f_\theta(X_t) = \langle \phi_1(X_t), \theta_1 \rangle = \langle \phi_{1,1}(X_{1,t}), \theta_{1,1} \rangle + \cdots + \langle \phi_{1,d_1}(X_{d_1,t}), \theta_{1,d_1} \rangle,$$

where ϕ_1 is the concatenation of the maps $\phi_{1,\ell}$, and θ_1 is the concatenation of the vectors $\theta_{1,\ell}$. Note that the maps $\phi_{1,\ell}$ and the vectors $\theta_{1,\ell}$ can be multidimensional, depending on the model. In this formulation, the effect of each feature is modeled by the function $g_\ell(x_\ell) = \langle \phi_{1,\ell}(x_\ell), \theta_{1,\ell} \rangle$, which can be either linear or nonlinear in x_ℓ. The empirical risk then takes the form

$$L(\theta) = \frac{1}{n}\sum_{j=1}^{n} |f_\theta(X_{t_j}) - Y_{t_j}|^2 + \sum_{\ell=1}^{d_1} \lambda_\ell \|M_\ell \theta_{1,\ell}\|_2^2, \tag{6}$$

where $\lambda_\ell > 0$ are hyperparameters and M_ℓ are regularization matrices. There are three types of effects that can be taken into account:

(i) A linear effect is obtained by setting $\phi_{1,\ell}(x_\ell) = x_\ell \in \mathbb{R}$. To regularize the parameter $\theta_{1,\ell}$, we set $M_\ell = 1$. This corresponds to a ridge penalty.

(ii) A nonlinear effect can be modeled using the Fourier map $\phi_{1,\ell}(x_\ell) = (\exp(ikx_\ell/2))_{-m \le k \le m}^\top$. To regularize the parameter $\theta_{1,\ell}$, we set M_ℓ to be the $(2m+1) \times (2m+1)$ diagonal matrix defined by $M_\ell = \mathrm{Diag}((\sqrt{1 + k^{2s}})_{-m \le k \le m})$, penalizing the Sobolev norm. A common choice for the smoothing parameter s, as used in GAMs, is $s = 2$ (see, e.g., Wood, 2017).

(iii) If x_ℓ is a categorical feature, i.e., x_ℓ takes values in a finite set E, we can define a bijection $\psi : E \to \{1, \dots, |E|\}$. The effect of x_ℓ can then be modeled as $g_\ell(x_\ell) = \langle \phi_{1,\ell}(x_\ell), \theta_1\rangle$, where $\phi_\ell = \phi \circ \psi$ and ϕ is the Fourier map with $m = \lfloor |E|/2 \rfloor$. To regularize the parameter $\theta_{1,\ell}$, we set M_ℓ as the identity matrix, which corresponds to applying a ridge penalty.

Overall, similar to GAMs, WeaKL can be optimized to fit additive models with both linear and nonlinear effects. The parameter $\hat{\theta}$ of the WeaKL can then be computed using (5) with

$$M = \begin{pmatrix} \sqrt{\lambda_1} M_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sqrt{\lambda_{d_1}} M_{d_1} \end{pmatrix}.$$

To stress that this WeaKL results from the enforcement of additive constraints, we call it the *additive WeaKL*. Note that, contrary to GAMs where identifiability issues must be addressed (Wood, 2017), WeaKL does not require further regularization, since $\hat{\theta}$ is the unique minimizer of the empirical risk L. Note that the hyperparameters λ_ℓ, along with the number m of Fourier modes and the choice of feature maps ϕ_ℓ, can be determined by model selection, as described in Appendix D.1.

Online adaption after a break: Online WeaKL. For many time series, the dependence of Y on X can vary over time. For example, the behavior of Y may change rapidly following extreme events, resulting in structural breaks. A notable example is the shift in electricity demand during the COVID-19 lockdowns, as illustrated in use case 1. To provide a clear mathematical framework, we assume that the distribution of (X, Y) follows an additive model that evolves smoothly over time. Formally, considering (t, X_t) as a feature vector, we assume that

$$f^\star(t, x_1, \dots, x_{d_1}) = h_0^\star(t) + \sum_{\ell=1}^{d_1} (1 + h_\ell^\star(t)) g_\ell^\star(x_\ell), \tag{7}$$

where $g_\ell^\star$ and $h_\ell^\star$ are univariate functions. This model forms the core of the Kalman-Viking algorithm (de Vilmarest and Wintenberger, 2024), which has demonstrated state-of-the-art performance in forecasting electricity demand and renewable energy production (Obst et al., 2021; de Vilmarest and Goude, 2022; de Vilmarest et al., 2024).

We assume that we have at hand estimators $\hat{g}_\ell$ of $g_\ell^\star$ that we want to update over time. For example, these estimators can be obtained by fitting an additive WeaKL model, initially assuming $h_\ell^\star = 0$. The functions $h_\ell^\star$ are then estimated by minimizing the empirical risk

$$L(\theta) = \frac{1}{n} \sum_{j=1}^{n} \Big| h_{\theta_0}(t_j) + \sum_{\ell=1}^{d_1} (1 + h_{\theta_\ell}(t_j)) \hat{g}_\ell(X_{\ell, t_j}) - Y_{t_j} \Big|^2 + \sum_{0 \le \ell \le d_1} \lambda_\ell \|h_{\theta_\ell}\|_{H^s}^2, \tag{8}$$

where $\lambda_\ell > 0$ are hyperparameters regularizing the smoothness of the functions h_{θ_ℓ}. Here, $h_\theta(t) = \langle \phi(t), \theta\rangle$, and ϕ is the Fourier map $\phi(t) = (\exp(ikt/2))^\top_{-m \le k \le m}$. The prior $h_{\theta_\ell} \simeq 0$ reflects the idea that the best a priori estimate of Y's behavior follows the stable additive model. Defining $W_t = Y_t - \sum_{\ell=1}^{d_1} \hat{g}_\ell(X_{\ell,t})$, the empirical risk can be reformulated as

$$L(\theta) = \frac{1}{n} \sum_{j=1}^{n} |\langle \phi_1(t_j, X_{t_j}), \theta\rangle - W_{t_j}|^2 + \|M\theta\|_2^2,$$

where $\phi_1(t, X_t) = ((\exp(ikt/2))_{-m\leq k\leq m}, (\hat{g}_\ell(X_{\ell,t}) \exp(ikt/2))_{-m\leq k\leq m})_{\ell=1}^{d_1})^\top \in \mathbb{C}^{(2m+1)(d_1+1)}$,

$$M = \begin{pmatrix} \sqrt{\lambda_0}D & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sqrt{\lambda_{d_1}}D \end{pmatrix},$$

and D is the $(2m+1)\times(2m+1)$ diagonal matrix $D = \mathrm{Diag}((\sqrt{1+k^{2s}})_{-m\leq k\leq m})$. From (5), we deduce that the unique minimizer of the empirical loss L is

$$\hat{\theta} = (\Phi^*\Phi + nM^*M)^{-1}\Phi^*\mathbb{W}, \tag{9}$$

where $\mathbb{W} = (W_{t_1}, \ldots, W_{t_n})^\top \in \mathbb{R}^n$.

This formulation allows to forecast the time series Y at the next time step, t_{n+1}, using

$$\begin{aligned} \hat{Y}_{t_{n+1}} &= f_{\hat{\theta}}(t_{n+1}, X_{t_{n+1}}) = \langle \phi_1(t_{n+1}, X_{t_{n+1}}), \hat{\theta}\rangle \\ &= h_{\hat{\theta}_0}(t_{n+1}) + \sum_{\ell=1}^{d_1}(1 + h_{\hat{\theta}_\ell}(t_{n+1}))\hat{g}_\ell(X_{\ell,t_{n+1}}). \end{aligned}$$

Since the model is continuously updated over time, this corresponds to an online learning setting. To emphasize that Equation (9) arises from an online adaptation process, we refer to this model as the *online WeaKL*. Unlike the Viking algorithm of de Vilmarest and Wintenberger (2024), which approximates the minimizer of the empirical risk through an iterative process, online WeaKL offers a closed-form solution and exploits GPU parallelization for significant speedups. As shown in Section 3.2, our approach leads to improved performance in electricity demand forecasting.

3.2 Application to electricity load forecasting

In this subsection, we apply shape constraints to two use cases in electricity demand forecasting and demonstrate the effectiveness of our approach. In these electricity demand forecasting problems, the focus is on short-term forecasting, with particular emphasis on the recent non-stationarities caused by the COVID-19 lockdowns and by the energy crisis.

Electricity load forecasting and non-stationarity. Accurate demand forecasting is critical due to the costly nature of electricity storage, coupled with the need for supply to continuously match demand. Short-term load forecasting, especially for 24-hour horizons, is particularly valuable for making operational decisions in both the power industry and electricity markets. Although the cost of forecasting errors is difficult to quantify, a 1% reduction in error is estimated to save utilities several hundred thousand USD per gigawatt of peak demand (Hong and Fan, 2016). Recent events such as the COVID-19 shutdown have significantly affected electricity demand, highlighting the need for updated forecasting models (Zarbakhsh et al., 2022).

Use case 1: Load forecasting during COVID. In this first use case, we test the performance of our WeaKL on the IEEE DataPort Competition on Day-Ahead Electricity Load Forecasting (Farrokhabadi et al., 2022). Here, the goal is to forecast the electricity demand of an unknown country during the period following the Covid-19 lockdown. The winning model of this competition was the Viking model of Team 4 (de Vilmarest and Goude, 2022), with a mean absolute error (MAE) of 10.9 gigawatts (GW). For comparison, a direct translation of their model into the online WeaKL framework—using the same features and maintaining the same additive effects—results in an MAE of 10.5 GW. In parallel, we also apply the online WeaKL methodology without relying on the variables selected by de Vilmarest and Goude (2022). Instead, we determine the optimal hyperparameters λ_ℓ and select the feature maps ϕ_ℓ through a hyperparameter tuning process (see Appendix D.1). This leads to a different selected model with a MAE of 9.9 GW (see Appendix D.4 for a complete description of the models). Thus, the online WeaKL given by (9) outperforms the state-of-the-art by 9%. As done in the IEEE competition (Farrokhabadi et al., 2022), we assess the significance of this result by evaluating the MAE skill score using a block bootstrap approach (see Appendix D.4). It shows that the online WeaKL outperforms the winning model proposed by de Vilmarest and Goude (2022) with a probability above 90%. The updated results of the competition are presented in Table 1. Note that a great variety of models were benchmarked in this competition, like Kalman filters (Team 4), autoregressive models (Teams 4 and 7), random forests (Teams 4 and 6), gradient boosting (Teams 6 and 36), deep residual networks (Team 19), and averaging (Team 13).

Table 1: Performance of the online WeaKL and of the top 10 participants of the IEEE competiton. A specific bootstrap test shows that the WeaKL significantly outperform the winning team.

Team	WeaKL	4	14	7	36	19	23	9	25	13	26
MAE (GW)	**9.9**	10.9	11.8	11.9	12.3	12.3	13.9	14.2	14.3	14.6	15.4

Use case 2: Load forecasting during the energy crisis. In this second use case, we evaluate the performance of our WeaKL within the open source benchmark framework proposed by Doumèche et al. (2023). This benchmark provides a comprehensive evaluation of electricity demand forecasting models, incorporating the GAM boosting model of Taieb and Hyndman (2014), the GAM of Obst et al. (2021), the Kalman models of de Vilmarest and Goude (2022), the time series random forests of Goehry et al. (2023), and the Viking model of de Vilmarest et al. (2024). The goal here is to forecast the French electricity demand during the energy crisis in the winter of 2022-2023. Following the war in Ukraine and maintenance problems at nuclear power plants, electricity prices reached an all-time high at the end of the summer of 2022. In this context, French electricity demand decreased by 10% compared to its historical trends (Doumèche et al., 2023). This significant shift in electricity demand can be interpreted as a structural break, which justifies the application of the online WeaKL given by (9).

In this benchmark, the models are trained from 8 January 2013 to 1 September 2022, and then evaluated from 1 September 2022 to 28 February 2023. The dataset consists of temperature data from the French meteorological administration Météo-France (2023), and electricity demand data from the French transmission system operator RTE (2023), sampled with a half-hour resolution. This translates into the feature variable

$$X = (\mathrm{Load}_1, \mathrm{Load}_7, \mathrm{Temp}, \mathrm{Temp}_{950}, \mathrm{Temp}_{\max 950}, \mathrm{Temp}_{\min 950}, \mathrm{ToY}, \mathrm{DoW}, \mathrm{Holiday}, t),$$

where Load_1 and Load_7 are the electricity demand lagged by one day and seven days, Temp is the temperature, and Temp_{950}, $\mathrm{Temp}_{\max 950}$, and $\mathrm{Temp}_{\min 950}$ are smoothed versions of Temp. The time of year $\mathrm{ToY} \in \{1, \ldots, 365\}$ encodes the position within the year. The day of the week $\mathrm{DoW} \in \{1, \ldots, 7\}$ encodes the position within the week. In addition, $\mathrm{Holiday}$ is a boolean variable set to one during holidays, and t is the timestamp. Here, the target $Y = \mathrm{Load}$ is the electricity demand, so $d_1 = 10$ and $d_2 = 1$.

We compare the performance of two of our WeaKLs against this benchmark. First, our additive WeaKL is a direct translation of the GAM formula proposed by Obst et al. (2021) into the additive WeaKL framework given by (6). Thus, $f_\theta(x) = \sum_{\ell=1}^{10} g_\ell(x_\ell)$, where:

- the effects g_1, g_2, and g_{10} of Load_1, Load_7, and t are linear,
- the effects $g_3, \dots, g_7$ of Temp, Temp_{950}, $\mathrm{Temp}_{\max 950}$, $\mathrm{Temp}_{\min 950}$, and ToY are nonlinear with $m = 10$,
- the effects g_8 and g_9 of DoW and Holiday are categorical with $|E| = 7$ and $|E| = 2$.

The weights θ are learned using data from 2013 to 2021, while the optimal hyperparameters $\lambda_1, \dots, \lambda_{10}$ are tuned using a validation set covering the period from 2021 to 2022. Once the additive WeaKL is learned, it becomes straightforward to interpret the impact of each feature on the model. For example, the effect $\hat{g}_3 : \mathrm{Temp} \mapsto \langle \phi_{1,3}(\mathrm{Temp}), \hat{\theta}_{1,3} \rangle$ of the rescaled temperature feature ($\mathrm{Temp} \in [-\pi, \pi]$) is illustrated in Figure 1.

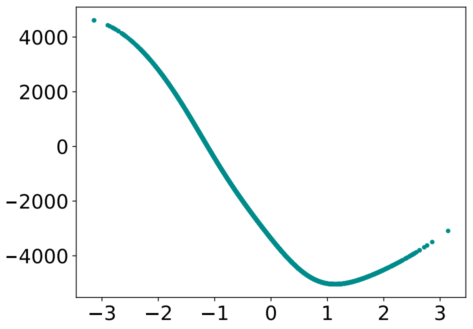

Figure 1: Effect in MW of the temperature in the additive WeaKL.

Second, our online WeaKL is the online adaptation of f_θ in response to a structural break, as described by (9). The hyperparameters $\lambda_0, \dots, \lambda_{10}$ in (8) are chosen to minimize the error over a validation period from 1 April 2020 to 1 June 2020, corresponding to the first COVID-19 lockdown. Note that this validation period does not immediately precede the test period, which is uncommon in time series analysis. However, this choice ensures that the validation period contains a structural break, making it as similar as possible to the test period. Next, the functions $h_0, \dots, h_{10}$ in (7) are trained on a period starting from 1 July 2020, and updated online.

The results are summarized in Table 2. The errors and their standard deviations are assessed by stationary block bootstrap (see Appendix D.2). Since holidays are notoriously difficult to predict, performance is evaluated over the entire period (referred to as *Including holidays*), and separately excluding holidays and the days immediately before and after (referred to as *Excluding holidays*). Over both test periods, the additive WeaKL significantly outperforms the GAM, while the online WeaKL outperforms the state-of-the-art by more than 10% across all metrics.

Figure 2 shows the errors of the WeaKLs as a function of time during the test period, which includes holidays. During the sobriety period, electricity demand decreased, causing the additive WeaKL to overestimate demand, resulting in a negative bias. Interestingly, this bias is effectively corrected by the online WeaKL, which explains its strong performance. This shows that the online update of the effects effectively corrects biases caused by shifts in the data distribution.

Then, we compare the running time of the algorithms. Note that, during hyperparameter tuning, the GPU implementation of WeaKL makes it possible to train 1.6×10^5 additive WeaKL over a period of eight years in less than five minutes on a single standard GPU (NVIDIA $L4$).

Table 2: Benchmark for load forecasting during the energy crisis

	Including holidays		Excluding holidays	
	RMSE (GW)	MAPE (%)	RMSE (GW)	MAPE (%)
Statistical model				
Persistence (1 day)	4.0±0.2	5.5±0.3	4.0±0.2	5.0±0.3
SARIMA	2.4±0.2	3.1±0.2	2.0±0.2	2.6±0.2
GAM	2.3±0.1	3.5±0.2	1.70±0.06	2.6±0.1
Data assimilation				
Static Kalman	2.1±0.1	3.1±0.2	1.43±0.05	2.20±0.08
Dynamic Kalman	1.4±0.1	1.9±0.1	1.10±0.04	1.58±0.05
Viking	1.5±0.1	1.8±0.1	0.98±0.04	1.33±0.04
Aggregation	1.4±0.1	1.8±0.1	0.96±0.04	1.36±0.04
Machine learning				
GAM boosting	2.6±0.2	3.7±0.2	2.3±0.1	3.3±0.2
Random forests	2.5±0.2	3.5±0.2	2.1±0.1	3.0±0.1
Random forests + bootstrap	2.2±0.2	3.0±0.2	1.9±0.1	2.6±0.1
WeaKLs				
Additive WeaKL	1.95±0.08	3.0 ±0.1	1.55±0.06	2.32±0.09
Online WeaKL	**1.14±0.09**	**1.5±0.1**	**0.87±0.04**	**1.17±0.05**

As for the online WeaKL, the training is more computationally intensive because the model must be updated in an online fashion. However, training 9.2×10^3 online WeaKLs over a period of two years takes less than two minutes. This approach is faster than the Viking algorithm, which takes over 45 minutes to evaluate the same number of parameter sets on the same dataset, even when using 10 CPUs in parallel. A detailed comparison of the running times for all algorithms is provided in Appendix D.5.

Both use cases demonstrate that WeaKL models are very powerful. Not only are they highly interpretable—thanks to their ability to fit into a common framework and produce simple formulas—but they are also competitive with state-of-the-art techniques in terms of both optimization efficiency (they can run on GPUs) and performance (measured by MAPE and RMSE).

4 Learning constraints

4.1 Mathematical formulation

Section 3 focused on imposing constraints on the shape of the regression function $f^\star$. In contrast, the goal of the present section is to impose constraints on the parameter θ. We begin with a general method to enforce linear constraints on θ, and subsequently apply this framework to transfer learning, hierarchical forecasting, and differential constraints.

Linear constraints. Here, we assume that $f^\star$ satisfies a linear constraint. By construction of f_θ in (1), such a linear constraint directly translates into a constraint on θ. For example, the linear constraint $f^{\star 1}(X_t) = 2f^{\star 2}(X_t)$ can be implemented by enforcing $\theta_1 = 2\theta_2$. Thus, in the following,

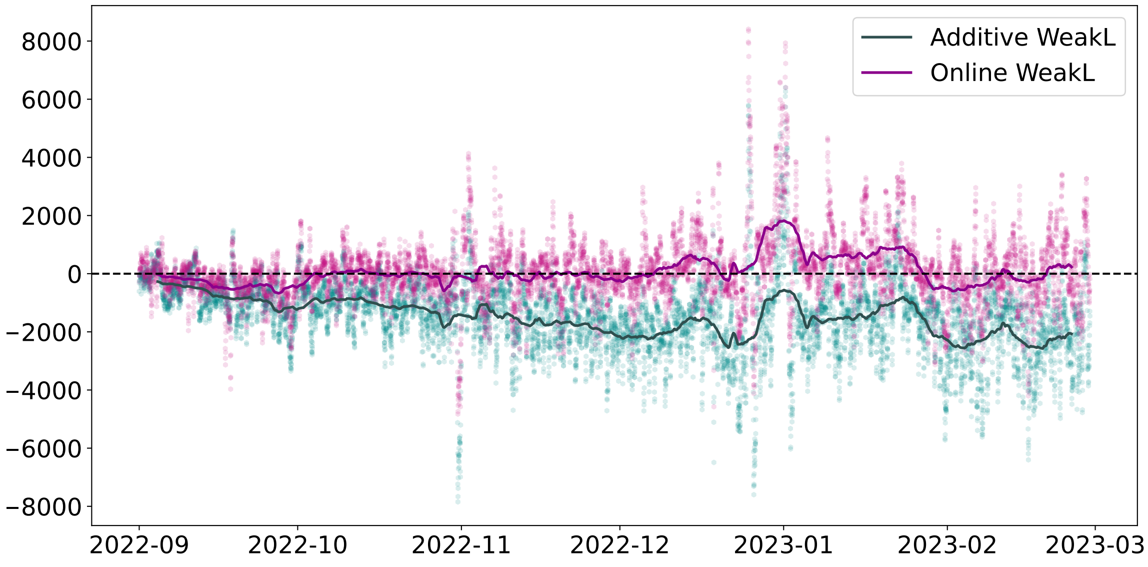

Figure 2: Error $Y_t - \hat{Y}_t$ in MW of the WeaKLs on the test period including holidays. Dots represent individual observations, while the bold curves indicate the one-week moving averages.

we assume a prior on θ in the form of a linear constraint. Formally, we want to enforce that $\theta \in \mathcal{S}$, where $\mathcal{S}$ is a known linear subspace of $\mathbb{C}^{\dim(\theta)}$. Given an injective $\dim(\theta) \times \dim(\mathcal{S})$ matrix P such that $\mathrm{Im}(P) = \mathcal{S}$, then, as shown in Lemma A.2, $\|C\theta\|_2^2$ is the square of the Euclidean distance between θ and $\mathcal{S}$, where $C = \mathrm{I}_{\dim(\theta)} - P(P^*P)^{-1}P^*$. In particular, $\|C\theta\|_2^2 = 0$ is equivalent to $\theta \in \mathcal{S}$, and $\|C\theta\|_2^2 = \|\theta\|_2^2$ if $\theta \in \mathcal{S}^\perp$. From this observation, there are two ways to enforce $\theta \in \mathcal{S}$ in the empirical risk (2).

On the one hand, suppose that $f^\star$ exactly satisfies the linear constraint. This happens in particular when the constraint results from a physical law. For example, to build upon the use cases of Section 3.2, assume that we want to forecast the electricity load of different regions of France, i.e., the target $Y \in \mathbb{R}^3$ is such that Y_1 is the load of southern France, Y_2 is the load of northern France, and $Y_3 = Y_1 + Y_2$ is the national load. This prototypical example of hierarchical forecasting is presented in Section C, where we show how incorporating even a simple constraint can significantly improve the model's performance. In this example, we know that $f^\star$ satisfies the constraint $f^{\star 3} = f^{\star 1} + f^{\star 2}$. When dealing with such exact priors, a sound approach is to consider only parameters θ such that $C\theta = 0$, or equivalently, $\theta = P\theta'$. Letting Π_ℓ be the $D_\ell \times \dim(\theta)$ projection matrix such that $\theta_\ell = \Pi_\ell\theta$, we have $\langle \phi_\ell(X_t), \theta_\ell \rangle = \langle \phi_\ell(X_t), \Pi_\ell\theta \rangle = \langle P^*\Pi_\ell^*\phi_\ell(X_t), \theta' \rangle$. Thus, minimizing the empirical risk (2) over $\theta' \in \mathbb{C}^{\dim(\mathcal{S})}$ simply requires changing ϕ_ℓ to $P^*\Pi_\ell^*\phi_\ell$, which is equivalent to replacing Φ_t with $\Phi_t P$ in (3).

On the other hand, suppose that the linear constraint serves as a good but inexact prior. For example, building on the last example, let X_t be the average temperature in France at time t. We expect the loads Y_1 in southern France and Y_2 in northern France to behave similarly. In both regions, lower temperatures lead to increased heating usage (and thus higher loads), while higher temperatures result in increased cooling usage (also leading to higher loads). Therefore, $f^{\star 1}$ and $f^{\star 2}$ share the same shape, resulting in the prior $f^{\star 1} \simeq f^{\star 2}$. This prototypical example of transfer learning is explored in the following paragraphs. Such inexact constraints can be enforced by adding a penalty $\lambda\|C\theta\|_2^2$ in the empirical risk (2), where $\lambda > 0$ is an hyperparameter. (Equivalently, this only consists in replacing M with $(\sqrt{\lambda}C^\top \mid M^\top)^\top$ in (2).) This ensures that $\|C\hat{\theta}\|_2^2$ is small, while allowing the model to learn functions that do not exactly satisfy the constraint.

These approaches are statistically sound, since under the assumption that $Y_t = f_{\theta^\star}(X_t) + \varepsilon_t$, where $\theta^\star \in \mathcal{S}$, both estimators have lower errors compared to unconstrained regression. This is true in the sense that, almost surely,

$$\frac{1}{n}\sum_{j=1}^{n} \|f_{\theta^\star}(X_{t_j}) - f_{\hat{\theta}_C}(X_{t_j})\|_2^2 + \|M(\theta^\star - \hat{\theta}_C)\|_2^2 \leq \frac{1}{n}\sum_{j=1}^{n} \|f_{\theta^\star}(X_{t_j}) - f_{\hat{\theta}}(X_{t_j})\|_2^2 + \|M(\theta^\star - \hat{\theta})\|_2^2,$$

where $\hat{\theta}$ is the unconstrained WeaKL and $\hat{\theta}_C$ is a WeaKL integrating the constraint $C\theta^\star \simeq 0$ (see Proposition A.3 and Remark A.4).

Transfer learning. Transfer learning is a framework designed to exploit similarities between different prediction tasks when $d_2 > 1$. The simplest case involves predicting multiple targets $Y_1, \dots, Y_{d_2}$ with similar features $X_1, \dots, X_{d_2}$. For example, suppose we want to forecast the electricity demand of d_2 cities. Here, Y_ℓ is the electricity demand of the city ℓ, while X_ℓ is the average temperature in city ℓ. The general function $f^\star$ estimating $(Y_1, \dots, Y_{d_2})$ can be expressed as $f^\star(X) = f^\star(X_1, \dots, X_{d_2}) = (f^{\star 1}(X_1), \dots, f^{\star d_2}(X_{d_2}))$. The transfer learning assumption is $f^{\star 1} \simeq \cdots \simeq f^{\star d_2}$. Equivalently, this corresponds to the linear constraint $\theta \in \mathrm{Im}(P)$, where $P = (\mathrm{I}_{2m+1} \mid \cdots \mid \mathrm{I}_{2m+1})^\top$ is a $(2m+1)d_1 \times (2m+1)$ matrix. Thus, one can apply the framework of the last paragraph on linear constraints as inexact prior using P.

Hierarchical forecasting. Hierarchical forecasting involves predicting multiple time series that are linked by summation constraints. This approach was introduced by Athanasopoulos et al. (2009) to forecast Australian domestic tourism. Tourism can be analyzed at various geographic scales. For example, at time t, one could consider the total number $Y_{A,t}$ of tourists in Australia, and the number $Y_{S_i,t}$ of tourists in each of the seven Australian states $S_1, \dots, S_7$. By definition, $Y_{A,t}$ is the sum of the $Y_{S_i,t}$, which leads to the exact summation constraint $Y_{A,t} = \sum_{i=1}^{7} Y_{S_i,t}$. Furthermore, since each state S_i is composed of z_i zones $Z_{i,1}, \dots, Z_{i,z_i}$, an additional hierarchical level can be introduced. Note that the number of zones depends on the state, for a total of 27 zones. This results in another set of summation constraints $Y_{S_i,t} = Y_{Z_{i,1},t} + \cdots + Y_{Z_{i,z_i},t}$. Overall, the complete set of summation constraints can be represented by a directed acyclic graph, as shown in Figure 3. Alternatively, these constraints can be expressed by a 35×27 summation matrix S that connects the bottom-level series $Y_b = (Y_{Z_{1,1}}, \dots, Y_{Z_{7,z_7}})^\top \in \mathbb{R}^{27}$ to all hierarchical nodes $Y = (Y_{Z_{1,1}}, \dots, Y_{Z_{7,z_7}}, Y_{S_1}, \dots, Y_{S_7}, Y_A)^\top \in \mathbb{R}^{35}$ through the relation $Y = SY_b$. Thus, by letting $\mathbb{1} = (1, \dots, 1)^\top \in \mathbb{R}^{27}$, and defining $\mathbb{1}^{(j)} \in \mathbb{R}^{27}$ by $\mathbb{1}^{(j)}_i = \begin{cases} 1 & \text{if } \sum_{k=1}^{j-1} z_k \leq i \leq \sum_{k=1}^{j} z_k \\ 0 & \text{otherwise} \end{cases}$, we have that $S = (\mathrm{I}_{27} \mid \mathbb{1}^{(1)} \mid \cdots \mid \mathbb{1}^{(7)} \mid \mathbb{1})^\top$. The goal of hierarchical forecasting is to take advantage of the summation constraints defined by S to improve the predictions of the vector Y representing all hierarchical nodes.

This context can be easily generalized to many time series forecasting tasks. Spatial summation constraints, which divide a geographic space into different subspaces, have been applied in areas such as electricity demand forecasting (Brégère and Huard, 2022), electric vehicle charging demand

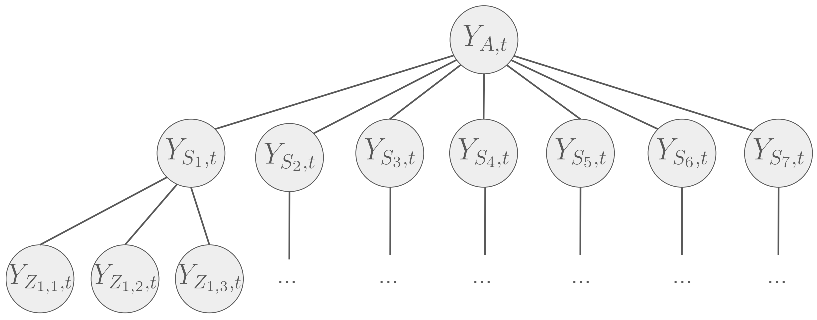

Figure 3: Graph representing the hierarchy of Australian domestic tourism.

forecasting (Amara-Ouali et al., 2024), and tourism forecasting (Wickramasuriya et al., 2019). Summation constraints also arise in multi-horizon forecasting, where, for example, an annual forecast must equal the sum of the corresponding monthly forecasts (Kourentzes and Athanasopoulos, 2019). Finally, they also appear when goods are categorized into different groups (Pennings and van Dalen, 2017).

There are two main approaches to hierarchical forecasting. The first, known as forecast reconciliation, attempts to improve an existing estimator $\hat{Y}$ of the hierarchical nodes Y by multiplying $\hat{Y}$ by a so-called reconciliation matrix P, so that the new estimator $P\hat{Y}$ satisfies the summation constraints. Formally, it is required that $\mathrm{Im}(P) \subseteq \mathrm{Im}(S)$, where S is the summation matrix. The goal is for $P\hat{Y}$ to have less error than $\hat{Y}$. The strengths of this approach are its low computational cost and its ability to seamlessly integrate with pre-existing forecasts. Various reconciliation matrices, such as the orthogonal projection $P = S(S^\top S)^{-1}S$ on $\mathrm{Im}(S)$ (see the paragraph above on linear constraints), have been shown to reduce forecasting errors and to even be optimal under certain assumptions (Wickramasuriya et al., 2019). Another complementary approach is to incorporate the hierarchical structure of the problem directly into the training of the initial estimator $\hat{Y}$ (Rangapuram et al., 2021). While this method is more computationally intensive, it provides a more comprehensive solution than reconciliation methods because it uses the hierarchy not only to shape the regression function, but also to inform the learning of its parameters. In this paper, we build on this approach to design three new estimators, all of which are implemented in Section 4.2.

As for now, we denote by ℓ_1 the total number of nodes and $\ell_2 \leq \ell_1$ the number of bottom nodes. Thus, $Y = (Y_\ell)^\top_{1\leq\ell\leq\ell_1}$ represents the global vector of all nodes, while $Y_b = (Y_\ell)^\top_{1\leq\ell\leq\ell_2}$ represents the vector of the bottom nodes. The $\ell_1 \times \ell_2$ summation matrix S is defined so that, for all time index t, the summation identity $Y_t = SY_{b,t}$ is satisfied.

Estimator 1. Bottom-up approach: WeaKL-BU. In the bottom-up approach, models are fitted only for the bottom-level series Y_b, resulting in a vector of estimators $\hat{Y}_b$. The remaining levels are then estimated by $\hat{Y} = S\hat{Y}_b$, where S is the summation matrix.

To achieve this, forecasts for each bottom node $1 \leq \ell \leq \ell_2$ are constructed using a set of explanatory variables $X_\ell \in \mathbb{R}^{d_\ell}$ specific to that node. Together, these explanatory variables $X_1, \ldots, X_{\ell_2}$ form the feature $X \in \mathbb{R}^{d_1+\cdots+d_{\ell_2}}$. A straightforward choice of features are the lags of the target variable, i.e., $X_{\ell,t} = Y_{\ell,t-1}$, though many other choices are possible. Next, for each bottom node $1 \leq \ell \leq \ell_2$, we fit a parametric model $f_{\theta_\ell}(X_{\ell,t})$ to predict the series $Y_{\ell,t}$. Each function f_{θ_ℓ} is parameterized by a mapping ϕ_ℓ (e.g., a Fourier map or an additive model) and a coefficient vector θ_ℓ, such that $f_{\theta_\ell}(X_{\ell,t}) = \langle \phi_\ell(X_{\ell,t}), \theta_\ell \rangle$. Therefore, the model for the lower nodes $Y_{b,t}$ can be expressed as $\Phi_t\theta$, where $\theta = (\theta_1, \dots, \theta_{\ell_2})^\top$ is the vector of all coefficients, and Φ_t is the feature matrix at time t defined in (4). Overall, the model for all levels $Y_t = SY_{b,t}$ is $S\Phi_t\theta$, and the empirical risk corresponding to this problem is given by

$$L(\theta) = \frac{1}{n}\sum_{j=1}^{n} \|\Lambda(S\Phi_{t_j}\theta - Y_{t_j})\|_2^2 + \|M\theta\|_2^2,$$

where Λ is a $\ell_1 \times \ell_1$ diagonal matrix with positive coefficients, and M is a penalty matrix that depends on the ϕ_ℓ mappings, as in Section 3.

Since Λ scales the relative importance of each node in the learning process, the choice of its coefficients plays a critical role in the performance of the estimator. In the experimental Section 4.2, Λ will be learned through hyperparameter tuning. Typically, $\Lambda_{\ell,\ell}$ should be large when $\mathrm{Var}(Y_\ell|X_\ell)$ is low—that is, the more reliable Y_ℓ is as a target (Wickramasuriya et al., 2019). From (4), we deduce that the minimizer $\hat\theta$ of the empirical risk is

$$\hat\theta = \Big(\Big(\sum_{j=1}^{n} \Phi_{t_j}^* S^*\Lambda^*\Lambda S\Phi_{t_j}\Big) + nM^*M\Big)^{-1} \sum_{j=1}^{n} \Phi_{t_j}^*\Lambda^*\Lambda Y_{t_j}. \quad (10)$$

We call $\hat\theta$ the *WeaKL-BU*. Setting $\Lambda = \mathrm{I}_{\ell_1}$, i.e., the identity matrix, results in treating all hierarchical levels equally, which is the setup of Rangapuram et al. (2021). On the other hand, setting $\Lambda_{\ell,\ell} = 0$ for all $\ell \geq \ell_2$ leads to learning each bottom node independently, without using any information from the hierarchy. This is the traditional bottom-up approach.

Estimator 2. Global hierarchy-informed approach: WeaKL-G. The context is similar to the bottom-up approach, but here models are fitted for all nodes $1 \leq \ell \leq \ell_1$, using local explanatory variables $X_\ell \in \mathbb{R}^{d_\ell}$, where $d_\ell \geq 1$. Thus, the model for Y_t is given by $\Phi_t\theta$, where $\theta = (\theta_1, \dots, \theta_{\ell_1})^\top$ is the vector of coefficients and Φ_t is the feature matrix at time t defined in (4). To ensure that the hierarchy is respected, we introduce a penalty term:

$$\|\Gamma(S\Pi_b\Phi_t\theta - \Phi_t\theta)\|_2^2 = \|\Gamma(S\Pi_b - \mathrm{I}_{\ell_1})\Phi_t\theta\|_2^2,$$

where Γ is a positive diagonal matrix and Π_b is the projection operator on the bottom level, defined as $\Pi_b\theta = (\theta_1, \dots, \theta_{\ell_2})^\top$. As in the bottom-up case, Γ encodes the level of trust assigned to each node. In Section 4.2, we learn Γ through hyperparameter tuning. This results in the empirical risk

$$L(\theta) = \frac{1}{n}\sum_{j=1}^{n} \|\Phi_{t_j}\theta - Y_{t_j}\|_2^2 + \frac{1}{n}\sum_{j=1}^{n} \|\Gamma(S\Pi_b - \mathrm{I}_{\ell_1})\Phi_{t_j}\theta\|_2^2 + \|M\theta\|_2^2.$$

where M is a penalty matrix that depends on the ϕ_ℓ mappings, as in Section 3. This empirical risk is similar to the one proposed by Zheng et al. (2023), where a penalty term is used to enforce hierarchical coherence during the learning process. From (4), we deduce that the minimizer is given by

$$\hat\theta = \Big(\sum_{j=1}^{n}(\Phi_{t_j}^*\Phi_{t_j} + \Phi_{t_j}^*(\Pi_b^*S^* - \mathrm{I}_{\ell_1})\Gamma^*\Gamma(S\Pi_b - \mathrm{I}_{\ell_1})\Phi_{t_j}) + nM^*M\Big)^{-1} \sum_{j=1}^{n} \Phi_{t_j}^* Y_t. \quad (11)$$

We refer to $\hat{\theta}$ as the *WeaKL-G*. The fundamental difference between (10) and (11) is that the WeaKL-BU estimator only learns parameters for the ℓ_2 bottom nodes, whereas the WeaKL-G estimators learns parameters for all nodes. We emphasize that WeaKL-BU and WeaKL-G follow different approaches. While WeaKL-BU adjusts the lower-level nodes and then uses the summation matrix S to estimate the higher levels, WeaKL-G relies directly on global information, which is subsequently penalized by S. In the next paragraph, we complement the WeaKL-BU estimator by adding transfer learning constraints.

Estimator 3. Hierarchy-informed transfer learning: WeaKL-T. In many hierarchical forecasting applications, the targets Y_ℓ are of the same nature throughout the hierarchy. Consequently, we often expect them to be explained by similar explanatory variables X_ℓ and to have similar regression functions estimators $f_{\hat{\theta}_\ell}$ (e.g., Leprince et al., 2023). For this reason, we propose an algorithm that combines WeaKL-BU with transfer learning.

Therefore, we assume that there is a subset $J \subseteq \{1, \dots, \ell_2\}$ of similar nodes and weights $(\alpha_i)_{i\in J}$ such that we expect $\alpha_i f_{\hat{\theta}_i}(X_{i,t}) \simeq \alpha_j f_{\hat{\theta}_j}(X_{j,t})$ for $i, j \in J$. In particular, there is an integer D such that $\theta_j \in \mathbb{C}^D$ for all $j \in J$. Therefore, denoting by Π_J the projection on J such that $\Pi_J\theta = (\theta_j)_{j\in J} \in \mathbb{C}^{D|J|}$, this translates into the constraint that $\Pi_J\theta \in \mathrm{Im}(M_J)$ where $M_J = (\alpha_1 \mathrm{I}_D, \dots, \alpha_{|J|}\mathrm{I}_D)^\top$. As explained in the paragraph on linear constraints, we enforce this inexact constraint by penalizing the empirical risk with the addition of the term $\|(\mathrm{I}_{D|J|} - P_J)\Pi_J\theta\|_2^2$, where $P_J = M_J(M_J^* M_J)^{-1}M_J^*$ is the orthogonal projection onto the image of M_J. This leads to the empirical risk

$$L(\theta) = \frac{1}{n}\sum_{j=1}^{n} \|\Lambda(S\Phi_{t_j}\theta - Y_{t_j})\|_2^2 + \lambda\|(\mathrm{I}_{D|J|} - P_J)\Pi_J\theta\|_2^2 + \|M\theta\|_2^2,$$

where M is a penalty matrix that depends on the ϕ_ℓ mappings, as in Section 3. We call *WeaKL-T* the minimizer $\hat{\theta}$ of L. It is given by

$$\hat{\theta} = \Big(\Big(\sum_{j=1}^{n} \Phi_{t_j}^* S^*\Lambda^*\Lambda S\Phi_{t_j}\Big) + n\lambda\Pi_J^*(\mathrm{I}_{D|J|} - P_J)\Pi_J + nM^*M\Big)^{-1}\sum_{j=1}^{n}\Phi_{t_j}^*\Lambda^*\Lambda Y_{t_j}. \qquad (12)$$

4.2 Application to tourism forecasting

Hierarchical forecasting and tourism. In this experiment, we aim to forecast Australian domestic tourism using the dataset from Wickramasuriya et al. (2019). The dataset includes monthly measures of Australian domestic tourism from January 1998 to December 2016, resulting in $n = 216$ data points. Each month, domestic tourism is measured at four spatial levels and one categorical level, forming a five-level hierarchy. At the top level, tourism is measured for Australia as a whole. It is then broken down spatially into 7 states, 27 zones, and 76 regions. Then, for each of the 76 regions, four categories of tourism are distinguished according to the purpose of travel: holiday, visiting friends and relatives (VFR), business, and other. This gives a total of five levels (Australia, states, zones, regions, and categories), with $\ell_2 = 76 \times 4 = 304$ bottom nodes, and $\ell_1 = 1 + 7 + 27 + 76 + \ell_2 = 415$ total nodes.

Benchmark. The goal is to forecast Australian domestic tourism one month in advance. Models are trained on the first 80% of the dataset and evaluated on the last 20%. Similar to Wickramasuriya et al. (2019), we only consider autoregressive models with lags from one month to two years. This setting is particularly interesting because, although each time series can be reasonably fitted using the 216 data points, the total number of targets ℓ_1 exceeds n. Consequently, the higher levels cannot be naively learned from the lags of the bottom level time series through linear regression.

The bottom-up (BU) model involves running 304 linear regressions $\hat{Y}^{\mathrm{BU}}_{\ell,t} = \sum_{j=1}^{24} a_{\ell,j} Y_{\ell,t-j}$ for $1 \leq \ell \leq \ell_2$, where $Y_{\ell,t-j}$ is the lag of $Y_{\ell,t}$ by j months. The final forecast is then computed as $\hat{Y}^{\mathrm{BU}}_{t} = S\hat{Y}^{\mathrm{BU}}_{\ell,t}$, where S is the summation matrix. The Independent (Indep) model involves running separate linear regressions for each target time series using its own lags. This results in 415 linear regressions of the form $\hat{Y}^{\mathrm{Indep}}_{\ell,t} = \sum_{j=1}^{24} a_{\ell,j} Y_{\ell,t-j}$ for $1 \leq \ell \leq \ell_1$. Rec-OLS is the estimator resulting from OLS adjustment of the Indep estimator, i.e., taking $P = S(S^*S)^{-1}S$ (Wickramasuriya et al., 2019). MinT refers to the estimator derived from the minimum trace adjustment of the Indep estimator (see MinT(shrinkage) in Wickramasuriya et al., 2019). PIKL-BU refers to the estimator (10), where, for all $1 \leq \ell \leq 304$, $X_{\ell,t} = (Y_{\ell,t-j})_{1\leq j\leq 24}$ and $\phi_\ell(x) = x$. PIKL-G is the estimator (11), where, for all $1 \leq \ell \leq 415$, $X_{\ell,t} = (Y_{\ell,t-j})_{1\leq j\leq 24}$ and $\phi_\ell(x) = x$. Finally, PIKL-T is the estimator (12), where $X_{\ell,t} = (Y_{\ell,t-j})_{1\leq j\leq 24}$ and $\phi_\ell(x) = x$. In the latter model, all the auto-regressive effects are penalized to enforce uniform weights, which means that $\alpha_\ell = 1$ and $J = \{1, \dots, \ell_2\}$ in (12). The hyperparameter tuning process to learn the matrix Λ for the WeaKLs is detailed in Appendix D.6.

Results. Table 3 shows the results of the experiment. The mean square errors (MSE) are computed for each hierarchical level and aggregated under *All levels*. Their standard deviations are estimated using block bootstrap with blocks of length 12. The models are categorized based on the features they utilize. We observe that the WeaKL-type estimators consistently outperform all other competitors in every case. This highlights the advantage of incorporating constraints to enforce the hierarchical structure of the problem, leading to an improved learning process.

Table 3: Benchmark in forecasting Australian domestic tourism

	MSE $(\times 10^6)$					
	Australia	States	Zones	Regions	Categories	All levels
Bottom data						
BU	5.3 ± 0.5	2.0 ± 0.2	1.37 ± 0.05	$\mathbf{1.19\pm0.02}$	$\mathbf{1.17\pm0.03}$	11.0 ± 0.7
WeaKL-BU	$\mathbf{4.5\pm0.5}$	$\mathbf{1.9\pm0.3}$	$\mathbf{1.34\pm0.05}$	$\mathbf{1.19\pm0.03}$	$\mathbf{1.17\pm0.03}$	$\mathbf{10.1\pm0.6}$
Own lags						
Indep	3.6 ± 0.6	1.8 ± 0.2	1.42 ± 0.05	1.23 ± 0.03	1.17 ± 0.03	9.2 ± 0.7
WeaKL-G	$\mathbf{3.6\pm0.5}$	$\mathbf{1.8\pm0.2}$	$\mathbf{1.37\pm0.05}$	$\mathbf{1.18\pm0.03}$	$\mathbf{1.15\pm0.03}$	$\mathbf{9.0\pm0.7}$
All data						
Rec-OLS	3.5 ± 0.5	1.8 ± 0.2	1.35 ± 0.05	1.18 ± 0.02	1.17 ± 0.03	8.9 ± 0.7
MinT	3.6 ± 0.4	1.7 ± 0.1	1.29 ± 0.05	$\mathbf{1.15\pm0.03}$	1.17 ± 0.03	8.9 ± 0.5
WeaKL-T	$\mathbf{3.1\pm0.3}$	$\mathbf{1.7\pm0.1}$	$\mathbf{1.27\pm0.05}$	$\mathbf{1.15\pm0.02}$	$\mathbf{1.12\pm0.03}$	$\mathbf{8.3\pm0.4}$

5 Conclusion

In this paper, we have shown how to design empirical risk functions that integrate common linear constraints in time series forecasting. For modeling purposes, we distinguish between shape constraints (such as additive models, online adaptation after a break, and forecast combinations) and learning constraints (including transfer learning, hierarchical forecasting, and differential constraints). These empirical risks can be efficiently minimized on a GPU, leading to the development of an optimized algorithm, which we call WeaKL. We have applied WeaKL to three real-world use cases—two in electricity demand forecasting and one in tourism forecasting—where it consistently outperforms current state-of-the-art methods, demonstrating its effectiveness in structured forecasting problems.

Future research could explore the integration of additional constraints into the WeaKL framework. For example, the current approach does not allow for forcing the regression function f_θ to be non-decreasing or convex. However, since any risk function L of the form (2) is convex in θ, the problem can be formulated as a linearly constrained quadratic program. While this generally increases the complexity of the optimization, it can also lead to efficient algorithms for certain constraints. In particular, when $d = 1$, imposing a non-decreasing constraint on f_θ reduces the problem to isotonic regression, which has a computational complexity of $O(n)$ (Wright and Wegman, 1980).

References

R. Agarwal, L. Melnick, N. Frosst, X. Zhang, B. Lengerich, et al. Neural additive models: Interpretable machine learning with neural nets. In *Advances in Neural Information Processing Systems*, volume 34, pages 4699–4711, 2021.

Y. Amara-Ouali, Y. Goude, N. Doumèche, P. Veyret, A. Thomas, D. Hebenstreit, et al. Forecasting electric vehicle charging station occupancy: Smarter mobility data challenge. *Journal of Data-centric Machine Learning Research*, 1(16):1–27, 2024.

A. Antoniadis, J. Cugliari, M. Fasiolo, Y. Goude, and J.-M. Poggi. *Statistical Learning Tools for Electricity Load Forecasting*. Springer, Cham, 2024.

G. Athanasopoulos, R. A. Ahmed, and R. J. Hyndman. Hierarchical forecasts for Australian domestic tourism. *International Journal of Forecasting*, 25:146–166, 2009.

N. H. Augustin, M. Musio, E. K. Klaus von Wilpert, S. N. Wood, and M. Schumacher. Modeling spatiotemporal forest health monitoring data. *Journal of the American Statistical Association*, 104:899–911, 2009.

G. Blanchard and N. Mücke. Kernel regression, minimax rates and effective dimensionality: Beyond the regular case. *Analysis and Applications*, 18:683–696, 2020.

M. Brégère and M. Huard. Online hierarchical forecasting for power consumption data. *International Journal of Forecasting*, 38:339–351, 2022.

A. Coletta, S. Gopalakrishnan, D. Borrajo, and S. Vyetrenko. On the constrained time-series generation problem. In *Advances in Neural Information Processing Systems*, volume 36, pages 61048–61059, 2023.

A. Daw, A. Karpatne, W. Watkins, J. Read, and V. Kumar. Physics-guided neural networks (PGNN): An application in lake temperature modeling. In *Knowledge Guided Machine Learning: Accelerating Discovery Using Scientific Knowledge and Data*, pages 352–372, New York, 2022. Chapman and Hall/CRC.

J. de Vilmarest and Y. Goude. State-space models for online post-covid electricity load forecasting competition. *IEEE Open Access Journal of Power and Energy*, 9:192–201, 2022.

J. de Vilmarest and O. Wintenberger. Viking: Variational Bayesian variational tracking. *Statistical Inference for Stochastic Processes*, 27:839–860, 2024.

J. de Vilmarest, J. Browell, M. Fasiolo, Y. Goude, and O. Wintenberger. Adaptive probabilistic forecasting of electricity (net-)load. *IEEE Transactions on Power Systems*, 39:4154–4163, 2024.

N. Doumèche, Y. Allioux, Y. Goude, and S. Rubrichi. Human spatial dynamics for electricity demand forecasting: The case of France during the 2022 energy crisis. *arXiv:2309.16238*, 2023.

N. Doumèche, F. Bach, G. Biau, and C. Boyer. Physics-informed machine learning as a kernel method. In *Conference on Learning Theory*, volume 247 of *Proceedings of Machine Learning Research*, pages 1399–1450, 2024a.

N. Doumèche, G. Biau, and C. Boyer. Convergence and error analysis of PINNs. *Bernoulli, in press*, 2024b.

N. Doumèche, F. Bach, G. Biau, and C. Boyer. Physics-informed kernel learning. *arXiv:2409.13786*, 2024.

M. Farrokhabadi, J. Browell, Y. Wang, S. Makonin, W. Su, and H. Zareipour. Day-ahead electricity demand forecasting competition: Post-COVID paradigm. *IEEE Open Access Journal of Power and Energy*, 9:185–191, 2022.

M. Fasiolo, S. N. Wood, M. Zaffran, R. Nedellec, and Y. Goude. Fast calibrated additive quantile regression. *Journal of the American Statistical Association*, 116:1402–1412, 2021.

P. Gaillard and Y. Goude. Opera: Online prediction by expert aggregation, 2016. Available: `https://CRAN.Rproject.org/package=opera.rpackage`.

P. Gaillard, G. Stoltz, and T. V. Erven. A second-order bound with excess losses. *Conference on Learning Theory*, pages 176–196, 2014.

R. W. Godahewa, C. Bergmeir, G. Webb, R. Hyndman, and P. Montero-Manso. Monash time series forecasting archive. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

B. Goehry, H. Yan, Y. Goude, P. Massart, and J.-M. Poggi. Random forests for time series. *REVSTAT-Statistical Journal*, 21:283–302, 2023.

T. Hastie and R. Tibshirani. Generalized additive models. *Statistical Science*, 1:297–310, 1986.

T. Hong and S. Fan. Probabilistic electric load forecasting: A tutorial review. *International Journal of Forecasting*, 32:914–938, 2016.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 2nd edition, 2012.

W. Ji, W. Qiu, Z. Shi, S. Pan, and S. Deng. Stiff-PINN: Physics-informed neural network for stiff chemical kinetics. *The Journal of Physical Chemistry A*, 125:8098–8106, 2021.

G. Jin, Y. Liang, Y. Fang, Z. Shao, J. Huang, J. Zhang, et al. Spatio-temporal graph neural networks for predictive learning in urban computing: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 36:5388–5408, 2024.

K. Kashinath, M. Mustafa, A. Albert, J.-L. Wu, C. Jiang, S. Esmaeilzadeh, et al. Physics-informed machine learning: Case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A*, 2021.

N. Kourentzes and G. Athanasopoulos. Cross-temporal coherent forecasts for Australian tourism. *Annals of Tourism Research*, 75:393–409, 2019.

S. N. Lahiri. *Resampling Methods for Dependent Data*. Springer, 2013.

J. Leprince, H. Madsen, J. K. Møller, and W. Zeiler. Hierarchical learning, forecasting coherent spatio-temporal individual and aggregated building loads. *Applied Energy*, 348:121510, 2023.

B. Lim and S. Zohren. Time-series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379:20200209, 2021.

G. Marra and S. N. Wood. Practical variable selection for generalized additive models. *Computational Statistics & Data Analysis*, 55:2372–2387, 2011.

G. Meanti, L. Carratino, L. Rosasco, and A. Rudi. Kernel methods through the roof: Handling billions of points efficiently. In *Advances in Neural Information Processing Systems*, volume 33, pages 14410–14422, 2020.

J. W. Messner, P. Pinson, J. Browell, M. B. Bjerregård, and I. Schicker. Evaluation of wind power forecasts – an up-to-date view. *Wind Energy*, 23:1461–1481, 2020.

M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. MIT Press, Cambridge, 2012.

Météo-France. Données SYNOP essentielles OMM. Available: public.opendatasoft.com/explore/dataset/, 2023.

R. Nickl and E. S. Titi. On posterior consistency of data assimilation with Gaussian process priors: The 2D-Navier–Stokes equations. *The Annals of Statistics*, 52:1825–1844, 2024.

D. Obst, J. de Vilmarest, and Y. Goude. Adaptive methods for short-term electricity load forecasting during COVID-19 lockdown in France. *IEEE Transactions on Power Systems*, 36:4754–4763, 2021.

C. L. Pennings and J. van Dalen. Integrated hierarchical forecasting. *European Journal of Operational Research*, 263:412–418, 2017.

F. Petropoulos, D. Apiletti, V. Assimakopoulos, M. Z. Babai, D. K. Barrow, S. Ben Taieb, et al. Forecasting: Theory and practice. *International Journal of Forecasting*, 38:705–871, 2022.

D. N. Politis and J. P. Romano. The stationary bootstrap. *Journal of the American Statistical Association*, 89:1303–1313, 1994.

M. Raissi, P. Perdikaris, and G. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

S. S. Rangapuram, L. D. Werner, K. Benidis, P. Mercado, J. Gasthaus, and T. Januschowski. End-to-end learning of coherent probabilistic forecasts for hierarchical time series. In *International Conference on Machine Learning*, pages 8832–8843, 2021.

P. Ravikumar, J. Lafferty, H. Liu, and L. Wasserman. Sparse additive models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 71:1009–1030, 2009.

C. Remlinger, C. Alasseur, M. Brière, and J. Mikael. Expert aggregation for financial forecasting. *The Journal of Finance and Data Science*, 9:100108, 2023.

RTE. éCO2mix. Available: rte-france.com/en/eco2mix, 2023. [Accessed: July 17, 2024].

M. G. Schultz, C. Betancourt, B. Gong, F. Kleinert, M. Langguth, L. H. Leufen, A. Mozaffari, and S. Stadtler. Can deep learning beat numerical weather prediction? *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379:20200097, 2021.

S. B. Taieb and R. Hyndman. A gradient boosting approach to the Kaggle load forecasting competition. *International Journal of Forecasting*, 30:382–394, 2014.

A. Timmermann. Chapter 4 forecast combinations. In G. Elliott, C. Granger, and A. Timmermann, editors, *Handbook of Economic Forecasting*, volume 1, pages 135–196. Elsevier, 2006.

S. L. Wickramasuriya, G. Athanasopoulos, and R. J. Hyndman. Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization. *Journal of the American Statistical Association*, 114:804–819, 2019.

S. N. Wood. *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC, New York, 2017.

S. N. Wood, Z. Li, G. Shaddick, and N. H. Augustin. Generalized additive models for gigadata: Modeling the U.K. black smoke network daily data. *Journal of the American Statistical Association*, 112:1199–1210, 2017.

I. W. Wright and E. J. Wegman. Isotonic, convex and related splines. *The Annals of Statistics*, 8: 1023–1035, 1980.

N. Zarbakhsh, M. Misaghian, and G. Mcardle. Human mobility-based features to analyse the impact of COVID-19 on power system operation of Ireland. *IEEE Open Access Journal of Power and Energy*, 9:213–225, 2022.

K. Zheng, H. Xu, Z. Long, Y. Wang, and Q. Chen. Coherent hierarchical probabilistic forecasting of electric vehicle charging demand. *IEEE Transactions on Industry Applications*, pages 1–12, 2023.

Appendix A. Proofs

The purpose of this appendix is to provide detailed proofs of the theoretical results presented in the main article. Appendix A.1 elaborates on the formula that characterizes the unique minimizer of the WeaKL empirical risks, while Appendix A.3 discusses the integration of linear constraints into the empirical risk framework.

A.1 A useful lemma

Lemma A.1 (Full rank) *The matrix*

$$\tilde{M} = \frac{1}{n}\Big(\sum_{j=1}^{n} \Phi_{t_j}^* \Lambda^* \Lambda \Phi_{t_j}\Big) + M^* M$$

is invertible. Moreover, for all $\theta \in \mathbb{C}^{\dim \theta}$, $\theta^\star \tilde{M}\theta \geq \lambda_{\min}(\tilde{M})\|\theta\|_2^2$, where $\lambda_{\min}(\tilde{M})$ is the minimum eigenvalue of $\tilde{M}$.

Proof First, we note that $\tilde{M}$ is a positive Hermitian square matrix. Hence, the spectral theorem guarantees that $\tilde{M}$ is diagonalizable in an orthogonal basis of $\mathbb{C}^{\dim(\theta)}$ with real eigenvalues. In particular, it admits a positive square root, and the min-max theorem states that $\theta^* \tilde{M}\theta = \|\tilde{M}^{1/2}\theta\|_2^2 \geq \lambda_{\min}(\tilde{M}^{1/2})^2\|\theta\|_2^2 = \lambda_{\min}(\tilde{M})\|\theta\|_2^2$. This shows the second statement of the lemma.

Next, for all $\theta \in \mathbb{C}^{\dim \theta}$, $\theta^* \tilde{M}\theta \geq \theta^* M^* M\theta$. Since M is full rank, $\operatorname{rank}(M) = \dim(\theta)$. Therefore, $\tilde{M}\theta = 0 \Rightarrow \theta^*\tilde{M}\theta = 0 \Rightarrow \theta^* M^* M\theta = 0 \Rightarrow \|M\theta\|_2^2 = 0 \Rightarrow M\theta = 0 \Rightarrow \theta = 0$. Thus, $\tilde{M}$ is injective and, in turn, invertible. ■

A.2 Proof of Proposition 2.1

The function $L : \mathbb{C}^{\dim(\theta)} \to \mathbb{R}^+$ can be written as

$$L(\theta) = \frac{1}{n}\Big(\sum_{j=1}^{n}(\Phi_{t_j}\theta - Y_{t_j})^* \Lambda^* \Lambda(\Phi_{t_j}\theta - Y_{t_j})\Big) + \theta^* M^* M\theta.$$

Recall that the matrices Λ and M are assumed to be injective. Observe that L can be expanded as

$$L(\theta + \delta\theta) = L(\theta) + 2\mathrm{Re}(\langle \tilde{M}\theta - \tilde{Y}, \delta\theta\rangle) + o(\|\delta\theta\|_2^2),$$

where $\tilde{Y} = \frac{1}{n}\sum_{j=1}^{n} \Phi_{t_j}^* \Lambda^* \Lambda Y_{t_j}$. This shows that L is differentiable and that its differential at θ is the function $dL_\theta : \delta\theta \mapsto 2\mathrm{Re}(\langle \tilde{M}\theta - \tilde{Y}, \delta\theta\rangle)$. Thus, the critical points θ such that $dL_\theta = 0$ satisfy

$$\forall\ \delta\theta \in \mathbb{C}^{\dim(\theta)},\ \mathrm{Re}(\langle \tilde{M}\theta - \tilde{Y}, \delta\theta\rangle) = 0.$$

Taking $\delta\theta = \tilde{M}\theta - \tilde{Y}$ shows that $\|\tilde{M}\theta - \tilde{Y}\|_2^2 = 0$, i.e., $\tilde{M}\theta = \tilde{Y}$. From Lemma A.1, we deduce that $\theta = \tilde{M}^{-1}\tilde{Y}$, which is exactly the $\hat{\theta}_n$ in (3).

From Lemma A.1, we also deduce that, for all θ such that $\|\theta\|_2$ is large enough, one has $L(\theta) \geq \lambda_{\min}(\tilde{M})\|\theta\|_2^2/2$. Since L is continuous, it has at least one global minimum. Since the unique critical point of L is $\hat{\theta}_n$, we conclude that $\hat{\theta}_n$ is the unique minimizer of L.

A.3 Orthogonal projection and linear constraints

Lemma A.2 (Orthogonal projection) *Let $\ell_1, \ell_2 \in \mathbb{N}^\star$. Let P be an injective $\ell_1 \times \ell_2$ matrix with coefficients in $\mathbb{C}$. Then $C = \mathrm{I}_{\ell_1} - P(P^*P)^{-1}P^*$ is the orthogonal projection on $\mathrm{Im}(P)^\perp$, where $\mathrm{Im}(P)$ is the image of P and I_{ℓ_1} is the $\ell_1 \times \ell_1$ identity matrix.*

Proof First, we show that P^*P is an $\ell_2 \times \ell_2$ matrix of full rank. Indeed, for all $x \in \mathbb{C}^{\ell_2}$, one has $P^*Px = 0 \Rightarrow x^*P^*Px = 0 \Rightarrow \|Px\|_2^2 = 0$. Since P is injective, we deduce that $\|Px\|_2^2 = 0 \Rightarrow x = 0$. This means that $\ker P^*P = \{0\}$, and so that P^*P is full rank. Therefore, $(P^*P)^{-1}$ is well defined.

Next, let $C_1 = P(P^*P)^{-1}P^*$. Clearly, $C_1^2 = C_1$, i.e., C_1 is a projector. Since $C_1^* = C_1$, we deduce that C_1 is an orthogonal projector. In addition, since $C_1 = P \times ((P^*P)^{-1}P^*)$, $\mathrm{Im}(C_1) \subseteq \mathrm{Im}(P)$. Moreover, if $x \in \mathrm{Im}(P)$, there exists a vector z such that $x = Pz$, and $C_1x = P(P^*P)^{-1}P^*Pz = Pz = x$. Thus, $x \in \mathrm{Im}(C_1)$. This shows that $\mathrm{Im}(C_1) = \mathrm{Im}(P)$. We conclude that C_1 is the orthogonal projection on $\mathrm{Im}(P)$ and, in turn, that $C = \mathrm{I}_{\ell_1} - C_1$ is the orthogonal projection on $\mathrm{Im}(P)^\perp$. ■

The following proposition shows that, given the exact prior knowledge $C\theta^\star = 0$, enforcing the linear constraint $C\theta = 0$ almost surely improves the performance of WeaKL.

Proposition A.3 (Constrained estimators perform better.) *Assume that $Y_t = f_{\theta^\star}(X_t) + \varepsilon_t$ and that $\theta^\star$ satisfies the constraint $C\theta^\star = 0$, for some matrix C. (Note that we make no assumptions about the distribution of the time series (X, ε).) Let Λ and M be injective matrices, and let $\lambda \geq 0$ be a hyperparameter. Let $\hat{\theta}$ be the WeaKL given by (3)and let $\hat{\theta}_C$ be the WeaKL obtained by replacing M with $(\sqrt{\lambda}C^\top \mid M^\top)^\top$ in (3).Then, almost surely,*

$$\frac{1}{n}\sum_{j=1}^{n} \|f_{\theta^\star}(X_{t_j}) - f_{\hat{\theta}_C}(X_{t_j})\|_2^2 + \|M(\theta^\star - \hat{\theta}_C)\|_2^2 \leq \frac{1}{n}\sum_{j=1}^{n} \|f_{\theta^\star}(X_{t_j}) - f_{\hat{\theta}}(X_{t_j})\|_2^2 + \|M(\theta^\star - \hat{\theta})\|_2^2.$$

Proof Recall from (3) that

$$\hat{\theta} = P^{-1}\sum_{j=1}^{n} \Phi_{t_j}^* \Lambda^* \Lambda Y_{t_j} \quad \text{and} \quad \hat{\theta}_C = \big(P + \lambda n C^*C\big)^{-1}\sum_{j=1}^{n} \Phi_{t_j}^* \Lambda^* \Lambda Y_{t_j},$$

where $P = (\sum_{j=1}^n \Phi_{t_j}^* \Lambda^* \Lambda \Phi_{t_j}) + nM^*M$. Since $C\theta^\star = 0$, we see that

$$\theta^\star = \big(P + \lambda n C^*C\big)^{-1} P\theta^\star. \tag{13}$$

Subtracting (13) to, respectively, $\hat{\theta}$ and $\hat{\theta}_C$, we obtain

$$\theta^\star - \hat{\theta} = P^{-1/2}\Delta \quad \text{and} \quad \theta^\star - \hat{\theta}_C = \big(P + \lambda n C^*C\big)^{-1} P^{1/2}\Delta,$$

where

$$\Delta = P^{-1/2}\Big(P\theta^\star - \sum_{j=1}^{n} \Phi_{t_j}^* \Lambda^* \Lambda Y_{t_j}\Big).$$

Moreover, according to the Loewner order (see, e.g., Horn and Johnson, 2012, Chapter 7.7), we have that $P^{-1/2}C^*CP^{-1/2} \geq 0$ and $(P^{-1/2}C^*CP^{-1/2})^2 \geq 0$. (Indeed, since P is Hermitian, so is $P^{-1/2}C^*CP^{-1/2}$.) Therefore, $(\mathrm{I}+\lambda n P^{-1/2}C^*CP^{-1/2})^2 \geq \mathrm{I}$ and $(\mathrm{I}+\lambda n P^{-1/2}C^*CP^{-1/2})^{-2} \leq \mathrm{I}$ (see, e.g., Horn and Johnson, 2012, Corollary 7.7.4). Consequently,

$$\|P^{1/2}(\theta^\star - \hat\theta_C)\|_2^2 = \Delta^*\big(\mathrm{I} + \lambda n P^{-1/2}C^*CP^{-1/2}\big)^{-2}\Delta \leq \|\Delta\|_2^2 = \|P^{1/2}(\theta^\star - \hat\theta)\|_2^2.$$

Observing that $\|P^{1/2}(\theta^\star - \hat\theta_C)\|_2^2 = \frac{1}{n}\sum_{j=1}^n \|f_{\theta^\star}(X_{t_j}) - f_{\hat\theta_C}(X_{t_j})\|_2^2 + \|M(\theta^\star - \hat\theta_C)\|_2^2$ and $\|P^{1/2}(\theta^\star - \hat\theta)\|_2^2 = \frac{1}{n}\sum_{j=1}^n \|f_{\theta^\star}(X_{t_j}) - f_{\hat\theta}(X_{t_j})\|_2^2 + \|M(\theta^\star - \hat\theta)\|_2^2$ concludes the proof. ■

Remark A.4 *Taking the limit $\lambda \to \infty$ in Proposition A.3 does not affect the result and corresponds to restricting the parameter space to $\ker(C)$, meaning that, in this case, $C\hat\theta_C = 0$.*

Note also that the proposition is extremely general, as it holds almost surely without requiring any assumptions on either X or ε. Here, the error of $\hat\theta$ is measured by

$$\frac{1}{n}\sum_{j=1}^{n} \|f_{\theta^\star}(X_{t_j}) - f_{\hat\theta}(X_{t_j})\|_2^2 + \|M(\theta^\star - \hat\theta)\|_2^2,$$

which quantifies both the error of $\hat\theta$ at the points X_{t_j} and in the M norm. Under additional assumptions on X and ε, this discretized risk can be shown to converge to the L^2 error, $\mathbb{E}\|f_{\theta^\star}(X) - f_{\hat\theta}(X)\|_2^2$, using Dudley's theorem (see, e.g., Theorem 5.2 in the Supplementary Material of Doumèche et al., 2024b).

However, the rate of this convergence of $\hat\theta$ to $\theta^\star$ depends on the properties of C and M, as well as the growth of $\dim(\theta)$ with n. For instance, when the penalty matrix M encodes a PDE prior, the analysis becomes particularly challenging and remains an open question in physics-informed machine learning. Therefore, we leave the study of this convergence outside the scope of this article.

Appendix B. More WeaKL models

B.1 Forecast combinations

To forecast a time series Y, different models can be used, each using different implementations and sets of explanatory variables. Let p be the number of models and let $\hat Y_t^1, \dots, \hat Y_t^p$ be the respective estimators of Y_t. The goal is to determine the optimal weighting of these forecasts, based on their performance evaluated over the time points $t_1 \leq \cdots \leq t_n$. Therefore, in this setting, $X_t = (t, \hat Y_t^1, \dots, \hat Y_t^p)$, and the goal is to find the optimal function linking X_t to Y_t. Note that, to avoid overfitting, we assume that the forecasts $\hat Y_t^1, \dots, \hat Y_t^p$ were trained on time steps before t_1. This approach is sometimes referred to as the online aggregation of experts (Remlinger et al., 2023; Antoniadis et al., 2024). Such forecast combinations are widely recognized to significantly improve the performance of the final forecast (Timmermann, 2006; de Vilmarest and Goude, 2022; Petropoulos et al., 2022; Amara-Ouali et al., 2024), as they leverage the strengths of the individual predictors.

Formally, this results in the model

$$f_\theta(X_t) = \sum_{\ell=1}^{p} (p^{-1} + h_{\theta_\ell}(t)) \hat{Y}_t^\ell,$$

where $h_{\theta_\ell}(t) = \langle \phi(t), \theta_\ell \rangle$, ϕ is the Fourier map $\phi(t) = (\exp(ikt/2))^\top_{-m \le k \le m}$, and $\theta_\ell \in \mathbb{C}^{2m+1}$. The p^{-1} term introduces a bias, ensuring that $h_{\theta_\ell} = 0$ corresponds to a uniform weighting of the forecasts $\hat{Y}^\ell$. The function $f^\star$ is thus estimated by minimizing the loss

$$L(\theta) = \frac{1}{n} \sum_{j=1}^{n} \Big| \Big(\sum_{\ell=1}^{p} (p^{-1} + h_{\theta_\ell}(t_j)) \hat{Y}_{t_j}^\ell \Big) - Y_{t_j} \Big|^2 + \sum_{\ell=1}^{p} \lambda_\ell \|h_{\theta_\ell}\|^2_{H^s},$$

where $\lambda_\ell > 0$ are hyperparameters. Again, a common choice for the smoothing parameter is to set $s = 2$. Let $\phi_1(X_t) = ((\hat{Y}_t^\ell \exp(ikt/2))_{-m \le k \le m})_{\ell=1}^{p})^\top \in \mathbb{C}^{(2m+1)p}$. The Fourier coefficients that minimize the empirical risk are given by

$$\hat{\theta} = (\Phi^* \Phi + n M^* M)^{-1} \Phi^* \mathbb{W},$$

where $\mathbb{W} = (W_{t_1}, \dots, W_{t_n})^\top$ is such that $W_t = Y_t - p^{-1} \sum_{\ell=1}^{p} \hat{Y}_t^\ell$,

$$M = \begin{pmatrix} \sqrt{\lambda_1} D & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sqrt{\lambda_{d_1}} D \end{pmatrix},$$

and D is the $(2m+1) \times (2m+1)$ diagonal matrix $D = \mathrm{Diag}((\sqrt{1 + k^{2s}})_{-m \le k \le m})$.

B.2 Differential constraints

As discussed in the introduction, some time series obey physical laws and can be expressed as solutions of PDEs. Physics-informed kernel learning (PIKL) is a kernel-based method developed by Doumèche et al. (2024) to incorporate such PDEs as constraints. It can be regarded as a specific instance of the WeaKL framework proposed in this paper. In effect, given a bounded Lipschitz domain Ω and a linear differential operator $\mathscr{D}$, using the model $f_\theta(x) = \langle \phi(x), \theta \rangle$, where $\phi(x) = (\exp(i\langle x, k \rangle/2))_{\|k\|_\infty \le m}$ is the Fourier map and θ represents the Fourier coefficients, the PIKL approach shows how to construct a matrix M such that

$$\int_\Omega \mathscr{D}(f_\theta, u)^2 \, du = \|M\theta\|_2^2.$$

Thus, to incorporate the physical prior $\forall x \in \Omega,\ \mathscr{D}(f^\star, x) = 0$ into the learning process, the empirical risk takes the form

$$L(\theta) = \frac{1}{n} \sum_{i=1}^{n} |f_\theta(X_{t_i}) - Y_{t_i}|^2 + \lambda \int_\Omega \mathscr{D}(f_\theta, u)^2 \, du = \frac{1}{n} \sum_{i=1}^{n} |f_\theta(X_{t_i}) - Y_{t_i}|^2 + \|\sqrt{\lambda} M \theta\|_2^2,$$

where $\lambda > 0$ is a hyperparameter. From (5) it follows that the minimizer of the empirical risk is $\hat{\theta} = (\Phi^* \Phi + nM)^{-1} \Phi^* \mathbb{Y}$. It is shown in Doumèche et al. (2024a) that, as $n \to \infty$, $f_{\hat{\theta}}$ converges to $f^\star$ under appropriate assumptions. Moreover, incorporating the differential constraint improves the learning process; in particular, $f_{\hat{\theta}}$ converges to $f^\star$ faster when $\lambda > 0$.

Appendix C. A toy-example of hierarchical forecasting

Setting. We evaluate the performance of WeaKL on a simple but illustrative hierarchical forecasting task. In this simplified setting, we want to forecast two random variables, Y_1 and Y_2, defined as follows:

$$Y_1 = \langle X_1, \theta_1 \rangle + \varepsilon_1, \quad Y_2 = \langle X_2, \theta_2 \rangle - \varepsilon_1 + \varepsilon_2,$$

where X_1, X_2, ε_1, and ε_2 are independent. The feature vectors are $X_1 \sim \mathcal{N}(0, \mathrm{I}_d)$ and $X_2 \sim \mathcal{N}(0, \mathrm{I}_d)$, with $d \in \mathbb{N}^\star$. The noise terms follow Gaussian distributions $\varepsilon_1 \sim \mathcal{N}(0, \sigma_1^2)$ and $\varepsilon_2 \sim \mathcal{N}(0, \sigma_2^2)$, with $\sigma_1, \sigma_2 > 0$. Note that the independence assumption aims at simplifying the analysis in this toy-example by putting the emphasis on the impact of the hierarchical constraints rather than on the autocorrelation of the signal, though in practice this assumption is unrealistic for most time series. This is why we will develop a use case of hierarchical forecasting with real-world time series in Section 4.2.

What distinguishes this hierarchical prediction setting is the assumption that $\sigma_1 \geq \sigma_2$. Consequently, conditional on X_1 and X_2, the sum $Y_1 + Y_2 = \langle X_1, \theta_1 \rangle + \langle X_2, \theta_2 \rangle + \varepsilon_2$ has a lower variance than either Y_1 or Y_2. We assume access to n i.i.d. copies $(X_{1,i}, X_{2,i}, Y_{1,i}, Y_{2,i})_{i=1}^n$ of the random variables (X_1, X_2, Y_1, Y_2). The goal is to construct three estimators $\hat{Y}_1$, $\hat{Y}_2$, and $\hat{Y}_3$ of Y_1, Y_2, and $Y_3 := Y_1 + Y_2$.

Benchmark. We compare four techniques. The *bottom-up (BU)* approach involves running two separate ordinary least squares (OLS) regressions that independently estimate Y_1 and Y_2 without using information about $Y_1 + Y_2$. Specifically,

$$\hat{Y}_1^{\mathrm{BU}} = \langle X_1, \hat{\theta}_1^{\mathrm{BU}} \rangle, \quad \hat{Y}_2^{\mathrm{BU}} = \langle X_2, \hat{\theta}_2^{\mathrm{BU}} \rangle,$$

where the OLS estimators are

$$\hat{\theta}_1^{\mathrm{BU}} = (\mathbb{X}_1^\top \mathbb{X}_1)^{-1} \mathbb{X}_1^\top \mathbb{Y}_1, \quad \hat{\theta}_2^{\mathrm{BU}} = (\mathbb{X}_2^\top \mathbb{X}_2)^{-1} \mathbb{X}_2^\top \mathbb{Y}_2.$$

Here, $\mathbb{X}_1 = (X_{1,1} \mid \cdots \mid X_{1,n})^\top$ and $\mathbb{X}_2 = (X_{2,1} \mid \cdots \mid X_{2,n})^\top$ are $n \times d$ matrices, while $\mathbb{Y}_1 = (Y_{1,1}, \ldots, Y_{1,n})^\top$ and $\mathbb{Y}_2 = (Y_{2,1}, \ldots, Y_{2,n})^\top$ are vectors of $\mathbb{R}^n$. To estimate Y_3, we simply set $\hat{Y}_3^{\mathrm{BU}} = \hat{Y}_1^{\mathrm{BU}} + \hat{Y}_2^{\mathrm{BU}}$.

The *Reconciliation (Rec)* approach involves running three independent forecasts of Y_1, Y_2, and Y_3, followed by using the constraint that the updated estimator $\hat{Y}_3^{\mathrm{Rec}}$ should be the sum of $\hat{Y}_1^{\mathrm{Rec}}$ and $\hat{Y}_2^{\mathrm{Rec}}$ (Wickramasuriya et al., 2019). To estimate Y_3, we run an OLS regression with $\mathbb{X} = (\mathbb{X}_1 \mid \mathbb{X}_2)$ and $\mathbb{Y} = \mathbb{Y}_1 + \mathbb{Y}_2$. In this approach,

$$\begin{pmatrix} \hat{Y}_{3,t}^{\mathrm{Rec}} \\ \hat{Y}_{1,t}^{\mathrm{Rec}} \\ \hat{Y}_{2,t}^{\mathrm{Rec}} \end{pmatrix} = S(S^T S)^{-1} S^T \begin{pmatrix} \langle X_t, (\mathbb{X}^\top \mathbb{X})^{-1} \mathbb{X}^\top \mathbb{Y} \rangle \\ \langle X_{1,t}, \hat{\theta}_1^{\mathrm{BU}} \rangle \\ \langle X_{2,t}, \hat{\theta}_2^{\mathrm{BU}} \rangle \end{pmatrix},$$

with $S = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$ and $X_t = (X_{1,t} \mid X_{2,t})$.

The *Minimum Trace (MinT)* approach is an alternative update method that replaces the update matrix $S(S^\top S)^{-1} S^T$ with $S(J - JWU(U^\top WU)^{-1} U^\top)$, $J = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, W the 3×3 covariance

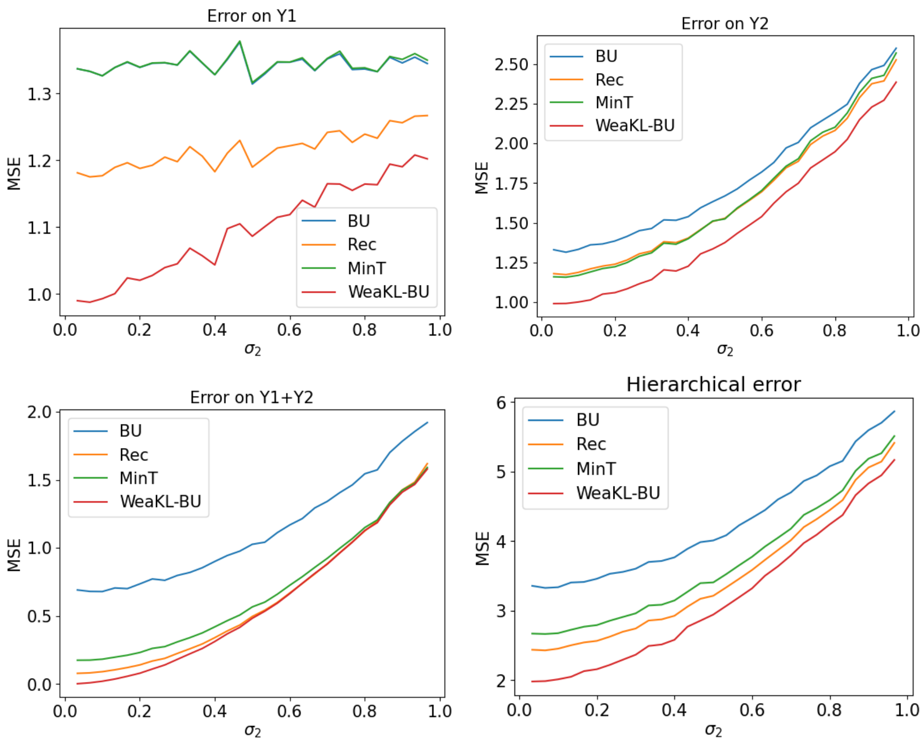

Figure 4: Hierarchical forecasting performance with $2d/n = 0.5$.

matrix of the prediction errors on the training data, and $U = \begin{pmatrix}-1 & 1 & 1\end{pmatrix}^\top$ (Wickramasuriya et al., 2019). This approach extends the linear projection onto $\mathrm{Im}(S)$ and better accounts for correlations in the noise of the time series. Finally, we apply the WeaKL-BU estimator (10) with $M = 0$, $\phi_1(x) = x$, $\phi_2(x) = x$, and $\Lambda = \mathrm{Diag}(1, 1, \lambda)$, where $\lambda > 0$ is a hyperparameter that controls the penalty on the joint prediction $Y_1 + Y_2$. It minimizes the empirical loss

$$L(\theta_1, \theta_2) = \frac{1}{n}\sum_{i=1}^{n} |\langle X_{1,i}, \theta_1\rangle - Y_{1,i}|^2 + |\langle X_{2,i}, \theta_2\rangle - Y_{2,i}|^2 + \lambda |\langle X_{1,i}, \theta_1\rangle + \langle X_{1,i}, \theta_2\rangle - Y_{1,i} - Y_{2,i}|^2,$$

In the experiments, we set $\lambda = \sigma_2^{-2}$ for simplicity, although it can be easily learned by cross-validation.

Monte Carlo experiment. To compare the performance of the different methods, we perform a Monte Carlo experiment. Since linear regression is invariant under multiplication by a constant, we set $\sigma_1 = 1$ without loss of generality. Since $\sigma_2 \leq \sigma_1$, we allow σ_2 to vary from 0 to 1. For each value of σ_2, we run 1000 Monte Carlo simulations, where each simulation uses $n = 80$ training samples and $\ell = 20$ test samples. In each Monte Carlo run, we independently draw $\theta_1 \sim \mathcal{N}(0, I_d)$, $\theta_2 \sim \mathcal{N}(0, I_d)$, $X_{1,i} \sim \mathcal{N}(0, I_d)$, $X_{2,i} \sim \mathcal{N}(0, I_d)$, $\varepsilon_{1,i} \sim \mathcal{N}(0, 1)$, and $\varepsilon_{2,i} \sim \mathcal{N}(0, \sigma_2^2)$, where $1 \leq i \leq n$. Note that, on the one hand, the L^2 error of an OLS regression on $Y_1 + Y_2$ is $\sigma_2^2(1 + 2d/n)$, while on the other hand, the minimum possible L^2 error when fitting $Y_1 + Y_2$ is σ_2^2. Thus, a large $2d/n$ is necessary to observe the benefits of hierarchical prediction. To achieve this, we set $d = 20$, resulting in $2d/n = 0.5$.

The models are trained on the n training data points, and their performance is evaluated on the ℓ test data points using the mean squared error (MSE). Given any estimator $(\hat{Y}_1, \hat{Y}_2, \hat{Y}_3)$ of (Y_1, Y_2, Y_1+Y_2), we compute the error $\ell^{-1}\sum_{j=1}^{\ell}|Y_{1,n+j}-\hat{Y}_{1,n+j}|^2$ on Y_1, the error $\ell^{-1}\sum_{j=1}^{\ell}|Y_{2,n+j}-\hat{Y}_{2,n+j}|^2$ on Y_2, and the error $\ell^{-1}\sum_{j=1}^{\ell}|Y_{1,n+j}+Y_{2,n+j}-\hat{Y}_{3,n+j}|^2$ on Y_1+Y_2. The hierarchical error is defined as the sum of these three MSEs, which are visualized in Figure 4.

Results. Figure 4 clearly shows that all hierarchical models (Rec, MinT, and WeaKL) outperform the naive bottom-up model for all four MSE metrics. Among them, our WeaKL consistently emerges as the best performing model, achieving superior results for all values of σ_2. Our WeaKL delivers gains ranging from 10% to 50% over the bottom-up model, solidifying its effectiveness in the benchmark.

The strong performance of WeaKL can be attributed to its approach, which goes beyond simply computing the best linear combination of linear experts to minimize the hierarchical loss, as reconciliation methods typically do. Instead, WeaKL directly optimizes the weights θ_1 and θ_2 to minimize the hierarchical loss. Another way to interpret this is that when the initial forecasts are suboptimal, reconciliation methods aim to find a better combination of those forecasts, but do so without adjusting their underlying weights. In contrast, the WeaKL approach explicitly recalibrates these weights, resulting in a more accurate and adaptive hierarchical forecast.

Extension to the over-parameterized limit. Another advantage of WeakL is that it also works for d such that $2n \geq 2d \geq n$. In this context, the Rec and MinT algorithms cannot be computed because the OLS regression of $\mathbb{Y}$ on $\mathbb{X}$ is overparameterized ($2d$ features but only n data points). To study the performance of the benchmark in the $n \simeq d$ limit, we repeated the same Monte Carlo experiment, but with $d = 38$, resulting in $d/n = 0.95$. The MSEs of the methods are shown in Figure 5. These results further confirm the superiority of the WeaKL approach in the overparameterized regime. Note that such overparameterized situations are common in hierarchical forecasting. For example, forecasting an aggregate index-such as electricity demand, tourism, or food consumption-at the national level using city-level data across $d \gg 1$ cities (e.g., local temperatures) often leads to an overparameterized model.

Extension to non-linear regressions. For simplicity, our experiments have focused on linear regressions. However, it is important to note that the hierarchical WeaKL can be applied to nonlinear regressions using exactly the same formulas. Specifically, in cases where $Y_1 = f_1(X_1) + \varepsilon_1$ and $Y_2 = f_2(X_2) - \varepsilon_1 + \varepsilon_2$, where f_1 and f_2 represent nonlinear functions, the WeaKL approach remains valid. This is because the connection to the linear case is straightforward: the WeaKL essentially performs a linear regression on the Fourier coefficients of X_1 and X_2, seamlessly extending its applicability to nonlinear settings.

Appendix D. Experiments

This appendix provides comprehensive details on the use cases discussed in the main text. Appendix D.1 describes our hyperparameter tuning technique. Appendix D.2 explains how we evaluate uncertainties. Appendix D.3 outlines our approach to handling sampling frequency in electricity demand forecasting applications. Appendix D.4 details the models used in Use case 1, while Appendix D.5 focuses on Use case 2, and Appendix D.6 covers the tourism demand forecasting use case.

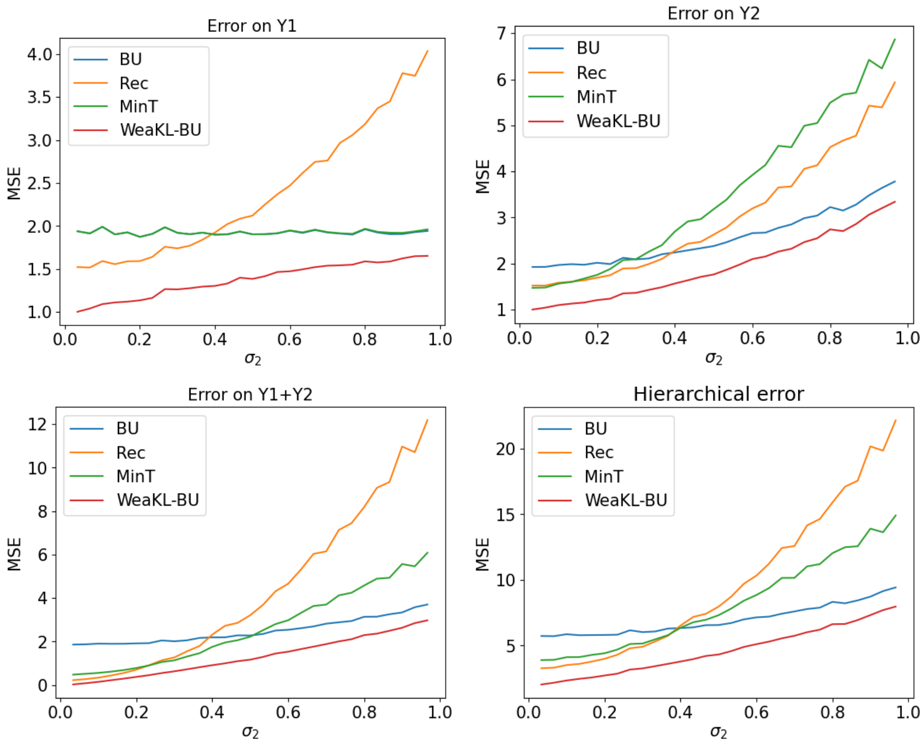

Figure 5: Hierarchical forecasting performance with $2d/n = 0.95$.

D.1 Hyperparameter tuning

Hyperparameter tuning of the additive WeaKL. Consider a WeaKL additive model

$$f_\theta(X_t) = \langle \phi_{1,1}(X_{1,t}), \theta_{1,1}\rangle + \cdots + \langle \phi_{1,d_1}(X_{d_1,t}), \theta_{1,d_1}\rangle,$$

where the type (linear, nonlinear, or categorical) of the effects are specified. Thus, as detailed in Section 3,

(i) If the effect $\langle \phi_{1,\ell}(X_{\ell,t}), \theta_{1,j}\rangle$ is assumed to be linear, then $\phi_{1,j}(X_{\ell,t}) = X_{\ell,t}$,

(ii) If the effect $\langle \phi_{1,\ell}(X_{\ell,t}), \theta_{1,\ell}\rangle$ is assumed to be nonlinear, then $\phi_{1,\ell}$ is a Fourier map with $2m_\ell + 1$ Fourier modes,

(iii) If the effect $\langle \phi_{1,\ell}(X_{\ell,t}), \theta_{1,\ell}\rangle$ is assumed to be categorical with values in E, then $\phi_{1,\ell}$ is a Fourier map with $2\lfloor |E|/2\rfloor + 1$ Fourier modes.

We let $\mathbf{m} = \{m_\ell \mid \text{the effect } \langle \phi_{1,\ell}(X_{\ell,t}), \theta_{1,\ell}\rangle \text{ is nonlinear}\}$ be the concatenation of the numbers of Fourier modes of the nonlinear effects. The goal of hyperparameter tuning is to find the best set of hyperparameters $\lambda = (\lambda_1, \dots, \lambda_{d_1})$ and $\mathbf{m}$ for the empirical risk (6) of the additive WeaKL.

To do so, we split the data into three sets: a training set, then a validation set, and finally a test set. These three sets must be disjoint to avoid overfitting, and the test set is the dataset on which the final performance of the method will be evaluated. The sets should be chosen so that the distribution of (X, Y) on the validation set resembles as much as possible the distribution of (X, Y) on the test set.

We consider a list of potential candidates for the optimal set of hyperparameters $(\lambda, \mathbf{m})_{\text{opt}}$. Since we have no prior knowledge about $(\lambda, \mathbf{m})$, we chose this list to be a grid of parameters. For each element $(\lambda, \mathbf{m})$ in the grid, we compute the minimizer $\hat\theta(\lambda, \mathbf{m})$ of the loss (6) over the training period. Then, given $\hat\theta(\lambda, \mathbf{m})$, we compute the mean squared error (MSE) of $f_{\hat\theta(\lambda,\mathbf{m})}$ over the validation period. This procedure is commonly referred to as grid search. The resulting estimate of the optimal hyperparameters $(\lambda, \mathbf{m})_{\text{opt}}$ corresponds to the values of $(\lambda, \mathbf{m})$ that minimize the MSE of $f_{\hat\theta(\lambda,\mathbf{m})}$ over the validation period. The performance of the additive WeaKL is then assessed based on the performance of $f_{\hat\theta(\lambda,\mathbf{m})_{\text{opt}}}$ on the test set.

Hyperparameter tuning of the online WeaKL. Consider an online WeaKL

$$f_\theta(t, x_1, \dots, x_{d_1}) = h_{\theta_0}(t) + \sum_{\ell=1}^{d_1} (1 + h_{\theta_\ell}(t))\hat g_\ell(x_\ell),$$

where the effects $\hat g_\ell$ are known, and the updates $h_{\theta_\ell}(t) = \langle \phi(t), \theta_\ell\rangle$ are such that ϕ is the Fourier map $\phi(t) = (\exp(ikt/2))^\top_{-m_j \le k \le m_j}$, with $m_j \in \mathbb{N}^\star$. We let $\mathbf{m} = \{m_j \mid 0 \le j \le d_1\}$ be the concatenation of the numbers of Fourier modes. The goal of hyperparameter tuning is to find the best set of hyperparameters $\lambda = (\lambda_0, \dots, \lambda_{d_1})$ and $\mathbf{m}$ for the empirical risk (8) of the online WeaKL.

To do so, we split the data into three sets: a training set, then a validation set, and finally a test set. This three sets must be disjoint to avoid overfitting. Moreover, the training set and the validation set must be disjoint from the data used to learn the effects $\hat g_\ell$. The test set must be the set on which the final performance of the method will be evaluated. The sets should be chosen so that the distribution of (X, Y) on the validation set resembles as much as possible the distribution of (X, Y) on the test set. Similarly to the hyperparameter tuning of the additive WeaKL, we then consider a list of potential candidates for the optimal hyperparameter $(\lambda, \mathbf{m})_{\text{opt}}$, which can be a grid. Then, we compute the minimizer $\hat{\theta}(\lambda, \mathbf{m})$ of the loss (8) on the training period, and the resulting estimation of $(\lambda, \mathbf{m})_{\text{opt}}$ is the set of hyperparameters $(\lambda, \mathbf{m})$ such that the MSE of $f_{\hat{\theta}(\lambda,\mathbf{m})}$ on the validation period is minimal. The performance of the online WeaKL is thus measured by the performance of $f_{\hat{\theta}(\lambda,\mathbf{m})_{\text{opt}}}$ on the test set.

D.2 Block bootstrap methods

Evaluating uncertainties with block bootstrap. The purpose of this paragraph is to provide theoretical tools for evaluating the performance of time series estimators. Formally, given a test period $\{t_1, \dots, t_n\}$, a target time series $(Y_{t_j})_{1\leq j\leq n}$, and an estimator $(\hat{Y}_{t_j})_{1\leq j\leq n}$ of Y, the goal is to construct confidence intervals that quantify how far $\text{RMSE}_n = (n^{-1}\sum_{j=1}^n |\hat{Y}_{t_j} - Y_{t_j}|^2)^{1/2}$ deviates from its expectation $\text{RMSE} = (\mathbb{E}|\hat{Y}_{t_1} - Y_{t_1}|^2)^{1/2}$, and how far $\text{MAPE}_n = n^{-1}\sum_{j=1}^n |\hat{Y}_{t_j} - Y_{t_j}||Y_{t_j}|^{-1}$ deviates from its expectation $\text{MAPE} = \mathbb{E}(|\hat{Y}_{t_1} - Y_{t_1}||Y_{t_1}|^{-1})$. Here, we assume that Y and $\hat{Y}$ are strongly stationary, meaning their distributions remain constant over time. Constructing such confidence intervals is non-trivial because the observations Y_{t_j} in the time series Y are correlated, preventing the direct application of the central limit theorem. The block bootstrap algorithm is specifically designed to address this challenge and is defined as follows.

Consider a sequence $Z_{t_1}, Z_{t_2}, \dots, Z_{t_n}$ such that the quantity of interest can be expressed as $g(\mathbb{E}(Z_{t_1}))$, for some function g. This quantity is estimated by $g(\bar{Z}_n)$, where $\bar{Z}_n = n^{-1}\sum_{j=1}^n Z_{t_j}$ is the empirical mean of the sequence. For example, $\text{RMSE} = g(\mathbb{E}(Z_{t_1}))$ and $\text{RMSE}_n = g(\bar{Z}_n)$ for $g(x) = x^{1/2}$ and $Z_{t_j} = (Y_{t_j} - \hat{Y}_{t_j})^2$, while $\text{MAPE} = g(\mathbb{E}(Z_{t_1}))$ and $\text{MAPE}_n = g(\bar{Z}_n)$ for $g(x) = x$ and $Z_{t_j} = |\hat{Y}_{t_j} - Y_{t_j}||Y_{t_j}|^{-1}$. The goal of the block bootstrap algorithm is to estimate the distribution of $g(\bar{Z}_n)$.

Given a length $\ell \in \mathbb{N}^\star$ and a starting time t_j, we say that $(Z_{t_j}, \dots, Z_{t_{j+\ell-1}}) \in \mathbb{R}^\ell$ is a block of length ℓ starting at t_j. We draw $b = \lfloor n/\ell \rfloor + 1$ blocks of length ℓ uniformly from the sequence $(Z_{t_1}, Z_{t_2}, \dots, Z_{t_n})$ and then concatenate these blocks to form the sequence $Z^* = (Z_1^*, Z_2^*, \dots, Z_{b\ell}^*)$. Thus, Z^* is a resampled version of Z obtained with replacement.

For convenience, we consider only the first n values of Z^* and compute the bootstrap version of the empirical mean: $\bar{Z}_n^* = \frac{1}{n}\sum_{j=1}^n Z_j^*$. By repeatedly resampling the b blocks and generating multiple instances of $\bar{Z}_n^*$, the resulting distribution of $\bar{Z}_n^*$ provides a reliable estimate of the distribution of $\bar{Z}_n$. In particular, under general assumptions about the decay of the autocovariance function of Z, choosing $\ell = \lfloor n^{1/4} \rfloor$ leads to

$$\sup_{x\in\mathbb{R}} |\mathbb{P}(T_n^* \leq x \mid Z_{t_1}, \dots, Z_{t_n}) - \mathbb{P}(T_n \leq x)| = O_{n\to\infty}(n^{-3/4}),$$

where $T_n^* = \sqrt{n}(\bar{Z}_n^* - \mathbb{E}(\bar{Z}_n^* \mid Z_{t_1}, \dots, Z_{t_n}))$ and $T_n = \sqrt{n}(\bar{Z}_n - \mathbb{E}(Z_{t_1}))$ (see, e.g. Lahiri, 2013, Theorem 6.7). Note that this convergence rate of $n^{-3/4}$ is actually quite fast, as even if the Z_{t_j} were i.i.d., the empirical mean $\bar{Z}_n$ would only converge to a normal distribution at a rate of $n^{-1/2}$ (by the Berry-Esseen theorem). This implies that the block bootstrap method estimates the distribution of $\bar{Z}_n$ faster than $\bar{Z}_n$ itself converges to its Gaussian limit.

The choice of ℓ plays a crucial role in this method. For instance, setting $\ell = 1$ leads to an underestimation of the variance of $\bar{Z}_n$ when the Z_{t_j} are correlated (see, e.g. Lahiri, 2013, Corollary

2.1). In addition, block resampling introduces a bias, as Z_{t_n} belongs to only a single block and is therefore less likely to be resampled than $Z_{t_{\lfloor n/2 \rfloor}}$. This explains why $\mathbb{E}(\bar{Z}_n^* \mid Z_{t_1}, \dots, Z_{t_n}) \neq \bar{Z}_n$. To address both problems, Politis and Romano (1994) introduced the stationary bootstrap, where the block length ℓ varies and follows a geometric distribution.

Comparing estimators with block bootstrap. Given two stationary estimators $\hat{Y}^1$ and $\hat{Y}^2$ of Y, the goal is to develop a test of level $\alpha \in [0,1]$ for the hypothesis $H_0 : \mathbb{E}|\hat{Y}_t^1 - Y_t| = \mathbb{E}|\hat{Y}_t^2 - Y_t|$. Using the previous paragraph, such a test could be implemented by estimating two confidence intervals I_1 and I_2 for $\mathbb{E}|\hat{Y}_t^1 - Y_t|$ and $\mathbb{E}|\hat{Y}_t^2 - Y_t|$ at level $\alpha/2$ using block bootstrap, and then rejecting H_0 if $I_1 \cap I_2 = \emptyset$. However, this approach tends to be conservative, potentially reducing the power of the test when assessing whether one estimator is significantly better than the other.

To create a more powerful test, Messner et al. (2020) and Farrokhabadi et al. (2022) suggest relying on the MAE skill score, which is defined by

$$\text{Skill} = 1 - \frac{\text{MAE}_1}{\text{MAE}_2},$$

where MAE_1 and MAE_2 are the mean average errors of $\hat{Y}^1$ and $\hat{Y}^2$, respectively. Note that Skill $= (\text{MAE}_2 - \text{MAE}_1)/\text{MAE}_2$ is the relative distance between the two MAEs. Thus, $\hat{Y}^1$ is significantly better than $\hat{Y}^2$ if Skill is significantly positive. A confidence interval for Skill can be obtained by block bootstrap. Indeed, consider the time series Z defined as $Z_{t_j} = (|\hat{Y}_{t_j}^1 - Y_{t_j}^1|, |\hat{Y}_{t_j}^2 - Y_{t_j}^2|)$, and let $g(x,y) = 1 - x/y$. We use the block bootstrap method over this sequence to estimate $g(\mathbb{E}(Z))$ by generating different samples of MAE_1 and MAE_2. In particular, in Appendix D.4, $\hat{Y}^1$ corresponds to WeakL, while $\hat{Y}^2$ is the estimator of the winning team of the IEEE competition.

D.3 Half-hour frequency

Short-term electricity demand forecasts are often estimated with a half-hour frequency, meaning that the objective is to predict electricity demand every 30 minutes during the test period. This applies to both Use case 1 and Use case 2. There are two common approaches to handling this frequency in forecasting models. One approach is to include the half-hour of the day as a feature in the models. The alternative, which yields better performance, is to train a separate model for each half-hour, resulting in 48 distinct models. This superiority arises because the relationship between electricity demand and conventional features (such as temperature and calendar effects) varies significantly across different times of the day. For instance, electricity demand remains stable at night but fluctuates considerably during the day. This variability justifies treating the forecasting problem at each half-hour as an independent learning task, leading to 48 separate models. Consequently, in both use cases, all models discussed in this paper—including WeaKL, as well as those from de Vilmarest and Goude (2022) and Doumèche et al. (2023)—are trained separately for each of the 48 half-hours, using identical formulas and architectures. This results in 48 distinct sets of model weights. For simplicity, and since the only consequence of this preprocessing step is to split the learning data into 48 independent groups, this distinction is omitted from the equations.

D.4 Precisions on the Use case 1 on the IEEE DataPort Competition on Day-Ahead Electricity Load Forecasting

In this appendix, we provide additional details on the two WeaKLs used in the benchmark for Use case 1 of the IEEE DataPort Competition on Day-Ahead Electricity Load Forecasting. The first model is a direct adaptation of the GAM-Viking model from de Vilmarest and Goude (2022) into the WeaKL framework. The second model is a WeaKL where the effects are learned through hyperparameter tuning.

Direct translation of the GAM-Viking model into the WeaKL framework. To build their model, de Vilmarest and Goude (2022) consider four primary models: an autoregressive model (AR), a linear regression model, a GAM, and a multi-layer perceptron (MLP). These models are initially trained on data from 18 March 2017 to 1 January 2020. Their weights are then updated using the Viking algorithm starting from 1 March 2020 (de Vilmarest and Goude, 2022, Table 3). The parameters of the Viking algorithm were manually selected by the authors based on performance monitoring over the 2020–2021 period (de Vilmarest and Goude, 2022, Figure7). To further refine the forecasts, the model errors are corrected using an autoregressive model, which they called the intraday correction and implemented as a static Kalman filter. The final forecast is obtained by online aggregation of all models, meaning that the predictions from different models are combined in a linear combination that evolves over time. The weights of this aggregation are learned using the ML-Poly algorithm from the `opera` package (Gaillard and Goude, 2016), trained over the period 1 July 2020 to 18 January 2021. The test period spans from 18 January 2021 to 17 February 2021. During this period, the aggregated model achieves a MAE of 10.9 GW, while the Viking-updated GAM model alone yields an MAE of 12.7 GW.

Here, to ensure a fair comparison between our WeaKL framework and the GAM-Viking model of de Vilmarest and Goude (2022), we replace their GAM-Viking with our online WeaKL in their aggregation. Our additive WeaKL model is therefore a direct translation of their offline GAM formulation into the WeaKL framework. Specifically, we consider the additive WeaKL based on the features $X = (\mathrm{DoW}, \mathrm{FTemps95}_{\mathrm{corr1}}, \mathrm{Load}_1, \mathrm{Load}_7, \mathrm{ToY}, t)$ corresponding to

$$Y_t = g_1^\star(\mathrm{DoW}_t) + g_2^\star(\mathrm{FTemps95}_{\mathrm{corr1}t}) + g_3^\star(\mathrm{Load}_{1t}) + g_4^\star(\mathrm{Load}_{7t}) + g_5^\star(\mathrm{ToY}_t) + g_6^\star(t) + \varepsilon_t,$$

where $g_1^\star$ is categorical with 7 values, $g_2^\star$ and $g_6^\star$ are linear, $g_3^\star$, $g_4^\star$, and $g_5^\star$ are nonlinear.

$\mathrm{FTemps95}_{\mathrm{corr1}}$ is a smoothed version of the temperature, while the other features remain the same as those used in Use case 2. The weights of the additive WeaKL model are determined using the hyperparameter selection technique described in Appendix D.1. The training period spans from 18 March 2017 to 1 November 2019, while the validation period extends from 1 November 2019 to 1 January 2020. During this grid search, the performance of $250,047$ sets of hyperparameters $(\lambda, \mathbf{m}) \in \mathbb{R}^7 \times \mathbb{R}^3$ is evaluated in less than a minute using a standard GPU (Nvidia L4 GPU, 24 GB RAM, 30.3 teraFLOPs for Float32). Notably, this optimization period exactly matches the training period of the primary models in de Vilmarest and Goude (2022), ensuring a fair comparison between the two approaches.

Then, we run an online WeaKL, where the effects $\hat{g}_\ell$, $1 \leq \ell \leq 7$, are inherited directly from the previously trained additive WeaKL. The weights of this online WeaKL are determined using the hyperparameter selection technique described in Appendix D.1. The training period extends from 1 February 2020 to 18 November 2020, while the validation period extends from 18 November 2020 to 18 January 2021, immediately preceding the final test period to ensure optimal adaptation. During this grid search, we evaluate 625 sets of hyperparameters $(\lambda, \mathbf{m}) \in \mathbb{R}^6 \times \mathbb{R}^6$ in less than a minute using a standard GPU. Since t is already included as a feature, the function h_0^* in Equation (7) is not required in this setting.

Table 4: Comparing GAM-Viking with its direct translation in the WeaKL framework on the final test period

Model GAM	GAM_{+}	$\text{GAM}_{+,\text{intra}}$	GAM_{on}	$\text{GAM}_{\text{on,intra}}$	GAM_{agg}
MAE (GW)	48.3	22.7	13.2	12.7	10.9
Model WeaKL	WeaKL_{+}	$\text{WeaKL}_{+,\text{intra}}$	WeaKL_{on}	$\text{WeaKL}_{\text{on,intra}}$	$\text{WeaKL}_{\text{agg}}$
MAE (GW)	58.0	23.4	11.2	11.3	10.5

Table 5: Comparing GAM with its direct translation in the WeaKL framework on a stationary test period.

Model	GAM_{+}	WeaKL_{+}	$\text{GAM}_{+,\text{intra}}$	$\text{WeaKL}_{+,\text{intra}}$
MAE (GW)	20.7	19.1	19.3	19.2

Finally, we evaluate the performance of our additive WeaKL (denoted as WeaKL_{+}), our additive WeaKL followed by intraday correction ($\text{WeaKL}_{+,\text{intra}}$), our online WeaKL ($\text{WeaKL}_{\text{on}}$), our online WeaKL with intraday correction ($\text{WeaKL}_{\text{on,intra}}$), and an aggregated model based on de Vilmarest and Goude (2022), where the GAM and GAM-Viking models are replaced by our additive and online WeaKL models ($\text{WeaKL}_{\text{agg}}$). The test period remains consistent with de Vilmarest and Goude (2022), spanning from 18 January 2021 to 17 February 2021. Their performance results are presented in Table 4 and compared to their corresponding translations within the GAM-Viking framework. Thus, GAM_{+} refers to the offline GAM, while $\text{GAM}_{+,\text{intra}}$ corresponds to the offline GAM with an intraday correction. Similarly, GAM_{on} represents the GAM-Viking model, and $\text{GAM}_{\text{on,intra}}$ denotes the GAM-Viking model with an intraday correction. Finally, GAM_{agg} corresponds to the final model proposed by de Vilmarest and Goude (2022).

The performance GAM_{+}, $\text{GAM}_{+,\text{intra}}$, WeaKL_{+}, and $\text{WeaKL}_{+,\text{intra}}$ in Table 4 alone is not very meaningful because the distribution of electricity demand differs between the training and test periods. To address this, Table 5 presents a comparison of the same algorithms, trained on the same period but evaluated on a test period spanning from 1 January 2020 to 1 March 2020. In this stationary period, WeaKL outperforms the GAMs.

Moreover, in Table 4, the online WeaKLs clearly outperform the GAM-Viking models, achieving a reduction in MAE of more than 10%. As a result, replacing the GAM-Viking models in the aggregation leads to improved overall performance. Notably, the WeaKLs are direct translations of the GAM-Viking models, meaning that the performance gains are due solely to model optimization and not to any structural changes.

Pure WeaKL. In addition, we trained an additive WeaKL using a different set of variables than those in the GAM model, aiming to identify an optimal configuration. Specifically, we consider the additive WeaKL with

$$X = (\text{FcloudCover\_corr1}, \text{Load1D}, \text{Load1W}, \text{DayType}, \text{FTemperature\_corr1}, \\ \text{FWindDirection}, \text{FTemps95\_corr1}, \text{Toy}, \text{t}),$$

where

(i) the effects of $\mathrm{FclouCover\_corr1}$, $\mathrm{Load1D}$, and $\mathrm{Load1W}$ are nonlinear,

(ii) the effect of $\mathrm{DayType}$ is categorical with 7 values,

(iii) the remaining effects are linear.

This model is trained using the hyperparameter tuning process detailed in Appendix D.1, with the training period spanning from 18 March 2017 to 1 January 2020, and validation starting from 1 October 2019. Next, we fit an online WeaKL model, with hyperparameters tuned using a training period from 1 March 2020 to 18 November 2020 and a validation period extending until 18 January 2021.

To verify that our pure WeaKL model achieves a significantly lower error than the best model from the IEEE competition, we estimate the MAE skill score by comparing our pure WeaKL to the model proposed by de Vilmarest and Goude (2022). To achieve this, we follow the procedure detailed in Appendix D.2, using block bootstrap with a block length of $\ell = 24$ and 3000 resamples to estimate the distribution of the MAE skill score, Skill. Here, $\hat{Y}^1$ represents the WeaKL, while $\hat{Y}^2$ corresponds to the estimator from de Vilmarest and Goude (2022). To evaluate the performance difference, we estimate the standard deviation σ_n of Skill_n and construct an asymptotic one-sided confidence interval for Skill. Specifically, we define $\mathrm{Skill}_n = 1 - (\sum_{j=1}^n |\hat{Y}^1_{t_j} - Y_{t_j}|)/(\sum_{j=1}^n |\hat{Y}^2_{t_j} - Y_{t_j}|)$ and consider the confidence interval $[\mathrm{Skill}_n - 1.28\sigma_n, +\infty[$, which corresponds to a confidence level of $\alpha = 0.1$. The resulting interval, $[0.007, +\infty[$, indicates that the Skill score is positive with at least 90% probability. Consequently, with at least 90% probability, the WeaKL chieves a lower MAE than the best model from the IEEE competition.

D.5 Precision on the use Use case 2 on forecasting the French electricity load during the energy crisis

This appendix provides detailed information on the additive WeaKL and the online WeaKL used in Use case 2, which focuses on forecasting the French electricity load during the energy crisis.

Additive WeaKL. As detailed in the main text, the additive WeaKL is built using the following features:

$$X = (\mathrm{Load}_1, \mathrm{Load}_7, \mathrm{Temp}, \mathrm{Temp}_{950}, \mathrm{Temp}_{\max950}, \mathrm{Temp}_{\min950}, \mathrm{ToY}, \mathrm{DoW}, \mathrm{Holiday}, t).$$

The effects of Load_1, Load_7, and t are modeled as linear. The effects of Temp, Temp_{950}, $\mathrm{Temp}_{\max950}$, $\mathrm{Temp}_{\min950}$, and ToY are modeled as nonlinear with $m = 10$. The effects of DoW and Holiday are treated as categorical, with $|E| = 7$ and $|E| = 2$, respectively. The model weights are selected through hyperparameter tuning, as detailed in Appendix D.1. The training period spans from 8 January 2013 to 1 September 2021, while the validation period covers 1 September 2021 to 1 September 2022. Notably, this is the exact same period used by Doumèche et al. (2023) to train the GAM. The objective of the hyperparameter tuning process is to determine the optimal values for $\lambda = (\lambda_1, \ldots, \lambda_{10}) \in (\mathbb{R}^+)^{10}$ and $\mathbf{m} = (m_3, m_4, m_5, m_6, m_7) \in (\mathbb{N}^\star)^5$ in Equation (6). As a result, the additive WeaKL model presented in Use case 2 is the outcome of this hyperparameter tuning process.

Online WeaKL. Next, we train an online WeaKL to update the effects of the additive WeaKL. To achieve this, we apply the hyperparameter selection technique detailed in Appendix D.1. The training period spans from 1 February 2018 to 1 April 2020, while the validation period extends from 1 April 2020 to 1 June 2020. These periods, although not directly contiguous to the test period, were specifically chosen because they overlap with the COVID-19 outbreaks. This is crucial, as it allows the model to learn from a nonstationary period. Moreover, since online models require daily updates, the online WeaKL is computationally more expensive than the additive WeaKL. The training period is set to two years and two months, striking a balance between computational efficiency and GPU memory usage. Using the parameters $(\lambda, \mathbf{m})$ obtained from hyperparameter tuning, we then retrain the model in an online manner with data starting from 1 July 2020, ensuring that the rolling training period remains at two years and two months.

Error quantification. Following the approach of Doumèche et al. (2023), the standard deviations of the errors are estimated using stationary block bootstrap with a block length of $\ell = 48$ and 1000 resamples.

Model running times. Below, we present the running times of various models in the experiment that includes holidays:

- GAM: 20.3 seconds.
- Static Kalman adaption: 1.7 seconds.
- Dynamic Kalman adaption: 48 minutes, for an hyperparameter tuning of 10^4 sets of hyperparameters (see Obst et al., 2021, II.A.2).
- Viking algorithm: 215 seconds (in addition to training the Dynamic Kalman model).
- Aggregation: 0.8 seconds.
- GAM boosting model: 6.6 seconds.
- Random forest model: 196 seconds.
- Random forest + bootstrap model: 34 seconds.
- Additive WeaKL: grid search of 1.6×10^5 hyperparameters: 257 seconds; training a single model: 2 seconds.
- Online WeaKL: grid search of 9.2×10^3 hyperparameters: 114 seconds; training a single model: 52 seconds.

D.6 Precisions on the use case on hierarchical forecasting of Australian domestic tourism with transfer learning

The matrices Λ for the WeaKL-BU, WeaKL-G, and WeaKL-T estimators are selected through hyperparameter tuning. Following the procedure detailed in Appendix D.1, the dataset is divided into three subsets: training, validation, and test. The training set comprises the first 60% of the data, the validation set the next 20%, and the test set the last 20%. The optimal matrix, Λ_{opt}, is chosen from a set of candidates by identifying the estimator trained on the training set that achieves the lowest

MSE on the validation set. The model is then retrained using both the training and validation sets with $\Lambda = \Lambda_{\text{opt}}$, and its performance is evaluated on the test set. Given that $d_1 = 415\times24 = 19,920$, WeaKL involves matrices of size $d_1^2 \simeq 4 \times 10^8$, requiring several gigabytes of RAM. Consequently, the grid search process is computationally expensive. For instance, in this experiment, the grid search over 1024 hyperparameter sets for WeaKL-T takes approximately 45 minutes.