ROLCH: Regularized Online Learning for Conditional Heteroskedasticity

Simon Hirsch[a,b,*], Jonathan Berrisch[b], Florian Ziel[b]

[a] *Statkraft Trading GmbH*
Germany
[b] *House of Energy Markets and Finance*
University of Duisburg-Essen
Germany

Preliminary Version, June 26, 2024

Abstract

Large-scale streaming data are common in modern machine learning applications and have led to the development of online learning algorithms. Many fields, such as supply chain management, weather and meteorology, energy markets, and finance, have pivoted towards using probabilistic forecasts, which yields the need not only for accurate learning of the expected value but also for learning the conditional heteroskedasticity. Against this backdrop, we present a methodology for online estimation of regularized linear distributional models for conditional heteroskedasticity. The proposed algorithm is based on a combination of recent developments for the online estimation of LASSO models and the well-known GAMLSS framework. We provide a case study on day-ahead electricity price forecasting, in which we show the competitive performance of the adaptive estimation combined with strongly reduced computational effort. Our algorithms are implemented in a computationally efficient `python` package.

Keywords: GAMLSS, probabilistic forecasting, adaptive learning, distributional regression, volatility modelling, volatility forecasting, weighted regression, iteratively reweighted least squares, coordinate descent, electricity price forecasting, EPF

JEL: C15; C18; C22; C32; C53; C55; C57; G17; Q47

*Corresponding author

Email addresses: `simon.hirsch@statkraft.com` (Simon Hirsch), `simon.hirsch@stud.uni-due.de` (Simon Hirsch), `jonathan.berrisch@uni-due.de` (Jonathan Berrisch), `florian.ziel@uni-due.de` (Florian Ziel)

1. Introduction

Motivation. Large-scale streaming data are common in modern applications of machine learning and have led to the development of online learning algorithms (Cesa-Bianchi & Orabona, 2021). For processes driven by a high-dimensional co-variate space, regularized algorithms have been presented by, e.g., Monti et al. (2018); Angelosante et al. (2010); Yang et al. (2023, 2010). In many settings, online statistical algorithms are used to issue forecasts. The advent of probabilistic forecasting in many fields, such as supply chain management, weather and meteorology, energy markets, and finance, yields the need not only for accurate learning of the expected value but also for learning the conditional heteroskedasticity (for a review on distributional regression see, e.g. Klein (2024) and Kneib et al. (2023), for probabilistic forecasting see Petropoulos et al., 2022; Gneiting & Katzfuss, 2014; Ziel et al., 2016; Nowotarski & Weron, 2018; Álvarez et al., 2021; Ziel, 2022). However, online learning approaches for the conditional heteroskedasticity remain sparse in the literature (see e.g. Álvarez et al. 2021 for adaptive heteroskedasticity estimation and Priouret et al. 2005; Dahlhaus & Subba Rao 2007; Werge & Wintenberger 2022; Hendrych & Cipra 2018; Cipra & Hendrych 2018 for online learning for conditional heteroskedasticity) and, to the best of our knowledge, there are no regularized online distributional regression approaches suitable for high-dimensional processes available so far.

Setting. In this paper, we provide a regularized online learning algorithm for the conditional distribution parameters of the response variable $\mathcal{Y}$ based on a combination of the online coordinate descent algorithm introduced by Angelosante et al. (2010) and Messner & Pinson (2019) and the generalized additive model for location, scale and shape (GAMLSS) models introduced by Rigby & Stasinopoulos (2005); Stasinopoulos & Rigby (2008); Stasinopoulos et al. (2018). Formally, the GAMLSS framework assumes that $i = 1, 2, ..., n$ independent observations $\mathcal{Y}_i$ have the probability density function (PDF)

$$f_{\mathcal{Y}}\left(y_i \mid \mu_i, \sigma_i, \nu_i, \tau_i\right)$$

with (up to) four distribution parameters, each of which can be a (linear) function of explanatory variables. The first two parameters, μ_i and σ_i, commonly characterize the location and scale of the distribution, while ν_i and τ_i are commonly denoted as the shape parameters describing the skewness and kurtosis. Rigby & Stasinopoulos (2005) define the original linear parametric GAMLSS as follows:

$$\mathcal{Y}_i \sim \mathcal{D}\left(\mu_i, \sigma_i, \nu_i, \tau_i\right) \Leftrightarrow \mathcal{D}\left(\theta_{i,1}, \theta_{i,2}, \theta_{i,3}, \theta_{i,4}\right) \tag{1}$$

for $i = 0, 1, ..., n$ independent observations of $\mathcal{Y}_i$, where $\mathcal{D}$ is any distribution function with (up to) p distribution parameters $\theta_{i,k}$ for $k = 1, ..., p$, where $p = 4$ commonly[1] and let $g_k(\cdot)$ be a known, monotonic

[1] Distributions with more than four parameters can be implemented nevertheless.

link function relating a distribution parameter to a predictor $\boldsymbol{\eta}_k$ and have

$$g_k(\theta_k) = \boldsymbol{\eta}_k = \mathcal{X}^k \boldsymbol{\beta}_k \tag{2}$$

where $\mathcal{X}^k$ is a known design matrix, $\boldsymbol{\beta}_k = (\beta_{k,1}, ..., \beta_{k,J_k})^\top$ is a parameter vector of length J_k and $\theta_k = (\theta_{0,k}, \theta_{1,k}, ..., \theta_{n,k})$. The linear parametric GAMLSS model, therefore, allows the modelling of all conditional distribution parameters as linear functions of the explanatory variables in $\mathcal{X}_k$ and is usually fitted using an iteratively reweighted algorithm on the cross-derivatives of the distribution function $\mathcal{D}$. This setup corresponds to the setting in which LASSO estimation was introduced to GAMLSS by Groll et al. (2019). Further regularized estimation approaches have been proposed by Ziel et al. (2021).

Online Setting. After having received the first n observations, we receive a new pair set of data Y_{n+1} and $X_{k,n+1}$. In the online setting, we are interested in updating the parameters of the model $\boldsymbol{\beta}_k$. The proposed algorithm uses the RS algorithm proposed by Rigby & Stasinopoulos (2005) in combination with the online coordinate descent (OCD, see Angelosante et al., 2010; Messner & Pinson, 2019) to update the regression coefficients based on the Gramian matrices. We employ online model selection based on information criteria.

Contributions. In summary, we make the following contributions to the current literature:

- We propose the *online* regularized linear GAMLSS as an efficient and scalable distributional regression framework for incremental learning. We discuss in-depth the implementation of the incremental update step, including regularization and online model selection (Section 2).
- We provide an open-source, ready-to-use `Python` implementation of the online GAMLSS model. Our implementation is based on `numpy` and `scipy` and employs `numba` just-in-time compilation for high computational efficiency. The code can be found here `https://github.com/simon-hirsch/rolch` and the package is available from PyPy (Section 3).
- We validate the proposed method in a forecasting study for electricity prices and demonstrate the competitive performance of our model, combined with strongly reduced estimation time compared to batch estimation (Section 4).

Future Research. Our work opens multiple avenues for future research. First, our implementation is constrained to linear parametric models. The inclusion of smooth terms and splines seems like a worthwhile extension. On a regular, non-adaptive grid, the inclusion is straightforward by adapting the design matrix $\mathcal{X}$. However, automatic, online knot selection for penalized splines is a non-trivial extension. Furthermore, the GAMLSS community has developed multiple extensions for the GAMLSS, e.g. using boosting to enhance the predictive performance or copulas to employ GAMLSS models in multivariate settings. Furthermore, the inclusion of autoregressive and cross-moments effects can provide valuable tools for modelling highly

complex processes such as weather, electricity markets or supply chain applications. Lastly, a thorough theoretical analysis of the error bounds of the online model compared to the batch solution should further increase the trust in the presented methods.

Paper Structure. The remainder of this paper is structured as follows: Section 2 briefly reviews the literature on online estimation of linear models with a focus on online coordinate descent (Section 2.2) and the RS algorithm for the linear parametric GAMLSS (Section 2.1) and presents the online GAMLSS model. Section 3 presents our open-source implementation. The following section presents our forecasting study setting (Section 4), and Section 5 presents the results. Finally, Section 6 concludes the paper.

2. Methodology

The following section introduces our methodology in detail. We first introduce the notation. The following three subsections present the GAMLSS' RS algorithm in a batch setting (Section 2.1), give an introduction to online estimation for (regularized) linear models (Section 2.2) and, finally, section 2.3 presents our algorithm `ROLCH` to enable regularized online learning for conditional heteroskedasticity.

Notation. Generally, we use a subscript n to indicate up to which observation data is available. Let $\boldsymbol{X}_n = (X_{n,1}, ..., X_{n,J})$ denote the n-th vector of J explanatory variables received and $\mathcal{X}_n = (\boldsymbol{X}_1, ..., \boldsymbol{X}_n)$ the full data set of n and J explanatory variables. In the online setting, rows $\boldsymbol{X}_n, \boldsymbol{X}_{n+1}, ...$ are received subsequently. Similarly, $\mathcal{Y}_n$ denotes all observations of the dependent variable $(Y_1, ..., Y_n)$ received by the forecaster so far. In the distributional setting, we have up to $k = 1, ..., p$ distribution parameters and accordingly up to p data sets $\mathcal{X}_{n,k}$ containing the J_k covariates for the distribution parameter k. We also use a `numpy`-style notation if we refer to a certain subset of matrices. For arbitrary matrix $\mathcal{A}$, $\mathcal{A}[n, :]$ denotes accessing the n-th row, while $\mathcal{A}[:, i]$ denotes accessing the i-the column. We start indices generally at 0. We denote changes across different iterations of our algorithm using a superscript $[j]$, for example $\widehat{\beta}_k^{[j]}$.

2.1. Batch GAMLSS

Preliminaries. The following exposition of the RS GAMLSS algorithm largely follows Appendix B of Rigby & Stasinopoulos (2005) and Stasinopoulos & Rigby (2008). Let us introduce some notation first. Generally, $\mathcal{L}_{\mathcal{D}}$ and $l_{\mathcal{D}}$ denote the likelihood and log-likelihood function of the distribution $\mathcal{D}$ given the data Y_i, $l(Y \mid \boldsymbol{\theta})$, where $\boldsymbol{\theta} = (\theta_1, ..., \theta_k)$ for k distribution parameters. Let

$$\boldsymbol{u}_k = \frac{\frac{\partial l}{\partial \theta_k}}{\frac{\partial \boldsymbol{\eta}_k}{\partial \theta_k}} \tag{3}$$

denote the score functions and

$$\boldsymbol{z}_k = \boldsymbol{\eta}_k + \boldsymbol{W}_{kk}^{-1} \boldsymbol{u}_k \tag{4}$$

be the adjusted dependent variables. $\boldsymbol{W}_{kk}$ is a diagonal, iterative weight matrix, which can have one of the following forms:

$$\boldsymbol{W}_{ks} = -\frac{\frac{\partial^2 l}{\partial \theta_k}}{\frac{\partial \boldsymbol{\eta}_k}{\partial \theta_k}\left(\frac{\partial \boldsymbol{\eta}_s}{\partial \theta_s}\right)^\top} \quad \text{or} \quad \boldsymbol{W}_{ks} = -E\left[\frac{\frac{\partial^2 l}{\partial \theta_k}}{\frac{\partial \boldsymbol{\eta}_k}{\partial \theta_k}\left(\frac{\partial \boldsymbol{\eta}_s}{\partial \theta_s}\right)^\top}\right] \quad \text{or} \quad \boldsymbol{W}_{ks} = -\text{diag}\left\{\frac{\frac{\partial l}{\partial \theta_k}}{\frac{\partial \boldsymbol{\eta}_k}{\partial \theta_k}}\frac{\frac{\partial l}{\partial \theta_k}}{\frac{\partial \boldsymbol{\eta}_s}{\partial \theta_s}}\right\} \tag{5}$$

over $i = 1, 2, ..., n$, i.e. the observed (Newton-Rapson), expected (Fisher-Scoring) or product (quasi Newton-Rapson) score function depending on which kind of algorithm is used in the RS algorithm. We use a Newton-Rapson scoring in our implementation. Hence, the adjusted observation vector $\boldsymbol{z}_k$ reads:

$$\boldsymbol{z}_k = \boldsymbol{\eta}_k + \frac{\frac{\partial l}{\partial \theta_k}}{\boldsymbol{W}_{kk}\left(\frac{\partial \boldsymbol{\eta}_k}{\partial \theta_k}\right)} = \boldsymbol{\eta}_k + \frac{\frac{\partial l}{\partial \theta_k}}{\left(-\frac{\frac{\partial \boldsymbol{\eta}_k}{\partial \theta_k}\left(\frac{\partial \boldsymbol{\eta}_s}{\partial \theta_s}\right)^\top}{\frac{\partial^2 l}{\partial \theta_k}}\right)\left(\frac{\partial \boldsymbol{\eta}_k}{\partial \theta_k}\right)} \tag{6}$$

Note that, in the implementation, we regularly employ that it holds:

$$\frac{\partial \boldsymbol{\eta}_k}{\partial \theta_k} = \frac{\partial \boldsymbol{\eta}_k}{\partial g^{-1}(\boldsymbol{\eta}_k)} = \frac{1}{\frac{\partial g^{-1}(\boldsymbol{\eta}_k)}{\partial \boldsymbol{\eta}_k}} \tag{7}$$

where $g^{-1}(\boldsymbol{\eta}_k)$ is the inverse of the link function (Rigby & Stasinopoulos, 2005; Stasinopoulos & Rigby, 2008; Stasinopoulos et al., 2024).

The Batch RS Algorithm. Recall that k is the parameter index and let m and r be the index for the outer and inner iteration of the RS algorithm.[2] We denote iterations as superscript, i.e. $\theta_k^{[m,r]}$ is the value of θ for the k-th distribution parameter at the outer iteration m and the inner iteration n. The RS algorithm consists of two cycles. The outer cycle maximises the penalized likelihood with respect to $\boldsymbol{\beta}_k$. The inner cycle is the iterative fitting for each distribution parameter. In each calculation of the algorithm, the most recent updated values of all quantities are used. We omit the back-fitting cycle for smooth terms in this description. The RS algorithm, as outlined in Rigby & Stasinopoulos (2005), is as as follows:

1. (*Initialisation:*) Initialise the fitted values $\theta_k^{[0,0]}$ for $k = 1, ..., p$. Evaluate the linear predictors $\boldsymbol{\eta}_k^{[0,0]} = g_k(\theta_k^{[0,0]})$ for $k = 0, ..., p$.
2. (*Outer Cycle:*) Start the outer cycle $m = 0, 1, ...$ until convergence. Iterate over $k = 1, ..., p$ and
   (a) Start the inner cycle $n = 0, 1, ...$ until convergence:

[2]Note that Rigby & Stasinopoulos (2005) define r as the outer iteration index, i as the inner iteration index and also i as in the observation index. However, overloading the observation index i leads to confusion in the online case, hence we deviate deliberately from their notation here.

- Evaluate $\boldsymbol{u}_k^{[m,r]}$, $\boldsymbol{W}_{kk}^{[m,r]}$ and $\boldsymbol{z}_k^{[m,r]}$
- Regress $\boldsymbol{z}_k^{[m,r]}$ against the design matrix $\mathcal{X}_k$ using the iterative weights $\boldsymbol{W}_{kk}^{[m,r]}$ to obtain the parameter estimates $\boldsymbol{\beta}_k^{[m,r+1]}$.
- Calculate the updated $\boldsymbol{\eta}_k^{[m,r+1]}$ and $\boldsymbol{\theta}_k^{[m,r+1]}$.
- Evaluate the convergence.

(b) End the inner cycle on the convergence of $\boldsymbol{\beta}_k^{[m,\dots]}$ and set $\boldsymbol{\beta}_k^{[m+1,0]} = \boldsymbol{\beta}_k^{[m,\dots]}$, set $\boldsymbol{\eta}_k^{[m+1,0]} = \boldsymbol{\eta}_k^{[m,\dots]}$ and set $\boldsymbol{\theta}^{[m+1,0]} = \boldsymbol{\theta}_k^{[m,\dots]}$. Otherwise, continue the inner cycle until convergence.

3. (*Next Parameter:*) Update k upon convergence.
4. (*End:*) End the outer cycle if the change in the penalized likelihood is sufficiently small, otherwise update m and continue the outer cycle.

Note that step 2 – b is the core estimation of the conditional distribution parameters and that the RS algorithm is agnostic to which (weighted) statistical learning method employed to regress $\boldsymbol{z}_k^{[m,r]}$ against the design matrix $\mathcal{X}_k$ using the iterative weights $\boldsymbol{W}_{kk}^{[m,r]}$. This has facilitated the development of multiple extensions to the batch GAMLSS, such as `gamlss.boost` (Mayr et al., 2012; Hofner et al., 2014) or regularized approaches as in Groll et al. (2019) and `gamlss.lasso` (Ziel et al., 2021). We will employ this agnosticism and use online coordinate descent (OCD) to update the distributional parameters online.

Relation to iteratively reweighted least squares. As an aside, it is possible to show that the RS algorithm, using the Newton-Rapson scoring, nests the iteratively reweighted least squares algorithm (IRLS) for conditional heteroskedasticity (Ziel et al., 2016; Ziel, 2016) if $\mathcal{D}$ is chosen as the mean-variance parameterization of the normal distribution and both $g_0(\cdot)$ and $g_1(\cdot)$ are the identity link functions.

Propagation of errors. Note that, as in many instances of distributional modelling, the model for the conditional mean $\hat{\mu} = \mathcal{X}_\mu \hat{\beta}_\mu$ is part of the model for the conditional variance due to the relation $\sigma^2 = (\mathcal{Y} - \hat{\mu})^2$. Therefore, a low-quality model for $\hat{\mu}$ will propagate through the estimation and overestimate the conditional variance and potentially higher moments (Ziel, 2022).

2.2. Online estimation of regularized linear models

Let us now move to the online or streaming setting. For the sake of simplicity, we assume to be in a regular regression setting and omit the subscript k for the distribution parameter used in the previous subsection. However, we are now interested in the subscript $i = 0, 1, ..., n$, which indicates which data the estimation algorithm has seen already. In the next Section 2.3, where we combine both components, we will employ both subscripts. The issue at hand can be summarized as follows: given some data $\mathcal{Y}_n$ and $\mathcal{X}_n$ we have estimated a set of coefficients β_n. We are now interested in updating the coefficients β_n given a new observation for Y_{n+1} and a new row of $\boldsymbol{X}_{n+1}$. The following paragraphs outline the algorithms for recursive

least squares, online LASSO and online LARS. For regularized estimation techniques, the issue of model selection arises. Therefore, the last paragraph discusses online model selection via information criteria.

Recursive Least Squares. We start with the well-known least-squares estimation for the coefficients β^{LS}:

$$\widehat{\boldsymbol{\beta}}_n^{\text{LS}} = (\mathcal{X}_n^\top \mathcal{X}_n)^{-1} \mathcal{X}_n^\top \mathcal{Y}_n.$$

given $\mathcal{Y}_n = (Y_1, \ldots, Y_n)^\top$ and $\mathcal{X}_n = (\boldsymbol{X}_1^\top, \ldots, \boldsymbol{X}_n^\top)$. In a repeated batch setting, we can use:

$$\widehat{\boldsymbol{\beta}}_{n+1}^{\text{LS}} = (\mathcal{X}_{n+1}^\top \mathcal{X}_{n+1})^{-1} \mathcal{X}_{n+1}^\top \mathcal{Y}_{n+1}.$$

to calculate $\widehat{\boldsymbol{\beta}}_{n+1}^{\text{LS}}$. Alternatively, employing the *Sherman-Morrison formula* and calculate β^{LS} via recursive least squares. For an invertible matrix $\boldsymbol{A}$ and vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ it holds

$$(\boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^\top)^{-1} = \boldsymbol{A}^{-1} - \frac{\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^\top \boldsymbol{A}^{-1}}{1 + \boldsymbol{v}^\top \boldsymbol{A}^{-1}\boldsymbol{u}}.$$

We also have: $\mathcal{X}_{n+1}^\top \mathcal{X}_{n+1} = \mathcal{X}_n^\top \mathcal{X}_n + \boldsymbol{X}_{n+1}^\top \boldsymbol{X}_{n+1}$ and therefore

$$\begin{aligned}(\mathcal{X}_{n+1}^\top \mathcal{X}_{n+1})^{-1} &= (\mathcal{X}_n^\top \mathcal{X}_n + \boldsymbol{X}_{n+1}^\top \boldsymbol{X}_{n+1})^{-1} \\ &= (\mathcal{X}_n^\top \mathcal{X}_n)^{-1} - \frac{(\mathcal{X}_n^\top \mathcal{X}_n)^{-1} \boldsymbol{X}_{n+1}^\top \boldsymbol{X}_{n+1} (\mathcal{X}_n^\top \mathcal{X}_n)^{-1}}{1 + \boldsymbol{X}_{n+1} (\mathcal{X}_n^\top \mathcal{X}_n)^{-1} \boldsymbol{X}_{n+1}^\top}. \end{aligned} \tag{8}$$

Thus, we can compute $(\mathcal{X}_{n+1}^\top \mathcal{X}_{n+1})^{-1}$ given $(\mathcal{X}_n^\top \mathcal{X}_n)^{-1}$ and $\boldsymbol{X}_{n+1}$ without much effort. Furthermore, we receive $\widehat{\boldsymbol{\beta}}_{n+1}^{\text{LS}}$ by noting that $\mathcal{X}_n^\top \mathcal{X}_n \widehat{\boldsymbol{\beta}}_n^{\text{LS}} = \mathcal{X}_n^\top \mathcal{Y}_n$:

$$\begin{aligned}\widehat{\boldsymbol{\beta}}_{n+1}^{\text{LS}} &= (\mathcal{X}_{n+1}^\top \mathcal{X}_{n+1})^{-1} \mathcal{X}_{n+1}^\top \mathcal{Y}_{n+1} \\ &= \widehat{\boldsymbol{\beta}}_n^{\text{LS}} + (\mathcal{X}_{n+1}^\top \mathcal{X}_{n+1})^{-1} \boldsymbol{X}_{n+1}^\top \underbrace{\left(Y_{n+1} - \boldsymbol{X}_{n+1} \widehat{\boldsymbol{\beta}}_n^{\text{LS}}\right)}_{\text{forecast error in } n+1} \end{aligned} \tag{9}$$

Hence, with (9) and (8), we can efficiently compute expanding window updates for the coefficient vector β^{LS}.

Weighted Recursive Least Squares. The RS algorithm presented in the previous Section 2.1 employs a weighted estimation. Hence, we will show in this paragraph how the weighted recursive least squares allow the online estimation of linear models. We define the weight matrix $\mathcal{W}$ with known weights for observations $1, ..., n, ...$ on the diagonal

$$\mathcal{W}_n = \begin{pmatrix} w_1 & 0 & \ldots & 0 \\ 0 & w_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & w_n \end{pmatrix} \tag{10}$$

Under the assumptions that (a) the weights are known and (b) the weights do not change for past observations, the weighted least-squares problem can be written as:

$$\begin{aligned}\widehat{\boldsymbol{\beta}}_n^{\text{WLS}} =& (\mathcal{X}_n^\top \mathcal{W}_n \mathcal{X}_n)^{-1} \mathcal{X}_n^\top \mathcal{W}_n \mathcal{Y}_n \\ =& \left((\mathcal{W}_n^{\frac{1}{2}} \mathcal{X}_n)^\top (\mathcal{W}_n^{\frac{1}{2}} \mathcal{X}_n) \right)^{-1} (\mathcal{W}_n^{\frac{1}{2}} \mathcal{X}_n)^\top \mathcal{W}_n^{\frac{1}{2}} \mathcal{Y}_n \end{aligned} \tag{11}$$

by noting that $\mathcal{W}_n$ is invertible as long as all weights are positive and non-zero. For convenience, we define the vectors

$$\mathcal{M}_n = (\mathcal{W}_n^{\frac{1}{2}} \mathcal{X}_n) \tag{12}$$

$$\mathcal{M}_{n+1} = (\mathcal{W}_{n+1}^{\frac{1}{2}} \mathcal{X}_{n+1}) \tag{13}$$

$$\boldsymbol{M}_{n+1} = \sqrt{w_{n+1}} \boldsymbol{X}_{n+1} \tag{14}$$

and have

$$\mathcal{M}_{n+1}^\top \mathcal{M}_{n+1} = \mathcal{M}_n^\top \mathcal{M}_n + \boldsymbol{M}_{n+1}^\top \boldsymbol{M}_{n+1} \tag{15}$$

so that we can again use the Sherman-Morrison Formula, we can re-write the weighted Gram Matrix $(\mathcal{X}_{n+1}^\top \mathcal{W}_{n+1} \mathcal{X}_{n+1})^{-1} = (\mathcal{M}'_{n+1} \mathcal{M}_{n+1})^{-1}$ for the next observation $n+1$ as

$$\begin{aligned}(\mathcal{M}_{n+1}^\top \mathcal{M}_{n+1})^{-1} &= (\mathcal{M}_n^\top \mathcal{M}_n)^{-1} - \frac{(\mathcal{M}_n^\top \mathcal{M}_n)^{-1} \boldsymbol{M}_{n+1}^\top \boldsymbol{M}_{n+1} (\mathcal{M}_n^\top \mathcal{M}_n)^{-1}}{1 + \boldsymbol{M}_{n+1} (\mathcal{M}_n^\top \mathcal{M}_n)^{-1} \boldsymbol{M}_{n+1}} \\ &= (\mathcal{X}_n^\top \mathcal{W}_n \mathcal{X}_n)^{-1} - \frac{w_{n+1} (\mathcal{X}_n^\top \mathcal{W}_n \mathcal{X}_n)^{-1} \boldsymbol{X}_{n+1}^\top \boldsymbol{X}_{n+1} (\mathcal{X}_n^\top \mathcal{W}_n \mathcal{X}_n)^{-1}}{1 + w_{n+1} \boldsymbol{X}_{n+1} (\mathcal{X}_n^\top \mathcal{W}_n \mathcal{X}_n)^{-1} \boldsymbol{X}'_{n+1}} \end{aligned} \tag{16}$$

and the regression equation as:

$$\begin{aligned}\widehat{\boldsymbol{\beta}}_{n+1}^{\text{WLS}} &= (\mathcal{X}_{n+1}^\top \mathcal{W}_{n+1} \mathcal{X}_{n+1})^{-1} \mathcal{X}_{n+1}^\top \mathcal{W}_{n+1} \mathcal{Y}_{n+1} \\ &= \widehat{\boldsymbol{\beta}}_n^{\text{WLS}} + (\mathcal{X}_{n+1}^\top \mathcal{W}_{n+1} \mathcal{X}_{n+1})^{-1} w_{n+1} \boldsymbol{X}_{n+1}^\top \underbrace{\left(Y_{n+1} - \boldsymbol{X}_{n+1} \widehat{\boldsymbol{\beta}}_n^{\text{LS}} \right)}_{\text{forecast error in } n+1} \end{aligned} \tag{17}$$

Again, combining (17) and (16) we can efficiently compute the update for the weighted recursive least-squares estimator $\widehat{\boldsymbol{\beta}}_{n+1}^{\text{WLS}}$.

Recursive least squares with exponential forget. The algorithm can be naturally extended to RLS under exponential forgetting

$$\hat{\beta} = \arg\min_{\beta} \left\{ \left\| \gamma^{T-t} (Y - \beta X) \right\|_2^2 \right\}, \tag{18}$$

where γ denotes the forgetting factor. Define the weight matrix Γ_n as

$$\Gamma_n = \begin{pmatrix} \gamma^{N-1} & 0 & \dots & 0 \\ 0 & \gamma^{N-2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \gamma \end{pmatrix} \tag{19}$$

and we note that the exponential discounting of older observations leads to

$$\mathcal{X}_{n+1}^{\top}\Gamma_{n+1}\mathcal{X}_{n+1} = \gamma(\mathcal{X}_{n}^{\top}\Gamma_{n}\mathcal{X}_{n}) + \boldsymbol{X}_{n+1}^{\top}\boldsymbol{X}_{n+1} \tag{20}$$

and the according update equation for the exponentially discounted Gram matrix as

$$(\mathcal{X}_{n+1}^{\top}\Gamma_{n+1}\mathcal{X}_{n+1})^{-1} = \frac{1}{\gamma}\left((\mathcal{X}_{n}^{\top}\Gamma_{n}\mathcal{X}_{n})^{-1} - \frac{(\mathcal{X}_{n}^{\top}\Gamma_{n}\mathcal{X}_{n})^{-1}\boldsymbol{X}_{n+1}^{\top}\boldsymbol{X}_{n+1}(\mathcal{X}_{n}^{\top}\Gamma_{n}\mathcal{X}_{n})^{-1}}{\gamma + \boldsymbol{X}_{n+1}(\mathcal{X}_{n}^{\top}\Gamma_{n}\mathcal{X}_{n})^{-1}\boldsymbol{X}_{n+1}^{\top}}\right). \tag{21}$$

Similar to the previous paragraph, we can efficiently update the coefficients by combining Equations (17) and (21).

Online coordinate Descent for LASSO. The LASSO estimator for linear models minimizes:

$$\widehat{\beta}_n = \arg\min_{\beta}\left\{\left\|\mathcal{W}^{\frac{1}{2}}\left(\mathcal{Y}_n - \beta\mathcal{X}_n\right)\right\|_2^2 + \lambda_n\left\|\beta\right\|_2\right\}.$$

Friedman et al. (2007, 2010) have introduced Path-wise cyclic coordinate descent (CCD) for LASSO, ridge and elastic net regression problems, and it remains the computationally efficient method to estimate coefficients. For a given regularization parameter λ, we repeatedly update the coefficient vector $\boldsymbol{\beta}$ of length J by taking

$$\beta_j \leftarrow \frac{S\left(\sum_{n=1}^{N} w_n x_{n,j}(y_n - \tilde{y}_n^{(j)}), \lambda\right)}{\sum_{n=1}^{N} w_n x_{n,j}^2} \tag{22}$$

where $S(\hat{\beta}, \lambda) = \text{sign}(\hat{\beta})(|\hat{\beta}| - \lambda)_+$ is the soft-thresholding function. Note that, implicitly, Friedman et al. (2010) assume that $\sum_{n=1}^{N} w_n = 1$. We use a numerical convergence criterion to break the algorithm after convergence. As proposed in Angelosante et al. (2010) and Messner & Pinson (2019), we rewrite the update Equation 22 to calculate the updated parameter from the Gramian matrix $\mathcal{G} = \mathcal{X}^{\top}\mathcal{X}$ and the y-Gram matrix $\mathcal{H} = \mathcal{X}^{\top}\mathcal{Y}$. In the univariate case, we can write the online coordinate descent (OCD) update rule for the LASSO estimator as:

$$\beta_j \leftarrow \frac{S\left(\mathcal{H}\left[j\right] - \mathcal{G}\left[j,:\right]\boldsymbol{\beta} + \mathcal{G}\left[j,j\right]\beta_j, \lambda\right)}{\mathcal{G}\left[j,j\right]} \tag{23}$$

Therefore, the algorithm only needs to store the $J \times J$ matrix $\mathcal{G}$ and the $J \times 1$ vector $\mathcal{H} = \mathcal{X}^{\top}\mathcal{Y}$. Similar to the classical coordinate descent, we can run the algorithm for a decreasing sequence of regularization strengths λ, starting with $\lambda_{\max} = \max|\mathcal{X}^{\top}\mathcal{Y}|$ as the element-wise maximum in $\mathcal{H}$ and using an exponential grid towards $\lambda_{\min} = \epsilon\lambda_{\max}$ with $\epsilon_{\lambda} = 0.001$ as typical values (Friedman et al., 2010, 2007; Messner & Pinson, 2019). Algorithm 1 presents the algorithm schematically. To improve the convergence speed in estimating the full λ-path, the $\boldsymbol{\beta}_{\lambda}$ for the previous regularization parameter is used as the start value. Also, we can employ potentially weighted Gramian matrices $\mathcal{G} = \mathcal{X}_n^{\top}\mathcal{W}_n\mathcal{X}_n$ and $\mathcal{H} = \mathcal{X}_n^{\top}\mathcal{W}_n\mathcal{Y}_n$ and the respectively exponentially discounted versions of $\mathcal{G}_n$ and $\mathcal{H}_n$ as in the previous paragraphs (Messner & Pinson, 2019). Our implementation in the software package `ROLCH` exploits active set iterations to speed up convergence, i.e. after the first full cycle through all features j, we only update non-zero coefficients (Meier et al., 2008;

Algorithm 1: Online LASSO

Data: $\boldsymbol{X}_{n+1}, Y_{n+1}, w_{n+1}$

Result: $\boldsymbol{\beta}_{n+1}$

Update the Gramian matrices $\mathcal{G}_{n+1}$ and $\mathcal{H}_{n+1}$ according to Equation 16.

for λ **do**

  Set starting coefficients $\boldsymbol{\beta}_{\lambda} \leftarrow \boldsymbol{\beta}_{\lambda[-1]}$

  forall $j \in 1, ..., J$ **do**

    update $\beta_{j,\lambda}$ according to Equation 23

  Check convergence for $\widehat{\boldsymbol{\beta}}_{n+1,\lambda}$ and proceed to next λ if converged.

Return: $\widehat{\boldsymbol{\beta}}_{n+1}$ for all λ

Friedman et al., 2010). A further overview on different implementations, tips and tricks can be found in Shi et al. (2016).

Online LARS. The original LARS algorithm introduced by Efron et al. (2004) is already using the Gramian matrix, so the adaption to an online setting is straightforward. Maheshwari et al. (2017) have also proposed a LARS-like online learning algorithm. However, compared to the OCD algorithm for the LASSO estimator, we cannot warm-start the online LARS implementation. Therefore, we prefer the LASSO estimator throughout the remainder of the paper.

Model selection and information Criteria. We select the regularization within the iterative re-weighting loop using information criteria. We define the generalised information criterion (GIC) as

$$\mathrm{GIC}(\hat{\mathcal{L}}, \nu) = -2\log(\hat{\mathcal{L}}) + \nu_0 k + \nu_1 k \log(n) + \nu_2 k \log(\log(n)) \tag{24}$$

where k denotes the number of estimated parameters, $\nu = (\nu_0, \nu_1, \nu_2)$ denotes a triplet of parameters, and $\hat{\mathcal{L}}$ denotes the maximized value of the likelihood function of the model. The most commonly used information criteria Akaike's Information Criterion (AIC, $\nu = (2, 0, 0)$), Bayesian Information Criterion (BIC, $\nu = (0, 1, 0)$) and the Hamann-Quinn Criterion (HQC, $\nu = (0, 0, 2)$) can be derived of the GIC and are implemented in our software package as tuning hyper-parameters Ding et al. (2018). Under the Gaussian assumption, the likelihood can be formulated as a function of the residual sum-of-squares

$$\log(\hat{\mathcal{L}}) = -\frac{n}{2} + \log\left(\frac{\mathrm{RSS}}{n}\right) + C \tag{25}$$

where $C = -\frac{n}{2}\left(1 + \log(2\pi)\right)$ is a constant which only depends on the data and hence can be neglected if the data underlying the model selection is the same for all models. This allows an efficient online update of the information criterion, including the option to account for exponential forget. Note that we use the effective training length $n = \frac{(1-\gamma^N)}{(1-\gamma)}$, where N is the number of data points seen by the model (so far). Alternatively,

Messner & Pinson (2019) propose to use the exponentially discounted in-sample root mean square error to select the optimal λ - essentially taking $\frac{1}{n}$RSS. However, this approach will converge towards the OLS solution.

2.3. Online GAMLSS

Having established the batch RS algorithm and the online coordinate descent for regularized linear models, this section proceeds by putting the pieces together and presenting the online linear parametric GAMLSS model.

Method. We aim to present the algorithm in the most generic form here. After having seen n observations, we have estimated coefficients $\widehat{\boldsymbol{\beta}}_{k,n}$ for $k = 1, .., p$ distribution parameters. Given some new data $\boldsymbol{Y}_{n+1}$ for the response and $\boldsymbol{X}_{k,n+1}$ covariate design matrices, we aim to update our coefficients towards $\widehat{\boldsymbol{\beta}}_{k,n+1}$ for all $k = 1, ..., p$. Let us recall the definition and update rule for the weighted, exponentially discounted Gramian matrices $\mathcal{G}_{k,n} = \mathcal{X}_{k,n}^\top \Gamma_{k,n} \mathcal{W}_{k,n} \mathcal{X}_{k,n}$ and $\mathcal{H}_{k,n} = \mathcal{X}_{k,n}^\top \Gamma_{k,n} \mathcal{W}_{k,n} \mathcal{Y}_{k,n}$ here, to which we will refer throughout the algorithm.

$$\mathcal{G}_{k,n+1} = \gamma \mathcal{G}_{k,n} + w_{k,n+1}(\boldsymbol{X}_{k,n+1}^\top \boldsymbol{X}_{k,n+1}) \tag{26}$$

$$\mathcal{H}_{k,n+1} = \gamma \mathcal{H}_{k,n} + w_{k,n+1}(\boldsymbol{X}_{k,n+1}^\top \boldsymbol{Y}_{k,n+1}) \tag{27}$$

On a high level, for updating the coefficients in the online GAMLSS, we do the following steps in the outer cycle: we initialise the $\boldsymbol{\eta}_{k,n+1}^{[0,0]} = g_k\left(\boldsymbol{\beta}_{k,n} \boldsymbol{X}_{k,n+1}^\top\right)$ and this allows us to evaluate $\boldsymbol{u}_{k,n+1}^{[0,0]}$, $\boldsymbol{W}_{kk,n+1}^{[0,0]}$ and $\boldsymbol{z}_{k,n+1}^{[0,0]}$. From there, we can update the weighted, exponentially discounted Gramian matrices $\mathcal{G}_{k,n} \to \mathcal{G}_{k,n+1}^{[0,0]}$ and $\mathcal{H}_{k,n} \to \mathcal{H}_{k,n+1}^{[0,0]}$ and subsequently run the online coordinate descent (OCD) algorithm to update the coefficient path in the inner iteration. As in the batch algorithm, we run the outer and inner cycles until convergence.

Detailed Update Algorithm. The following paragraph describes the update step for the algorithm in detail. We keep the structure from Section 2.1. To make the discussion clearer, we call an *update step* a full update for all distribution parameters $\theta_{k,n} \to \theta_{k,n+1}$ and associated regression coefficients $\boldsymbol{\beta}_{k,n} \to \boldsymbol{\beta}_{k,n+1}$ for $k = 1, ..., p$ while after receiving data $\boldsymbol{Y}_{k,n+1}$ and $\boldsymbol{X}_{k,n+1}$, while *step* might refer to any arbitrary step in the algorithm. The update step does the following:

1. (*Initialisation:*) Initialise the fitted values $\theta_{k,n+1}^{[0,0]} = \boldsymbol{\beta}_{k,n} \boldsymbol{X}_{k,n+1}^\top$ for $k = 1, ..., p$. Evaluate the linear predictors $\eta_{k,n+1}^{[0,0]} = g_k(\theta_{k,n+1}^{[0,0]})$ for $k = 0, ..., p$. Note that for a one-step update step, these values are scalars.
2. (*Outer Cycle:*) Start the outer cycle $m = 0, 1, ...$ until convergence. Iterate over $k = 1, ..., p$ and
   (a) Start the inner cycle $n = 0, 1, ...$ until convergence:

- Evaluate $u_{n+1,k}^{[m,r]}$, $w_{n+1,kk}^{[m,r]}$ and $z_{n+1,k}^{[m,r]}$ using Equations 4, 5 and 6.
- Update the Gramian Matrices $\mathcal{G}_{k,n} \to \mathcal{G}_{k,n+1}^{[m,r]}$ and $\mathcal{H}_{k,n} \to \mathcal{H}_{k,n+1}^{[m,r]}$ by taking the working vector $z_{n+1,k}^{[m,r]}$ as response variable and weights $w_{kk,n+1}^{[m,r]}$:

$$\mathcal{G}_{k,n+1}^{[m,r]} = \gamma \mathcal{G}_{k,n} + w_{kk,n+1}^{[m,r]} \left(\boldsymbol{X}_{k,n+1}^{\top} \boldsymbol{X}_{k,n+1} \right)$$
$$\mathcal{H}_{k,n+1}^{[m,r]} = \gamma \mathcal{H}_{k,n} + w_{kk,n+1}^{[m,r]} \left(\boldsymbol{X}_{k,n+1}^{\top} z_{n+1,k}^{[m,r]} \right)$$

- Update the coefficient path $\boldsymbol{\beta}_{n,k,\lambda}^{[m,r+1]} \to \boldsymbol{\beta}_{n,k,\lambda}^{[m,r+1]}$ using OCD based on $\mathcal{G}_{k,n+1}^{[m,r]}$ and $\mathcal{H}_{k,n+1}^{[m,r]}$ and select the optimal tuning parameter λ using information criteria.
  - Update the exponentially discounted weighted sum of squared residuals to approximate the likelihood (see Equation 25) by taking

$$\mathrm{RSS}_{k,n+1,\lambda} = \frac{\left(w_{kk,n+1} \left(z_{n+1,k}^{[m,r]} - \boldsymbol{\beta}_{n,k,\lambda}^{[m,r+1]} \boldsymbol{X}_{k,n+1}^{\top} \right) \right)^2 + (1-\gamma)\mathrm{RSS}_{n,k,\lambda}}{w_{kk,n+1} + (1-\gamma)\omega_{kk,n}} \tag{28}$$

    where $\omega_{kk,n}$ is the sum of discounted weights $\omega_{kk,n} = \sum_{i=0}^{n} \gamma^{i-1} w_{kk,i}$ up to n, which can easily be tracked online.
  - Select the $\lambda_{k,n+1}^{\mathrm{opt}} = \arg\min_{\lambda} \left(\mathrm{GIC}(\hat{\mathcal{L}}_{\lambda}, \nu) \right)$ and update the likelihood based on the RSS by Equation 25. Select the tuned coefficients $\boldsymbol{\beta}_{n+1,k}^{[m,r+1]} = \boldsymbol{\beta}_{n+1,k,\lambda^{\mathrm{opt}}}^{[m,r+1]}$.
- Calculate the updated $\boldsymbol{\eta}_{k,n+1}^{[m,r+1]}$ and $\boldsymbol{\theta}_{k,n+1}^{[m,r+1]}$.
- Evaluate the convergence.

(b) End the inner cycle on the convergence of $\boldsymbol{\beta}_{k,n+1}^{[m,\ldots]}$ and set $\boldsymbol{\beta}_{k,n+1}^{[m+1,0]} = \boldsymbol{\beta}_{k,n+1}^{[m,\ldots]}$, set $\boldsymbol{\eta}_{k,n+1}^{[m+1,0]} = \boldsymbol{\eta}_{k,n+1}^{[m,\ldots]}$ and set $\boldsymbol{\theta}_{k,n+1}^{[m+1,0]} = \boldsymbol{\theta}_{k,n+1}^{[m,\ldots]}$. Otherwise, continue the inner cycle until convergence.

3. (*Next Parameter:*) Update k upon convergence.
4. (*End:*) End the outer cycle if the change in the penalized likelihood is sufficiently small, otherwise update m and continue the outer cycle.

Note that within step 2 – a, in each inner iteration, we start at the Gramian matrices of the previous full fit, not at the previous iterations m or r of the algorithm, since this would imply adding the $\boldsymbol{X}_{k,n+1}$ and $z_{n+1,k}^{[m,r]}$ multiple times to the Gramian matrices within one update step. However, we can (and should) warm-start the OCD algorithm using the coefficient path from the previous iterations within each inner iteration of the update step.

Convergence Criteria. The main convergence criteria is the difference between the log-likelihood under the distribution $\mathcal{D}$, given the inner and outer iterations current fitted values $\hat{\boldsymbol{\eta}}_{k,n+1}^{[m,r]}$, both for the batch and online case. We track the (exponentially discounted) log-likelihood for checking the convergence in the online case. Additionally, we introduce a breaking criterion if the residual sum of squares increases during

the inner cycle by more than a factor of $\varepsilon_{\text{RSS}} = 1.5$ of the previous inner cycle's RSS. This criterion can be turned off by setting ε_{RSS} to an arbitrarily high number or infinity. This criterion can be used to 'force' the algorithm into a local minima with respect to the RSS in the inner cycle, compared to global minima in the deviance. This can improve the convergence, if the parametric distribution assumption does not fit the observed data well, especially in the case of outliers.

Mini-batch update steps. The algorithm above implements one-step updates, i.e. receiving new data row-by-row: in each update, we receive $\boldsymbol{X}_{k,n+1}$ and $\boldsymbol{Y}_{k,n+1}$. In principle, mini-batch update steps, in which we receive $\mathcal{X}_{k,n+1:n+t} = (\boldsymbol{X}_{k,n+1}, ..., \boldsymbol{X}_{k,n+t})^\top$ and $\mathcal{Y}_{n+1:n+t} = (Y_{n+1}, ..., Y_{n+t})$ in a single pass can be implemented in the same manner. Equations 26 and 27 natively handle mini-batch updates. Equation 28 needs to be adjusted to

$$\text{RSS}_{k,n+t,\lambda} = \frac{\left((1-\gamma)^{t-1}\boldsymbol{W}_{kk,n+t}\left(\boldsymbol{z}_{n+t,k}^{[m,r]} - \boldsymbol{\beta}_{n+t,k,\lambda}^{[m,r]}\mathcal{X}_{k,n+1:n+t}^\top\right)\right)^2 + (1-\gamma)^t \text{RSS}_{n,k,\lambda}}{\sum_{i=0}^{t-1}(1-\gamma)^i w_{kk,n+i} + (1-\gamma)^t \omega_{kk,n}}$$

to account for the correct discounting within the mini-batch update step. Mini-batch updates decrease the computational cost by running the OCD fewer times, but the warm-starting might not be as effective.

Trade-offs and Interactions. This paragraph will discuss some interactions for the parameters and their implications.

- We can potentially use different forget factors for γ_k for each distribution parameter. Generally, a higher forget leads to faster adaption of the coefficients. However, since the estimation of higher moments depends on the estimation of the first moment(s), we note that too aggressive adaption of the coefficients of the location will lead to an *underestimation* of the conditional heteroskedasticity and potentially higher moments.

- For the selection of the regularization, we suggest selecting a higher regularization for higher k to avoid overfitting in modelling the conditional scale, kurtosis and skewness (Ziel, 2022). We also note that the $\varepsilon_{\lambda,k}$ for higher moments needs to be smaller, otherwise the lower end of the coefficient path might not approach the OLS solution. Groll et al. (2019) analyse the impact of different shrinkage parameters on batch distributional regression.

Limitations. Finally, let us note an important limitation of our proposed algorithm. In the batch setting with repeated fits of the GAMLSS, the weight matrix $\mathcal{W}_{kk}^{[m,r]} = \text{diag}\left(w_{kk,0}, ..., w_{kk,n}\right)$ is updated in every iteration m, r and again, in the next batch fit for $n+1$, the weight matrix is updated for *all* for $w_{kk,i}$ from $i = 0, ..., n+1$ in all iterations. In the online setting, $\mathcal{W}_{kk,n}$ is the weight matrix after convergence of update

step $n-1 \to n$. In the update step $n \to n+1$, we cannot update $\mathcal{W}_{kk,n}$ anymore. Therefore, we set

$$\mathcal{W}_{kk,n+1}^{[m,r]} = \begin{pmatrix} (1-\gamma)\mathcal{W}_{kk,n} & 0 \\ 0 & w_{kk,n+1}^{[m,r]} \end{pmatrix}$$

and can only update $w_{kk,n+1}^{[m,r]}$. Note that due to the update rules in Equations 26 and 27, this effect is counter-weighted by the exponential forget γ. However, there is a delicate balance to strike to trade off the beneficial effect of the forget and increased instability in the coefficient estimation. This might lead to slower convergence compared to batch learning if the data is drawn from a stationary process. The online GAMLSS is, therefore, an *approximation* of the repeated batch GAMLSS, contrary to the case for, e.g. recursive least squares or online coordinate descent, where the update step leads to the exact coefficients.

3. Implementation in the Python Package `ROLCH`

We provide an open-source implementation of our algorithm in the `Python` package `ROLCH`. To the best knowledge of the authors, this package is the first `Python` implementation of GAMLSS and can, therefore, provide a basis for future extensions.[3] Although still in the early development phase, we provide the following features:

- (*Consistent API*) Our package provides an API comparable to other major machine learning packages in `Python` such as `sklearn` or `tensorflow` with very few dependencies (only `numpy`, `numba`, and `scipy` Harris et al., 2020; Virtanen et al., 2020; Lam et al., 2015).
- (*Automatic scaling*) We employ automatic mean-variance scaling for all covariates with an incremental calculation of the exponentially discounted mean and variance using Welford's Algorithm (Welford, 1962).
- (*Online Model selection*) We extend the model selection approach in Ziel et al. (2021) and allow for online automatic model selection using AIC, BIC, and HQC allowing for fast, efficient and theoretically grounded (local) model selection in the online coordinate descent.
- (*Performance*) We employ just-in-time compilation using `numba` to achieve a high-performance implementation and employ various computational tricks such as active set coordinate descent, different warm-starting options (using the previous fit or a mixture of previous fit and previous β_λ on the same coefficient path) and allow for random selection during coordinate descent to speed up the convergence of the coordinate descent (see e.g. Shi et al., 2016; Wright, 2015).

[3] The package `pyNM` implements GAMLSS for `Python` as binding to the `R` library `https://github.com/ppsp-team/PyNM`.

- (*Flexibility*) At the same time, our implementation offers multiple features for users such as choosing the set of regularized coefficients and the possibility to constrain coefficients within bounds, e.g. allow for positive/negative coefficients only. The update formula for the coordinate descent (Equation 22) is then

$$\beta_j \leftarrow \text{clip}\left(\frac{S\left(\mathcal{H}\left[j\right] - \mathcal{G}\left[j,:\right]\boldsymbol{\beta} + \mathcal{G}\left[j,j\right]\beta_j, \lambda\right)}{\mathcal{G}\left[j,j\right]}, l_j, u_j\right)$$

  where l_j and u_j are the user-chosen bounds for β_j. This corresponds to the implementation in the `glmnet R` package (Friedman et al., 2010; Tay et al., 2023).

Currently, our package implements the Gaussian distribution, Student's t-distribution, and Johnson's S_U distribution.[4] The code for our implementation can be found at `https://github.com/simon-hirsch/rolch` and the package is available at the Python Package Index (`https://pypi.org/project/rolch/`) under the MIT-License.

4. Forecasting Study for Electricity Price Forecasting

The following section presents an exemplary application in electricity price forecasting (EPF) for the online GAMLSS method. We employ the same setting as Marcjasz et al. (2023) in forecasting day-ahead electricity prices for the German short-term electricity market.

Market Description. The German electricity market consists of four major markets: The long-term futures market, the day-ahead spot market, the intraday market and the balancing market. Since our forecasting study is concerned with the day-ahead or spot market only, our description focuses on this market only. A detailed summary of the German electricity market can be found in Viehmann (2017). The spot market is the major reference price for the futures market and its trading volume is a multiple of the intraday and balancing market. Let $h \in \{0, 1, ..., H\}$ and $H = 23$ denote the 24 delivery hours and d denote the delivery day. The market is organized as a daily auction at $d-1$, 12:00 hours for all 24 delivery hours of the following day. Results are published at $d-1$, 12:42 hours. Market participants submit limit orders and block/linked/complex bids. The market is organized as a pay-as-cleared auction and cleared by the EUPHEMIA algorithm and maximises the European Welfare under consideration of constraints such as the available international transmission capacity (Viehmann, 2017; Marcjasz et al., 2023; Uniejewski & Weron, 2021; Ziel & Weron, 2018; Lago et al., 2018, 2021).

[4] Since the implementations derive from `scipy`, implementing further distributions is straightforward. Only the cross-derivatives and the initial guess for the $\theta_k^{[0,0]}$ need to be added manually.

Models. We employ a (linear) expert model type as it is common in electricity price forecasting Lago et al. (2018); Marcjasz et al. (2023); Ziel & Weron (2018). The model consists of 38 terms and captures the autoregressive price effects, seasonal effects, the fundamental effects of renewable generation and the influence of fuel prices. In the distributional framework, we model each distribution parameter $k = 1, ..., p$ for each delivery hour $h = \{0, 1, ..., H\}$ individually as function of linear predictors:

$$\begin{aligned} g_k(\theta^k_{d,h}) = \beta_{k,0,h} &+ \beta_{k,1,h} P_{d-1,h} + \beta_{k,2,h} P_{d-2,h} + \beta_{k,3,h} P_{d-7,h} + \beta_{k,4,h} P_{d-14,h} \\ &+ \sum_{s \in \{0,1,...,H\} \setminus h} \beta_{5+s,k,h} P_{d-1,k} + \beta_{k,29,h} \widehat{\text{Load}}_{d,h} + \beta_{k,30,h} \widehat{\text{RES}}_{d,h} \\ &+ \beta_{k,31,h} \, \text{EUA}_d + \beta_{k,32,h} \, \text{Gas}_d + \beta_{k,33,h} \, \text{Coal}_{d,h} + \beta_{k,34,h} \, \text{Oil}_d \\ &+ \beta_{k,35,h} \, \text{Mon}_{d,h} + \beta_{k,36,h} \, \text{Sat}_{d,h} + \beta_{k,37,h} \, \text{Sun}_{d,h} \end{aligned} \tag{29}$$

where $\beta_{k,0,h}$ is the intercept or bias, $\beta_{k,1,h}$ ot $\beta_{k,4,h}$ capture autoregressive effects, $\beta_{5,k,h}$ to $\beta_{28,k,h}$ capture the price level of all other hours of the previous day, $\beta_{k,29,h}$ and $\beta_{k,30,h}$ model the influence of system load or demand and the renewable generation in-feed, $\beta_{k,31,h}$ to $\beta_{k,34,h}$ capture the effects of the price level for European Emission Allowances (EUAs), natural gas, coal and oil prices and finally $\beta_{k,35,h}$ to $\beta_{k,38,h}$ capture the weekly seasonality for Mondays, Saturdays and Sundays. Data is retrieved from ENTSO-E and from Marcjasz et al. (2023). Similar to their setting, the training data set from 2015-01-15 to 2018-12-26, our test data ranges from 2018-12-27 to 2020-12-31.

Scoring Rules. We benchmark forecasts using established, proper probabilistic scoring rules (Gneiting, 2008, 2011; Gneiting & Katzfuss, 2014; Nowotarski & Weron, 2018). For the mean prediction, we employ the Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE). We evaluate the Coverage (CR) and the Interval Score (IS, also known as Winkler Score, see Bracher et al. (2021)) for the 50%, 75%, 90% and 95% prediction intervals (PI). For the full predictive distribution, we evaluate the Log Score (LS) and the continuous ranked probability score (CRPS) using the approximation via the Pinball Score (PS) on a dense grid of quantiles $\mathcal{Q} = \{0.01, 0.02, 0.03, ..., 0.99\}$. The implementation of the scoring rules is provided by the `scoringrules` package (Zanetta & Allen, 2024). We evaluate the statistical significance of the difference in predictive accuracy using the Diebold-Mariano test Diebold & Mariano (2002); Diebold (2015).

For the mean price forecast $\widehat{P}_{d,h}$, the predictive distribution $\widehat{\mathcal{D}}^P_{d,h}$ and the realized electricity price $P_{d,h}$, the scoring rules are defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{DH} \sum_{d}^{D} \sum_{h}^{H} \left(\widehat{P}_{d,h} - P_{d,h} \right)^2} \tag{30}$$

$$\text{MAE} = \frac{1}{DH} \sum_{d}^{D} \sum_{h}^{H} \left| \widehat{\mathcal{D}}^{-1}_{P_{d,h}}(0.5) - P_{d,h} \right| \tag{31}$$

where $\widehat{\mathcal{D}}^{-1}_{P_{d,h}}(p)$ is the quantile or percentage point function. For a $(1-\alpha)\times 100\%$-PI defined by the lower and upper bounds $\widehat{L}_{d,s} = \widehat{\mathcal{D}}^{-1}_{P_{d,h}}(\alpha/2)$ and $\widehat{U}_{d,s} = \widehat{\mathcal{D}}^{-1}_{P_{d,h}}(1-\alpha/2)$, the CR and IS are defined as

$$\mathrm{CR}_{\alpha} = \frac{1}{DH}\sum_{d}^{D}\sum_{h}^{H} \mathbf{1}_{\widehat{L}_{d,s}\leq P_{d,s}\leq \widehat{U}_{d,s}} \tag{32}$$

$$\mathrm{IS}_{\alpha} = \frac{1}{DH}\sum_{d}^{D}\sum_{h}^{H} \begin{cases} (\widehat{U}_{d,s} - \widehat{L}_{d,s}) + \frac{\alpha}{2}(\widehat{L}_{d,s} - P_{d,s}) & \text{if } \widehat{L}_{d,s} > P_{d,s}, \\ (\widehat{U}_{d,s} - \widehat{L}_{d,s}) + \frac{\alpha}{2}(P_{d,s} - \widehat{U}_{d,s}) & \text{if } \widehat{U}_{d,s} < P_{d,s}, \\ (\widehat{U}_{d,s} - \widehat{L}_{d,s}) & \text{else.} \end{cases} \tag{33}$$

Let us note that for a skewed distribution and the central prediction intervals, the Interval Score might be misleading since there can be non-central prediction intervals with the same coverage but smaller width (and hence lower IS). The CRPS is approximated by the PS on a dense grid of quantiles

$$\mathrm{PS}_{\alpha,d,s} = \begin{cases} \alpha\left(P_{d,s} - \widehat{\mathcal{D}}^{-1}_{P_{d,h}}(\alpha)\right) & \text{if } P_{d,s} \geq \widehat{\mathcal{D}}^{-1}_{P_{d,h}}(\alpha), \\ (1-\alpha)\left(\widehat{\mathcal{D}}^{-1}_{P_{d,h}}(\alpha) - P_{d,s}\right) & \text{else.} \end{cases} \tag{34}$$

$$\mathrm{CRPS} = \frac{2}{|\mathcal{Q}|}\frac{1}{DH}\sum_{\alpha\in\mathcal{Q}}\sum_{d}^{D}\sum_{h}^{H}\mathrm{PS}_{\alpha,d,s} \tag{35}$$

Furthermore, we report the Log Score (LS) as the negative log-likelihood of the true value under the predictive distribution. The LS is defined as:

$$\mathrm{LS} = \frac{1}{DH}\sum_{d}^{N}\sum_{h}^{H} -\log\left(\widehat{\boldsymbol{d}}_{P_{d,h}}(y_{d,h})\right) \tag{36}$$

where $\widehat{\boldsymbol{d}}_{P_{d,h}}(x)$ is the probability density function of the predictive distribution $\widehat{\mathcal{D}}^{P}_{d,h}$ and observation x. Let us note that the RMSE, MAE, IS, LS and the CRPS[5] are strictly proper scoring rules. For all of them, lower scores correspond to a better forecast. The CR is a measure of calibration for probabilistic forecasts only. Lastly, we evaluate the computation time for all forecasting studies.

Diebold-Mariano Test. We evaluate the statistical significance of the differences in predictive accuracy using the Diebold-Mariano (DM-) test (Diebold & Mariano, 2002; Diebold, 2015). For two models A and B, and the 24-dimensional vectors of scores $\mathcal{S}^{A}_{d} = (S^{A}_{d,0}, S^{A}_{d,1}, ..., S^{A}_{d,H})^{\top}$ and $\mathcal{S}^{B}_{d} = (S^{B}_{d,0}, S^{B}_{d,1}, ..., S^{B}_{d,H})^{\top}$ the DM-test employs the loss differential

$$\boldsymbol{\Delta}^{A,B}_{d} = \|\mathcal{S}^{A}_{d,h}\|_1 - \|\mathcal{S}^{B}_{d,h}\|_1,$$

[5]Furthermore, note that within the EPF community, the first fraction in the formulation of the CRPS is often $1/|\mathcal{Q}|$ instead of $2/|\mathcal{Q}|$. This formulation corresponds to $0.5\times$CRPS and has initially been used this way in the GEFcom 2014 as the Average Pinball Score (Hong et al., 2016; Marcjasz et al., 2023; Nowotarski & Weron, 2018).

Method	Distribution	Estimation	Location	Scale	Tail	Skew	MAE	RMSE	CR50	CR90	IS50	IS90	LS	CRPS	Time (Min)
Batch	Student-t	OLS	Eq. (29)	Eq. (29)	-	-	4.400	7.473	0.708	0.961	15.450	27.022	3.259	3.451	26.296
		LASSO	Eq. (29)	Eq. (29)	-	-	5.078	8.834	**0.510**	0.881	16.961	22.787	3.469	3.850	167.578
	Johnson-S_U	OLS	Eq. (29)	-	-	-	4.529	7.139	0.555	0.916	14.910	21.434	3.340	3.371	458.131
				Eq. (29)	Eq. (29)	Eq. (29)	**4.158**	9.163	0.465	0.850	**13.558**	17.255	3.163	**3.057**	1670.944
	Gaussian $\mathcal{N}$	OLS	Eq. (29)	-	-	-	4.649	6.956	0.784	0.982	17.156	33.637	3.476	3.780	13.177
				Eq. (29)	-	-	4.634	6.949	0.689	0.963	16.087	28.445	3.349	3.578	27.702
		LASSO	Eq. (29)	Eq. (29)	-	-	9.596	14.243	0.589	0.912	31.841	47.327	4.032	7.154	479.687
Online	Student-t	OLS	Eq. (29)	-	-	-	4.262	6.980	0.557	0.920	14.164	21.146	3.165	3.217	1.610
				Eq. (29)	-	-	4.216	7.034	0.533	**0.905**	13.957	20.041	**3.122**	3.153	1.910
					Eq. (29)	-	4.239	7.073	0.591	0.914	15.058	24.145	3.256	3.348	1.904
		LASSO	Eq. (29)	-	-	-	4.257	6.980	0.557	0.921	14.151	21.126	3.164	3.214	2.790
				Eq. (29)	-	-	4.285	7.027	0.550	0.932	14.598	24.986	3.163	3.304	3.266
					Eq. (29)	-	4.296	7.063	0.514	0.914	14.527	21.730	3.183	3.274	10.291
	Johnson-S_U	OLS	Eq. (29)	-	-	-	4.396	7.147	0.561	0.912	14.566	20.888	3.326	3.294	2.170
				Eq. (29)	-	-	4.328	7.556	0.510	0.877	14.290	19.625	3.183	3.227	3.051
					Eq. (29)	-	4.294	9.342	0.455	0.836	14.075	**17.224**	3.190	3.183	3.742
						Eq. (29)	4.388	9.360	0.470	0.850	14.416	18.528	3.160	3.259	4.270
	Gaussian $\mathcal{N}$	OLS	Eq. (29)	-	-	-	4.486	**6.871**	0.654	0.951	15.198	24.958	3.384	3.414	**1.015**
				Eq. (29)	-	-	4.282	6.954	0.625	0.926	19.336	34.899	3.390	4.086	1.237

Table 1: Models and Scores. Combinations of models analysed in the forecasting study. Distribution refers to the assumed parametric form of the response distribution. Setting refers to whether models are estimated incrementally or re-trained on the increasing window training set. The best value in each column is marked **bold**. Note that the timing corresponds to a full forecasting study, i.e. the estimation of $736 \times 24 = 17.664$ models. The colour gradient runs on a logarithmic scale.

where $\|\cdot\|_1$ represents the ℓ_1-norm (see e.g. Nowotarski & Weron, 2018; Berrisch & Ziel, 2024; Lago et al., 2018, 2021). We test the H_0 that $1/D \sum_{d=1}^{D} \mathcal{S}_d^A \leq 1/D \sum_{d=1}^{D} \mathcal{S}_d^B$, i.e. the average score of model A is lower or equal the average score of model B. Rejecting the H_0, therefore, implies that the forecasts of model B are significantly better than the forecasts of model A.

5. Results and Discussion

Table 1 presents the tested models in the different settings, estimation methods and the results of the different scoring rules. Figure 1 gives the p-value of the pairwise Diebold-Mariano test. In our case study, online learning models deliver competitive performance and, to a certain extent, can even outperform rolling batch estimation while reducing the estimation time by some 1-2 orders of magnitude. The following paragraphs discuss some observations in more detail.

Normal Distribution. The normal distribution is common as a simple benchmark and serves this purpose also in this forecasting study. However, in the GAMLSS framework (in the batch and online setting, in the `R` and our `Python` implementation), we have encountered issues with the estimation of models where the specified distribution does not match the empirical distribution of the data (see e.g. Robledo, 2022). Since electricity prices are clearly non-Gaussian, this also applies in this case.

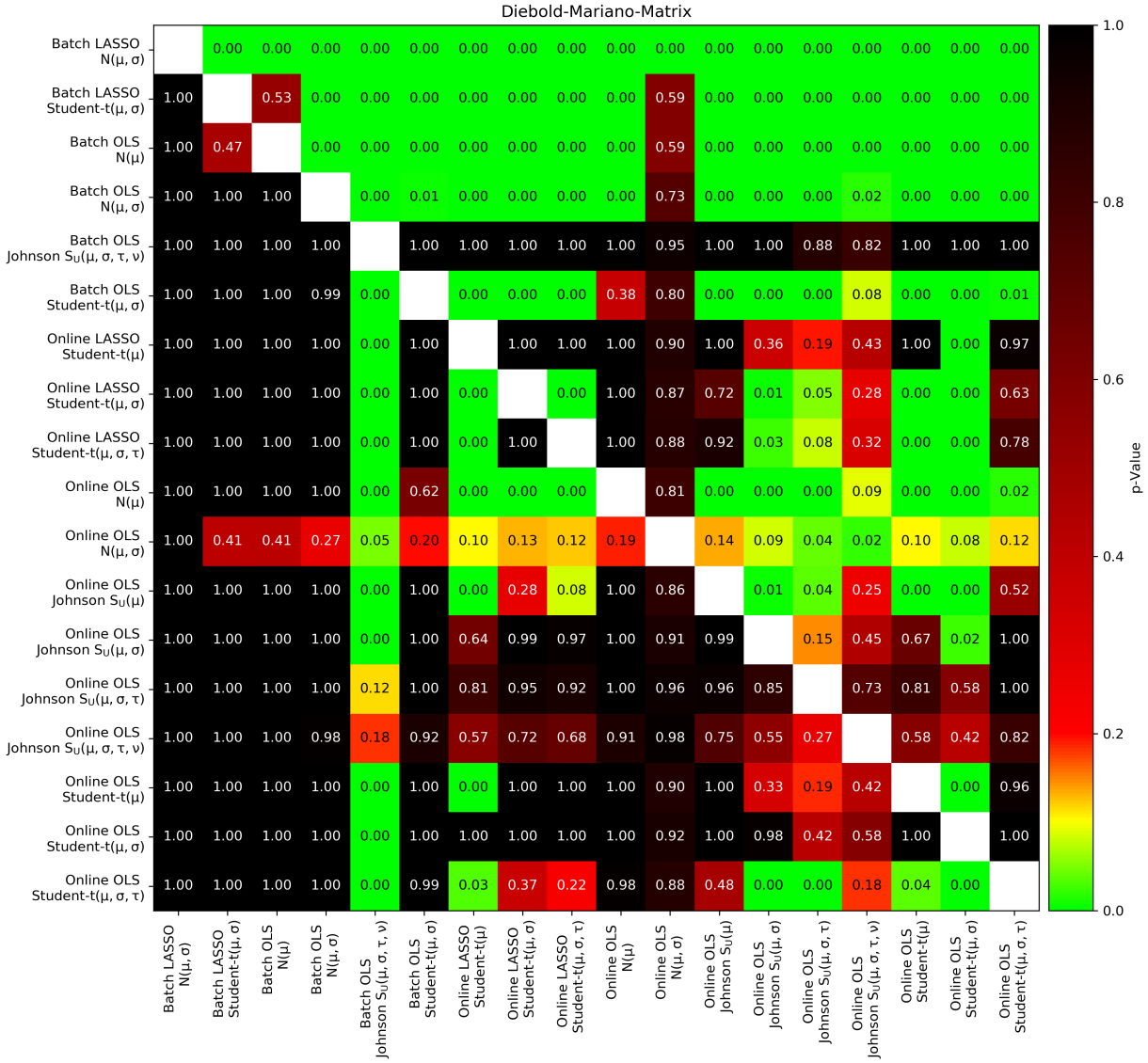

Figure 1: DM-test Results. p-Values of the pairwise Diebold-Mariano test. A p-value of $p < \alpha$, $\alpha = 0.05$ implies that we can reject the H_0 and hence conclude that the model on the column has a significantly better forecasting performance than the model on the row.

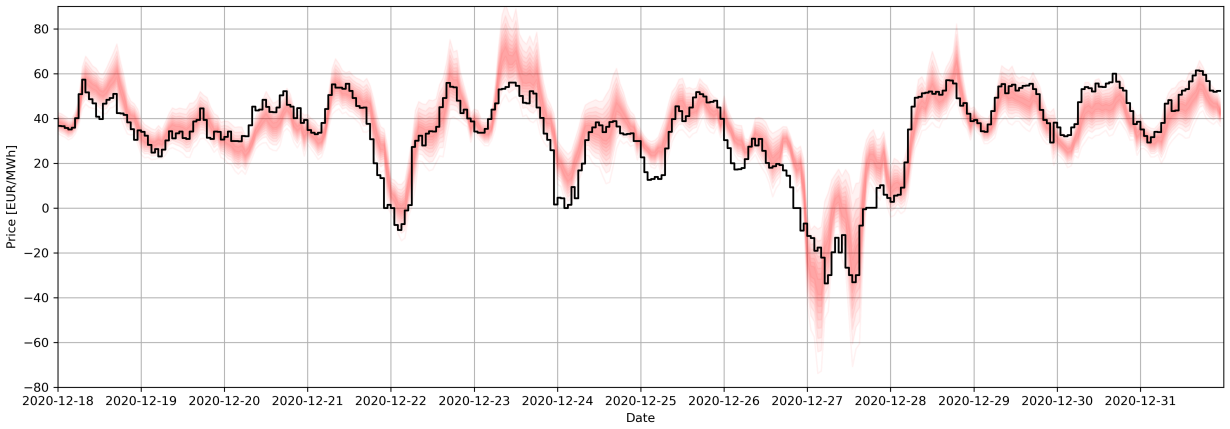

Figure 2: Illustrative Forecast. Prediction Intervals derived from the probabilistic model. Here, we assume the power prices to follow Johnson's S_U distribution and model all moments conditionally on Equation 29. The prediction intervals correspond to the $\{0.05, 0.1, ..., 0.95\}$ quantiles of the predictive distribution. Note that the period with extremely low prices corresponds to Christmas.

LASSO estimation. We note that LASSO estimation does not necessarily improve the performance of the model. This can be seen, e.g. in the direct comparison between the location-scale t model in the online and batch settings. This is in contrast to the literature, where LASSO estimated models typically outperform OLS models (Uniejewski & Weron, 2021; Nowotarski & Weron, 2018). However, these results hold primarily for models in the expected value and little research is done on model selection in the higher moments. Additionally, we have not (yet) performed any tuning with respect to the model selection, but estimated all models taking the BIC.

6. Conclusion

Methodological contribution. This paper presents an efficient, scalable approach to online distributional regression respectively online regression for conditional heteroskedastic data. We implement an incremental algorithm for the well-known, linear-parametric GAMLSS modelling approach, including regularized estimation Rigby & Stasinopoulos (2005); Stasinopoulos & Rigby (2008); Stasinopoulos et al. (2018); Klein (2024) and provide an open-source `Python` implementation of our approach.

Empirical Results. We validate our approach in a forecasting study for the German day-ahead electricity prices, a highly volatile data set, where we see that the online GAMLSS deliver competitive forecasting accuracy in terms of the CRPS while reducing the computation time by some 1-2 orders of magnitude.

Future research. Our research opens up multiple avenues for future research. First, while our empirical results are promising, theoretical results on the regret bounds of the online algorithm would further increase

the trust in the proposed method. Secondly, we present a regularized method for the online estimation, but the (online) model selection in distributional regression is a relatively untapped field, and advances here will directly benefit practical applications. Furthermore, our implementation is constrained to the linear parametric GAMLSS so far. While simple splines can be implemented directly in the design matrix, an interesting practical extension would be the ability to incrementally fit penalized splines and automatically select grid points.

CRediT authorship contribution statement

SH: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization.

JB: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization.

FZ: Conceptualization; Methodology; Validation; Formal analysis; Writing – review and editing; Supervision.

Acknowledgements

Simon Hirsch is employed by Statkraft Trading GmbH and gratefully acknowledges the support and funding received. Simon Hirsch is grateful to Daniel Gruhlke for many helpful discussions. This work contains the author's opinions and does not necessarily reflect Statkraft's position. The authors declare no conflict of interest.

Appendix A. Relationship between GAMLSS and IRLS

The following section sheds light on the relationship between GAMLSS and the IRLS for fitting regression models for (conditional) heteroskedasticity (see e.g. Dette & Wagener, 2013; Ziel, 2016; Ziel et al., 2016).

For the GAMLSS part, we assume the response to be distributed following the mean-variance parameterization of the normal distribution. This corresponds to the `NO2` implementation in the GAMLSS `R` package. Furthermore, the link functions $g_1(\cdot)$ and $g_2(\cdot)$ are the identity functions and hence $\eta_1 = g_1(\theta_1) = \theta_1$ and $\eta_2 = g_2(\theta_2) = \theta_2$ For the distribution parameters $\theta_1 = \mu$ and $\theta_2 = \sigma^2$, derivatives of the likelihood are given by:

$$\frac{\partial l}{\partial \mu} = \frac{1}{\sigma^2}(y-\mu) \qquad \frac{\partial l}{\partial (\sigma^2)} = \frac{1}{2}\frac{(y-\mu)^2 - \sigma^2}{\sigma^4}$$

$$\frac{\partial^2 l}{\partial \mu^2} = -\frac{1}{\sigma^4} \qquad \frac{\partial^2 l}{\partial (\sigma^2)^2} = -\frac{1}{2}\frac{1}{\sigma^4}$$

we therefore have the weights $\boldsymbol{W}_{11} = 1/\sigma^2$ and $\boldsymbol{W}_{11} = 1/(2\sigma^4)$ since $\partial\eta_k/\partial\theta_k = 1$ for the identity link function. The working vector is given by Equation 6 and can be written as:

$$z_1 = z_\mu = \mu + \frac{\frac{1}{\sigma^2}(y-\mu)}{\frac{1}{\sigma^2}} = y$$

$$z_2 = z_\sigma^2 = \sigma^2 + \frac{1}{2}\frac{\frac{(y-\mu)^2 - \sigma^2}{\sigma^4}}{-\left(-\frac{1}{2\sigma^4}\right)} = (y-\mu)^2$$

Therefore, the working vectors reduce to the response variable y and the squared residuals $(y-\mu)^2$, a natural estimator for the variance. The regression for the mean is weighted by the inverse of the estimated variance.

References

Álvarez, V., Mazuelas, S., & Lozano, J. A. (2021). Probabilistic load forecasting based on adaptive online learning. *IEEE Transactions on Power Systems*, *36*, 3668–3680.

Angelosante, D., Bazerque, J. A., & Giannakis, G. B. (2010). Online adaptive estimation of sparse signals: Where rls meets the ℓ_1-norm. *IEEE Transactions on signal Processing*, *58*, 3436–3447.

Berrisch, J., & Ziel, F. (2024). Multivariate probabilistic crps learning with an application to day-ahead electricity prices. *International Journal of Forecasting*, .

Bracher, J., Ray, E. L., Gneiting, T., & Reich, N. G. (2021). Evaluating epidemic forecasts in an interval format. *PLoS computational biology*, *17*, e1008618.

Cesa-Bianchi, N., & Orabona, F. (2021). Online learning algorithms. *Annual review of statistics and its application*, *8*, 165–190.

Cipra, T., & Hendrych, R. (2018). Robust recursive estimation of garch models. *Kybernetika*, *54*, 1138–1155.

Dahlhaus, R., & Subba Rao, S. (2007). A recursive online algorithm for the estimation of time-varying arch parameters. *Bernoulli*, *13*, 389–422.

Dette, H., & Wagener, J. (2013). Least squares estimation in high dimensional sparse heteroscedastic models. In *Robustness and Complex Data Structures: Festschrift in Honour of Ursula Gather* (pp. 135–147). Springer.

Diebold, F. X. (2015). Comparing predictive accuracy, twenty years later: A personal perspective on the use and abuse of diebold–mariano tests. *Journal of Business & Economic Statistics*, *33*, 1–1.

Diebold, F. X., & Mariano, R. S. (2002). Comparing predictive accuracy. *Journal of Business & economic statistics*, *20*, 134–144.

Ding, J., Tarokh, V., & Yang, Y. (2018). Model selection techniques: An overview. *IEEE Signal Processing Magazine*, *35*, 16–34.

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics*, *32*, 407–499.

Friedman, J., Hastie, T., Höfling, H., & Tibshirani, R. (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics*, *1*.

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, *33*, 1.

Gneiting, T. (2008). Probabilistic forecasting.

Gneiting, T. (2011). Making and evaluating point forecasts. *Journal of the American Statistical Association*, *106*, 746–762.

Gneiting, T., & Katzfuss, M. (2014). Probabilistic forecasting. *Annual Review of Statistics and Its Application*, *1*, 125–151.

Groll, A., Hambuckers, J., Kneib, T., & Umlauf, N. (2019). Lasso-type penalization in the framework of generalized additive models for location, scale and shape. *Computational Statistics & Data Analysis*, *140*, 59–73.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Fernández del Río, J., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., & Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*, 357–362. doi:10.1038/s41586-020-2649-2.

Hendrych, R., & Cipra, T. (2018). Self-weighted recursive estimation of garch models. *Communications in Statistics-Simulation and Computation*, *47*, 315–328.

Hofner, B., Mayr, A., & Schmid, M. (2014). gamboostlss: An r package for model building and variable selection in the gamlss framework. *arXiv preprint arXiv:1407.1774*, .

Hong, T., Pinson, P., Fan, S., Zareipour, H., Troccoli, A., & Hyndman, R. J. (2016). Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond.

Klein, N. (2024). Distributional regression for data analysis. *Annual Review of Statistics and Its Application*, *11*.

Kneib, T., Silbersdorff, A., & Säfken, B. (2023). Rage against the mean–a review of distributional regression approaches. *Econometrics and Statistics*, *26*, 99–123.

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.

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. doi:https://doi.org/10.1016/j.apenergy.2021.116983.

Lam, S. K., Pitrou, A., & Seibert, S. (2015). Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC* (pp. 1–6).

Maheshwari, J., Jariwala, R., Pradhan, S., & George, N. V. (2017). Online least angle regression algorithm for sparse system identification. In *2017 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)* (pp. 191–195). IEEE.

Marcjasz, G., Narajewski, M., Weron, R., & Ziel, F. (2023). Distributional neural networks for electricity price forecasting. *Energy Economics*, *125*, 106843.

Mayr, A., Fenske, N., Hofner, B., Kneib, T., & Schmid, M. (2012). Generalized additive models for location, scale and shape for high dimensional data—a flexible approach based on boosting. *Journal of the Royal Statistical Society Series C: Applied Statistics*, *61*, 403–427.

Meier, L., Van De Geer, S., & Bühlmann, P. (2008). The group lasso for logistic regression. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *70*, 53–71.

Messner, J. W., & Pinson, P. (2019). Online adaptive lasso estimation in vector autoregressive models for high dimensional wind power forecasting. *International Journal of Forecasting*, *35*, 1485–1498.

Monti, R. P., Anagnostopoulos, C., & Montana, G. (2018). Adaptive regularization for lasso models in the context of nonstationary data streams. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, *11*, 237–247.

Nowotarski, J., & Weron, R. (2018). Recent advances in electricity price forecasting: A review of probabilistic forecasting. *Renewable and Sustainable Energy Reviews*, *81*, 1548–1568.

Petropoulos, F., Apiletti, D., Assimakopoulos, V., Babai, M. Z., Barrow, D. K., Taieb, S. B., Bergmeir, C., Bessa, R. J., Bijak, J., Boylan, J. E. et al. (2022). Forecasting: theory and practice. *International Journal of Forecasting*, *38*, 705–871.

Priouret, P., Moulines, E., & Roueff, F. (2005). On recursive estimation for time varying autoregressive processes. *Annals of Statistics*, *33*, 2610–2654.

Rigby, R. A., & Stasinopoulos, D. M. (2005). Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society Series C: Applied Statistics*, *54*, 507–554.

Robledo, K. P. (2022). *Statistical algorithms for semi-parametric variance regression with application to biomarkers*. Ph.D. thesis Macquarie University.

Shi, H.-J. M., Tu, S., Xu, Y., & Yin, W. (2016). A primer on coordinate descent algorithms. *arXiv preprint arXiv:1610.00040*, .

Stasinopoulos, D. M., & Rigby, R. A. (2008). Generalized additive models for location scale and shape (gamlss) in r. *Journal of Statistical Software*, *23*, 1–46.

Stasinopoulos, M. D., Kneib, T., Klein, N., Mayr, A., & Heller, G. Z. (2024). *Generalized Additive Models for Location, Scale and Shape: A Distributional Regression Approach, with Applications* volume 56. Cambridge University Press.

Stasinopoulos, M. D., Rigby, R. A., & Bastiani, F. D. (2018). Gamlss: A distributional regression approach. *Statistical Modelling*, *18*, 248–273.

Tay, J. K., Narasimhan, B., & Hastie, T. (2023). Elastic net regularization paths for all generalized linear models. *Journal of statistical software*, *106*.

Uniejewski, B., & Weron, R. (2021). Regularized quantile regression averaging for probabilistic electricity price forecasting. *Energy Economics*, *95*, 105121.

Viehmann, J. (2017). State of the german short-term power market. *Zeitschrift für Energiewirtschaft*, *41*, 87–103.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., & SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. doi:10.1038/s41592-019-0686-2.

Welford, B. (1962). Note on a method for calculating corrected sums of squares and products. *Technometrics*, *4*, 419–420.

Werge, N., & Wintenberger, O. (2022). Adavol: An adaptive recursive volatility prediction method. *Econometrics and Statistics*, *23*, 19–35.

Wright, S. J. (2015). Coordinate descent algorithms. *Mathematical programming*, *151*, 3–34.

Yang, H., Xu, Z., King, I., & Lyu, M. R. (2010). Online learning for group lasso. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (pp. 1191–1198).

Yang, S., Yan, Y., Zhu, X., & Sun, Q. (2023). Online linearized lasso. In *International Conference on Artificial Intelligence and Statistics* (pp. 7594–7610). PMLR.

Zanetta, F., & Allen, S. (2024). Scoringrules: a python library for probabilistic forecast evaluation. URL: https://github.com/frazane/scoringrules.

Ziel, F. (2016). Iteratively reweighted adaptive lasso for conditional heteroscedastic time series with applications to ar–arch type processes. *Computational Statistics & Data Analysis*, *100*, 773–793.

Ziel, F. (2022). M5 competition uncertainty: Overdispersion, distributional forecasting, gamlss, and beyond. *International Journal of Forecasting*, *38*, 1546–1554.

Ziel, F., Croonenbroeck, C., & Ambach, D. (2016). Forecasting wind power–modeling periodic and non-linear effects under conditional heteroscedasticity. *Applied Energy*, *177*, 285–297.

Ziel, F., Muniain, P., & Stasinopoulos, M. (2021). gamlss. lasso: Extra lasso-type additive terms for gamlss. *R package version*, (pp. 1–0).

Ziel, F., & Weron, R. (2018). Day-ahead electricity price forecasting with high-dimensional structures: Univariate vs. multivariate modeling frameworks. *Energy Economics*, *70*, 396–420.