Matrix Completion for Panel Data Forecasting

A dissertation submitted in partial fulfilment of the requirements for the degree of Master of Science in Statistical Science

September 15, 2024

Clemens Tobias Schnabel, St. Edmund Hall
Academic Year 2023/24
Supervisor: Prof. Frank Windmeijer

DEPARTMENT OF
STATISTICS

Abstract

This dissertation explores the potential of Matrix Completion with Nuclear Norm Minimization (MC-NNM) for panel data forecasting, with a focus on electricity price prediction. I implement and evaluate, for the first time, the autocorrelated errors extension to MC-NNM proposed by Athey et al. (2021), demonstrating its benefits in a forecasting context. A new, open-source Python implementation of MC-NNM is introduced, offering unprecedented flexibility in handling various covariate types. I conduct a comprehensive comparison of MC-NNM against state-of-the-art forecasting methods using a novel, open-source forecast engine and a rich dataset of European electricity prices. My results show that MC-NNM offers competitive performance and superior stability, particularly with longer time windows, in a notoriously challenging forecasting domain. This work not only advances the understanding of Matrix Completion methods in statistics but also pioneers the application of these techniques to electricity price forecasting. By treating this as a panel data problem, I open new avenues for leveraging cross-sectional dependencies in energy markets. My findings validate the theoretical properties of MC-NNM in a practical setting and suggest its potential as a robust tool for panel data forecasting across various domains.

This is my own work (except where otherwise indicated).

Candidate: Clemens Tobias Schnabel

Signed: 

Date: September 15, 2024

Acknowledgements

I would like to thank Frank Windmeijer for his close, thoughtful, and patient supervision. I am grateful to Kyra Pauly, Michael Haas, and Zafer Toubasi for proofreading. All remaining errors are mine alone.

Contents

List of Figures

List of Tables

List of Definitions

List of Assumptions

List of Estimators

1 Introduction

Matrix completion methods, which impute unobserved entries in data matrices, have seen limited application in observational statistics. Athey et al. (2021) adapt these methods, originally developed for settings with random missingness, to applications where missing data patterns are non-random. They introduce the Matrix Completion with Nuclear Norm Minimization (MC-NNM) estimator, which can be used for causal inference by imputing unobserved counterfactuals. Matrix completion methods can capture complex patterns in the data without imposing explicitly linear relationships, while maintaining a degree of interpretability and becoming more accurate with more observed data. These properties suggest potential applications in panel data analysis beyond those currently explored in the literature. By using MC-NNM to impute *future* outcomes instead of *counterfactual* ones, we can use the flexibility of this estimator to forecast panel data. Building on the work of Athey et al. (2021), this dissertation makes several main contributions to the literature in statistics and forecasting:

1. It implements the autocorrelated errors extension described in Athey et al. (2021). While this extension may not be suitable for causal inference due to violation of the assumption of independent errors, this dissertation explores its potential benefits in the context of forecasting. Neither this extension nor the application of the estimator to forecasting have been done before.

2. To do so, it provides a new, open-source implementation of the MC-NNM estimator in Python / `JAX`. Unlike existing implementations, which are scarcely documented and difficult to modify, my implementation is fully documented and designed for extensibility. It is also the only implementation that allows for time-and unit-specific specific covariates and autocorrelated errors.

3. It assesses the forecasting performance of the MC-NNM estimator for panel data, both with and without autocorrelation extensions. This novel application addresses some of the limitations of MC-NNM for causal inference, such as its row-column exchangeability property, and explores its potential in a domain where its theoretical properties suggest it could excel.

4. It introduces an open-source forecast engine, complete with a rich, public dataset of European electricity prices. This engine provides a standardized framework for comparing various forecasting methods, including my implementation of MC-NNM, in a real-world panel data setting.

The following sections describe the motivation of Matrix Completion estimators and the panel data use case, introduce and discuss the adapted missingness structure that yields MC-NNM, and assess its performance in a horse race comparison to prevalent forecasting methods.

2 Setup and Context

Matrix completion (MC) methods originate in the fields of computer science and statistics, where they have been shown to efficiently recover the missing entries of a real-valued matrix with bounded error. Such a reconstruction is useful in many applications, ranging from recommender systems to Global Positioning Systems, remote sensing problems, and applications with corrupted data (Candes & Plan, 2010).

2.1 Recovery of Low Rank Matrices and Optimality Guarantees

The key property that enables accurate reconstruction of missing or corrupted data, even in high-dimensional settings, is that the matrix to be completed has *low rank*. A low-rank matrix can be represented as the product of two much smaller matrices, thereby capturing the essential structure of the data with fewer parameters. In the context of observed data, assuming a matrix has low rank implies that the data exhibits a certain level of regularity or structure (Candès & Recht, 2009). Said low-rank property allows for efficient estimation of missing entries, as it reduces the effective number of parameters to be estimated. It also provides a form of regularization, helping to prevent overfitting and allowing for more robust predictions of unobserved outcomes.

In a foundational paper in the MC literature, Candes and Tao (2010) derive an information theoretic limit on the minimum number of matrix entries that have to be observed for exact matrix recovery in the case of a deterministic matrix observed without noise. Specifically, they show that for an $n \times n$ matrix of rank r, at least $O(nrlogn)$ randomly sampled entries are necessary to ensure exact recovery with high probability. This result was quickly extended by Candes and Plan (2010), who show that matrix completion is provably accurate even when the few observed entries are corrupted with a small amount of noise. They demonstrate that an unknown $n \times n$ matrix of low rank r can be recovered from $nr\log^2(n)$ noisy samples with an error that is proportional to the noise level. Such a low-rank approximation $\boldsymbol{L}$ is at the heart of modern matrix completion methods.

We require some additional notation to formalize this problem. Let $\boldsymbol{Y} \in \mathbb{R}^{N \times T}$ be a data matrix, the elements of $Y_{i,t}$ which are either *observed* ($Y_{i,t} \in \mathcal{O}$) or *missing* ($Y_{i,t} \in \mathcal{M}$).

Definition 1: Orthogonal Projection
The matrix $P_{\mathcal{O}}(A)$ is the orthogonal projection which contains all known (observed) information about A; $P^{\perp}_{\mathcal{O}}(A)$ is its complement:

$$P_{\mathcal{O}}(A)_{i,t} = \begin{cases} A_{i,t} & \text{if } (i,t) \in \mathcal{O} \\ 0 & \text{if } (i,t) \in \mathcal{M} \end{cases} \quad \text{and} \quad P^{\perp}_{\mathcal{O}}(A)_{i,t} = \begin{cases} 0 & \text{if } (i,t) \in \mathcal{O} \\ A_{i,t} & \text{if } (i,t) \in \mathcal{M} \end{cases} \tag{1}$$

(Candes & Tao, 2010, p. 2054)

The matrix $\boldsymbol{Y}$ can (in principle) be recovered from $\boldsymbol{P_{\mathcal{O}}(Y)}$ if it is the *unique matrix of rank* $\leqslant r$ that is consistent with the data, that is, if $\boldsymbol{Y}$ is the *unique solution* to equation

$$\min \operatorname{rank}(\boldsymbol{L^*}) \; s.t. \; \boldsymbol{P_{\mathcal{O}}(L^*)} = \boldsymbol{P_{\mathcal{O}}(Y)} \tag{2}$$

(Candes & Tao, 2010, p. 2054). Rank minimization is in general an NP-hard problem, which makes (2) computationally intractable for practical purposes. Fortunately, Mazumder, Hastie, and Tibshirani (2010) prove that exact matrix completion is instead possible by modifying the objective function and solving the following *convex* relaxation:

$$\min \|\boldsymbol{L^*}\|_* \; s.t. \; \boldsymbol{P_{\mathcal{O}}(L^*)} = \boldsymbol{P_{\mathcal{O}}(Y)} \tag{3}$$

In (3), $\| \circ \|_*$ denotes a particular *matrix norm*. Matrix norms serve several purposes in MC, one of which is similar to the purpose of vector norms in modern statistical methods of *shrinkage estimation*, also called *regularization*. Certain methods, such as for example *LASSO*, *shrink* estimators toward zero, in the hope of trading off unbiasedness to gain efficiency and achieve smaller mean squared error than best unbiased estimators. Vector and matrix norms, intuitively speaking, measure *distances*:

Definition 2: ℓ_1 Norm

$\|x\|_1 = \sum_{i=1}^{n} |x_i|$ (Golub & Van Loan, 2013, p. 69)

The ℓ_1 norm is used in *LASSO* to impose sparsity on parameter estimates. The ℓ_2 Norm, commonly used in *Ridge* regression to penalize coefficient size, is conventionally used to measure the physical distance between two points in $\mathbb{R}^n$:

Definition 3: ℓ_2 Norm

$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$ (Golub & Van Loan, 2013, p. 69)

To generalize these norms to matrices, we need the inner product:

Definition 4: Inner Product

$\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \mathbf{trace}(A^\top B)$ (Strang, 2006, p. 23)

The *Fröbenius Norm* is the matrix equivalent to the ℓ_2 norm:

Definition 5: Fröbenius Norm

$\|\boldsymbol{A}\|_{\boldsymbol{F}} = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2} = \sqrt{\langle A, A \rangle}$ (Golub & Van Loan, 2013, p. 71)

Intuitively, this norm allows us to measure the *distance* between two matrices: what for linear regression is the $SSR = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$, the Fröbenius Norm is for matrices, that is, we

can use it to measure how well a low-rank approximation represents the original matrix ($\boldsymbol{Y}$). It is, however, unsuitable as an objective function for our minimization problem:

$$\min_L \frac{1}{|\mathcal{O}|} \sum_{(i,t)\in\mathcal{O}} (Y_{i,t} - L_{i,t})^2 \iff \min_L \frac{1}{|\mathcal{O}|} \|\boldsymbol{P_{\mathcal{O}}(Y - L)}\|_F^2 \tag{4}$$

For $(i,t) \in \mathcal{M} \implies \notin \mathcal{O}$, the objective function does not depend on $L_{i,t}$ as $L_{i,t} \notin \boldsymbol{P_{\mathcal{O}}(Y - L)}$, but for $(i,t) \in \mathcal{O}$, minimization is achieved by setting $L = Y$, so this estimator would just return $\boldsymbol{Y}$ (Athey et al., 2021, p. 1721). This is where the final building block comes into play: based on the singular values of $\boldsymbol{Y}$, we can compute the *Nuclear Norm*:

Definition 6: Singular Value Decomposition (SVD)
Any matrix $A \in \mathbb{R}^{m\times n}$ can be factored into $A = U\Sigma V^\top$ where the columns of $U_{m\times m}$ are the eigenvectors of $AA^\top$, the columns of $V_{n\times n}$ are the eigenvectors of $A^\top A$, and the r singular values σ_i on the diagonal of $\Sigma_{m\times n}$ are the square roots of the nonzero eigenvalues of both $AA^\top$ and $A^\top A$ (Strang, 2006, p. 367)

Definition 7: Nuclear Norm
$\|\boldsymbol{A}\|_* = \sum_{i=1}^{r} \sigma_i(A)$ (Golub & Van Loan, 2013, p. 71)

This norm is particularly useful to us as it allows us to write (3) as the tightest possible relaxation of the NP-hard original problem (2). Adding a Nuclear Norm *penalty term* $\lambda\|\boldsymbol{L}\|_*$ to (3) solves our earlier problems: for $(i,t) \in \mathcal{M} \implies \notin \mathcal{O}$, the objective function now *does* depend on $L_{i,t}$ as $L_{i,t} \in \|\boldsymbol{L}\|_*$, and for $(i,t) \in \mathcal{O}$, minimization is *no longer* simply achieved by setting $L = Y$, so that in this case, it no longer just returns $\boldsymbol{Y}$ and becomes useful: the solution to (3) is the low rank matrix we want to recover:

Estimator 1: `SOFT-IMPUTE` (Mazumder, Hastie, & Tibshirani, 2010)

$$\hat{\boldsymbol{L}}_{\boldsymbol{MHT}} = \arg\min_L \left[\sum_{(i,t)\in\mathcal{O}} \frac{(\boldsymbol{Y_{i,t} - L_{i,t}})^2}{|\mathcal{O}|} + \lambda\|\boldsymbol{L}\|_* \right]$$

The computation of this estimator is possible in several different ways, the most efficient of which is the algorithm proposed by Mazumder, Hastie, and Tibshirani (2010), which uses clever initialization strategies in an expectation–maximization type algorithm that iteratively updates the missing elements of $\boldsymbol{L}$ with those obtained from a soft-thresholded Singular Value Decomposition (SVD). With warm starting, this allows efficient computation of an entire regularization path of solutions on a grid of values of the regularization parameter. The computationally intensive part of the algorithm is the computation of the SVD of a dense matrix. Exploiting the problem

structure, the authors show that the task can be performed with a complexity linear in the matrix dimensions. Further details are relegated to the Appendix.

2.2 Use Case: Causal Inference on Panel Data

The common thread in the statistical and mathematical literature on MC is that entries are assumed to be missing (uniformly) at random, which is a reasonable assumption in applications concerning sensor data. For other applications, especially those found in the social sciences, however, this assumption is less reasonable. A 2021 paper by Athey, Bayati, Doudchenko, Imbens, and Khosravi (hereafter Athey et al. (2021)) introduces an MC estimator that allows for deterministic patterns of missing data, which renders it useful for causal inference on certain types of observed data. That estimator is the main focus of this dissertation and is introduced in Section 3.

This dissertation focuses on the case of *panel data*, also sometimes referred to as *longitudinal* data, which consists of observations on multiple units (e.g., individuals, firms, countries) over several time periods, allowing researchers to study both cross-sectional and temporal variations simultaneously. Unlike cross-sectional data (which provides a snapshot of multiple units at a single point in time), repeated cross-sectional data (which observes different units from the same population at different times), or time series data (which tracks a single unit over time), panel data provides a multidimensional view that captures both differences between units and changes within units over time, enabling more robust analysis of dynamic relationships and individual-specific effects. While panel data shares similarities with multivariate time series data, the key distinction lies in the structure and interpretation: multivariate time series typically focus on the interdependencies among multiple variables for a single unit over time, whereas panel data emphasizes the behavior of multiple units, each potentially influenced by its own set of variables, observed over time.

I will consider a simple type of panel data: consider a real-valued outcome of interest $Y_{i,t}$, where $i = 1, \dots, n$ denote the units of observation and $t = 1, \dots, T$ denote time periods. The set $\boldsymbol{Y}_{N \times T}$ is a panel data set. Panel data sets are frequently used to draw causal inference on the effects of (policy) interventions, typically called *treatment effects* (Card & Krueger, 2000; Imbens, 2004). The main reason for their popularity is that they are amenable to the potential outcomes model popularized by Rubin (1974). Central to this model is the concept of potential outcomes (PO), where each unit has multiple possible outcomes depending on the treatment received: in each period, a unit may or may not be exposed to a binary treatment, denoted by W_{it}. Here, $W_{it} = 1$ indicates that the unit receives the treatment, which corresponds to the PO $Y_{i,t}(1)$, while $W_{it} = 0$ signifies no treatment, which corresponds to the PO $Y_{i,t}(0)$. For each unit and period, we observe the pair (W_{it}, Y_{it}), where Y_{it} represents the realized outcome, defined as $Y_{it} = Y_{it}(W_{it})$.

Assumption 1: Stable Unit Treatment Value
The potential outcomes for any unit do not vary with the treatments assigned to other units, and, for each unit, there are no different forms or versions of each treatment level, which lead to different PO (Imbens & Rubin, 2015, p. 10)

If assumption 1 holds, we can define our estimand of interest, the average treatment effect for the treated (ATT):

Definition 8: ATT over Potential Outcomes

$$\tau = \Big(\sum_{(i,t):W_{i,t}=1} [Y_{i,t}(1) - Y_{i,t}(0)] \Big) / \sum_{i,t} W_{i,t}$$

Framing the problem of treatment effect estimation as a missing data problem is what makes MC methods intuitively appealing. If we can accurately impute the unobserved elements of the matrix $\mathbf{Y}(\mathbf{0})$, we can directly compute treatment effects, as opposed to estimating them using methods that make additional identifying assumptions such as difference-in-differences and synthetic control. Doing so requires modifications to the MC methods that originate in the statistical and mathematical literatures and brings us to the estimator proposed by Athey et al. (2021).

3 The MC-NNM Estimator

To omit very frequent citation, I note that the notation and most definitions in this section are taken entirely from Athey et al. (2021).

3.1 Setup and Outcome Model

As before, we observe N units over T periods, where $Y_{i,t}(0)$ is the *observed* untreated PO, and unit $Y_{i,t}(1)$ is the *observed* PO if unit i has been exposed to the binary treatment. Assuming (SUTVA), for each unit in each period, we observe $(W_{i,t}, Y_{i,t})$, where the treatment indicator

$$W_{i,t} = \begin{cases} 1 & \text{if } (i,t) \in \mathcal{M}issing \\ 0 & \text{if } (i,t) \in \mathcal{O}bserved \end{cases}$$

indicates missingness of $Y_{i,t}$ in the matrix $\mathbf{Y} = \mathbf{Y}(\mathbf{0})$ of ***untreated*** outcomes. The estimator exclusively leverages information in this outcome matrix, which notably *does not include observations of treated units*. Athey et al. (2021) detail how one can incorporate those values into the estimation process, but this comes at a cost: in order to use the observed treated values, we would have to assume that the treatment effect is either constant or have a *low-rank pattern*

that would have to be specified for estimation (Athey et al., 2021, Section 8.2). Assuming this, however, in a way defeats one of the main appeals of this method, which is its reliance on only one identifying assumption in addition to 1 and 2, as compared to the four to five additional assumptions found in other popular methods for this type of causal inference. The matrix $\mathbf{W}$ records treatment assignment. Taken together,

$$\mathbf{Y_{N\times T}} = \begin{pmatrix} Y_{1,1} & Y_{12} & ? & \dots & Y_{1,T} \\ ? & ? & Y_{23} & \dots & ? \\ Y_{3,1} & ? & Y_{3,3} & \dots & ? \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ Y_{N,1} & ? & Y_{N,3} & \dots & ? \end{pmatrix} \text{ and } \mathbf{W}_{N\times T} = \begin{pmatrix} 0 & 0 & 1 & \dots & 0 \\ 1 & 1 & 0 & \dots & 1 \\ 0 & 1 & 0 & \dots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & 0 & \dots & 1 \end{pmatrix}$$

represent the outcome data in the form of one incomplete matrix ($\mathbf{Y}$), and one complete matrix ($\mathbf{W}$). One of the main contributions of Athey et al. (2021) is to allow for non-random patterns of missingness in $\mathbf{Y}$, including those with time-series dependency structures.

I directly introduce the most complete version of the outcome model, which corresponds to the most complex version of the MC-NNM estimator. In the following, for any random variable $\cdot$, $\cdot^*$ denotes a true value, with $\widehat{\cdot}$ as the corresponding estimated values. $\mathbf{1}_D$ denotes a 1-vector of length D.

Athey et al. (2021) model the outcome matrix $\boldsymbol{Y}_{N\times T}$ as the sum of real-valued matrices, each with dimensions $N \times T$. The core piece of this sum is the *low-rank matrix* $\boldsymbol{L}^*$:

Definition 9: Low-rank Matrix $\boldsymbol{L}^*$
$\boldsymbol{L}^*_{N\times T}$ captures the low-rank components of $\boldsymbol{Y}$

Additionally, the outcome model contains at least one, and possibly all, of the following:

Definition 10: Unit Fixed Effects
$\mathbf{1}_T^\top$ captures the unobserved, possibly heterogeneous, time-invariant characteristics of each individual observation (Gelman, 2005, Definition 1, p. 20). Its coefficient is $\boldsymbol{\Gamma}^*_{N\times 1}$

Definition 11: Time Fixed Effects
$\mathbf{1}_N^\top$ captures the unobserved, possibly heterogeneous, period-specific effects on all observations (Gelman, 2005, Definition 1, p. 20). Its coefficient is $\boldsymbol{\Delta}^*_{T\times 1}$

Definition 12: Unit-Specific Covariates
For unit i we observe a vector X_i and $\boldsymbol{X}_{N\times P}$ with row $i = X_i^\top$ where P denotes the number of available covariates

Definition 13: Time-Specific Covariates
For period t we observe a vector Z_t and $\boldsymbol{Z}_{T\times Q}$ with row $t = Z_t^\top$ where Q denotes the number of available covariates

Definition 14: Covariate Coefficient Matrix
$\boldsymbol{H}^* \in \mathbb{R}^{P\times Q}$ **contains the coefficients of** $\boldsymbol{X}$ **and** $\boldsymbol{Z}$**:** $\boldsymbol{X}_{N\times P}\boldsymbol{H^*}_{P\times Q}\boldsymbol{Z}^\top_{Q\times T} \in \mathbb{R}^{N\times T}$

$\boldsymbol{H}$ is the coefficient matrix for $\boldsymbol{X}$ and $\boldsymbol{Z}$. We can augment $\boldsymbol{X}$ and $\boldsymbol{Z}$:

Definition 15: Augmented Covariate Matrices
$\tilde{\boldsymbol{X}}_{\boldsymbol{N\times(P+N)}} = [X_{N\times P}\ ;\ I_{N\times N}]$ **is** $\boldsymbol{X}$ **appended by an Identity matrix** $\boldsymbol{I}_{N\times N}$
$\tilde{\boldsymbol{Z}}_{\boldsymbol{T\times(Q+T)}} = [Z_{T\times Q}\ ;\ I_{T\times T}]$ **is** $\boldsymbol{Z}$ **appended by an Identity matrix** $\boldsymbol{I}_{T\times T}$

and additionally augment the coefficient matrix $\boldsymbol{H}$:

Definition 16: Augmented Covariate Coefficient Matrix
$\tilde{\boldsymbol{H}}^* = \begin{bmatrix} \boldsymbol{H}^*_{\boldsymbol{X},\boldsymbol{Z}} & \boldsymbol{H}^*_{\boldsymbol{X}} \\ \boldsymbol{H}^*_{\boldsymbol{Z}} & 0 \end{bmatrix}$ **with** $\boldsymbol{H}^*_{\boldsymbol{X},\boldsymbol{Z}} \in \mathbb{R}^{\boldsymbol{P\times Q}}, \boldsymbol{H}^*_{\boldsymbol{Z}} \in \mathbb{R}^{\boldsymbol{N\times Q}}, \boldsymbol{H}^*_{\boldsymbol{X}} \in \mathbb{R}^{\boldsymbol{P\times T}}$
$\tilde{\boldsymbol{H}}^*_{(N+P)\times(T+Q)}$ **contains the coefficients of the augmented covariate matrices:**
$\tilde{\boldsymbol{X}}_{N\times(P+N)}\tilde{\boldsymbol{H}}^*_{(N+P)\times(T+Q)}\tilde{\boldsymbol{Z}}^\top_{(Q+T)\times T} + \tilde{\boldsymbol{H}}^*_{\boldsymbol{Z}}\boldsymbol{Z}^\top_{(Q+T)\times T} + \boldsymbol{X}_{N\times(P+N)}\tilde{\boldsymbol{H}}^*_{X} \in \mathbb{R}^{N\times T}$ [a]

[a] The red and blue matrices are incorrectly given as $\tilde{\boldsymbol{H}}^*_{X,Z}$ and $\tilde{\boldsymbol{Z}}^\top$ in Athey et al. (2021, (18))

Augmenting these matrices allows for a flexible modeling of covariate effects that can capture interactions between unit-specific and time-specific covariates, as well as their direct linear effects, enabling the model to account for complex relationships between covariates and the outcome variable $\boldsymbol{Y}$ across both the cross-sectional and temporal dimensions of the panel data. Finally, Athey et al. (2021) describe a third type of covariate:

Definition 17: Unit-Time-Specific Covariates
For period t and unit i we observe $V_{i,t} \in \mathbb{R}_{J\times 1}$ where J denotes the number of available covariates. In matrix form, $\boldsymbol{V}_{N\times T\times J}$ is a three-dimensional tensor, which is combined with coefficient vector $\boldsymbol{\beta}^*_{J\times 1}$ yields an $N \times T$ matrix: $\left[V_{i,t}^\top \beta^*\right]_{i,t}$

Taken together, these components yield the outcome model:

Definition 18: MC-NNM Outcome Model

$$\boldsymbol{Y} = \boldsymbol{L}^* + \tilde{\boldsymbol{X}}\tilde{\boldsymbol{H}}^*_{X,Z}\tilde{\boldsymbol{Z}}^\top + \tilde{\boldsymbol{H}}^*_Z\tilde{\boldsymbol{Z}}^\top + \tilde{\boldsymbol{X}}\tilde{\boldsymbol{H}}^*_X + \boldsymbol{\Gamma}^*\mathbf{1}_T^\top + \mathbf{1}_N(\boldsymbol{\Delta}^*)^\top + \left[V_{i,t}^\top\boldsymbol{\beta}^*\right]_{i,t} + \boldsymbol{\epsilon}_{N\times T}$$

The measurement error $\boldsymbol{\epsilon_{N\times T}}$ is assumed to be conditionally mean 0: $\mathbb{E}[\boldsymbol{\epsilon}|\boldsymbol{L}^*, \tilde{\boldsymbol{H}}^*, \boldsymbol{\Gamma}^*, \boldsymbol{\Delta}^*, \boldsymbol{\beta}^*] = 0$. To facilitate the definition of the estimator, we can additionally define:

Definition 19: MC-NNM Estimated Outcome Model

$$\widehat{\boldsymbol{Y}} = \widehat{\boldsymbol{L}} + \tilde{\boldsymbol{X}}\widehat{\tilde{\boldsymbol{H}}}_{X,Z}\tilde{\boldsymbol{Z}}^\top + \widehat{\tilde{\boldsymbol{H}}}_Z\tilde{\boldsymbol{Z}}^\top + \tilde{\boldsymbol{X}}\widehat{\tilde{\boldsymbol{H}}}_X + \widehat{\boldsymbol{\Gamma}}\mathbf{1}_T^\top + \mathbf{1}_N(\widehat{\boldsymbol{\Delta}})^\top + \left[V_{i,t}^\top\widehat{\boldsymbol{\beta}}\right]_{i,t} + \boldsymbol{\epsilon}_{N\times T}$$

Which, finally, brings us to the estimator proposed by Athey et al. (2021):

Estimator 2: MC-NNM (Athey et al., 2021)
To estimate $\left(\widehat{\tilde{\boldsymbol{H}}}, \widehat{\boldsymbol{L}}, \widehat{\boldsymbol{\Gamma}}, \widehat{\boldsymbol{\Delta}}, \widehat{\boldsymbol{\beta}}\right)$ in the Outcome Model, we solve the convex program

$$\min_{\boldsymbol{H},\boldsymbol{L},\boldsymbol{\Gamma},\boldsymbol{\Delta},\boldsymbol{\beta}} \left[\frac{1}{|\mathcal{O}|}\left\|\boldsymbol{P}_{\mathcal{O}}\left(\boldsymbol{Y} - \widehat{\boldsymbol{Y}}\right)\right\|_F^2 + \lambda_L\|\boldsymbol{L}\|_* + \lambda_H\|\boldsymbol{H}\|_{1,\ell}\right]$$

where $\|\boldsymbol{H}\|_{1,\ell} = \sum_{i,t}|H_{i,t}|$ denotes the element-wise ℓ_1, $\|\cdot\|_F$ the Fröbenius, and $\|\cdot\|_*$ the nuclear norm. Adapted from Athey et al. (2021, (18))

3.2 Inference and Limitations of MC-NNM

Athey et al. (2021) make only one assumption in addition to (SUTVA), which requires another definition:

Definition 20: σ-sub-Gaussian Random Variable
A random variable X with mean $\mu = \mathbb{E}[X]$ is σ-sub-Gaussian if
$\exists\,\sigma > 0$ **s.t.** $\mathbb{E}\left[e^{\lambda(X-\mu)}\right] \leqslant e^{\frac{\sigma^2\lambda^2}{2}} \quad \forall\,\lambda \in \mathbb{R}$ (Wainwright, 2019, p. 23)

A σ-sub-Gaussian random variable follows a distribution with tails that decay at least as quickly as the tails of a Gaussian distribution, that is, exponentially. Athey et al. (2021) use this assumption in one of their main results, a theoretical upper bound on the estimation error of their proposed estimator. As this bound is given for the covariate-free case and relies on additional assumptions and notation, repeating it here is not informative for my purposes. Broadly speaking, it states that the estimation error is decreasing in N, T, and the number of non-missing observations. Under the following assumption, MC-NNM identifies the ATT, provided that treatment effects are constant over time:

Assumption 2: Unconfoundedness / Exogeneity
In the Outcome Model, $\boldsymbol{\epsilon} \perp\!\!\!\perp \boldsymbol{L}^*$, $\epsilon_{i,t} \in \boldsymbol{\epsilon}_{N\times T}$ are independent of each other and σ-sub-Gaussian

Assumption 2 is necessary only for the identification of the ATT, not for estimation. If treatment effects are not constant over time, this "changes the interpretation of the estimand, but does not in general invalidate a causal interpretation" (Athey et al., 2021, p. 1717).

This estimator offers several key strengths for causal inference in panel data settings. It makes fewer assumptions than other common methods and can handle complex dependency structures across both time and units and is flexible in accommodating various types of covariates (unit-specific, time-specific, and unit-time specific). It adapts well to different shapes of data matrices, performing well whether there are many units and few time periods or vice versa. By leveraging both cross-sectional and temporal patterns in the data, it can potentially provide more accurate imputations of missing counterfactuals compared to common regression-based methods that focus solely on one dimension. Additionally, the estimator is computationally efficient, making it feasible for large-scale applications, and it has been shown to outperform traditional synthetic control and unconfoundedness-based methods in various simulated scenarios (Athey et al., 2021).

These advantages, however, come at a cost. MC-NNM has two main drawbacks. The first is that uncertainty quantification is very challenging. Athey et al. (2021) provide theoretical guarantees in the form of concentration bounds on the Root Mean Squared Error of the estimator that cannot be evaluated in practice. Recent advances in conformal prediction offer methods of obtaining confidence intervals, but those methods are not currently implemented for this estimator and require exchangeability of the data as well as assumptions on missingness patterns (Chen et al., 2019; Gui, Barber, & Ma, 2023).

The second drawback, which is shared by many widely used estimators including difference-in-differences and synthetic control, is that these estimators treat the matrices as row and column exchangeable (Aldous, 1981). This means that swapping columns corresponding to different periods would not change the estimates. Column-exchangeable estimators ignore the temporal structure of the data, which means that when estimating outcomes form say, 2020, the estimator considers observations from 1980-1999 to be equally valuable as observations from 2001-2019, contradicting our understanding of how many processes captured by panel data typically evolve. Such an implausible assumption of column exchangeability can lead to incorrect inferences, especially in settings where time trends, policy changes, or other temporal factors play a crucial role in the outcome of interest. It limits the ability of these models to capture and account for important time-dependent patterns in the data, potentially leading to misleading conclusions about causal effects.

3.3 Beyond Row-Column Exchangeability: MC-NNM for Forecasting

Despite these hurdles, there is a compelling case for exploring the application of this estimator in panel data forecasting, an area where it has not been studied before. Similar to imputing (potential) outcomes that are missing due to treatment, MC-NNM can also be used to impute outcomes which are missing because they have not been realized yet. This application is particularly promising for settings with large T, as the concentration bounds provided by Athey et al. (2021) suggest improved performance as the number of time periods increases. Moreover, matrix completion offers a unique advantage in its ability to simultaneously leverage both cross-sectional and time-series correlation patterns. As noted in the empirical comparison of MC-NNM and different types of linear models conducted by Athey et al. (2021), MC-NNM can potentially bridge the gap between these methods, avoiding the need to choose between focusing on cross-sectional or temporal correlations. This versatility in capturing multidimensional patterns makes matrix completion especially appealing for panel data forecasting, where both unit-specific and time-varying factors play crucial roles. By exploiting these complex relationships in the data, matrix completion could provide more accurate and robust forecasts compared to traditional methods that primarily focus on one dimension of the data structure.

A useful extension to MC-NNM described in Athey et al. (2021, Section 8.3) allows us to take temporal correlations in the error term $\boldsymbol{\epsilon}$ into account when estimating MC-NNM. This extension only requires a straightforward modification to the objective function of MC-NNM: the introduction of a variance-covariance matrix $\boldsymbol{\Omega}$. This is analogous to Generalized Least Squares estimators, which use an inverted covariance matrix to weight observations (Pesaran, 2015, Chapter 5.4). While this matrix can in principle account for cross-sectional dependency structures in the error term as well as temporal ones, the extension described by Athey et al. (2021) focuses on temporal structures. Exploring the modelling of cross-sectional dependencies instead of (or in addition to) temporal ones is an interesting topic for further research.

Definition 21: Covariance Matrix
Let $\boldsymbol{Y}_i$ denote the i-th row of $\boldsymbol{Y}$ and $\boldsymbol{\epsilon_i} \in \boldsymbol{\epsilon}_{N\times T}$ the corresponding rows of the error matrix, both of shape $1 \times T$. $\mathbb{E}[\boldsymbol{\epsilon}_i^\top \boldsymbol{\epsilon}_i] = \boldsymbol{\Omega}_{T\times T}$ reflects the (temporal) structure in the components $\boldsymbol{\epsilon}_i$ of the error term $\boldsymbol{\epsilon}$

In the simple example of an autoregressive process of order 1 (AR(1)) with autocorrelation coefficient ρ given by Athey et al. (2021, Section 8.3), this would correspond to $\Omega_{t,s} = \sigma^2\rho^{|t-s|}$. However, $\boldsymbol{\Omega}$ can also capture more complex error structures, such as higher-order AR(p) processes, moving average (MA) processes, and mixed ARMA processes. For MA(q) processes, $\boldsymbol{\Omega}$ would have non-zero elements up to the q-th off-diagonal. In ARMA(p, q) cases, Ω would reflect a combination of both AR and MA components. This flexibility allows the model to accommodate a wide range of error structures, from simple autocorrelation to more complex time series dynamics.

The structure of the covariance matrix is more complex than the simple AR(1) example suggests. For general ARMA processes, van der Leeuw (1994) provides a comprehensive closed-form solution for $\boldsymbol{\Omega}$ derived from a complex matrix equation involving special types of Toeplitz matrices. While more intricate than the toy example above, this formulation offers valuable insights into the structure of the covariance matrix for stationary ARMA processes. It also provides a basis for computing the inverse and determinant of the covariance matrix efficiently. However, it is important to note that this approach is designed for stationary processes and does not address nonstationary cases, meaning it is not applicable to time series where statistical properties such as mean, variance, or autocorrelations change over time. Nonstationary stochastic processes, such as those with unit roots or time-varying parameters, require different analytical techniques, such as differencing, time-varying parameter models, or state-space representations. Such nonstationary stochastic processes are beyond the scope of this dissertation and are discussed in detail for example in Pesaran (2015, Chapter 15).

Analytically, incorporating the covariance matrix into MC-NNM is simple: rewriting the objective function in terms of sums rather than matrices, and re-using our earlier definition of $\widehat{\boldsymbol{Y}}$, we can define what I will call MC-NNM with **T**emporally **S**tructured **R**esiduals:

Estimator 3: MC-NNM-TSR (Athey et al., 2021)
Let $\boldsymbol{Y_i}$ and $\hat{\boldsymbol{Y}}_i$ denote the i-th rows of $\boldsymbol{Y}$ and $\hat{\boldsymbol{Y}}$, respectively. To estimate $\left(\widehat{\widetilde{\boldsymbol{H}}}, \widehat{\boldsymbol{L}}, \widehat{\boldsymbol{\Gamma}}, \widehat{\boldsymbol{\Delta}}, \widehat{\boldsymbol{\beta}}\right)$ in the Outcome Model given a known covariance matrix $\boldsymbol{\Omega}$, we solve the convex program

$$\min_{\boldsymbol{H},\boldsymbol{L},\boldsymbol{\Gamma},\boldsymbol{\Delta},\boldsymbol{\beta}} \left[\frac{1}{|\mathcal{O}|} \sum_{(i,t)\in\mathcal{O}} \sum_{(i,s)\in\mathcal{O}} (Y_{it} - \hat{Y}_{it})[\Omega^{-1}]_{ts}(Y_{is} - \hat{Y}_{is}) + \lambda_L \|\boldsymbol{L}\|_* + \lambda_H \|\boldsymbol{H}\|_{1,\ell} \right]$$

where $\|\boldsymbol{H}\|_{1,\ell} = \sum_{i,t} |H_{i,t}|$ denotes the element-wise ℓ_1, $\|\cdot\|_F$ the Fröbenius, and $\|\cdot\|_*$ the nuclear norm. Summation over entries in $\mathcal{O}$ is equivalent to using the orthogonal projection operator as in MC-NNM.
Adapted from Athey et al. (2021, (18) and Section 8.3)

This estimator is equivalent to MC-NNM if $\boldsymbol{\Omega} = \boldsymbol{I}_{T\times T}$. If $\boldsymbol{\Omega} \neq \boldsymbol{I}_{T\times T}$, the estimates produced by the two estimators will differ. It is important to note that MC-NNM-TSR can **not** be used for causal inference in the same way that MC-NNM can, as imposing any temporal structure on the residuals violates identifying Assumption 2: the $\epsilon_{i,t}$ are no longer independent if we specify a temporal structure, which means the estimate for τ produced by MC-NNM-TSR no longer matches the MC-NNM estimand, the ATT. Utilizing the additional information contained in $\boldsymbol{\Omega}$ could, however, be helpful in forecasting application in which we do not necessarily care about causal interpretability. I introduce such an application in Section 4.

3.4 Implementation

First, however, I will briefly discuss the practical implementation of both MC-NNM and MC-NNM-TSR. For the former, two implementations exist: `fect` (Liu, Wang, & Xu, 2024) in R[1] and `causaltensor` in Python. Both implement MC-NNM as part of a larger suite of causal inference methods. Neither implements the unit-time-specific covariates introduced above. In forecasting tasks, these covariates can be crucial to utilizing predictors that vary across both time and units. For MC-NNM-TSR, no implementations exist.

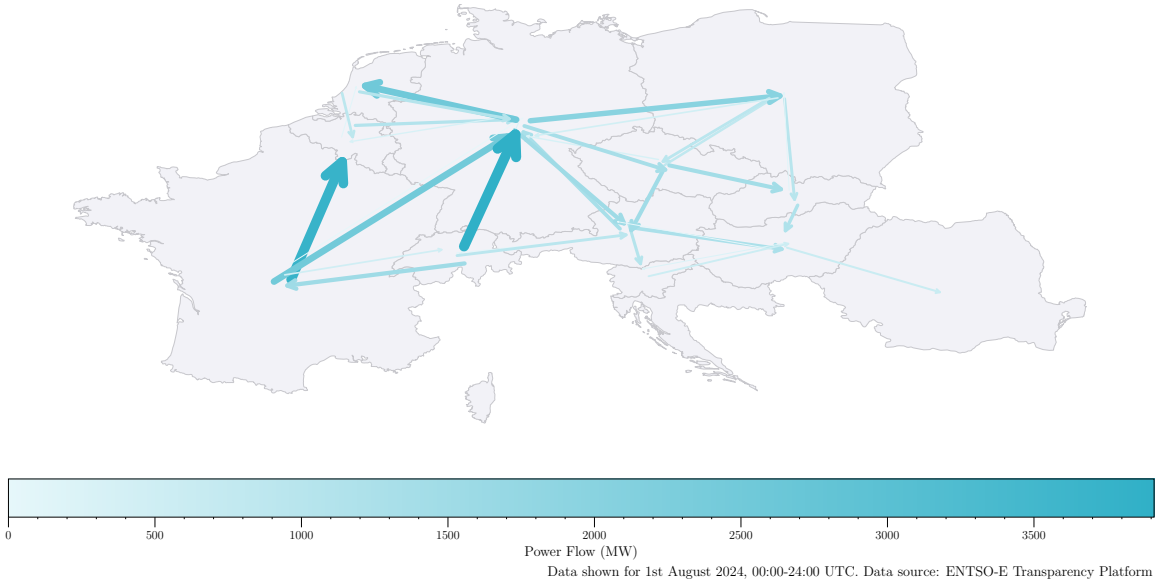
To address the lack of both of these capabilities in existing implementations, I have implemented MC-NNM-TSR from scratch following the verbal description in Athey et al. (2021) in Python using `JAX` (Bradbury et al., 2018), a high-performance numerical computing library. `JAX` is widely adopted in modern machine learning frameworks due to its GPU compatibility and focus on performance optimization. It offers just-in-time (JIT) compilation and efficient vectorization capabilities, which can significantly speed up numerical computations, especially for operations involving large matrices. My implementation is available as an open-source package named `lightweight-mcnnm`. It is easy to use, rigorously tested, available for the latest three python versions and for all operating systems. The source code along with examples demonstrating its functionality and a comparison to both `fect` and `causaltensor` can be found on Github (Schnabel, 2024a). A detailed documentation that explains the workings of every internal function is available here (Schnabel, 2024b). Additional details about my implementation, including the soft-thresholding operator introduced by Mazumder, Hastie, and Tibshirani (2010), as well as design choices I have made are relegated to the Appendix. There are three specific architectural choices that will become relevant in the application. First, all core estimation functions are JIT compiled, which means that they are converted to optimized machine code at runtime. This compilation occurs the first time a function is called with a given set of matrix shapes, resulting in significantly faster subsequent executions, which is particularly beneficial for iterative algorithms like the cyclic coordinate descent used in MC-NNM. It can, however, increase execution time when the functions are called several times on input matrices of varying sizes, as the functions have to be recompiled in their entirety. Second, I have enabled 64-bit precision for floating point numbers throughout the implementation. This can help with numerical stability and estimation precision, but comes at the cost of doubling memory usage and slower computation times compared to 32-bit precision. Both JIT compilation and 64-bit precision can be turned off by any user of the package, the former leading to slower and the latter to faster execution times in nearly all cases. Lastly, I have implemented a new holdout validation approach as an alternative to the cross-validation described by Athey et al. (2021). This is described in the Appendix and, unlike cross-validation, respects the time series nature of observations.

[1] Both `fect` and its precursor, `gsynth` (also maintained by Liu, Wang, and Xu (2024)), are largely identical to the original implementation of Athey et al. (2021), which is a functional but entirely undocumented `Rcpp` package called `MCPanel` that has not been updated since 2017.

4 Application: Electricity Price Forecasting

To test the viability of using both MC-NNM and MC-NNM-TSR for panel data forecasting, I evaluate their performance on publicly available electricity price data for 2,007 days comprising 13 European countries: Austria, Belgium, the Czech Republic, France, Germany and Luxembourg, Hungary, The Netherlands, Poland, Romania, Slovakia, Slovenia, and Switzerland. These countries form the 12 bidding zones (Germany and Luxembourg are one joint bidding zone) in the so-called Core Capacity Calculation Region (CORE CCR) of integrated electricity markets. The reason why I choose this application is the transmission of electricity within this region. In many panel data sets, researchers can hypothesize about (causal) mechanisms relating one unit's observed outcome to those of other units. For example, it is widely accepted that stock prices of assets within a certain sector tend to be correlated, and a crude hypothesis of a cross-sectional mechanisms influencing the stock price of a single asset is that large movements in the prices of other assets will influence the price of an asset in the same sector. In the case of electricity market data, however, these mechanisms are *physical* and *observable*: we know that electricity (power) is traded and physically transmitted between countries. Figure 1 below shows the physical transmission volume between the 12 bidding zones I consider on August 1, 2024.

Figure 1: Cross-Border Flows



4.1 Application Setup

Most empirical literature treats the problem of forecasting electricity prices as a univariate time-series forecasting task. The way that the market on which these prices are determined works is as follows: On **day D-1**, Market participants such as Grid operators, utilities, and other power traders submit bids and offers for delivery and receipt of physical power on power exchanges such as the European Power Exchange (EPEX). This process occurs on every day of

the year, and all bids and offers must be submitted by a deadline called *gate closure* of 12 noon. The bids and offers are for each hour of the following **day D**, creating a day-ahead market. The power exchange then uses an algorithm to match supply and demand, determining the market clearing price for each hour. This price is then published and becomes the reference price for electricity transactions. The goal of this forecasting application is to accurately forecast the hourly day-ahead prices for **day D** for each bidding zone on **day D-1**, measured in Euros per megawatt hour (MWh).

The interconnected nature of the European electricity grid means that prices in one country can significantly affect those in neighboring countries. For instance, excess renewable energy production in one country might lead to lower prices, which can then influence prices in connected markets. This is corroborated by Lago, De Ridder, and De Schutter (2018) and Lago et al. (2018), who find that including information about French energy production improves the forecasting performance for a model predicting Belgian day-ahead prices. The only recent work in Electricity Price Forecasting (EPF) literature I could find that combines data about several markets into one dataset is Tschora et al. (2022, Section 4.2, Table 4), who combine Belgian, French, and German data and find that a range of models perform as well or better on this combined dataset than on the individual country datasets.

There are several plausible reasons why there is not more published work using panel data for EPF. Panel data introduces additional complexity and dimensionality, especially when combining the day-ahead prices with exogenous predictors, as most models do. They are also more computationally intensive and require harmonization of input data. Publication bias could result in studies using panel data, but not finding a significant performance gain over siloed single-country forecasts, and therefore not publishing those results. Speculation on these and other reasons is outside the scope of this dissertation. To the extent that the practical challenges affect the estimators used in my comparison, my methodology, which I will describe shortly, ensures that these challenges affect all methods I compare equally.

4.2 Data Description

The dataset used in this application comprises hourly day-ahead electricity prices for the 12 European bidding zones covering 13 countries (Luxembourg and Germany constitute a joint bidding zone) listed above from January 1, 2019, 00:00 to June 30, 2024, 23:00, resulting in $578,028$ observations. In addition to the primary target variable of day-ahead electricity prices, I incorporate several exogenous variables to enhance the forecasting models' performance:

- Coal futures settlement prices for API2 European (thermal) coal futures of the evening of day D-2 (forward-filled for each hour of day D-1). Measured in US Dollars per $1,000$ metric tons

- Gas futures settlement prices for TTF natural gas futures of the evening of day D-2 (forward-filled for each hour of day D-1). Measured in Euros per MWh.
- Calendar information: hour of the day (0-23), day of the week (0-6, where 0 is Monday), day of the year (1-366), month (1-12), quarter (1-4), week of the year (1-53), and a "weekend" dummy are all included as numerical features, following standard EPF practice (available long before day D-1)
- Day-ahead generation forecasts of the total power generation per bidding zone released on day D-1 . Measured in Megawatts (MW).
- Generation forecasts for wind and solar power per bidding zone released on day D-1. Measured in MW.
- Day-ahead load forecasts of the total power demand per bidding zone released on day D-1 . Measured in MW.

All variables are available at hourly resolution. Coal and gas prices as well as load (demand) and generation (supply) forecasts are useful to forecast prices as they contain information about input prices for coal and gas power plants, as well as the expected levels of supply and demand that directly influence the clearing price. The coal and gas futures prices and calendar variables serve as time-specific covariates shared across all countries, while the remaining variables are country-specific. All data is publicly available, specific sources are described in the Appendix. The dataset contains very few missing values, as shown in Table 1 below.

Table 1: Data Quality Summary

Dataset	Total Entries	Missing Entries	Missing %	Longest Missing Streak	Overlapping Missing Entries
Day-Ahead Prices	578.028	0	0.0	0	0
Generation Forecast	578.028	1.155	0.1998%	384	193
Load Forecast	578.028	361	0.0625%	25	119
Wind and Solar Forecast	578.028	1.111	0.1922%	96	264
Coal and Gas Prices	96.338	0	0.0	0	0

Table 7 in the Appendix contains detailed information on when the respective variables become available.[2] In short, only two variables I use as exogenous predictors are not publicly available before gate closure, which is when a market participant interested in forecasting the day-ahead electricity prices to inform their trading would need them. Those market actors tend to have their own proprietary versions of these forecasts, and this application is not intended to illustrate the use of any of these models in a trading environment. I therefore use both generation forecasts as a proxy for non-public forecasts, while noting here that I cannot be certain that

[2]It also lists installed generation capacity as a variable, which I queried but did not ultimately use.

these exogenous predictors are not influenced by the forecasting target, the day-ahead price.

Figure 2 visualizes the distribution of the day-ahead prices of each bidding zone. We can see that each distribution has a relatively thin right tail, as well as a thinner left tail. All countries exhibit a right (positive) skew. It is notable that day-ahead prices can be, and sometimes are, negative.

Figure 2: Distribution of Day-Ahead Prices by Country

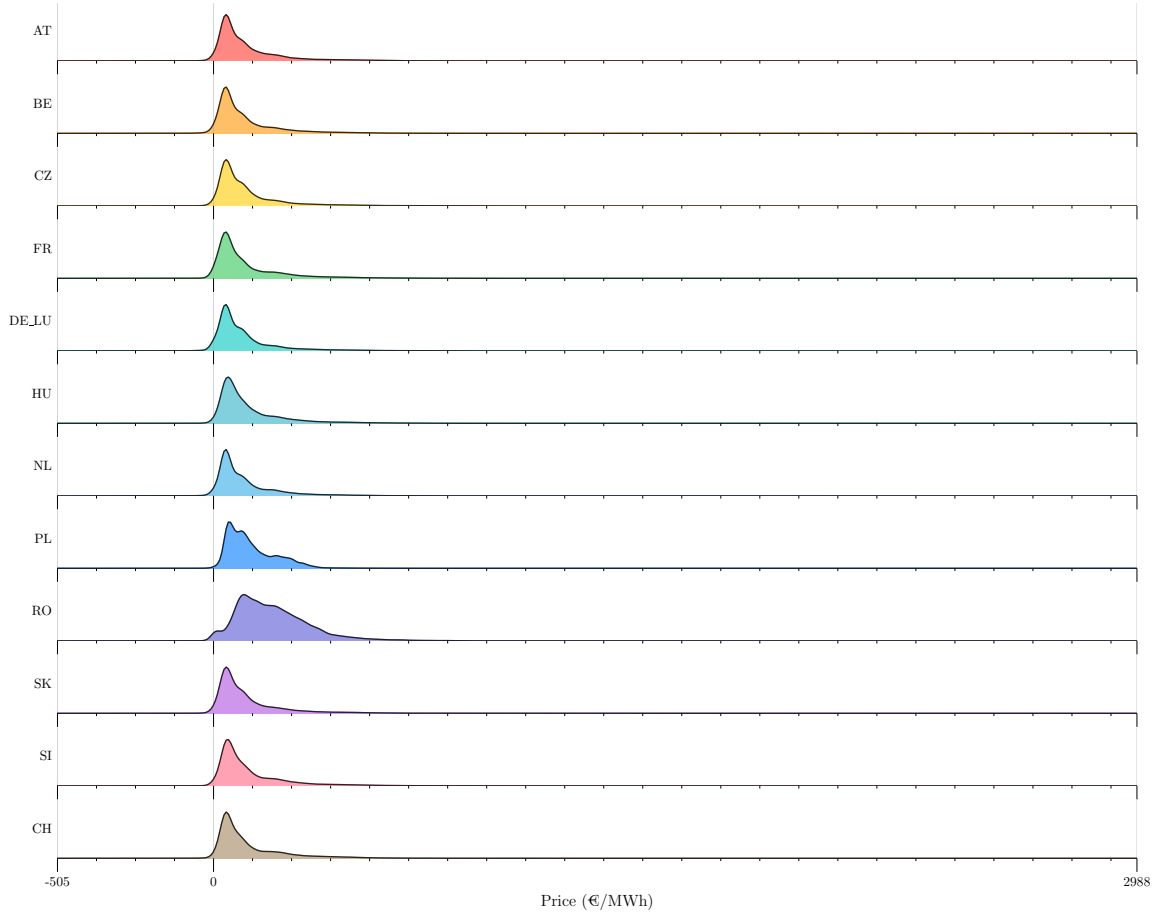
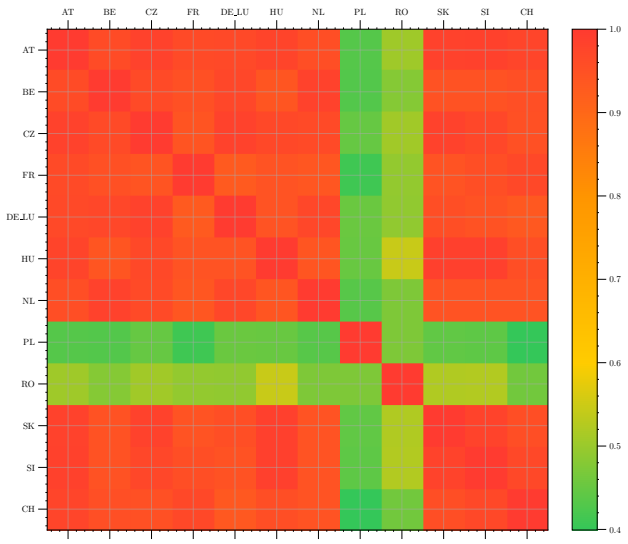


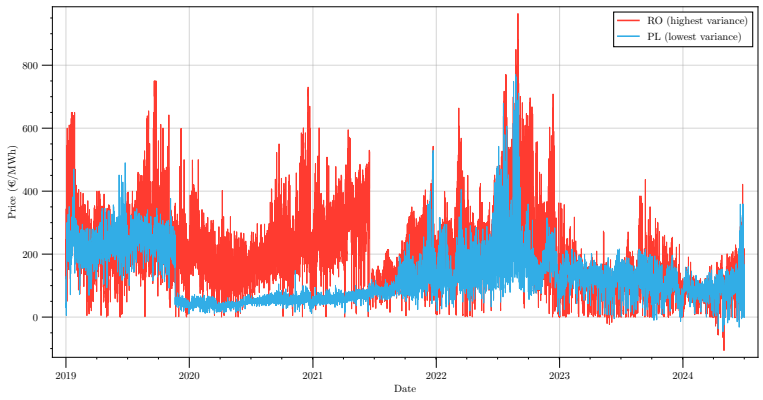
Figure 3 below visualizes the correlation in the day-ahead price between these 12 bidding zones. With the notable exception of Poland and Romania, which in Figure 2 exhibit a more pronounced asymmetric shape than other bidding zones, all other zones have very strongly correlated day-ahead prices.

Figure 3: Correlation Matrix of Day-Ahead Prices



Poland and Romania also, interestingly, exhibit the lowest and highest variance, respectively, in the day-ahead prices of all bidding zones in the dataset, as visualized in Figure 4 below.

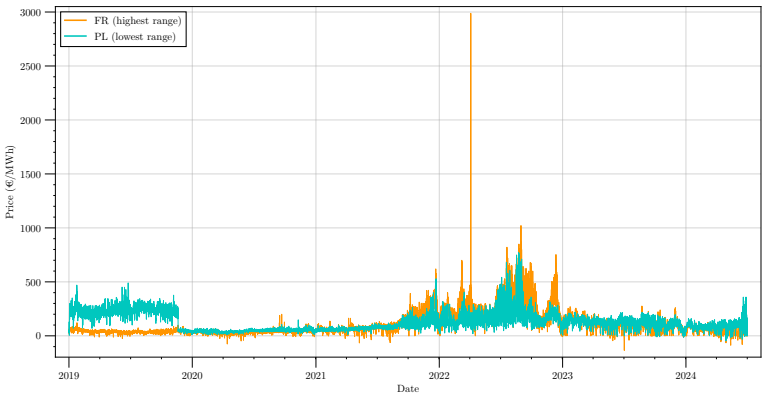
Figure 4: Largest and Smallest Variance of Day-Ahead Prices



The large variance in Romanian day-ahead prices becomes especially clear when compared to the two bidding zones with the lowest and highest ranges in day-ahead prices: Poland and France, visualized in Figure 5. Poland is a very interesting bidding zone, as it is characterized by a comparatively weak correlation in day-ahead prices to the other bidding zones, while simultaneously possessing the lowest range and variance among all bidding zones. We can also observe a break in the price level for Poland in late 2019 in both Figure 4 and Figure 5. These

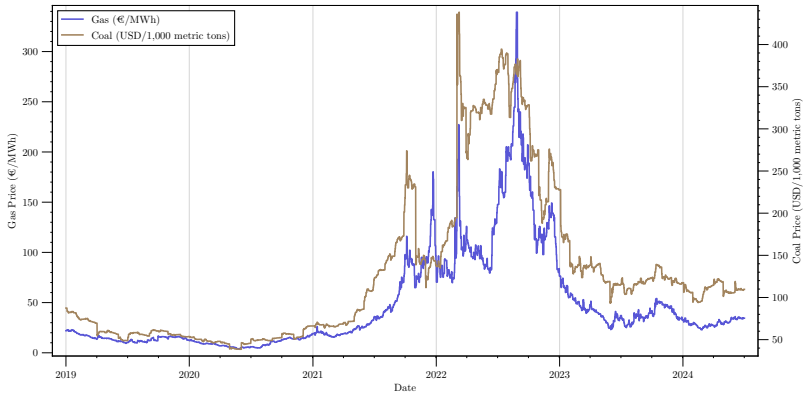
combined factors should make it difficult for any model to accurately forecast Polish day-ahead prices as part of a larger panel dataset.

Figure 5: Largest and Smallest Range of Day-Ahead Prices



Comparing figures 4 and 5 to Figure 6 below, a common pattern in electricity prices and the prices of the input used by gas and coal power plants emerges: a decline in prices early in 2020 during the Covid-19 pandemic, and a drastic increase in 2022 that coincides with the Russian war of aggression against Ukraine.

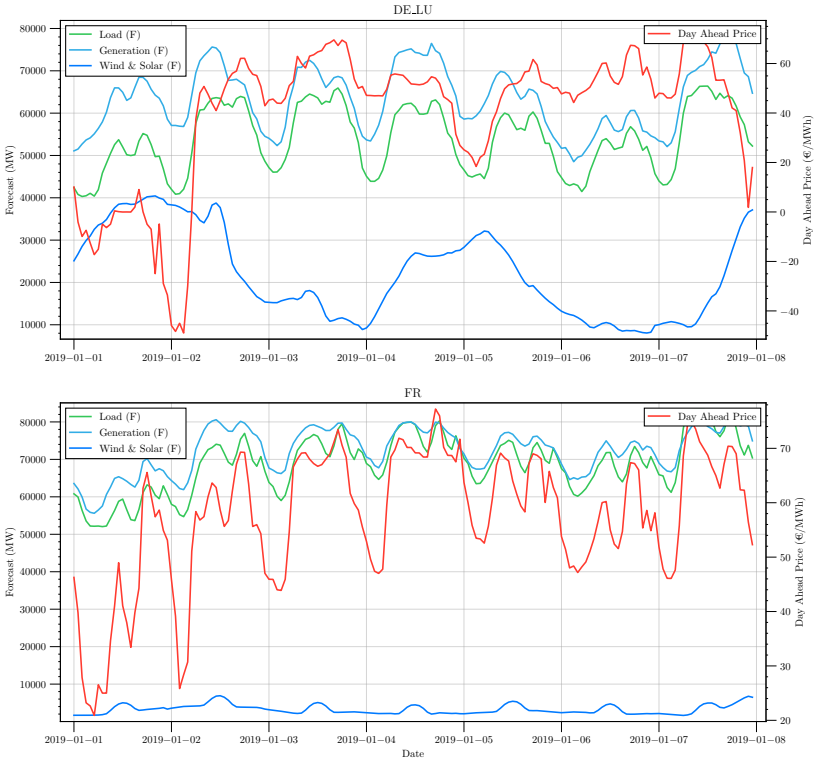
Figure 6: Coal and Gas Prices



These patterns in and of themselves are interesting from an EPF perspective, but they are not the focus of this dissertation. I impute the very few missing values for the exogenous predictors / covariates to ensure that no estimator is computed on missing data. This is a preprocessing step which itself could be done using matrix completion, making MC-NNM and MC-NNM-TSR very robust to data anomalies: one can simply complete covariate/predictor matrices and then complete the outcome matrix. This is not possible with other methods discussed below. My imputation strategy is detailed in the Appendix. Given the very low number of missing covariate

observations and the fact that no observations of the forecasting target are missing, I do not discuss it in detail here. Finally, Figure 7 below shows the first week of data for two neighboring bidding zones with markedly different patterns: the combined zone for Germany and Luxembourg (upper panel) and France (lower panel). The green and blue lines show the load, generation, and wind and solar forecasts, while the red line shows the corresponding day-ahead prices. We can observe that Wind and Solar generation (dark blue line) at times makes up a significant portion of the total power forecasted to be generated in Germany and Luxembourg, while it plays an insignificant role in France, which heavily relies on a stable base generation of Nuclear power, the excess of which it frequently exports to Germany (as shown in Figure 1). The hourly day-ahead prices are volatile and negative for most of the first day of the data in Germany and Luxembourg as a result of the high generation of cheap wind and solar energy, while they vary in a smaller range but are positive throughout the first week of the data in France.

Figure 7: Data Excerpt for Germany/Luxembourg and France



My final dataset spans January 1, 2019 00 : 00 to June 30, 24 23 : 00 for a total of 12 bidding zones$*$24 hours per day$*2,007$ days $= 578,016$ observations for each of the forecasting target and the 4 exogenous predictors. This very high dimensionality and the patterns in the data discussed just now make this a challenging dataset for a forecasting task, and therefore an interesting application. I do not test for the stationarity of any variable for three reasons. First,

such tests are subjective in that they depend on the time frame used. Second, panel versions of such tests, such as the one defined in Pesaran (2012) are not implemented in python currently. Most importantly, I compare MC-NNM and MC-NNM-TSR to both linear and nonlinear methods, for each of which the ideal preprocessing steps to mitigate possible non-stationarity differ. I therefore choose to sidestep this topic and treat possible momentary non-stationarity as a robustness exercise for all estimators.

4.3 Performance Comparison Implementation

As this dissertation is about evaluating Matrix Completion for forecasting, rather than EPF in particular, I will focus on only a few of the many models commonly used in the EPF literature in selecting comparisons to MC-NNM and MC-NNM-TSR. In particular, I exclude Neural Networks (NNs) from my comparison for three reasons. First, unlike all other estimators I use, NNs are black-box models, which means I could only speculate about specific reasons why they would or would not perform well. Second, although they tend to perform well in EPF applications focusing on univariate price series, I have been unable to find configurations of the current state-of-the art EPF NNs such as `NBEATSx` for panel data (Olivares et al., 2023), and modifying existing configurations to work well with $12 * 4 + 9$ input series rather than the typical 3-4 would have been squarely outside the scope of this dissertation. Lastly, as Lago et al. (2021) elaborate in their review of the EPF literature, it is reasonable to compare models whose computational requirements are roughly comparable: "[i]f its computational time is large, i.e. in the order of minutes, the model should indeed be more accurate than all state-of-the-art models, e.g. [Deep NNs]. If its computational time is small, i.e. in the order of seconds, the model should be more accurate than the state-of-the-art models with low computational requirements, e.g. LEAR" (Lago et al., 2021, p. 17). As I will show, both MC-NNM and MC-NNM-TSR belong in the latter category.

In constructing this application, I carefully followed the critique of common pitfalls that make the results of EPF methods difficult to compare across papers voiced by Lago et al. (2021) in designing their forecasting benchmark. Such pitfalls include using very short test periods, avoiding comparisons with well-established methods, not providing computational costs, using non-public datasets and/or model configurations, evaluating on a single market, using inadequate accuracy metrics, non-reproducible results, not recalibrating models regularly, performing ex-post hyperparameter optimization, failing to provide sufficient details for replication, and data contamination between training and test sets (Lago et al., 2021, Section 1).

My implementation of the comparison addresses these pitfalls. While Lago et al. (2021) offer their own open-source evaluation framework for both existing and new models on EPF data, the estimators and data they offer are all designed for univariate price series. Therefore, I emulate their approach and extend it in certain respects. I implement a custom Forecast "engine"

in Python that offers a standardized interface through which estimators can access the data and return predictions to the engine. This engine is designed to ensure a fair and comprehensive comparison of different forecasting methods. It initializes with the following methods: `fit()`, `predict()`, and `optimize()`. This design ensures consistency across all estimators and facilitates easy integration of new models. The engine is configured with several key parameters, including minimum and maximum training window sizes, an optimization wait period, and optimization trigger conditions such as a performance degradation threshold. These parameters allow for flexible and adaptive –but standardized– model training and evaluation. For each forecast day, the engine performs a series of operations. It first retrieves and prepares the input data for each estimator, adjusting the training window size within the specified limits. Crucially, the engine ensures that estimators do not have access to the test prices (last 24 hourly prices for all countries) for training or prediction, eliminating the possibility of data leakage. It then checks if the optimization trigger condition is met, which is the case if the optimization metric specified for each estimator has degraded by $\geqslant 10\%$ compared to the average value of the metric in the three days prior. If optimization is triggered and the wait period has elapsed since the last optimization, the engine performs hyperparameter tuning on the training data known up to that point. This process utilizes the `Optuna` package to conduct efficient Bayesian optimization over the hyperparameter space specific to each estimator for an estimator-specific number of trials (cf. Bergstra et al., 2011). This optimization is fully automated and therefore avoids cherry-picking of favorable hyperparameters. The execution time of this optimization is recorded. The best hyperparameters, based on validation performance, are then selected and used in subsequent iterations until the next optimization is triggered. Following optimization (or if optimization is not triggered), the engine calls the `fit()` method of each estimator to train on the available data using the current hyperparameters. It then uses the `predict()` method to generate forecasts for the next 24 hours (using the covariates for this test period, which are available at prediction time). Both the fitting and prediction processes are timed. The engine computes and stores various performance metrics listed below for each estimator. These metrics provide a comprehensive view of each model's performance across different aspects of forecast accuracy. The entire implementation, including the forecast engine, all individual estimators, data, results, and replication instructions are available on GitHub. This ensures seamless reproducibility of all results and allows for transparent evaluation and potential extension of my work by other researchers.

There is some disagreement in the EPF literature on how to evaluate forecasting performance. Lago et al. (2021) discuss the merits and drawbacks of various error metrics, emphasizing the importance of selecting appropriate measures for assessing forecast accuracy. They argue that research in EPF should prioritize certain metrics over others, particularly advocating for the use of the relative Mean Absolute Error (rMAE) in conjunction with other measures. Based on their recommendations and the specific characteristics of electricity price data, my analysis

will focus on three key metrics: Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and relative Mean Absolute Error (rMAE).

Definition 22: Root Mean Square Error (RMSE)
RMSE $= \sqrt{\frac{1}{24N_d}\sum_{d=1}^{N_d}\sum_{h=1}^{24}(p_{d,h} - \hat{p}_{d,h})^2}$ **where** $p_{d,h}$ **represents the actual price,** $\hat{p}_{d,h}$ **is the forecasted price,** d **is the day index,** h **is the hour index, and** N_d **is the number of days in the out-of-sample test period, which always equals 1 in my setup.**

RMSE is widely used and penalizes larger errors more heavily, providing insight into the overall magnitude of forecast errors.

Definition 23: Mean Absolute Error (MAE)
MAE $= \frac{1}{24N_d}\sum_{d=1}^{N_d}\sum_{h=1}^{24}|p_{d,h} - \hat{p}_{d,h}|$

MAE offers a linear measure of error and is less sensitive to outliers compared to RMSE, making it particularly useful in the context of electricity prices, which can exhibit significant volatility and occasional extreme values. Finally, the relative Mean Absolute Error (rMAE) is calculated as:

Definition 24: Relative Mean Absolute Error (rMAE)
rMAE $= \frac{\frac{1}{24N_d}\sum_{d=1}^{N_d}\sum_{h=1}^{24}|p_{d,h} - \hat{p}_{d,h}|}{\frac{1}{24N_d}\sum_{d=1}^{N_d}\sum_{h=1}^{24}|p_{d,h} - \hat{p}_{d,h}^{\textbf{naive}}|}$ **where** $\hat{p}_{d,h}^{\textbf{naive}}$ **represents a naive forecast, typically using the price from 1 or 7 days prior. The rMAE normalizes the MAE by the MAE of this naive forecast, facilitating comparisons across different time periods. I use** $\hat{p}_{d,h}^{naive} = p_{d-1,h}$ **as the naive forecast.**

These metrics are preferable to other common metrics for several reasons. First, they provide a comprehensive view of forecast accuracy, capturing both overall performance (RMSE) and average error magnitude (MAE). Second, rMAE allows for fair comparisons across different datasets and time periods, addressing some limitations of absolute error metrics. Thirdly, they avoid issues associated with percentage-based errors like Mean Absolute Percentage Error (MAPE), which can be problematic when dealing with prices close to zero or negative prices, both of which occur in electricity markets. Lastly, these metrics align well with the linear nature of electricity costs and profits, making them more relevant for practical applications in electricity price forecasting (Lago et al., 2021). Unlike Lago et al. (2021), I do not implement statistical tests for differences in forecasting accuracy. The complex nature of the panel data presents challenges for traditional tests like *Diebold-Mariano* or *Giacomini-White* (Giacomini & White, 2006). These tests, designed for univariate time series, are not easily interpretable in this high-dimensional context. My approach instead provides a comprehensive assessment of practical forecasting performance, which is more relevant for real-world applications in electricity price

forecasting than statistical significance alone. The engine standardizes data preparation, model fitting, prediction, and performance evaluation across all estimators, offering insights into the relative strengths of different models in this complex panel data setting.

Finally, I make three design choices that differ from the framework of Lago et al. (2021). First, while my engine allows for expanding-window forecasts that ingest more training data at each iteration, I implement 3 different sliding window configurations for each estimator listed below, which means that I limit each model to 56, 84, and 112 days of training data. This corresponds to two, three, and four months, respectively. Lago et al. (2021) use 56, 84, 1092, and 1456 days, corresponding to 2 and 3 months as well as 3 and 4 years. I choose to omit the longer two windows for two reasons. First, the data Lago et al. (2021) spans 2011 to 2017, which excludes the periods of the Covid-19 pandemic and the Russian war against Ukraine, both of which drastically change the patterns in energy prices as shown in Figures 5, 4, and 6. Second, their single-country datasets use one price series and two exogenous variables, while I use 12 countries with more covariates per country. This means that the dimensionality of my data is much higher, which would render the computation times for 3 and 4 years prohibitively high on consumer-grade hardware.

Second, I do not begin training at 56, 84, or 112 days of data. Rather, I begin at the minimum required for each estimator and expand the data window until the limits of 56, 84, and 112 days, respectively, are met, at which point the window size stays fixed and the window "slides" over the data by dropping the oldest 24 hourly points while appending the most recent 24 points for each variable. I do this to assess how MC-NNM, MC-NNM-TSR, and the other estimators discussed below perform on various data sizes, similar to the original comparison in Athey et al. (2021). This additionally enables me to compare the performance of all estimators during different periods and mitigates the impact of structural breaks such as the Covid-19 pandemic and the Russian war against Ukraine.

Third, I do not transform the data that is fed to each estimator. Lago et al. (2021) use preprocessing such as *arc hyperbolic sine* (asinh) variance-stabilizing transformation for their LEAR models (see below), but not for the NNs they evaluate. This has been shown to mitigate the impact of price spikes. Similar to my reasons to not test for stationarity, I choose not to perform such data preprocessing as a robustness exercise for the estimators. Using any kind of transformation would immediately open up a debate as to which transformation is used and how it is applied in a panel setting (for each country's series individually or across countries). I sidestep this problem by giving all estimators the exact same un-transformed data and evaluating their performance.

4.4 Estimators Compared

I now briefly introduce the estimators I will compare MC-NNM and MC-NNM-TSR against. I will keep my descriptions of these established models brief and refer to the literature where appropriate. Estimator hyperparameter ranges and optimization details are given in Table 2 at the end of this section. To facilitate this characterization, I first define some common notation, which I will align to MC-NNM's notation. Forecasting runs daily for $D \in \{2, 2007\}$. Let $\boldsymbol{Y}_{12\times(D\times24)}$ be the matrix of hourly day-ahead prices. On the forecast day D, the last 24 columns of this matrix are unknown/ missing: they are the forecast target. Define $\boldsymbol{Z}_{9\times(D\times24)}$ to be the matrix of hourly time-varying covariates (coal and gas futures settlement prices and calendar variables) shared by all units. Define $\boldsymbol{V}_{3\times12\times(D\times24)}$ as the 3-dimensional tensor containing the three hourly unit-time specific covariates (generation, wind and solar generation, and load forecasts for each country). The last 24 hours of the covariates are known at forecast time.

MC-NNM

I use MC-NNM as described earlier, using the holdout validation method described in the Appendix with the following validation parameters: an initial training window of all but the last 24 columns (which contain the prices to be forecasted, which are not available and masked out), 3 holdout folds with horizon 1 and step size 8. During optimization, the holdout validation methods searches over $18^2 = 324$ (λ_L, λ_H) pairs using $1,000$ maximum iterations and a convergence tolerance of 1.0. Between optimizations, the optimal (λ_L, λ_H) values selected by the holdout validation are used to fit the estimator directly, using $10,000$ maximum iterations and a convergence threshold of 0.01.

MC-NNM_32

Identical to MC-NNM, but using 32-bit floating point precision instead of the default 64-bit precision. This halves memory usage and reduces computation time.

MC-NNM-TSR

Identical to MC-NNM_32, but using covariance matrix $\boldsymbol{\Omega}$ computed as follows: I demean $\boldsymbol{Y}$ by subtracting the mean of each country's time series. I then compute autocorrelations for lags 1 to 24 for each country, and average these across all countries to obtain a single autocorrelation function. I then compute a decay rate λ based on the $24th$ lag average autocorrelation, ensuring a positive value by taking the maximum of the autocorrelation and a small constant (10^{-8}). The $\boldsymbol{\Omega}$ matrix is then constructed using an exponential decay function: $\Omega_{ij} = e^{-\lambda|i-j|}$ for $i, j = 1, \ldots, T$ where T is the number of time periods available. This approach captures the average autocorrelation structure across all countries while ensuring the positive definiteness of $\boldsymbol{\Omega}$ through the exponential decay function, which is required for the inversion of $\boldsymbol{\Omega}$ within

MC-NNM-TSR. As discussed earlier, $\boldsymbol{\Omega}$ can in principle accommodate very complex temporal dependency patterns. My approach to constructing it is fairly simple and hence limited. The reason I focus on autocorrelations is the extensive use of autoregressive structures in estimators popular in the EPF literature, such as the LEAR estimators discussed below.

Elastic Net

This is one of the estimators Athey et al. (2021) compare MC-NNM to in a prediction benchmark. The authors use a financial dataset of daily returns for 2453 stocks over 10 years (3082 days). They create 50 subsamples by selecting N randomly sampled stocks and their first T daily returns, with $N \times T = 4900$. They vary (N, T) from very thin to very fat matrices, ranging from (490, 10) to (10, 490), with 25% of entries missing randomly in each case. The comparison is between Elastic Net regression and MC-NNM. The performance is evaluated using average RMSE. Their results show that MC-NNM outperforms both Elastic Net estimators for all matrix shapes. To validate these findings in a forecasting context, I implement a *horizontal* Elastic Net (horizontal in the framing of Athey et al. (2021) means using the rows of $\boldsymbol{Y}$ as units of observation to analyze patterns over time within units) that is specified as follows:

Estimator 4: Elastic Net

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} (Y_{it} - X_{it}^T \boldsymbol{\beta})^2 + \lambda \left[\alpha |\boldsymbol{\beta}|_{\ell_1} + (1-\alpha) |\boldsymbol{\beta}|_{\ell_2}^2 \right] \right\}$$

where $\|\cdot\|_{\ell_1}$ denotes the ℓ_1 and $\|\cdot\|_{\ell_2}$ denotes the ℓ_2 norm, λ denotes the mixing parameter and α the regularization parameter. Y_{it} is the day-ahead price for unit i at time t, X_{it} is the feature matrix comprising generation forecasts from $\boldsymbol{V}_{1,i,t}$, load forecasts from $\boldsymbol{V}_{2,i,t}$, wind and solar generation forecasts from $\boldsymbol{V}_{3,i,t}$, and coal and gas futures settlement prices and calendar variables from $\boldsymbol{Z}_{:,t}$. $\boldsymbol{\beta}$ is the vector of coefficients to be estimated

For optimization trials, this estimator uses the last 20% of observations (rounded to whole days) as a validation set. This is the only estimator I implement that does not use lagged day-ahead prices as features, but rather attempts to forecast solely based on the exogenous variables. This approach allows me to assess the predictive power of the exogenous forecasts, and is potentially more robust to sudden structural changes in the price series such as the one in late 2019 in Poland, shown in Figure 4.

LASSO

As a bridge between Elastic Net and typical EPF methods, I implement a LASSO regression. While both estimators induce sparsity, LASSO tends to select one feature from a group of correlated features. In contrast, Elastic Net is more likely to include all correlated

features with smaller coefficients, inducing less sparsity but offering greater stability and a more balanced approach to feature selection. In a high-dimensional forecasting problem such as this application, we would *ex ante* expect both sparsity and stability to be beneficial, which is why I include both estimators (Hastie, 2017). The LASSO is specified as follows, with the lag structure and optimization metric mimicking the EPF literature (Lago et al., 2021):

Estimator 5: LASSO

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} (Y_{it} - X_{it}^T \boldsymbol{\beta})^2 + \lambda |\boldsymbol{\beta}|_{\ell_1} \right\}$$

where $|\cdot|_{\ell_1}$ denotes the ℓ_1 norm, λ denotes the regularization parameter, Y_{it} is the day-ahead price for unit i at time t, X_{it} is the feature matrix comprising price lags (24, 48, 72, 168 hours), generation forecasts (current, 24 and 168 hour lags) from $\boldsymbol{V}_{1,i,t}$, load forecasts (current, 24 and 168 hour lags) from $\boldsymbol{V}_{2,i,t}$, wind and solar generation forecasts (current, 24 and 168 hour lags) from $\boldsymbol{V}_{3,i,t}$, coal and gas futures settlement prices from $\boldsymbol{Z}_{:,t}$, calendar variables (day of week, hour dummies), and a country indicator. $\boldsymbol{\beta}$ is the vector of coefficients to be estimated.

The optimization metric is the AIC, defined as $\text{AIC} = 2k + n\log(\text{MSE})$, where k is the number of non-zero coefficients, n is the number of observations, and $\text{MSE} = \text{RMSE}^2$. For optimization trials, this estimator uses the last 20% of observations (rounded to whole days) as a validation set.

LEAR

Next, I implement a method widely used in EPF and originally introduced by Uniejewski, Nowotarski, and Weron (2016) named LEAR (*LASSO Estimated AutoRegressive*). In short, this model estimates a separate LASSO regression for each country and hour of the day, for a total of $24 \times 12 = 288$ models a day. In the existing EPF literature such as Lago et al. (2021), which only considers individual markets, this reduces to 24 models a day. This is the only method I implement that completely ignores the panel structure and instead treats each individual country as if we had no information about any other countries. The reason I implement it is that taken together with the next model, it will provide us with a relatively clear indication of whether treating EPF as a panel data problem is beneficial in terms of forecasting performance. The lag structure and performance metric is adopted from Lago et al. (2021).

Estimator 6: LEAR

$$\hat{\boldsymbol{\beta}}_{c,h} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{N} \sum_{i=1}^{N} (Y_{i,c,h} - X_{i,c,h}^{T} \boldsymbol{\beta})^2 + \lambda_{c,h} |\boldsymbol{\beta}|_{\ell_1} \right\}$$

where $|\cdot|_{\ell_1}$ denotes the ℓ_1 norm, $\lambda_{c,h}$ denotes the regularization parameter for country c and hour h, $Y_{i,c,h}$ is the day-ahead price for observation i, country c, and hour h, $X_{i,c,h}$ is the feature matrix comprising price lags (24, 48, 72, 168 hours), generation forecasts (current, 24 and 168 hour lags) from $\boldsymbol{V}_{1,i,t}$, load forecasts (current, 24 and 168 hour lags) from $\boldsymbol{V}_{2,i,t}$, wind and solar generation forecasts (current, 24 and 168 hour lags) from $\boldsymbol{V}_{3,i,t}$, coal and gas futures settlement prices from $\boldsymbol{Z}_{:,t}$, and calendar variables (day of week, hour dummies). $\boldsymbol{\beta}_{c,h}$ is the vector of coefficients to be estimated for country c and hour h.

The optimization metric is the average AIC, defined as $\text{AIC}_{c,h} = 2kc, h + \log(\text{MSE}_{c,h})$, where kc, h is the number of non-zero coefficients for country c and hour h, and $\text{MSE}_{c,h}$ is the Mean Squared Error for country c and hour h. The average AIC is computed across all countries and hours. The fitting of the 288 models per day is parallelized. For optimization trials, this estimator uses the last 20% of observations (rounded to whole days) as a validation set.

Panel LEAR

Finally, building upon the LEAR model, I implement a panel version that estimates a separate LASSO regression for each hour of the day, but pools data across countries. This approach, which I call Panel LEAR, results in 24 models per day instead of the 288 in the original LEAR. By pooling data across countries, this model leverages the panel structure of the data, potentially capturing cross-country dependencies and improving forecast accuracy. If there is a benefit in using panel data instead of treating the bidding zones as separate markets, we would *ex ante* expect this specification to perform better than the univariate LEAR.

Estimator 7: Panel LEAR

$$\hat{\boldsymbol{\beta}}_h = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{NC} \sum_{c=1}^{C} \sum_{i=1}^{N} (Y_{i,c,h} - X_{i,c,h}^T \boldsymbol{\beta})^2 + \lambda_h |\boldsymbol{\beta}|_{\ell_1} \right\}$$

where $|\cdot|_{\ell_1}$ denotes the ℓ_1 norm, λ_h denotes the regularization parameter for hour h, $Y_{i,c,h}$ is the day-ahead price for observation i, country c, and hour h, $X_{i,c,h}$ is the feature matrix comprising price lags (24, 48, 72, 168 hours), generation forecasts (current, 24 and 168 hour lags) from $\boldsymbol{V}_{1,i,t}$, load forecasts (current, 24 and 168 hour lags) from $\boldsymbol{V}_{2,i,t}$, wind and solar generation forecasts (current, 24 and 168 hour lags) from $\boldsymbol{V}3,i,t$, coal and gas futures settlement prices from $\boldsymbol{Z}_{:,t}$, calendar variables (day of week, hour dummies), and country indicators. $\boldsymbol{\beta}_h$ is the vector of coefficients to be estimated for hour h, shared across all countries.

The optimization metric used is the average AIC computed across all countries for each hour. The fitting of the 24 models per day is parallelized. For optimization trials, this estimator uses the last 20% of observations (rounded to whole days) as a validation set.

Table 2: Estimator Optimization Parameters

Estimator	Min. Wait	Min. Data	Perf. Metric	Perf. Threshold	Min. Opt. Freq.	Parameter Range
MC-NNM	7	1	MAE	+10%	30d	$\lvert(\lambda_H, \lambda_L)\rvert = 18^2$
MC-NNM_32	7	1	MAE	+10%	30d	$\lvert(\lambda_H, \lambda_L)\rvert = 18^2$
MC-NNM-TSR	7	1	MAE	+10%	30d	$\lvert(\lambda_H, \lambda_L)\rvert = 18^2$
Elastic Net	7	1	MAE	+10%	30d	$\lambda \in \log[10^{-5}, 1]$ $\alpha \in [0, 1]$, 3 trials
LASSO	7	7	AIC	+10%	30d	$\lambda \in \log[10^{-5}, 10]$ 3 trials
LEAR	14	7	Avg. AIC	+10%	30d	$\lambda_{i,c} \in \log[10^{-5}, 10]$ $\forall i \in \{1, 24\}$ $\forall c \in \{1, 12\}$ 2 trials
LEAR Panel	14	7	Avg. AIC	+10%	30d	$\lambda_i \in \log[10^{-5}, 10]$ $\forall i \in \{1, 24\}$ 2 trials

5 Results and Discussion

In all tables in this section, the rows of each table correspond to estimators, while the columns present summary statistics for each metric as well as execution times. The best (lowest) value for each summary statistic (for example mean RMSE) is highlighted in green, the second-best (second lowest) value per column in cyan. The visualizations in this section display rMAE due to its intuitive interpretation as a relative performance metric.

I begin by comparing the performance of MC-NNM and MC-NNM_32. I use the first year of data (1 initial day of training data, forecasting for days $2, \ldots, 364$). As Table 3 below shows, the 32-bit configuration has marginally higher RMSE and MAE than the 64-bit configuration, while having lower rMAE. This counterintuitive result can be attributed to several factors. First, the increased precision of 64-bit calculations may lead to the accumulation of small rounding errors over many iterations, which can compound into larger discrepancies. In contrast, 32-bit calculations may benefit from more frequent rounding to zero for very small values, potentially preventing error propagation. Second, `JAX`, the underlying library used in my implementation, is optimized for 32-bit operations by default, which may contribute to more stable computations in the 32-bit version. Additionally, when dealing with very close numbers, as often occurs in matrix completion tasks, 32-bit precision may sometimes prevent numerical instabilities by rounding small differences to zero more readily than 64-bit precision.

Table 3: Comparison of 32-bit and 64-bit MC-NNM Estimators On The First Year of Data

Estimator	RMSE				MAE				rMAE				Mean Execution Time (s)		
	min	mean	median	max	min	mean	median	max	min	mean	median	max	`fit()`	`predict()`	`optimize()`
MC-NNM (32)	14.245	33.065	30.275	105.003	8.242	18.739	17.650	45.308	0.447	1.750	1.545	4.957	0	5.031	2.245
MC-NNM (64)	13.322	33.033	30.578	105.003	7.547	18.721	17.585	46.387	0.435	1.754	1.565	4.957	0	5.041	15.646

Notably, the 32-bit version offers a significant computational advantage, with average prediction times being lower by 0.01 seconds per iteration and optimization times reduced on average by approximately 86% compared to the 64-bit version. Over the full dataset, which has $2,006$ forecast days, such small differences accumulate to an enormous difference in total execution time. Given the minimal difference in forecasting accuracy and the substantial reduction in computational time, I choose to use the 32-bit version (MC-NNM_32) for subsequent analyses and comparisons with other estimators. This choice allows me to conduct more extensive experiments within my computational constraints without trading off substantially in terms of accuracy. In the remainder of this section, all references to MC-NNM and MC-NNM-TSR therefore refer to results obtained using 32-bit floating point precision.

Results Using a 56-day Sliding Window

Table 4 and Figure 8 below present the performance comparison of the estimators using a 56-day sliding window. The Elastic Net estimator demonstrates the best overall performance across all metrics, achieving the lowest minimum, mean, and median RMSE, MAE, and rMAE. This suggests that for this shorter window size, the Elastic Net's ability to balance between the ℓ_1 and ℓ_2 penalties provides an effective approach to capturing the underlying patterns in the data.

Table 4: Comparison of Estimators With 56-Day Sliding Window

Estimator	RMSE				MAE				rMAE				Mean Execution Time (s)		
	min	mean	median	max	min	mean	median	max	min	mean	median	max	`fit()`	`predict()`	`optimize()`
MC-NNM	8.411	49.331	34.793	320.343	5.752	38.460	23.114	301.834	0.278	1.887	1.534	10.365	0	5.796	2.529
MC-NNM-TSR	8.032	49.140	34.614	319.978	5.328	38.273	23.110	301.834	0.278	1.876	1.560	9.807	0	10.442	5.734
ElasticNet	3.764	25.972	17.371	711.283	2.785	17.768	11.687	273.206	0.138	0.930	0.771	31.205	0.871	0.000	1.092
LASSO	15.714	81.371	69.208	446.215	12.245	57.267	44.296	300.592	0.375	3.507	2.797	19.674	5.079	0.000	1.667
LEAR	6.186	393.940	47.033	103 176.328	3.776	80.316	23.454	13 831.132	0.264	5.076	1.255	1151.798	1.202	0.025	0.065
LEAR-Panel	6.434	503.392	46.835	180 758.706	3.791	96.716	23.568	25 112.787	0.271	6.120	1.252	2031.926	1.185	0.025	0.064

MC-NNM-TSR and MC-NNM show very similar performance, ranking second and third in terms of mean and median RMSE and MAE. Interestingly, the are best and second best, respectively, in terms of maximum RMSE and rMAE. The incorporation of temporal structure in MC-NNM-TSR offers a consistent, albeit marginal, improvement over the standard MC-NNM for this window size. In terms of execution time, the Elastic Net is the fastest for both fitting and prediction. It is worth noting that the fit time for both MC-NNM estimators is reported as 0 because fitting and prediction are inseparable steps in these methods, with the `fit()` method merely passing control to the prediction stage. The LASSO estimator performs notably worse than the Elastic Net and MC-NNM approaches. This may indicate that the strict sparsity induced by LASSO is too restrictive for capturing the complex dependencies in electricity prices over this time horizon.

Figure 8: Daily rMAE of Estimators With 56-Day Sliding Window

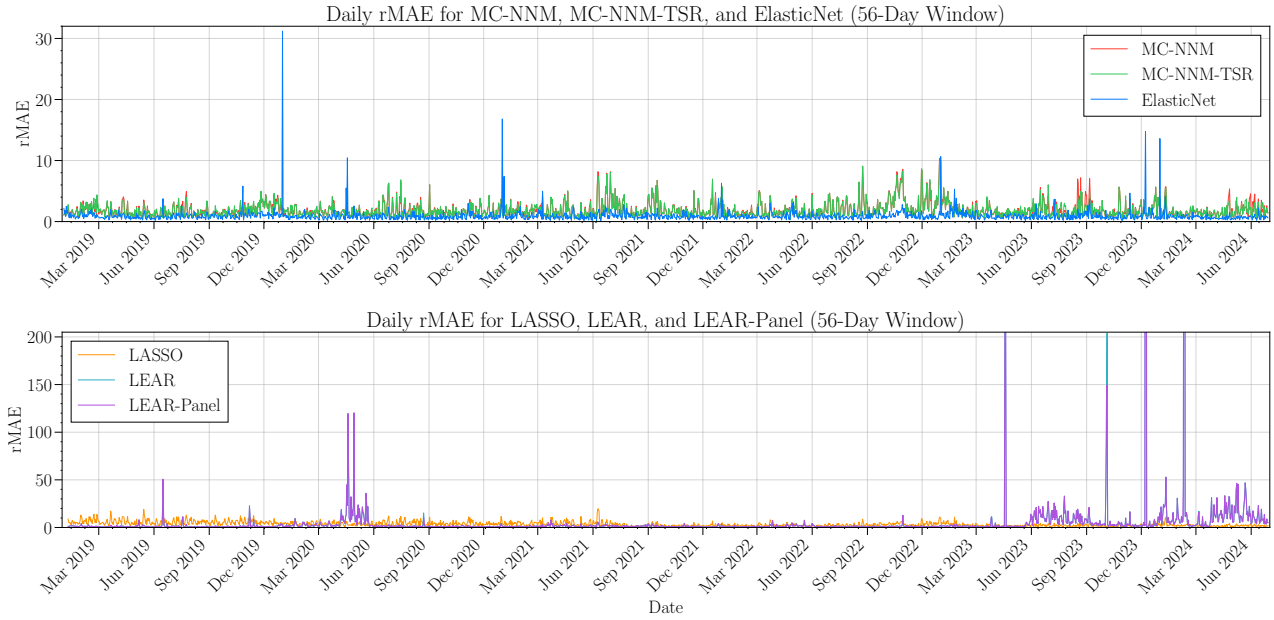


Figure 8 is split into two panels due to the significant difference in rMAE scales between the three top-performing estimators and the three LASSO-based methods. The y-axes have different scales, which is also reflected in the maximum rMAE values reported in Table 4. Both LEAR and LEAR Panel exhibit extremely high maximum errors, suggesting they may be prone to occasional severe mispredictions. However, their median performance is competitive, particularly in terms of rMAE.

An interesting observation is the superior stability of both MC-NNM estimators compared to the other methods. They show no visible dramatic spikes in rMAE and maintain more consistent performance across the entire 2,006 forecast days. This stability comes at the expense of having, on average, higher rMAE than the Elastic Net. The enhanced stability of the MC-NNM estimators might be due to their use of fixed effects and different regularization approach compared to the linear models. The estimated low-rank matrix may capture idiosyncratic shocks more effectively, making the results less susceptible to drastic spikes. Comparing LEAR and LEAR Panel, we can observe that their performances are quite similar, with LEAR Panel showing slightly higher mean and maximum rMAE. This is somewhat surprising, as one might expect the panel version to perform better by leveraging cross-country information. The similar performance suggests that the benefits of pooling data across countries may be offset by the loss of country-specific modeling in this context. The LEAR estimator(s), despite their simple structure, have relatively long fitting times due to the need to estimate separate models for each hour and country. Overall, these results suggest that for a 56-day sliding window, the Elastic Net provides the best balance of accuracy and computational efficiency. However, the MC estimators offer competitive performance with

potentially greater robustness to volatility, making them attractive alternatives, especially in unstable market conditions.

Results Using an 84-day Sliding Window

Table 5 and Figure 9 below present the performance comparison of the estimators using an 84-day sliding window. Many of the trends observed in the 56-day window persist, but there are some notable differences.

Table 5: Comparison of Estimators With 84-Day Sliding Window

Estimator	RMSE				MAE				rMAE				Mean Execution Time (s)		
	min	mean	median	max	min	mean	median	max	min	mean	median	max	`fit()`	`predict()`	`optimize()`
MC-NNM	8.064	51.841	36.789	383.996	4.897	40.630	24.910	369.120	0.286	2.001	1.631	13.186	0	16.858	12.856
MC-NNM-TSR	7.908	51.861	36.486	375.507	4.950	40.662	24.777	360.373	0.296	2.004	1.655	13.117	0	27.125	19.785
ElasticNet	3.523	26.208	18.110	661.197	2.406	18.173	11.983	370.660	0.127	0.956	0.788	28.481	1.111	0.000	1.409
LASSO	17.368	84.523	74.656	361.459	12.490	59.708	46.076	320.579	0.368	3.689	2.927	20.924	7.361	0.000	2.521
LEAR	6.164	297.727	45.463	71 530.336	3.929	64.211	24.031	9458.683	0.266	4.035	1.201	793.786	1.847	0.027	0.218
LEAR-Panel	6.049	294.556	45.389	62 382.795	3.855	61.266	23.942	5960.928	0.269	3.685	1.203	468.175	1.819	0.025	0.216

The Elastic Net continues to demonstrate the best overall performance, maintaining the lowest mean and median RMSE, MAE, and rMAE. However, the gap between Elastic Net and the MC-NNM estimators has narrowed slightly, particularly in terms of rMAE. MC-NNM and MC-NNM-TSR show improved performance relative to the other estimators with this longer window. The incorporation of temporal structure in MC-NNM-TSR continues to offer improvements over MC-NNM, but the difference remains small.

Figure 9: Daily rMAE of Estimators With 84-Day Window

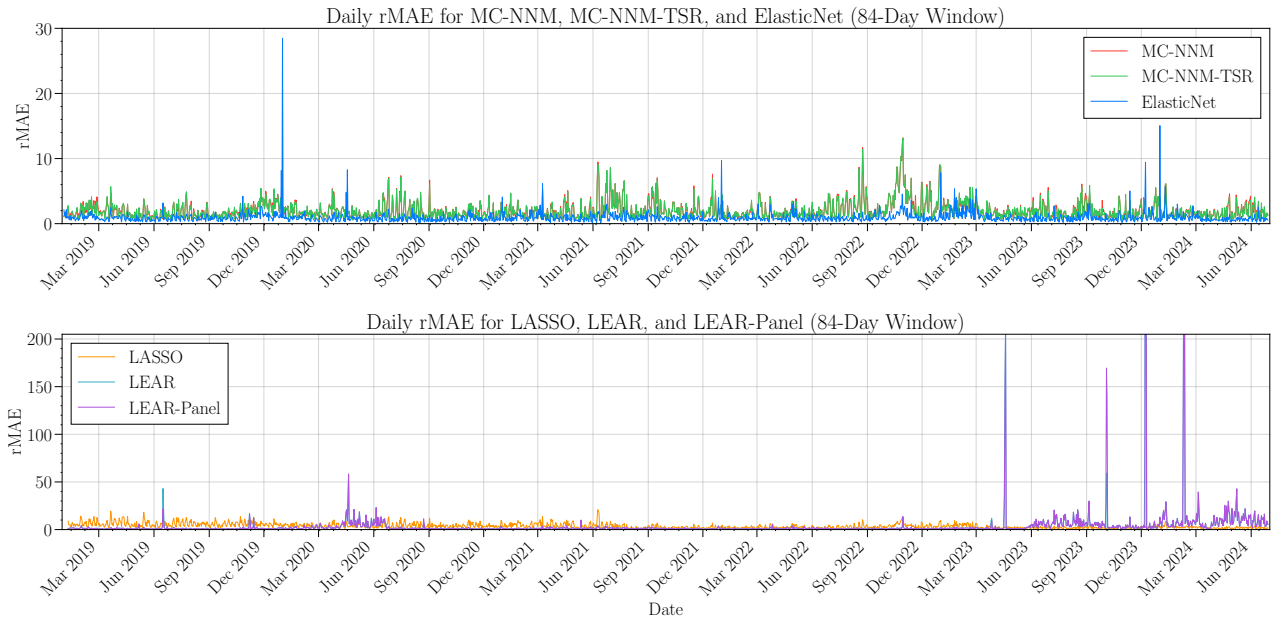


Figure 9 reveals that the stability advantage of the MC-NNM estimators is more pronounced with the 84-day window. The Elastic Net shows increased volatility in its rMAE, while both MC-NNM estimators maintain a more consistent performance. The LASSO estimator's performance has improved relative to the 56-day window, but it still lags behind the Elastic Net and MC-NNM approaches. This suggests that the longer window allows LASSO to capture more relevant features, but its strict sparsity remains a limitation. It does, however, show a dramatic decrease in maximum RMSE and MAE, making it the best estimator in terms of the maximum of both metrics. Interestingly, the performance gap between LEAR and LEAR Panel has widened slightly with the 84-day window. LEAR now shows lower mean and maximum rMAE compared to LEAR Panel, indicating that the country-specific modeling becomes more advantageous with a longer historical window. LEAR Panel is, however, the second-best performing estimator in terms of minimum RMSE and MAE. In terms of execution time, the relative performance of the estimators remains similar to the 56-day window, with Elastic Net remaining the fastest. However, the absolute fitting, prediction, and optimization times have increased for all estimators due to the larger data window. Overall, while the Elastic Net still provides the best balance of accuracy and computational efficiency, the MC-NNM estimators' improved relative performance and enhanced stability with the longer window make them increasingly attractive alternatives, especially for forecasting during periods of high volatility.

Results Using a 112-day Sliding Window

Table 6 and Figure 10 below present the performance comparison of the estimators using a 112-day sliding window. The trends observed in the 56- and 84-day windows generally continue, with some further notable shifts in relative performance.

Table 6: Comparison of Estimators With 112-Day Sliding Window

Estimator	RMSE				MAE				rMAE				Mean Execution Time (s)		
	min	mean	median	max	min	mean	median	max	min	mean	median	max	`fit()`	`predict()`	`optimize()`
MC-NNM	7.825	53.550	37.992	401.004	5.678	42.105	25.980	386.470	0.300	2.100	1.732	12.992	0	50.471	31.906
MC-NNM-TSR	7.277	53.391	37.727	401.004	5.131	41.940	25.874	386.470	0.295	2.094	1.729	13.639	0	51.182	33.835
ElasticNet	3.566	26.458	19.024	368.116	2.252	18.496	12.468	294.846	0.117	0.978	0.802	14.195	1.360	0.000	1.742
LASSO	19.991	86.776	79.405	412.320	16.242	61.541	48.840	365.495	0.356	3.780	3.012	22.281	9.251	0.000	3.083
LEAR	6.809	174.872	42.637	40 532.072	4.300	48.550	23.487	5246.787	0.245	2.823	1.179	159.449	2.420	0.027	0.276
LEAR-Panel	6.870	186.739	42.721	33 253.808	4.308	48.753	23.480	3293.416	0.255	2.855	1.186	191.374	2.439	0.027	0.278

The Elastic Net maintains its position as the best overall performer, but the gap between it and the MC-NNM estimators has narrowed even further. In particular, MC-NNM-TSR now performs second-best across RMSE, MAE, and rMAE, suggesting that the incorporation of temporal structure becomes more beneficial with a longer window. Interestingly, MC-NNM shows the lowest maximum rMAE followed by MC-NNM-TSR. This suggests that MC-NNM-TSR is marginally less robust to spikes in the forecast error compared to its time-agnostic sibling, which as displayed in Figure 10 is attributable to 4 dates on which MC-NNM-TSR has a larger

rMAE than MC-NNM. Both MC-NNM and MC-NNM-TSR continue to demonstrate improved performance relative to the other estimators.

Figure 10: Daily rMAE of Estimators With 112-day Window

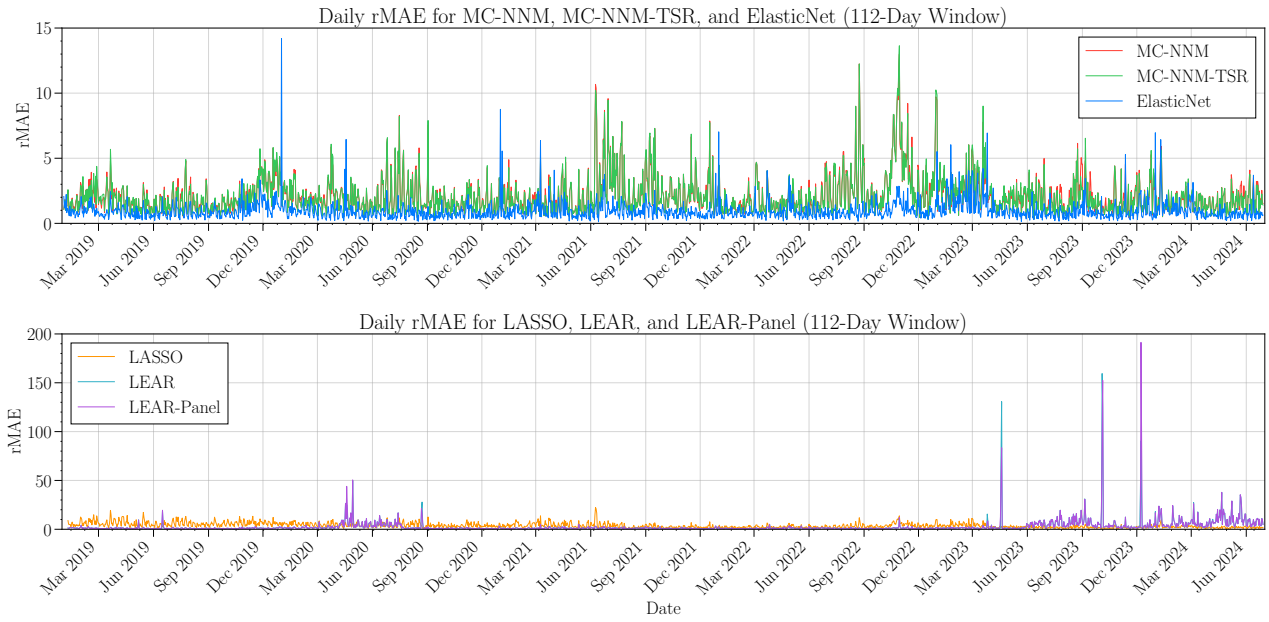


Figure 10 further emphasizes the stability advantage of the MC-NNM estimators. With the 112-day window, both MC-NNM variants show remarkably consistent performance, even during periods of extreme market volatility. The Elastic Net, while still performing well on average, exhibits more pronounced spikes in rMAE. The LASSO estimator continues to improve with the longer window, maintaining its advantage in terms of maximum RMSE and MAE. This suggests that for longer time horizons the ℓ_1 sparsity constraint becomes less restrictive and more effective at capturing stable, long-term relationships in the data. The performance gap between LEAR and LEAR Panel has continued to widen. LEAR now significantly outperforms LEAR Panel across all metrics, further reinforcing the importance of country-specific modeling when longer historical data is available. Execution times have increased proportionally for all estimators, with the MC-NNM variants showing the most significant increase in prediction time. This reflects the growing computational complexity of these methods as the data window expands and the matrices they have to decompose double in size. In conclusion, while the Elastic Net remains the most balanced performer in terms of accuracy and speed, the 112-day window results demonstrate the increasing competitiveness of the MC-NNM estimators, particularly MC-NNM-TSR. Their superior stability and improving accuracy with longer time horizons suggest they may be preferable for long-term forecasting or in markets characterized by frequent volatility spikes.

Limitations

While this dissertation provides novel insights into the performance of various forecasting methods for electricity prices, it is important to explicitly acknowledge several of its limitations:

1. Computational constraints: Due to the high computational demands of some estimators, particularly the MC-NNM variants, I had to limit the number of iterations and cross-validation folds. This may have affected the robustness of the results, especially for hyperparameter tuning. It is plausible that the gap between Elastic Net and MC-NNM-TSR would further shrink with longer sliding windows, and that the LEAR estimators might outperform the others if given multiple years of training data.

2. Limited market coverage: My comparison focuses on the CORE CCR, which, while significant, does not represent the entire European electricity market. The performance of these estimators may vary in other regions with different market structures or regulations, or other applications with better-behaved data (by which I mean no zero or negative values in the forecasting target, fewer spikes, no sudden breaks, and known stationarity).

3. Limited exploration of feature engineering: I use a straightforward approach to feature selection and did not extensively explore potential transformations or interactions that might improve forecasting performance.

4. Absence of formal statistical tests: While my implementation compares error metrics, it lacks formal statistical tests for the significance of differences between estimators' performances. I believe that the tests used in the EPF literature are difficult to construct and interpret in panel settings, but this would be a natural future extension of my comparison.

5. Simplified treatment of temporal dependencies: My approach to incorporating temporal structure into MC-NNM-TSR is relatively basic. More sophisticated methods for modeling time series dependencies might yield better results. One of the main aims of this dissertation was to investigate whether this extension could improve forecasting performance. My results suggest that it does, but larger improvements might be possible with more sophisticated covariance matrix specifications.

6. Focus on point forecasts: This dissertation exclusively evaluates point forecasts without considering prediction intervals or probabilistic forecasts, which are increasingly important in quantifying forecast uncertainty.

Addressing these limitations in future research could provide a more comprehensive understanding of the relative strengths and weaknesses of these forecasting methods in the context of electricity price prediction.

6 Conclusion

This dissertation makes several contributions to the literature in statistics and forecasting:

1. It provides the first implementation and evaluation of the autocorrelated errors extension to MC-NNM described by Athey et al. (2021). While this extension may violate key assumptions for causal inference, this work demonstrates its potential benefits in panel data forecasting.

2. It introduces a new, open-source implementation of the MC-NNM estimator in Python/`JAX`. This implementation is fully documented, designed for extensibility, and uniquely allows for time-and unit-specific covariates.

3. It offers the first assessment of MC-NNM's forecasting performance, both with and without autocorrelation extensions. This novel application addresses some limitations of MC-NNM for causal inference and explores its potential in a domain where its theoretical properties suggest it could excel.

4. It presents an open-source forecast engine with a rich, public dataset of European electricity prices, providing a standardized framework for comparing various forecasting methods in a real-world panel data setting.

It is additionally the first investigation that I am aware of into explicitly treating multi-country Electricity Price Forecasting as a panel data problem that I could find, and the first to apply Matrix Completion methods to EPF. This novel approach opens up new avenues for leveraging cross-sectional dependencies in electricity markets, potentially improving forecast accuracy and our understanding of market dynamics.

My results demonstrate that while MC-NNM does not consistently outperform all other methods, it offers competitive performance and superior stability, particularly with longer time windows. This suggests it may be preferable for long-term forecasting or in applications characterized by frequent volatility spikes. Notably, the electricity price data I use is characterized by high volatility, occasional zero and negative prices, and structural breaks, making it particularly challenging to forecast. The competitive performance of MC-NNM on this data suggests that it can offer good performance in general panel data forecasting applications. Furthermore, the empirical results validate that the theoretical error bound given by Athey et al. (2021), which becomes tighter with more data, is directionally applicable for forecasting tasks.

These contributions collectively advance our understanding of Matrix Completion methods in statistics, particularly in the realm of panel data forecasting. By providing robust, open-source tools and a comprehensive evaluation framework, this dissertation facilitates further research and practical applications of these methods in complex, real-world settings.

7 References

Aldous, D. J. (1981). Representations for partially exchangeable arrays of random variables. *Journal of Multivariate Analysis*, *11*(4), 581–598. https://doi.org/10.1016/0047-259X(81)90099-3 (cit. on p. 14).

Athey, S., Bayati, M., Doudchenko, N., Imbens, G., & Khosravi, K. (2021). Matrix Completion Methods for Causal Panel Data Models [Appendix at https://arxiv.org/pdf/1710.10251.pdf]. *Journal of the American Statistical Association*, *116*(536), 1716–1730. https://doi.org/10.1080/01621459.2021.1891924 (cit. on pp. 0, 5, 8–17, 28, 30, 41).

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for Hyper-Parameter Optimization. *Advances in Neural Information Processing Systems*, *24*. Retrieved August 17, 2024, from https://proceedings.neurips.cc/paper_files/paper/2011/hash/86e8f7ab32cfd12577bc2619bc635690-Abstract.html (cit. on p. 26).

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). JAX: Composable transformations of Python+NumPy programs. http://github.com/google/jax (cit. on p. 17).

Candes, E. J., & Plan, Y. (2010). Matrix Completion With Noise [Conference Name: Proceedings of the IEEE]. *Proceedings of the IEEE*, *98*(6), 925–936. https://doi.org/10.1109/JPROC.2009.2035722 (cit. on p. 6).

Candes, E. J., & Tao, T. (2010). The Power of Convex Relaxation: Near-Optimal Matrix Completion [Conference Name: IEEE Transactions on Information Theory]. *IEEE Transactions on Information Theory*, *56*(5), 2053–2080. https://doi.org/10.1109/TIT.2010.2044061 (cit. on pp. 6, 7).

Candès, E. J., & Recht, B. (2009). Exact Matrix Completion via Convex Optimization. *Foundations of Computational Mathematics*, *9*(6), 717–772. https://doi.org/10.1007/s10208-009-9045-5 (cit. on p. 6).

Card, D., & Krueger, A. B. (2000). Minimum Wages and Employment: A Case Study of the Fast-Food Industry in New Jersey and Pennsylvania: Reply. *American Economic Review*, *90*(5), 1397–1420. https://doi.org/10.1257/aer.90.5.1397 (cit. on p. 9).

Chen, Y., Fan, J., Ma, C., & Yan, Y. (2019). Inference and Uncertainty Quantification for Noisy Matrix Completion [arXiv:1906.04159 [cs, eess, math, stat]]. *Proceedings of the National*

Academy of Sciences, *116*(46), 22931–22937. https://doi.org/10.1073/pnas.1910053116 (cit. on p. 14).

Gelman, A. (2005). Analysis of Variance: Why It Is More Important than Ever [Publisher: Institute of Mathematical Statistics]. *The Annals of Statistics*, *33*(1), 1–31. Retrieved June 4, 2023, from https://www.jstor.org/stable/3448650 (cit. on p. 11).

Giacomini, R., & White, H. (2006). Tests of Conditional Predictive Ability [Publisher: [Wiley, Econometric Society]]. *Econometrica*, *74*(6), 1545–1578. Retrieved September 10, 2024, from https://www.jstor.org/stable/4123083 (cit. on p. 27).

Golub, G. H., & Van Loan, C. F. (2013). *Matrix computations* (4. ed) [OCLC: 835290390]. Johns Hopkins Univ Press. Retrieved June 2, 2023, from http://bvbr.bib-bvb.de:8991/F?func=service&doc_library=BVB01&local_base=BVB01&doc_number=025701078&line_number=0001&func_code=DB_RECORDS&service_type=MEDIA (cit. on pp. 7, 8).

Gui, Y., Barber, R. F., & Ma, C. (2023, October). Conformalized matrix completion [arXiv:2305.10637 [stat]]. https://doi.org/10.48550/arXiv.2305.10637 (cit. on p. 14).

Hastie, T. (2017). *The elements of statistical learning: Data mining, inference, and prediction* (Second edition.). Springer. (Cit. on p. 31).

Imbens, G. (2004). Nonparametric Estimation of Average Treatment Effects under Exogeneity: A Review. *Review of Economics and Statistics*. https://scholar.harvard.edu/imbens/publications/nonparametric-estimation-average-treatment-effects-under-exogeneity-review (cit. on p. 9).

Imbens, G., & Rubin, D. B. (2015). *Causal inference for statistics, social, and biomedical sciences: An introduction*. Cambridge University Press. Retrieved April 25, 2023, from http://catdir.loc.gov/catdir/enhancements/fy1513/2014020988-t.html (cit. on p. 10).

Lago, J., De Ridder, F., & De Schutter, B. (2018). Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms. *Applied Energy*, *221*, 386–405. https://doi.org/10.1016/j.apenergy.2018.02.069 (cit. on p. 19).

Lago, J., De Ridder, F., Vrancx, P., & De Schutter, B. (2018). Forecasting day-ahead electricity prices in Europe: The importance of considering market integration. *Applied Energy*, *211*, 890–903. https://doi.org/10.1016/j.apenergy.2017.11.098 (cit. on p. 19).

Lago, J., Marcjasz, G., De Schutter, B., & Weron, R. (2021). Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark.

Applied Energy, *293*, 116983. https://doi.org/10.1016/j.apenergy.2021.116983 (cit. on pp. 25–28, 31).

Liu, L., Wang, Y., & Xu, Y. (2024). A Practical Guide to Counterfactual Estimators for Causal Inference with Time-Series Cross-Sectional Data. *American Journal of Political Science*, *68*(1), 160–176. https://doi.org/10.1111/ajps.12723 (cit. on p. 17).

Mazumder, R., Hastie, T., & Tibshirani, R. (2010). Spectral Regularization Algorithms for Learning Large Incomplete Matrices. *Journal of Machine Learning Research*, *11*(80), 2287–2322. Retrieved April 24, 2023, from http://jmlr.org/papers/v11/mazumder10a.html (cit. on pp. 7, 8, 17, 46, 47, 49).

Olivares, K. G., Challu, C., Marcjasz, G., Weron, R., & Dubrawski, A. (2023). Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with NBEATSx. *International Journal of Forecasting*, *39*(2), 884–900. https://doi.org/10.1016/j.ijforecast.2022.03.001 (cit. on p. 25).

Pesaran, M. H. (2012). On the interpretation of panel unit root tests. *Economics Letters*, *116*(3), 545–546. https://doi.org/10.1016/j.econlet.2012.04.049 (cit. on p. 25).

Pesaran, M. H. (2015). *Time Series and Panel Data Econometrics* (1st ed.). University Press. https://doi.org/10.1093/acprof:oso/9780198736912.001.0001 (cit. on pp. 15, 16).

Rubin, D. B. (1974). Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, *66*(5), 688–701. https://doi.org/10.1037/h0037350 (cit. on p. 9).

Schnabel, T. (2024a). Lightweight-mcnnm [Github Repository. Version used in this dissertation: https://github.com/tobias-schnabel/mcnnm/releases/tag/v1.0.2]. https://github.com/tobias-schnabel/mcnnm (cit. on p. 17).

Schnabel, T. (2024b). Lightweight-mcnnm Documentation. https://mcnnm.readthedocs.io/en/latest/ (cit. on p. 17).

Strang, G. (2006, January). *Linear Algebra and Its Applications, 4th Edition* (4th ed.). Cengage Learning. (Cit. on pp. 7, 8).

Tschora, L., Pierre, E., Plantevit, M., & Robardet, C. (2022). Electricity price forecasting on the day-ahead market using machine learning. *Applied Energy*, *313*, 118752. https://doi.org/10.1016/j.apenergy.2022.118752 (cit. on p. 19).

Uniejewski, B., Nowotarski, J., & Weron, R. (2016). Automated Variable Selection and Shrinkage for Day-Ahead Electricity Price Forecasting [Number: 8 Publisher: Multidisciplinary Digital Publishing Institute]. *Energies*, *9*(8), 621. https://doi.org/10.3390/en9080621 (cit. on p. 31).

van der Leeuw, J. (1994). The covariance matrix of ARMA errors in closed form. *Journal of Econometrics*, *63*(2), 397–405. https://doi.org/10.1016/0304-4076(94)90032-9 (cit. on p. 16).

Wainwright, M. J. (2019). *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*. Cambridge University Press. https://doi.org/10.1017/9781108627771 (cit. on p. 13).

A Appendix

A.1 Details of Python Package `lightweight-mcnnm`

This appendix provides additional details about the implementation of the Matrix Completion with Nuclear Norm Minimization (MC-NNM) and MC-NNM with Time Series Regularization (MC-NNM-TSR) models in the `lightweight-mcnnm` package.

Soft-Thresholding Operator and Singular Value Thresholding

The implementation uses the soft-thresholding operator introduced by Mazumder, Hastie, and Tibshirani (2010). This operator is applied in the Singular Value Thresholding (SVT) step of the algorithm, which is crucial for the nuclear norm regularization of the low-rank matrix L. The soft-thresholding operator is defined as:

$$S_\lambda(x) = \text{sign}(x) \max(|x| - \lambda, 0),$$

where λ is the threshold parameter. In the context of SVT, this operator is applied to the singular values of the matrix. The SVT procedure is as follows:

1. Compute the Singular Value Decomposition (SVD) of the input matrix: $M = U\Sigma V^T$
2. Apply the soft-thresholding operator to the singular values: $\Sigma' = S_\lambda(\Sigma)$
3. Reconstruct the thresholded matrix: $M' = U\Sigma' V^T$

This process effectively promotes low-rank solutions by shrinking smaller singular values towards zero.

Covariance Matrix Ω

This part of MC-NNM-TSR is inverted automatically within the package. If the user does not supply a covariance matrix, the implementation uses a default of $\boldsymbol{\Omega}^{-1} = \boldsymbol{I}_{T\times T}$, which corresponds to the regular MC-NNM estimator. If the user does supply such a matrix, it is inverted and used as described in MC-NNM-TSR.

Augmented Covariate Matrices

The current implementation always augments the covariate matrices X and Z to $\tilde{\boldsymbol{X}}$ and $\tilde{\boldsymbol{Z}}$ as described earlier. I plan on making this optional in a future version. The additional flexibility offered by this choice is likely to be beneficial in completing a complex matrix as accurately as possible, while complicating an interpretation of the estimated coefficient matrix $\tilde{\boldsymbol{H}}$, which is irrelevant in the forecasting application at the heart of this dissertation.

Validation Methods

The implementation supports two validation methods for selecting optimal regularization parameters. Both methods validate over a grid of values for λ_L and λ_H values that is initialized by the method `initialize_coefficients` described here and in Mazumder, Hastie, and Tibshirani (2010, Section 7)

Cross-Validation This is the default method, implemented in the `cross_validate` function. It performs K-fold cross-validation to select optimal regularization parameters (λ_L and λ_H). The process works as follows:

1. The data is randomly split into K folds by using `jax.random.bernoulli` to generate a random boolean mask which is multiplied with the mask of observed entries to split the data into a training and validation set for each fold. Folds in which no elements in the training set are treated are ignore (this would correspond to a fully observed matrix, which cannot be completed as it is already complete)
2. For each pair of lambda values in the provided grid:
   (a) The model is trained and evaluated on each fold's training set
   (b) The loss is computed for each fold's validation set
   (c) The average loss across all valid folds is calculated
3. The lambda pair that results in the lowest average validation loss across folds is selected.

Key parameters include:

- `K`: The number of folds (default is 5)
- `num_lam`: The number of lambda values to consider in the grid search

Holdout Validation This method, implemented in the `holdout_validate` function, uses a time-based holdout strategy. It creates multiple train-test splits based on specified time windows and evaluates the model's performance over these splits. The process includes:

1. Creating K holdout masks based on the specified time windows
2. For each lambda pair in the grid:
   (a) The model is trained on the training set for each holdout fold
   (b) The holdout RMSE is computed for the validation set of each fold
3. The average holdout RMSE across all folds is calculated for each lambda pair

4. The lambda pair with the lowest average RMSE is selected

Key parameters include:

- `initial_window` (int): Sets the number of initial time periods to use for the first training set.
  - Example: If `initial_window=50` and you have 100 time periods, the first training set will use periods 1-50.
  - Utility: Controls how much historical data is considered relevant for initial prediction.
- `step_size` (int): Determines how many time periods to move forward for each subsequent split.
  - Example: If `step_size=10`, after the initial training, the next split will start at period 10, then 20, and so on.
  - Utility: Smaller step sizes provide more validation points but increase computation time. Larger step sizes are faster but may miss important temporal patterns.
- `horizon` (int): Sets the number of future time periods to predict (forecast horizon).
  - Example: If `horizon=5`, each validation step will predict 5 time periods ahead.
  - Utility: Allows tailoring the validation to specific forecasting needs. A longer horizon tests the model's long-term predictive power, while a shorter horizon focuses on immediate future predictions.
- `K` (int): Sets the number of folds (splits) to use in the time-based validation.
  - Example: If `K=5`, the function will create 5 different train-test splits to evaluate the model.
  - Utility: More folds provide a more robust evaluation but increase computation time. Fewer folds are faster but may be less reliable.
- `max_window_size` (Optional[int]): Sets the maximum size of the window to consider. If None, all data is used.
  - Example: If `max_window_size=80` and you have 100 time periods, only the most recent 80 periods will be used for any training set.
  - Utility: Effectively limits how far back in time the model will look for training data, useful when very old data might not be relevant to current predictions.

After either validation process is complete, the model is re-fit to the whole dataset using

the shortest path between the two highest values in the grid of λ_L and λ_H values and the pair of values selected by the regularization. This warm-starting helps improve numerical stability and speed and is described in detail in Mazumder, Hastie, and Tibshirani (2010, Algorithm 1)

Cyclic Coordinate Descent Algorithm

The core estimation algorithm uses cyclic coordinate descent to iteratively update the model parameters. The main steps in each iteration are:

1. Update the low-rank matrix $\boldsymbol{L}$ using Singular Value Thresholding (SVT)
2. Update the covariate coefficients matrix $\tilde{\boldsymbol{H}}$ using element-wise soft-thresholding
3. Update unit fixed effects $\boldsymbol{\Gamma}$ (if used)
4. Update time fixed effects $\boldsymbol{\Delta}$ (if used)
5. Update unit-time-specific covariate coefficients $\boldsymbol{\beta}$ (if used)

The algorithm iterates through these steps until convergence or the maximum number of iterations (default $1e4$) is reached. Convergence is determined by comparing the relative change in the objective function value between iterations to a specfied tolerance (default is $1e-4$).

Computational Considerations

The implementation is designed to handle large matrices efficiently, leveraging JAX for potential GPU acceleration. However, the estimation process can be computationally intensive, especially for large datasets or when using cross-validation with many folds. While it is in principle possible to optimize the package's code to run on large, distributed clusters, such optimizations require expertise that I do not possess. Additionally, the main bottleneck of this algorithm is the computation of Singular Value Decompositions, which as best I can tell are already fairly optimized in `JAX`. Users can adjust various parameters to balance computational resources and estimation precision, including:

- The number of folds in cross-validation
- The number of lambda values to consider in the grid search
- The maximum number of iterations for the optimization algorithm
- The convergence tolerance

The package also provides options to toggle JIT compilation and precision settings, allowing users to fine-tune performance based on their specific use case and available computational resources.

A.2 Data Definitions

The day-ahead generation and load forecasts, as well as wind and solar generation forecasts, are provided for each market time unit of the following day, aligning with the day-ahead nature of the electricity price predictions. The data source for each bidding zone's forecasts are the Transmission System Operators that control that bidding zone. The data source for the day-ahead prices are the European power exchanges. All electricity data is queried from the public transparency platform of the European Network of Transmission System Operators for Electricity (ENTSOE) via a publicly available API. The futures settlement prices are queried from the freely available yahoo finance API.

Table 7: Data Definitions

Variable	Code	Publication	Relative to Gate Closure[a]
Day-ahead Prices	12.1.D	D-1	+1h
Load Forecast	6.1.B	D-1	–2h
Generation Forecast	14.1.C	D-1	+6h
Wind & Solar Generation Forecast	14.1.D	D-1	+6h
Installed Capacity	14.1.A	annual	N/A
Coal Futures Settlement Price	(API2) CIF ARA	D-1	–12h
Gas Futures Settlement Price	TTF	D-1	–12h

[a] Gate Closure is 12 noon on D-1, the day before physical delivery of electricity.
Futures data extracted using Yahoo Finance API, electricity data extracted using ENTSO-E API.

A.3 Data Imputation Strategy

My imputation strategy leverages the cross-sectional relationships between countries to estimate missing values in covariates / predictors as follows:

Methodology

1. For each day with missing data:
   (a) Identify missing values for each country.
   (b) For each missing value:
      - Use data from the same hour of the previous day to compute weights for other countries.
      - Calculate a weighted average of other countries' values for the missing time point.
      - Impute the missing value with this weighted average.
2. Weights are determined by the relative values of other countries in the previous day's corresponding hour.
3. Countries with missing data in the previous day are excluded from the calculation to ensure reliability.

Advantages

- Adapts to changing relationships between countries over time
- Preserves the relative influence of different countries
- Handles missing data in a way that's consistent with the panel structure of the dataset
- Model-free: this strategy does not impose fixed relationships between countries or assume any specific time series model

Limitations

- Assumes relative stability in cross-country relationships from one day to the next
- May not capture longer-term trends or seasonal patterns
- Effectiveness may be reduced if there are extended periods of missing data across multiple countries