Automated Smoothing Parameter Estimation for Quantile Additive Models:

Bertrand Nortier

A thesis submitted for the degree of
Doctor of Philosophy in Mathematics

University of Bristol
June 20, 2021

ABSTRACT

Quantile Additive Models (QAMs) are statistical models where conditional quantiles of an observed variable are modelled as a sum of functions. These functions are univariate or multivariate smooth functions of a set of covariates. The model is penalized with a quadratic regularization term that is used to control the degree of smoothness of the functions via a vector of hyper-parameters or smoothing parameters. Determining these smoothing parameters is particularly difficult in the case of QAMs. This is especially true when trying to fit a model of extreme quantiles. The more extreme the quantile, the more the dataset used to fit the model is imbalanced. In this thesis, a fully automated method to determine these smoothing parameters and the vector of parameters of the model is proposed. To achieve this goal, several contributions are made. First, a cross validation criterion that addresses some of the shortcomings of previously proposed criteria is introduced: the Quantile Generalized Approximate Cross Validation (QGACV) criterion. Then, it is proposed to replace the exact pinball loss function by its expected value, itself replaced by a rounded surrogate loss. The degree of rounding of the surrogate loss is estimated with a method aimed at minimizing a squared distance between the surrogate loss and the expected loss, which is estimated by an empirical loss. This is followed by a method to optimize both the QGACV criterion based on graduated optimization and Generalized Iteratively Reweighted Least Squares. Last, a method to obtain confidence intervals based on non-parametric bootstrap, together with a method to relax the smoothing parameters to obtain better coverage are introduced. The full implementation of the methodology including automated initialization methods have been implemented in an R package named 'qgacv'. Package 'qgacv' is then compared to existing R packages that can be used to fit QAMs. The method presented is competitive and has several advantages compared to the existing methods and R packages.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Professor Simon N. Wood for his guidance, my examiners Professor Roger Koenker and Doctor Song Liu for their comments as well as Doctor Matteo Fasiolo, Professor Stéphane Chrétien and Doctor Neil Oxtoby for comments and help. Part of this research was carried out during a PhD enrichment scheme at the Alan Turing Institute (ATI). I would like to thank the ATI for research support.

AUTHOR'S DECLARATION

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED:..... DATE:.....

CONTENTS

List of Symbols

$\#$ cardinal of a set, page 43

$\tilde{a}_i$, $\tilde{a}_{\alpha,i}$ diagonal elements of the hat matrix $\tilde{\mathbf{A}} = \tilde{\mathbf{A}}_\alpha$. We also have $\tilde{a}_{\alpha,i} = \frac{\partial \tilde{f}_{\alpha,i}}{\partial y_i}$. Note that by definition, $\tilde{a}_{\alpha,i} = a_{(\alpha_1=\alpha,\alpha_2=\alpha,i)}$, page 15

$a_{\alpha_1,\alpha_2,i}$ derivative of the predicted value (with the surrogate loss rounded using $\alpha = \alpha_1$) with respect to y_i. The derivative is calculated away from the minimum of the penalized empirical loss. It is evaluated at the predicted value resulting from the optimum of the penalized empirical loss using the surrogate loss rounded at α_2. $a_{\alpha_1,\alpha_2,i} = \left(\frac{\partial \tilde{f}_{\alpha_1,i}}{\partial y_i}\right)_{|\tilde{f}_{\alpha_1,i}=\tilde{f}_{\alpha_2,i}}$. The $a_{\alpha_1,\alpha_2,i}$ are the diagonal elements of matrix $\mathbf{A}_{\boldsymbol{\alpha_1},\boldsymbol{\alpha_2}}$, page 36

$a_{\alpha_1,0,i}$ derivative of the predicted value (with the surrogate loss rounded using $\alpha = \alpha_1$) with respect to y_i. The derivative is calculated away from the minimum of the penalized empirical loss. It is calculated at the predicted value at the minimum of the penalized empirical loss based on the exact pinball loss. $a_{\alpha_1,0,i} = \lim_{\alpha\to 0} a_{\alpha_1,\alpha,i} = \left(\frac{\partial \tilde{f}_{\alpha_1,i}}{\partial y_i}\right)_{|\tilde{f}_{\alpha_1,i}=\lim_{\alpha\to 0}\tilde{f}_{\alpha,i}} = \left(\frac{\partial \tilde{f}_{\alpha_1,i}}{\partial y_i}\right)_{|\tilde{f}_{\alpha_1,i}=\hat{f}_i}$. The $a_{\alpha_1,0,i}$ are the diagonal elements of matrix $\mathbf{A}_{\boldsymbol{\alpha_1},\mathbf{0}}$, page 36

a_i short hand notation a_i designates the sensitivity of the predicted value to y_i obtained with a penalized empirical loss using the surrogate loss with a rounding of $\hat{\alpha}$ (Formula 3.15) calculated at the predicted value estimated with the penalized empirical loss with the exact pinball loss $\hat{f}_i$. $a_i = a_{\hat{\alpha},0,i} = \lim_{\alpha\to 0} a_{\hat{\alpha},\alpha,i} = \left(\frac{\partial \tilde{f}_{\hat{\alpha},i}}{\partial y_i}\right)_{|\tilde{f}_{\hat{\alpha},i}=\lim_{\alpha\to 0}\tilde{f}_{\alpha,i}} = \left(\frac{\partial \tilde{f}_{\hat{\alpha},i}}{\partial y_i}\right)_{|\tilde{f}_{\hat{\alpha},i}=\hat{f}_i}$. The a_i are the diagonal elements of matrix $\mathbf{A} = \mathbf{A}_{\hat{\alpha},\mathbf{0}}$, page 36

$\hat{a}_{PL,i}$ sensitivity of the predicted value calculated with the exact pinball loss with respect to y_i: $\hat{a}_{PL,i} = \frac{\partial \hat{f}_i}{\partial y_i}$, page 61

$\mathcal{A}$ the set of datapoints that are above the model. For the exact pinball loss, $\mathcal{A} = \left\{i \in 1,...,n; y_i > \hat{f}_i\right\}$. For the surrogate loss, $\mathcal{A} = \left\{i \in 1,...,n; y_i > \tilde{f}_{\alpha,i}\right\}$, page 24

α tuning parameter that determines the degree of rounding of the surrogate loss $\mathcal{L}_{\alpha,\tau}(u) = \tau(u+\gamma) + \alpha \log\left(1+\exp\left(-\frac{u+\gamma}{\alpha}\right)\right)$, page 16

$\hat{\alpha}$ value of the tuning parameter of the surrogate loss α used to calculate the hat matrix. This hat matrix is then used to calculate the effective degrees of freedom of the model. $\hat{\alpha}$ is calculated using formula 3.15, page 32

α_{start} value of the tuning parameter for the surrogate loss $\mathcal{L}_{\alpha,\tau}(u)$, that is a first estimate of an starting value to run a Generalized IRLS algorithm to estimate the vector of parameters $\boldsymbol{\beta}$ (see section 7.2), page 105

α_{end} value of the tuning parameter for the surrogate loss $\mathcal{L}_{\alpha,\tau}(u)$, that is a first estimate of an stopping value to run a Generalized IRLS algorithm to estimate the vector of parameters $\boldsymbol{\beta}$ (see section 7.2), page 105

α_U value of the tuning parameter α for the surrogate loss $\mathcal{L}_{\alpha,\tau}(u)$, that is a stopping value for the optimization of the QGACV (see section 7.6), page 112

α_L value of the tuning parameter α for the surrogate loss $\mathcal{L}_{\alpha,\tau}(u)$, that is a stopping value for the Generalized IRLS (see section 7.6), page 112

$\alpha_{\alpha Stable}$ parameter of the alpha Stable distribution, page 21

$\tilde{\mathbf{A}}$,$\tilde{\mathbf{A}}_\alpha$ hat matrix defined as $\tilde{\mathbf{A}} = \mathbf{X}(\mathbf{X}^T\tilde{\mathbf{W}}\mathbf{X} + \mathbf{S}_{\boldsymbol{\lambda}})^{-1}(\mathbf{X}^T\tilde{\mathbf{W}})$. Alternative notations for matrix $\tilde{\mathbf{A}}$ are $\tilde{\mathbf{A}}_\alpha$ to indicate that matrix $\tilde{\mathbf{A}}$ is calculated with the surrogate loss with rounding α, page 15

$\mathbf{A}_{\alpha_1,\alpha_2}$ hat matrix defined as $\mathbf{A}_{\alpha_1,\alpha_2} = \mathbf{X}\left(\mathbf{X}^T\mathbf{W}_{\alpha_1,\alpha_2}\mathbf{X} + \mathbf{S}_{\boldsymbol{\lambda}}\right)^{-1}\left(\mathbf{X}^T\mathbf{W}_{\alpha_1,\alpha_2}\right)$. The diagonal elements of $\mathbf{A}_{\alpha_1,\alpha_2}$ are $a_{\alpha_1,\alpha_2,i}$, page 37

$\mathbf{A}_{\alpha_1,0}$ hat matrix defined as $\mathbf{A}_{\alpha_1,0} = \lim_{\alpha\to 0}\mathbf{A}_{\alpha_1,\alpha} = \lim_{\alpha\to 0}\mathbf{X}\left(\mathbf{X}^T\mathbf{W}_{\alpha_1,\alpha}\mathbf{X} + \mathbf{S}_{\boldsymbol{\lambda}}\right)^{-1}\left(\mathbf{X}^T\mathbf{W}_{\alpha_1,\alpha}\right)$. The diagonal elements of $\mathbf{A}_{\alpha_1,0}$ are $a_{\alpha_1,0,i}$, page 37

$\mathbf{A}$ short hand notation for $\mathbf{A}_{\hat{\alpha},0}$: $\mathbf{A} = \mathbf{A}_{\hat{\alpha},0} = \lim_{\alpha\to 0}\mathbf{A}_{\hat{\alpha},\alpha} = \lim_{\alpha\to 0}\mathbf{X}\left(\mathbf{X}^T\mathbf{W}_{\hat{\alpha},\alpha}\mathbf{X} + \mathbf{S}_{\boldsymbol{\lambda}}\right)^{-1}\left(\mathbf{X}^T\mathbf{W}_{\hat{\alpha},\alpha}\right)$, page 37

B number of non-parametric boostrap samples for smoothing parameter relaxation, page 95

B_0 number of non-parametric bootstrap sample to estimate the confidence interval, page 90

$\mathcal{B}$ the set of datapoints that are below the model. For the exact pinball loss, $\mathcal{B} = \left\{i \in 1,...,n; y_i < \hat{f}_i\right\}$. For the surrogate loss, $\mathcal{B} = \left\{i \in 1,...,n; y_i < \tilde{f}_{\alpha,i}\right\}$, page 24

β_{00} intercept of the model, page 5

$\boldsymbol{\beta}_{01}$ vector of coefficients of the linear part of the model, page 5

$\boldsymbol{\beta}_{02}$ vector of coefficients for the smooth components before transformation to take into account identifiability constraints, page 8

$\boldsymbol{\beta}_0$ vector of parameters of the model before identifiability constraint transformation. This vector is composed of an intercept β_{00}, a vector of parameters for the linear part of the model $\boldsymbol{\beta}_{01}$ and a vector of parameters for the smooths of the model $\boldsymbol{\beta}_{02}$: $\boldsymbol{\beta}_0 = \left(\beta_{00}, \boldsymbol{\beta}_{01}^T, \boldsymbol{\beta}_{02}^T\right)^T$. The coefficients are assumed to be the coefficients of the model matrix $\mathbf{X}_0$ before transformation to take into account indentifiability constraints, page 12

$\hat{\boldsymbol{\beta}}_0$ estimate of $\boldsymbol{\beta}_0$, page 12

$\boldsymbol{\beta}$ vector of parameters of the model after model matrix $\mathbf{X}_0$ and penalty $\mathbf{S}_{\boldsymbol{\lambda}}^0$ were transformed (to $\mathbf{X}$ and $\mathbf{S}_{\boldsymbol{\lambda}}$) to take into account identifiability constraint, page 13

$\hat{\boldsymbol{\beta}}$ estimate of $\boldsymbol{\beta}$, page 13

$\beta_{\alpha Stable}$ parameter of the alpha Stable distribution, page 21

$\hat{\boldsymbol{\beta}}^{*b}$ vector of parameters estimated using the dataset resampled with replacement $\mathcal{X}^{*b}$ with $b = 1, ..., B_0$ for CI or $b = 1, ..., B$ for smoothing parameter relaxation, page 90

$\overline{\beta}^*$ mean of the non-parametric bootstrap estimated vector of parameters $\overline{\boldsymbol{\beta}}^* = \frac{1}{B}\sum_{b=1}^{B} \hat{\boldsymbol{\beta}}^{*b}$, page 91

$\hat{\boldsymbol{\beta}}_{E,\boldsymbol{\lambda}}$ penalized empirical loss where the pinball loss function is replaced by its expectation: $\hat{\boldsymbol{\beta}}_{E,\boldsymbol{\lambda}} \in \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} E_{y_i|\underline{\mathbf{x}}_i}\left[\mathcal{L}_\tau\left(y_i - \mathbf{x}_i^T\boldsymbol{\beta}\right)\right] + \frac{1}{2}\boldsymbol{\beta}^T S_{\boldsymbol{\lambda}}\boldsymbol{\beta}$, page 28

$\tilde{\boldsymbol{\beta}}$, $\tilde{\boldsymbol{\beta}}_\alpha$ estimate of the vector of parameters with surrogate loss rounded with parameter α, page 15

c constant used in the QGACV formula: $c = (1 + 2\phi)$, page 44

$\delta_{\alpha Stable}$ parameter of the alpha Stable distribution, page 21

$\grave{\mathbf{d}}$ vector subtracted from $\hat{\boldsymbol{\rho}}$ to obtain $\grave{\boldsymbol{\rho}}$, the relaxed vector of smoothing parameters, page 96

$\frac{\partial h(x)}{\partial x}$ partial derivative of a given differentiable function h with respect to x, page 195

$\frac{dh(x)}{dx}$ total derivative of a given differentiable function h with respect to x, page 195

$\partial_x h(x)$ subdifferential of lower semi-continuous convex function h with respect to x, page 55

$\mathcal{F}$ space of functions: domain of function f, page 5

f function modelling the conditional response. $f(\underline{\boldsymbol{x}})$ can be decomposed in an intercept β_{00}, a linear part $\underline{\boldsymbol{\pi}}^T\boldsymbol{\beta}_{01}$ and a smooth part $g(\underline{\boldsymbol{z}})$, page 5

$\boldsymbol{f}$ the vector of predicted values: $\boldsymbol{f}(\underline{\mathbf{X}}) = \mathbf{X}\boldsymbol{\beta} = (f(\underline{\boldsymbol{x}}_1), \ldots, f(\underline{\boldsymbol{x}}_n))^T$, page 5

$\hat{f}_{E,\boldsymbol{\lambda}}(\underline{\boldsymbol{x}}_i)$ predicted value calculated using the vector of parameter estimated with the penalized expected empirical loss: $\hat{f}_{E,\boldsymbol{\lambda}}(\underline{\boldsymbol{x}}_i) = \mathbf{x}_i^T\hat{\boldsymbol{\beta}}_{E,\boldsymbol{\lambda}}$, page 28

$\hat{f}$ estimate of the function modelling the conditional quantile of the response. The predicted value is $\hat{f}_i = \hat{f}(\underline{\boldsymbol{x}}_i)$, page 5

$\hat{f}_{\boldsymbol{\lambda}}$ alternative notation for $\hat{f}$. We use this notation to accentuate the fact that $\hat{f}$ depends on $\boldsymbol{\lambda}$. The predicted value is $\hat{f}_{\boldsymbol{\lambda}}(\underline{\boldsymbol{x}}_i) = \hat{f}_{\boldsymbol{\lambda},i}$, page 14

$\hat{f}_i^{[-i]}$ leave one out predicted value: $\hat{f}_i^{[-i]} = \mathbf{x}_i^T\hat{\boldsymbol{\beta}}^{[-i]}$,
where $\hat{\boldsymbol{\beta}}^{[-i]} \in \underset{\beta}{\operatorname{argmin}} \frac{1}{n}\sum_{j=1,j\neq i}^{n} \mathcal{L}_\tau\left(y_j - \mathbf{x}_j^T\boldsymbol{\beta}\right) + \mathcal{P}_{\boldsymbol{\lambda}}(\boldsymbol{\beta})$.
Alternative Notation: $\hat{f}_{\boldsymbol{\lambda},i}^{[-i]}$, page 14

f_i alternative notation for $f(\underline{\boldsymbol{x}}_i)$. $f_i = f(\underline{\boldsymbol{x}}_i)$, page 5

$\hat{f}_i^{*b}$ $\hat{f}_i^{*b} = \mathbf{x}_i^T\hat{\boldsymbol{\beta}}^{*b}$, page 90

$\tilde{\mathbf{F}}_{\hat{\alpha},\alpha}$ As discussed in Wood [2017], section 6.1.2, pp.251-252, matrix calculating the amount of shrinkage of the coefficients.
$\tilde{\mathbf{F}}_{\hat{\alpha},\alpha} = \left(\mathbf{X}^T\tilde{\mathbf{W}}_{\hat{\alpha},\alpha}\mathbf{X} + \mathbf{S}_{\boldsymbol{\lambda}}\right)^{-1}\left(\mathbf{X}^T\tilde{\mathbf{W}}_{\hat{\alpha},\alpha}\mathbf{X}\right)$. This matrix can be used as a more computationally efficient version as hat matrix $\mathbf{A}_{\hat{\alpha},\alpha}$ as we have: $tr(\mathbf{A}_{\hat{\alpha},\alpha}) = tr(\mathbf{F}_{\hat{\alpha},\alpha})$, page 196

$\tilde{f}_i$,$\tilde{f}_{\alpha,i}$ fitted value with the vector of parameters $\boldsymbol{\beta}$ at value $\boldsymbol{\beta} = \tilde{\boldsymbol{\beta}}$, the estimed vector of parameters with the surrogate loss $\mathcal{L}_{\alpha,\tau}$. We have: $\tilde{f}_i = \mathbf{x}_i^T\tilde{\boldsymbol{\beta}}$. We use also the alternative notation $\tilde{f}_{\alpha,i}$ to indicate that the function is parametrized by parameter α, page 15

g smooth part of function f. f is decomposed in $f(\underline{\boldsymbol{x}}_i) = \beta_{00} + \underline{\boldsymbol{\pi}}_i^T\beta_{01} + g(\underline{\boldsymbol{z}}_i)$ were β_{00} is an intercept, $\underline{\boldsymbol{\pi}}$ is a sub-vector of predictors that have a linear relationship with the response, $\underline{\boldsymbol{z}}$ is a sub-vector of predictors that have a smooth relationship with the response, page 5

γ Parameter used to center the loss function: $\gamma = \alpha\log\left(\frac{1-\tau}{\tau}\right)$, page 16

$\gamma_{\alpha Stable}$ parameter of the alpha Stable distribution, page 21

$\mathcal{G}_{\alpha_1,\alpha_2}^{Pen}$ gradient of the penalized empirical loss $l_{\alpha_1,\alpha_2}^{Pen}$, page 195

$\mathcal{G}^{UnPen}_{\alpha_1,\alpha_2}$ gradient of the unpenalized empirical loss $l^{UnPen}_{\alpha_1,\alpha_2}$, page 195

$\mathcal{H}^{Pen}_{\alpha_1,\alpha_2}$ Hessian of the penalized empirical loss $l^{Pen}_{\alpha_1,\alpha_2}$, page 195

$\mathcal{H}^{UnPen}_{\alpha_1,\alpha_2}$ Hessian of the unpenalized empirical loss $l^{Pen}_{\alpha_1,\alpha_2}$, page 195

I Indicator function: $I(condition) = 1$ if condition is true, 0 otherwise, page 5

$\mathcal{I}$ the set of datapoints that are interpolating the model. For the exact pinball loss, $\mathcal{I} = \left\{ i \in 1, ..., n; y_i = \hat{f}_i \right\}$. For the surrogate loss, $\mathcal{I} = \emptyset$, page 24

K_{Gauss} Gaussian Kernel: $K_{Gauss}(u) = \frac{\exp\left(-\frac{u^2}{2}\right)}{\sqrt{2\pi}}$, page 184

K_{Log} Logistic Kernel: $K_{Logistic}(u) = \frac{\exp(-u)}{(1+\exp(-u))^2}$, page 16

K_{SLog} Shifted Logistic Kernel: $K_{SLog,\tau}(u) = \frac{\left(\frac{1-\tau}{\tau}\right)\exp(-u)}{\left(1+\left(\frac{1-\tau}{\tau}\right)\exp(-u)\right)^2}$, page 16

K_{Sig} Sigmoid Kernel: $K_{Sig}(u) = \frac{1}{2\pi}\frac{1}{\exp(-u)+\exp(u)}$, page 184

$\boldsymbol{\lambda}$ vector of smoothing parameters. In this document, we interchangeably use $\boldsymbol{\lambda}$ or $\boldsymbol{\rho}$ to designate the vector of smoothing parameters with $\boldsymbol{\rho} = \log(\boldsymbol{\lambda}) \Leftrightarrow \boldsymbol{\lambda} = \exp(\boldsymbol{\rho})$, page 7

$\grave{\boldsymbol{\lambda}}$ relaxed vector of smoothing parameters, page 96

$\hat{\boldsymbol{\lambda}}$ the "optimal" vector of smoothing parameters, i.e. the vector $\boldsymbol{\lambda}$ that leads to the minimum of the QGACV function. Note that we use interchangeably $\hat{\boldsymbol{\lambda}}$ and $\hat{\boldsymbol{\rho}}$, the vector of log-smoothing parameters with $\hat{\boldsymbol{\rho}} = \log\hat{\boldsymbol{\lambda}}$, page 81

$\mathcal{L}_{\alpha,\tau}$ variant of the surrogate loss of Zheng [2011] used in this document $\mathcal{L}_{\alpha,\tau}(y_i - f_i) = \tau(y_i - f_i + \gamma) + \alpha\log\left(1 + \exp\left(-\frac{y_i - f_i + \gamma}{\alpha}\right)\right)$, where $\gamma = \alpha\log\left(\frac{1-\tau}{\tau}\right)$, page 16

$\mathcal{L}_\tau$ pinball loss function $\mathcal{L}_\tau(u) = u \times (\tau - I(u < 0))$, where I is the indicator function, page 5

$l^{Pen}_{\alpha_1,\alpha_2}$ penalized empirical loss: $l^{Pen}_{\alpha_1,\alpha_2} = \frac{1}{n}\sum_{i=1}^n \mathcal{L}_{\alpha_1,\tau}(u_{\alpha_2,i}) + \frac{1}{2}\boldsymbol{\beta}^T\left(\sum_{k=1}^q e^{\rho_k}\mathbf{S}_k\right)\boldsymbol{\beta}$, page 88

$l^{UnPen}_{\alpha_1,\alpha_2}$ unpenalized empirical loss $l^{UnPen}_{\alpha_1,\alpha_2} = \frac{1}{n}\sum_{i=1}^n \mathcal{L}_{\alpha_1,\tau}(\tilde{u}_{\alpha_2,i})$, page 196

m number of smooths/dimension of the vector of smoothing parameters $\boldsymbol{\lambda}$, page 7

n number of observations, page 5

$n.eff$ "effective" number of datapoints used in the QGACV formula: $n.eff = 2n\phi$, where $\phi = \min(\tau, 1-\tau)$, page 44

o order of splines, page 8

$\mathbb{P}$ probability of an event, page 5

p Number of columns of the model matrix $\mathbf{X}$. The size of $\mathbf{X}$ is then $n \times p$, page 13

ϕ min between τ and $1-\tau$ used in the QGACV formula: $\phi = \min(\tau, 1-\tau)$, page 44

$\underline{\boldsymbol{\pi}}_i$ sub-vector of the vector predictors $\underline{\boldsymbol{x}}_i$ that have a linear relationship with the response. $\underline{\boldsymbol{x}}_i = \left(\underline{\boldsymbol{\pi}}_i^T, \underline{\boldsymbol{z}}_i^T\right)^T$, page 5

$\mathcal{P}(\boldsymbol{\lambda})$ penalty parametrized by the vector of smoothing parameters $\boldsymbol{\lambda}$, page 7

ψ_q^o basis function where o is the order of the spline and q the number of the basis function, q=1,...,Q, page 8

Q number of coefficients of a single univariate smooth, page 8

Q_j number of basis functions for smooth j, $j = 1, ..., m$, page 9

$QGACV$ Quantile Generalized Approximate Cross Validation criterion: $QGACV = \sum_{i=1}^{n} \frac{\mathcal{L}_\tau(y_i - f_i)}{n.eff - cTr(\mathbf{A})}$, page 44

$\boldsymbol{\rho}$ vector of log-smoothing parameters. In this document, we interchangeably use $\boldsymbol{\lambda}$ or $\boldsymbol{\rho}$ to designate the vector of smoothing parameters with $\boldsymbol{\rho} = \log(\boldsymbol{\lambda}) \Leftrightarrow \boldsymbol{\lambda} = \exp(\boldsymbol{\rho})$, page 88

$\grave{\boldsymbol{\rho}}$ relaxed vector of log-smoothing parameters, page 96

$\hat{\boldsymbol{\rho}}$ the "optimal" vector of log-smoothing parameters, i.e. the vector $\boldsymbol{\rho}$ that leads to the minimum of the QGACV function. Note that we use interchangeably $\hat{\boldsymbol{\rho}}$ and $\hat{\boldsymbol{\lambda}}$, the vector of smoothing parameters with $\hat{\boldsymbol{\lambda}} = \exp(\hat{\boldsymbol{\rho}})$, page 95

$\mathbf{S}_{\boldsymbol{\lambda}}^g$ block diagonal penalty matrix parametrized by the vector of smoothing parameters $\boldsymbol{\lambda}$ made only of the smooth elements of the model. We also have $\mathbf{S}_{\boldsymbol{\lambda}}^g = \begin{bmatrix} (\lambda_1 S_1) & 0 & \cdots & 0 \\ 0 & (\lambda_2 S_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & (\lambda_m S_m) \end{bmatrix}$ where the $S_1, ..., S_m$ are matrices, page 11

$\mathbf{S}^0_{\boldsymbol{\lambda}}$ Extended block diagonal penalty matrix adapted to the extended model matrix that comprises the intercept, the linear part and the smooth part before transformation to take into account identifiability constraints: $\mathbf{S}^0_{\boldsymbol{\lambda}} = \begin{bmatrix} 0 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & 0 \\ 0 & \dots & 0 & \mathbf{S}^g_{\boldsymbol{\lambda}} \end{bmatrix}$, page 13

$\mathbf{S}_{\boldsymbol{\lambda}}$ Matrix $\mathbf{S}^0_{\boldsymbol{\lambda}}$ transformed to take into account identifiability constraints., page 13

s.t. such that, page 7

τ quantile parameter of the loss function (see loss function), page 5

$\mathbf{u}$ vector of errors $u_i = y_i - f_i$, $i = 1, ..., n$, page 5

u_i error: $u_i = y_i - f_i$, page 16

$\hat{u}_i$ error when the predicted value is calculated using the exact pinball loss $\mathcal{L}_\tau$: $\hat{u}_i = y_i - \hat{f}_i$, page 16

$\tilde{u}_i, \tilde{u}_{\alpha,i}$ error when the predicted value is calculated using the surrogate loss with rounding α: $\mathcal{L}_{\alpha,\tau}$ $\tilde{u}_{\alpha,i} = y_i - \tilde{f}_{\alpha,i}$. To simplify the notation, we may also remove the α index: $\tilde{u}_i = y_i - \tilde{f}_i$, page 16

$\boldsymbol{v}$ square root of the diagonal of the precision matrix of the QGACV used to obtain the $\grave{\mathbf{d}}$ with $\grave{\mathbf{d}} = -\grave{\zeta}\sqrt{diag\left(\left(\mathcal{H}^{QGACV}_{\hat{\boldsymbol{\rho}}}\right)^{-1}\right)} = -\grave{\zeta}\boldsymbol{v}$, page 96

$V_0(\boldsymbol{\lambda})$ leave one out cross validation criterion. $V_0(\boldsymbol{\lambda}) = \frac{1}{n}\sum_{i=1}^n \mathcal{L}_\tau\left(y_i - \hat{f}^{[-i]}_{\boldsymbol{\lambda},i}\right)$, page 14

$V_1(\boldsymbol{\lambda})$ Approximate cross validation criterion as derived in Nychka et al. [1995]: $V_1 = \frac{1}{n}\sum_{i=1}^n \mathcal{L}_{\alpha,\tau}\left(\frac{y_i - \tilde{f}_{\alpha,i}}{1-\tilde{a}_i}\right)$. To calculate this criterion, we use surrogate loss 3.7 instead of the surrogate loss used in Nychka et al. [1995]., page 15

$V_2(\boldsymbol{\lambda})$ Generalized Approximate cross validation criterion as derived in Yuan [2006]: $V_2 = \sum_{i=1}^n \frac{\mathcal{L}_{\alpha,\tau}\left(y_i - \tilde{f}_{\alpha,i}\right)}{n - Tr\left(\tilde{\mathbf{A}}_\alpha\right)}$. To calculate this criterion, we use surrogate loss 3.7 instead of the surrogate loss used in Yuan [2006] or Nychka et al. [1995], page 21

$\breve{V}_2(\boldsymbol{\lambda})$ GACV similar to Yuan [2006] without the approximation $\mathcal{L}_{\alpha,\tau}\left(\frac{y_i - \tilde{f}_{\alpha,i}}{1 - \frac{Tr(\tilde{\mathbf{A}})}{n}}\right) \approx \frac{\mathcal{L}_{\alpha,\tau}\left(y_i - \tilde{f}_{\alpha,i}\right)}{1 - \frac{Tr(\tilde{\mathbf{A}})}{n}}$. We have: $\breve{V}_2 = \frac{1}{n}\sum_{i=1}^n \mathcal{L}_{\alpha,\tau}\left(\frac{y_i - \tilde{f}_{\alpha,i}}{1 - \frac{Tr(\tilde{\mathbf{A}})}{n}}\right)$, page 65

$V_3(\boldsymbol{\lambda})$ Criterion $V_3(\boldsymbol{\lambda})$ is the QGACV, page 44

$V_4(\boldsymbol{\lambda})$ Smooth loss GACV (SGACV) criterion. This criterion is similar to the GACV derived in Yuan [2006] but the loss function is replaced by $\mathcal{L}_{\alpha,\tau}$ (equation 3.7) and we use a high level of rounding $\hat{\alpha}$, page 74

$V_{0,E}(\boldsymbol{\lambda})$ LOOCV where the difference between the predicted value with and without removing observation i is estimated using the expected loss. $V_{0,E}(\boldsymbol{\lambda}) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_\tau\left(y_i - \hat{f}_{\boldsymbol{\lambda}}(\underline{\boldsymbol{x}}_i) + \left(\hat{f}_{E,\boldsymbol{\lambda}}(\underline{\boldsymbol{x}}_i) - \hat{f}^{[-i]}_{E,\boldsymbol{\lambda}}(\underline{\boldsymbol{x}}_i)\right)\right)$, page 28

$\tilde{V}_0(\boldsymbol{\lambda})$ LOOCV criterion where the difference between the predicted value with and without removing observation i is calculated with the surrogate loss $\mathcal{L}_{\hat{\alpha},\tau}$, i.e. with a rounding $\hat{\alpha}$, page 33

$\tilde{V}_1(\boldsymbol{\lambda})$ ACV criterion derived in this document where the exact pinball loss is used for all part of the criterion apart from the calculation of the trace of the hat matrix that uses a rounding $\hat{\alpha}$ and is evaluated at the predicted value calculated at the minimum of the penalized empirical exact pinball loss: $\tilde{V}_1 = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_\tau\left(\frac{y_i-\hat{f}_i}{1-a_i}\right) = \frac{1}{n}\sum_{i=1}^{n}\frac{\mathcal{L}_\tau\left(y_i-\hat{f}_i\right)}{1-a_i}$, page 38

$\tilde{V}_2(\boldsymbol{\lambda})$ GACV criterion we could derive by averaging the weights of ACV criterion $\tilde{V}_1(\boldsymbol{\lambda})$: $\tilde{V}_2 = \sum_{i=1}^{n}\frac{\mathcal{L}_\tau\left(y_i-\hat{f}_i\right)}{n-tr(\mathbf{A})}$, page 39

$\tilde{\tilde{V}}_0(\boldsymbol{\lambda})$ LOOCV criterion where the difference between the predicted value with and without removing observation i is calculated with the surrogate loss $\mathcal{L}_{\hat{\alpha},\tau}$, but calculated at with a predicted value estimated at the minimum of the exact pinball loss: $\tilde{\tilde{V}}_0 = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_\tau\left(y_i - \hat{f}_i + \left(\tilde{f}_{\hat{\alpha},i} - \tilde{f}^{[-i]}_{\hat{\alpha},i}\right)|_{\tilde{f}_{\hat{\alpha},i}=\hat{f}_i}\right)$, page 33

$\tilde{\tilde{V}}_1(\boldsymbol{\lambda})$ ACV criterion $\tilde{V}_1(\boldsymbol{\lambda})$ after subsampling on one side of the quantile, page 43

$\tilde{\tilde{V}}_2(\boldsymbol{\lambda})$ GACV criterion obtained after averaging the weights of ACV criterion with subsampling $\tilde{\tilde{V}}_2$, page 44

$\overset{\bullet}{V}_0(\boldsymbol{\lambda})$ LOOCV where all parts are calculated with the surrogate loss rounded at $\hat{\alpha}$, page 74

$\overset{\bullet}{V}_1(\boldsymbol{\lambda})$ ACV criterion from Nychka et al. [1995] where the loss function is replaced by loss function of equation 3.7 and we use a high level of rounding $\hat{\alpha}$, page 74

$\tilde{\mathbf{W}}$, $\tilde{\mathbf{W}}_\alpha$ $\tilde{\mathbf{W}}$ is a diagonal weight matrix. The diagonal elements of the matrix are the second derivative of the loss function. An alternative is $\tilde{\mathbf{W}}_\alpha$ where α is the rounding of the surrogate loss used to calculate the matrix, page 15

$\mathbf{W}_{\alpha_1,\alpha_2}$ diagonal weight matrix. The diagonal elements of the matrix are the second derivative of the loss function. α_1 is the rounding for the loss function used to calculate the second derivative of the surrogate loss: $\mathcal{L}''_{\alpha_1,\tau}$. α_2 is the rounding of the loss function used to estimate the error that is provided to the second derivative of the surrogate loss.
$\mathbf{W}_{\alpha_1,\alpha_2} = diag\left(w_{\alpha_1,\alpha_2,1},...,w_{\alpha_1,\alpha_2,n}\right)$, where $w_{\alpha_1,\alpha_2,i} = \mathcal{L}''_{\alpha_1,\tau}(\tilde{u}_{\alpha_2,i})$, page 37

$\mathbf{W}_{\alpha_1,0}$ diagonal weight matrix. The diagonal elements of the matrix are the second derivative of the loss function. α_1 is the rounding for the loss function used to calculate the second derivative of the surrogate loss: $\mathcal{L}''_{\alpha_1,\tau}$. The second value of $\alpha = 0$ indicates that the error used in the second derivative of the loss function is calculated with the exact pinball loss: $w_{\alpha_1,0,i} = \lim_{\alpha\to 0}\mathcal{L}''_{\alpha_1,\tau}(\tilde{u}_{\alpha,i}) = \mathcal{L}''_{\alpha_1,\tau}(\hat{u}_i)$.
$\mathbf{W}_{\alpha_1,\alpha_2} = diag\left(w_{\alpha_1,0,1},...,w_{\alpha_1,0,n}\right)$, page 37

$\mathbf{W}$ short hand notiation for $\mathbf{W}_{\hat{\alpha},0}$. The diagonal elements of the matrix are the second derivative of the loss function. α_1 is the rounding for the loss function used to calculate the second derivative of the surrogate loss: $\mathcal{L}''_{\alpha_1,\tau}$. The second value of $\alpha = 0$ indicates that the error used in the second derivative of the loss function is calculated with the exact pinball loss: $w_i = w_{\hat{\alpha},0,i} = \lim_{\alpha\to 0}\mathcal{L}''_{\hat{\alpha},\tau}(\tilde{u}_{\alpha,i}) = \mathcal{L}''_{\hat{\alpha},\tau}(\hat{u}_i)$.
$\mathbf{W} = \mathbf{W}_{\hat{\alpha},0} = diag\left(w_{\hat{\alpha},0,1},...,w_{\hat{\alpha},0,n}\right) = diag\left(w_1,...,w_n\right)$, page 37

$\tilde{w}$,$\tilde{w}_\alpha$ Diagonal elements of $\tilde{\mathbf{W}}$.see $\tilde{\mathbf{W}}$, page 37

$w_{\alpha_1,\alpha_2,i}$ see $\mathbf{W}_{\alpha_1,\alpha_2}$, page 37

$w_{\alpha_1,0,i}$ see $\mathbf{W}_{\alpha_1,0}$, page 37

w_i see $\mathbf{W}$, page 37

X random vector of predictors, page 5

$\underline{\mathbf{X}}$ matrix of observations of the predictors, we have $\underline{\mathbf{X}} = \left[\ \underline{\boldsymbol{x}}_1,\quad \underline{\boldsymbol{x}}_2,\quad \ldots\quad ,\underline{\boldsymbol{x}}_n\ \right]^T$, page 5

$\mathbf{X}_0$ complete model matrix that includes the intercept, the linear part and the smooth part before transformation to take into account idenfitiability contraints. We have $\mathbf{X}_0 = \left(\mathbf{x}_{0,1},\mathbf{x}_{0,2},...,\mathbf{x}_{0,n}\right)^T$., page 12

$\mathbf{X}$ transformation of matrix $\mathbf{X}_0 = \left(\mathbf{x}_{0,1},\mathbf{x}_{0,2},...,\mathbf{x}_{0,n}\right)^T$ to take into account identifiability constraints, page 13

$\mathbf{x}_i$ elements of model matrix $\mathbf{X}$, page 13

$\mathbf{x}_{0,i}$ elements of model matrix $\mathbf{X}_0$. We have $\mathbf{x}_{0,i} = \left(1,\underline{\boldsymbol{\pi}}_i^T,\mathbf{z}_i^T\right)^T$, page 12

$\mathcal{X}$ dataset: $\mathcal{X} = \left(\underline{\boldsymbol{x}}_i,y_i\right)_{i=1,...,n}$, page 90

$\mathcal{X}^{*b}$ dataset resampled with replacement for non-parametric bootstrap for calculating CI ($b = 1, ..., B_0$) or for smoothing parameter relaxation ($b = 1, ..., B$), page 90

$\underline{\boldsymbol{x}}_i$ vector of observations of the predictors, page 5

Y random univariate response variable, page 5

$\mathbf{y}$ vector of observations of the response, page 5

y_i observation i of the response ($i = 1, ..., n$), page 5

$\mathbf{Y}$ random vector of observations, page 53

$\mathbf{Z}$ model matrix for smooth components, page 9

$\mathbf{z}_i$ elements of the model matrix for smooth components: $\mathbf{Z} = (\mathbf{z}_1, ..., \mathbf{z}_n)^T$. Matrix $\mathbf{Z}$ is of size $n \times (Q_1 + ... + Q_m)$. Each Q_j is the number of basis functions for smooth j, $j = 1, ..., m$, page 9

$\grave{\zeta}$ multipler of the square root of the diagonal of the precision matrix of the QGACV to obtain $\grave{\mathbf{d}}$, page 96

$\underline{\boldsymbol{z}}_i$ sub-vector of the vector of predictors $\underline{\boldsymbol{x}}_i$ that have a smooth relationship with the response. $\underline{\boldsymbol{x}}_i = \left(\underline{\boldsymbol{\pi}}_i^T, \underline{\boldsymbol{z}}_i^T\right)^T$, page 5

1. INTRODUCTION

Quantile Additive Models (QAMs) combine two important statistical concepts: Quantile regression and Additive Modelling. Quantile Regression (QR) was first developed in Koenker and Bassett Jr [1978] as a way to model quantiles of a response variable conditional on some predictors. This type of model is well adapted to linear and potentially heteroscedastic conditional responses. However, in cases where the response is non-linear, QR fails to accurately model the relationship between the response and covariates.

Quantile Additive Models (QAMs) are multivariate non-linear extensions of the linear QR based on the concept of (Generalized) Additive Models (Hastie and Tibshirani [1987], Hastie and Tibshirani [1990]). An additive model is a linear model where the predictors are smooth, possibly non-linear functions of the covariates (Wood [2017], preface). The advantage of this approach is that it renders the model interpretable and computationally tractable. Smooth functions in QAMs are generated using basis functions. The linear quantile regression has a unique solution (that can be a single point or a whole set of solutions). For a QAM, the solution depends on the number of basis functions used (see for example Koenker [2020c]). In general, once the user has decided the number of basis functions to use, a smoothing penalty is added to the loss function. This penalty allows the user to fine tune the degree of smoothness of the model without having to change the number of basis functions. Determining automatically this vector of smoothing penalties is an arduous task. Because of this difficulty, at the time of starting this project, packages that included non-linear quantile regression were mostly letting the user choose the number of basis functions and/or smoothing penalty, or were selecting the parameter(s) by optimizing a criterion on a grid[1]. Letting the user choose the smoothing penalty weight can lead to fits that are somewhat arbitrary and open to potential overfit or underfit. It also makes the function somewhat unpractical. Optimizing a criterion on a grid is not scalable beyond one or two dimensions with a regular computer.

The aim of this thesis is to propose a fully automated method to estimate both the vector of smoothing parameters and the vector of parameters of a Quantile Additive Model (QAM) together with an implementation in R. The estimation of QAMs is difficult for several reasons. First, given a fixed vector

[1]for example, package 'cosso' Lin et al. [2013] uses K-fold cross validation to tune the models and function 'rqss' in package 'quantreg' lets the user choose the regularization parameter.

of smoothing parameters, the loss function is not differentiable. Koenker and Bassett Jr [1978] use linear programming directly to estimate a linear quantile regression. However, in the case of QAMs, the penalty is a quadratic function. It is then not possible to use linear programming to solve the problem. Furthermore, popular techniques such as primal coordinate descent cannot be used to solve a QAM (Cevher [2016]). This is because the loss function of QAMs contains a linear transform of the vector of parameters.

Then the estimation of the vector of smoothing parameters is difficult. Cross validation is a method that can be used to estimate the generalization error of the model. However, we are facing two issues when using cross validation to estimate the generalization error. First, the loss function is non-differentiable. Removing a single datapoint from the dataset as in Leave-One-Out Cross Validation (LOOCV) often does not change the fit. Second, when we are fitting non-central quantiles, we have an unbalanced dataset, with more datapoints on one side than the other. This unbalanced dataset leads to problems when trying to approximate the LOOCV.

To obtain an estimate of the generalization error that has a lower computational cost than the LOOCV, a Generalized Approximate Cross Validation has been proposed in Yuan [2006]. To obtain the smoothing parameters automatically, we could optimize a GACV-type criterion. However, this function is non-smooth and non-convex. Therefore, most standard optimization algorithm cannot be used to optimize this criterion.

The contributions of this thesis then include: the principled derivation of a new GACV-type criterion specifically adapted to QAMs: the Quantile GACV (QGACV), the proposal of a surrogate loss function that is infinitely smooth and has a minimum at 0 to approximate the pinball loss used for QR for both the estimation of the vector of parameters and the estimation of the smoothing parameters, the proposal of a method to estimate both the vector of smoothing parameters and optimize the QGACV criterion, a method to estimate the degree of rounding of the loss function to calculate the effective degrees of freedom (edf)[2] in the QGACV formula, a method to estimate the confidence intervals for the model based on non-parametric bootstrap, a method to relax the smoothing parameters to obtain better coverage of the confidence intervals, and finally, the implementation of these methods in an R package: package 'qgacv'. Also, the methods proposed previously were mostly focusing on unidimensional models (Quantile Smoothing Splines Bloomfield and Steiger [1983]). Our methodology extends this to the multivariate setting (QAMs).

[2]There are two definitions for the effective degrees of freedom that are equal under certain conditions(see section 4.2). In this document, we will use definition 4.4, i.e. the average sensitivity of the predicted value to changes in the observations of the response variable Ye [1998] (Ye refers to Generalized degrees of freedom) (see also Efron [2004], Kaufman and Rosset [2014])

This thesis is structured as follows (see also the Flowchart on Figure 1.1): after this chapter, we introduce Quantile Regression and Quantile Additive Models (QAMs) in chapter 2. Then in chapter 3, we derive the new QGACV criterion and discuss the replacement of the exact pinball loss with a smooth surrogate for part of the criterion. Chapter 4 contains further comments about the derivation of the QGACV, loss functions and other topics. It adds to some points made in Chapter 3. After having derived the QGACV criterion, in Chapter 5, we note that the problem to solve is in fact a bilevel problem where the upper level problem is non-smooth and non-convex and the lower level problem is non-smooth and convex. A method is then proposed to solve both the upper and lower level problems alternatively. In Chapter 6, we use non-parametric bootstrap to obtain confidence intervals (CIs). As the coverage of the CIs does not seem sufficient, a smoothing parameter relaxation is proposed. In Chapter 7 the general algorithm to estimate the model integrating the contents of Chapters 3, 5 and 6 is presented. Additionally, algorithms to obtain different starting and stopping values for the rounding of the surrogate loss together with initialization algorithms are discussed. The last three chapters (8, 9 and 10) are an introduction to R package 'qgacv' implementing the algorithms of Chapter 7, a comparison with other packages and a conclusion.

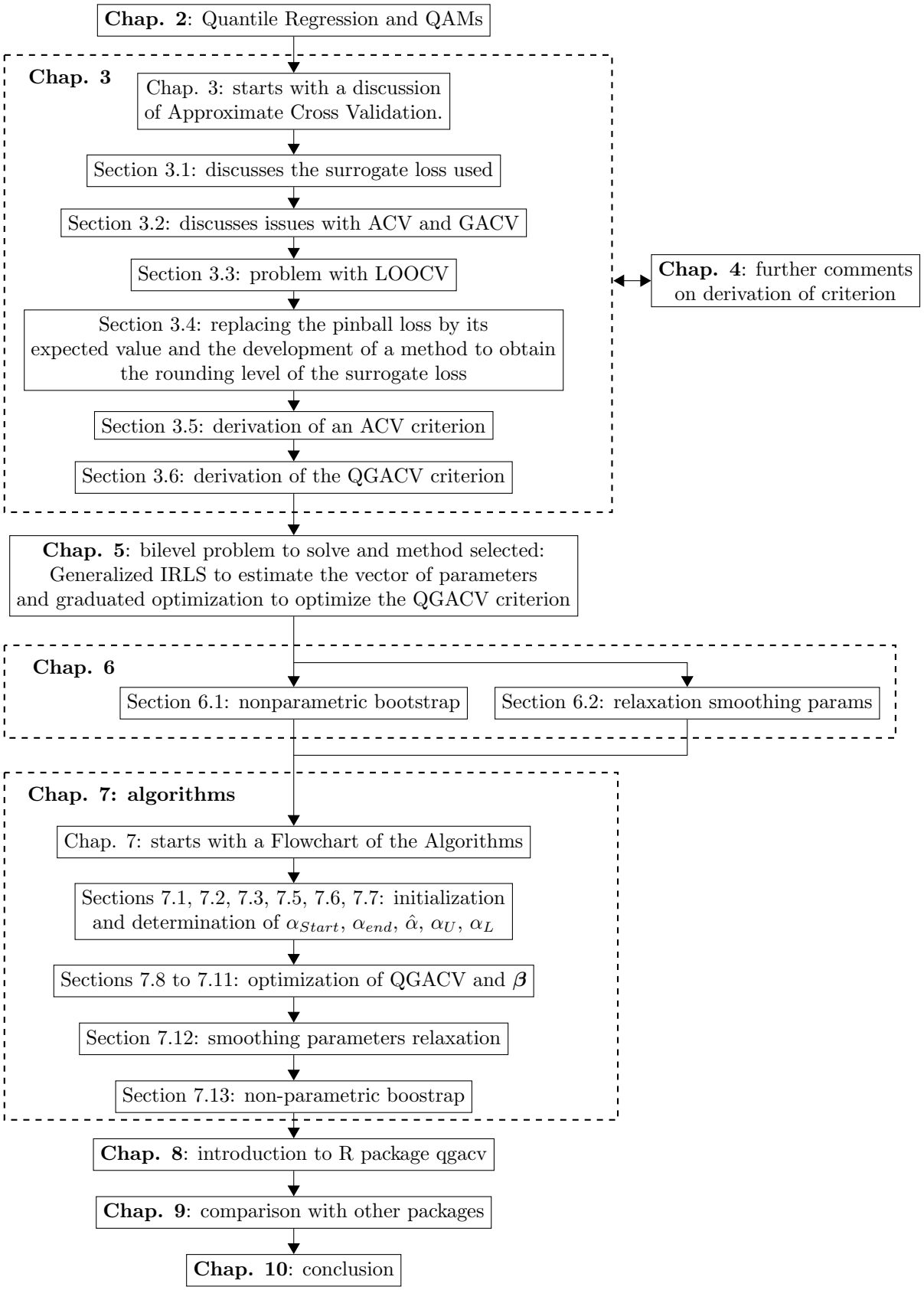


Fig. 1.1: Flowchart of the thesis

2. QUANTILE REGRESSION AND QUANTILE ADDITIVE MODELS

Boscovich [1757] proposed a method to fit a regression line to some data by minimizing the L1 distance between the model and the observations. This type of regression can be referred to as Least Absolute Deviation (LAD) amongst others (see Bloomfield and Steiger [1983] p.34 for an exhaustive list). The Quantile Regression (QR) (Koenker and Bassett Jr [1978]) is an extension of this concept. It comes from the realization that if the absolute value function is tilted around the origin, instead of obtaining the conditional median we obtain a conditional model of the cumulative density function of the response.

2.1 Quantile Regression

In a mean regression, we assume that the expected response conditional on the covariates is a function of the covariates. In a QR, instead of modelling the conditional expected value, we model a quantile $\tau \in (0,1)$ of the response conditional on some covariates $\underline{\boldsymbol{x}}$ (note that in this document, observations of predictors will be indicated by an underline (e.g. $\underline{\boldsymbol{x}}_i$) whereas elements of the model matrix will not have an underscore (e.g. $\mathbf{x}_i$)):

$$\mathbb{P}\left(Y \leq f\left(\underline{\boldsymbol{x}}\right) | X = \underline{\boldsymbol{x}}\right) = \tau \tag{2.1}$$

If we have a vector of observations $\mathbf{y}$ of random variable Y and a matrix of observations $\underline{\mathbf{X}} = (\underline{\boldsymbol{x}}_1, ..., \underline{\boldsymbol{x}}_n)^T$ of the vector of covariates X, it can be shown that f can be estimated by solving the following convex optimization problem:

$$\begin{aligned} \hat{f} &\in \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - f\left(\underline{\boldsymbol{x}}_i\right)\right), \\ \mathcal{L}_\tau(u_i) &= u_i \left(\tau - I\left(u_i < 0\right)\right) \end{aligned} \tag{2.2}$$

where $\mathcal{F}$ is a space of functions, I is the indicator function, $\mathbf{u} = (u_1, ..., u_n)^T$ is a vector of errors with $u_i = y_i - f(\underline{\boldsymbol{x}}_i)$. The sign $\in$ is due to the potential non-unique solution of the problem. We will also use the notation $\boldsymbol{f}(\underline{\mathbf{X}}) = (f(\underline{\boldsymbol{x}}_1), \ldots, f(\underline{\boldsymbol{x}}_n))^T$ to indicate the vector of predicted value. This vector of predicted values is a function of the matrix of observations of all predictors $\underline{\mathbf{X}}$. The notation for predicted value will be also simplified as f_i to mean

$f(\underline{\boldsymbol{x}}_i)$.

In this document, we will assume that f can be decomposed in 3 parts: an intercept that we call β_{00}, a linear part $\left(\underline{\boldsymbol{\pi}}_i^T \boldsymbol{\beta}_{01}\right)$ and smooth part $g(\underline{\boldsymbol{z}}_i)$, where g is a smooth function. $\underline{\boldsymbol{\pi}}_i$ is the sub-vector of vector $\underline{\boldsymbol{x}}_i$ consisting of the predictors that have a linear relationship with the response. $\underline{\boldsymbol{z}}_i$ is the sub-vector of covariates $\underline{\boldsymbol{x}}_i$ consisting of the predictors that have a smooth relationship with the response. If $g = 0$, then, we obtain the linear quantile regression. Figure 2.1 shows an example of linear quantile regression empirical loss function when there is a single covariate. The loss function is piecewise linear. The linear quantile regression is:

$$\left(\hat{\beta}_{00}^{LQR}, \hat{\boldsymbol{\beta}}_{01}^{LQR}\right)^T \in \underset{(\beta_{00}, \boldsymbol{\beta}_{01})}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - \beta_{00} - \underline{\boldsymbol{\pi}}_i^T \boldsymbol{\beta}_{01}\right) \tag{2.3}$$

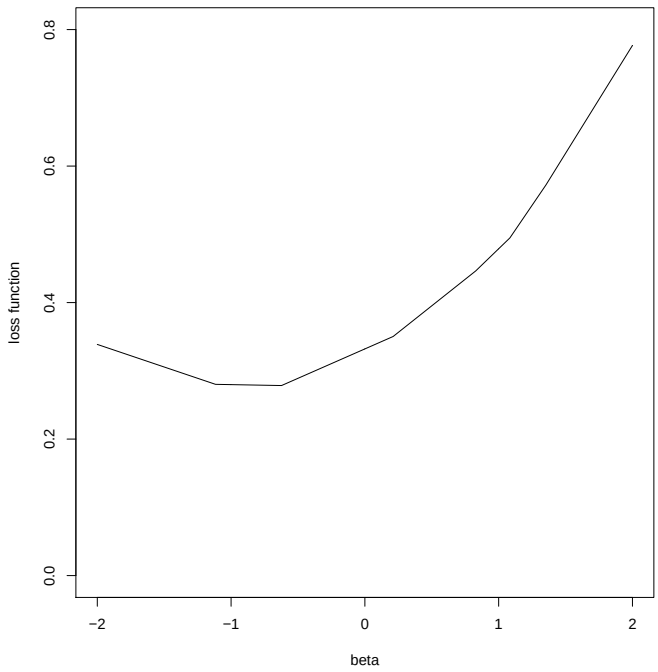


Fig. 2.1: This chart shows an example of empirical loss $\frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_\tau(y_i - \underline{\pi}_i \beta_{01})$ as a function of β_{01}. We assume that $\beta_{00} = 0$ and that there is a single predictor with scalar coefficient β_{01}. The function is a piecewise linear function.

2.2 Quantile Additive Model

In the case of Quantile Additive Models, we assume that multivariate function g is non-zero and can be decomposed into a sum of simpler functions that can be univariate or multivariate, for example:

$$g(\underline{\boldsymbol{z}}_i) = g_1(\underline{z}_{1,i}) + g_2(\underline{z}_{2,i}) + g_3(\underline{z}_{3,i}, \underline{z}_{4,i}) + ...$$

In the following, to simplify the notation, we will decompose function g generally as:

$$g(\underline{\boldsymbol{z}}_i) = \sum_{j=1}^{m} g_j(\underline{\boldsymbol{z}}_i) \tag{2.4}$$

where each g_j can be a univariate function of one of the coordinates of vector $\underline{\boldsymbol{z}}_i$ or a multivariate function of several of the coordinates of vector $\underline{\boldsymbol{z}}_i$. Replacing in equation 2.2:

$$\left(\hat{\beta}_{00}, \hat{\boldsymbol{\beta}}_{01}, \hat{g}_1, ..., \hat{g}_m\right) \in \underset{(\beta_{00}, \boldsymbol{\beta}_{01}, g_1, ..., g_m)}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - \beta_{00} - \underline{\boldsymbol{\pi}}_i^T \boldsymbol{\beta}_{01} - \sum_{j=1}^{m} g_j\left(\underline{\boldsymbol{z}}_i\right) \right) \tag{2.5}$$

The functions g_j are assumed to be sufficiently flexible to adapt to the data. Because of this flexibility, in many cases, the model would overfit. It is then often proposed to add a penalty $\mathcal{P}_{\boldsymbol{\lambda}}(f)$ to regulate the degree of smoothness of each function. Going back to the original notation of equation 2.2, we can write this as: [1]

$$\hat{f} \in \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - f\left(\underline{\boldsymbol{x}}_i\right) \right) + \mathcal{P}_{\boldsymbol{\lambda}}\left(f\right) \tag{2.6}$$

Where $\boldsymbol{\lambda}$ is a vector of regularization parameters allowing to tune the penalty. Penalty $\mathcal{P}_{\boldsymbol{\lambda}}\left(f\right)$ is often decomposed into one penalty on each smooth g_j and no penalty on the linear part: $\mathcal{P}_{\boldsymbol{\lambda}}\left(f\right) = \sum_{j=1}^{m} \lambda_j \mathcal{P}_j\left(g_j\right)$. Koenker et al. [1994], Koenker [2011] take $\mathcal{P}_j\left(g_j\right)$ to be total variation penalties. If function g_j is a univariate function, this total variation penalty is written:

$$\mathcal{P}_j\left(g_j\right) = \int \left| g_j^{''}\left(z\right) \right| dz$$

[1] Note that following Reinsch [1967], Aravkin et al. [2019], this problem can also be written equivalently:

$$\hat{f} \in \underset{f \in \mathcal{F}}{\operatorname{argmin}} \mathcal{P}_{\boldsymbol{\lambda}}\left(f\right) \qquad s.t. \quad \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - f\left(\underline{\boldsymbol{x}}_i\right) \right) \leq \sigma$$

or can also be seen as the Lagrangian form of the following constrained optimization problem:

$$\hat{f} \in \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - f\left(\underline{\boldsymbol{x}}_i\right) \right) \qquad s.t. \quad \mathcal{P}_{\boldsymbol{\lambda}}\left(f\right) \leq \tau$$

Fasiolo et al. [2020b] assume the penalties to be quadratic penalties. If function g_j is univariate, an example of quadratic penalty is:

$$\mathcal{P}_j\left(g_j\right) = \int g_j^{''}\left(z\right)^2 dz \tag{2.7}$$

For a thin plate spline of two predictors, the equivalent penalty is (see Wood [2017] p.216):

$$\mathcal{P}_j\left(g_j\right) = \int\int \left(\frac{\partial^2 g_j}{\partial z_1^2}\right)^2 + 2\left(\frac{\partial^2 g_j}{\partial z_1 \partial z_2}\right)^2 + \left(\frac{\partial^2 g_j}{\partial z_2^2}\right)^2 dz_1 dz_2 \tag{2.8}$$

2.2.1 Representation of the functions

Several methods could be employed to represent the functions of QAMs. One of the most used representation is the cubic spline basis expansion. One reason for this popularity comes from the fact that this representation has good approximation theoretic properties (Reinsch [1967], De Boor [1978]).

A popular method for spline basis expansion is the Basis spline (B-spline) expansion (De Boor [1978]). One reason for this popularity is the proposal of Eilers and Marx [1996] to add a tractable finite difference penalty to the model (Penalized Spline or P-spline).

We here follow Wood [2017] pp.204-208. The value of the spline at point $\underline{z}_i$ is defined as a sum of basis functions. If there are Q basis functions, and assuming that there is a single covariate $\underline{z}_i$ for the smooth, the value of the spline function at $\underline{z}_i$ is:

$$g\left(\underline{z}_i\right) = \sum_{q=1}^{Q} \beta_{02,q} \psi_q^o\left(\underline{z}_i\right) \tag{2.9}$$

Where o is the order of the spline and the $\beta_{02,q}$ are the coefficients associated with each basis function. We use notation $\boldsymbol{\beta}_{02}$ to designate the coefficients for the smooths of the model. For a cubic spline, $o = 2$. The value of basis $\psi_q^o\left(\underline{z}_i\right)$ can be calculated by means of the Cox-De Boor-Mansfield recursive algorithm. First, we define $Q + o + 2$ knots over the support of $\underline{z}_i$. We here call the knots $s_1 < ... < s_{Q+o+2}$. Then the recursive algorithm starts from the functions ψ at order -1. Each $\psi_q^{-1}(\underline{z}_i)$ is equal to 1 over $[s_q, s_{q+1})$ and 0 outside of this interval.

$$\psi_q^{-1}(\underline{z}_i) = \begin{cases} 1 & s_q \leq \underline{z}_i < s_{q+1} \\ 0 & \text{otherwise} \end{cases}$$

Then $\psi_q^0(\underline{z}_i)$, $\psi_q^1(\underline{z}_i)$ and $\psi_q^2(\underline{z}_i)$ are calculated recursively with the following formula:

$$\psi_q^o(\underline{z}_i) = \frac{\underline{z}_i - s_q}{s_{q+o+1} - s_q}\psi_q^{o-1}(\underline{z}_i) + \frac{s_{q+o+2} - \underline{z}_i}{s_{q+o+2} - s_{q+1}}\psi_{q+1}^{o-1}(\underline{z}_i) \quad q = 1, ..., Q.$$

g can be re-written in the format of a vector of parameters times an $n \times Q$ model matrix $\mathbf{Z}$:

$$\mathbf{Z} = (\mathbf{z}_1, ..., \mathbf{z}_n)^T$$

We use notation $\mathbf{Z}$ for the model matrix that only includes smooth components as we will use notation $\mathbf{X}_0$ for an extended version of the model matrix that includes the intercept and the linear components and $\mathbf{X}$ for an extended version of the model matrix that includes the intercept, linear components and identifiability constraints (see section 2.2.4). For a cubic spline (o=2):

$$\mathbf{z}_i = \left(\psi_1^2(\underline{z}_i), \ldots, \psi_Q^2(\underline{z}_i)\right)^T$$

Then equation 2.9 can be re-written:

$$\left(g\left(\underline{z}_1\right), ..., g\left(\underline{z}_n\right)\right)^T = \mathbf{Z}\boldsymbol{\beta}_{02}$$

where $\boldsymbol{\beta}_{02}$ is the vector of Q coefficients. For multiple smooths, this can be written in a similar way:

$$\left(g\left(\underline{\boldsymbol{z}}_1\right), ..., g\left(\underline{\boldsymbol{z}}_n\right)\right)^T = \mathbf{Z}\boldsymbol{\beta}_{02}$$

where here matrix $\mathbf{Z}$ is the juxtaposition of the columns of the model matrix for each smooth. Matrix $\mathbf{Z}$ is of size $n \times (Q_1 + ... + Q_m)$. Each Q_j is the number of basis functions for smooth j, $j = 1, ..., m$.

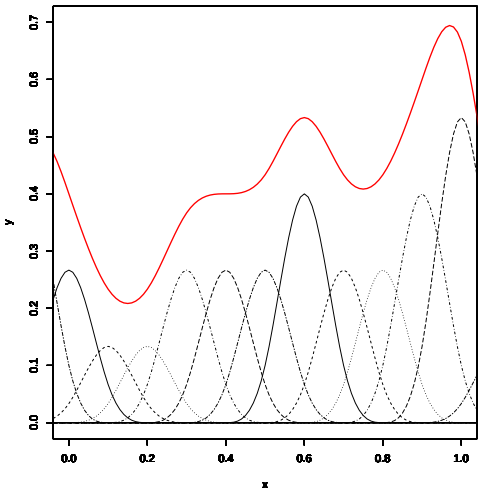


Fig. 2.2: This figure shows how a unidimensional spline is constructed. The basis functions are represented in black, in plain or dotted lines. Each individual basis function is multiplied by a weight. The sum of these weighted basis functions results in the red curve, the constructed spline with the basis functions. This figure is similar to Figure 5.6, page 205 in Wood [2017]

2.2.2 Multivariate Smooths

Several methods can be used to represent multivariate functions (see e.g. Wood [2017], chapter 5). A simple method that can be used consists in using tensor product of basis functions (see e.g Eilers et al. [2006]). If we take the tensor products of basis functions we obtain a new basis that can be used to obtain a multivariate smooth. For example:

$$f_{\underset{\sim}{z}_1\underset{\sim}{z}_2}\left(\underset{\sim}{z}_1, \underset{\sim}{z}_2\right) = \sum_{q_1=1}^{Q_1}\sum_{q_2=1}^{Q_2} \beta_{02,q_1,q_2}\psi^o_{q_1}\left(\underset{\sim}{z}_1\right)\psi^o_{q_2}\left(\underset{\sim}{z}_2\right) \tag{2.10}$$

The issue with tensor product splines is their scalability. If, for example, we use $Q_1 = 10$ and $Q_2 = 10$, a 2 dimensional smooth would have 100 coefficients. Then if we wanted to use this technique to represent a 3 dimensional smooth, we would have 1,000 coefficients. It is then difficult to use this technique beyond 2 or 3 dimensions. Note that some methods have been proposed to obtain more efficient multivariate smooth (for example Kim and Gu [2004] Siebenborn and Wagner [2021])

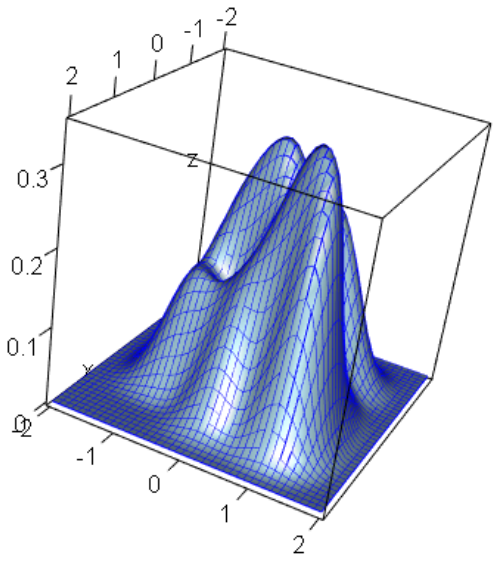


Fig. 2.3: An example of multivariate smooth created using a tensor product of smooths

2.2.3 Penalty

Eilers and Marx [1996] introduce a penalty based on the spline representation.

They denominate this model as a P-spline model. In 1 dimension, following Wood [2017], p.206, the penalty is simply built by difference of coefficients:

$$\Delta\boldsymbol{\beta}_{02} = \left[\left(\beta_{02,2} - \beta_{02,1}\right), \left(\beta_{02,3} - \beta_{02,2}\right), ..., \left(\beta_{02,Q} - \beta_{02,Q-1}\right)\right]^T \tag{2.11}$$

$\Delta\boldsymbol{\beta}_{02}$ can also be written:

$$(\Delta\boldsymbol{\beta}_{02}) = P\boldsymbol{\beta}_{02} = \begin{bmatrix} -1 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 \\ \vdots & 0 & \ddots & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & \cdots & \cdots & 0 & -1 \end{bmatrix} \begin{bmatrix} \beta_{02,1} \\ \beta_{02,2} \\ \vdots \\ \vdots \\ \vdots \\ \beta_{02,Q} \end{bmatrix} \tag{2.12}$$

The squared smoothing penalty is then:

$$\left(\Delta\boldsymbol{\beta}_{02}\right)^T \left(\Delta\boldsymbol{\beta}_{02}\right) = \boldsymbol{\beta}_{02}^T P^T P \boldsymbol{\beta}_{02} = \boldsymbol{\beta}_{02}^T S \boldsymbol{\beta}_{02} = \boldsymbol{\beta}_{02}^T \begin{bmatrix} -2 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \cdots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & \cdots & 0 & 1 & -2 \end{bmatrix} \boldsymbol{\beta}_{02} \tag{2.13}$$

This squared penalty approximates the integral of the square of the first derivative as described in Eilers et al. [2006]. The penalty is simply expressed as $\boldsymbol{\beta}_{02}^T S \boldsymbol{\beta}_{02}$ where S is a penalty matrix. For tensor product of smooths, the univariate penalties created following the finite difference methods can be combined (Wood [2010]):

If we have observations of two covariates $\underline{\boldsymbol{z}}_1$ and $\underline{\boldsymbol{z}}_2$, then the univariate smoothing penalties in each direction derived using finite differences can be written: $\boldsymbol{\beta}_{02,\underline{z}_1}^T S_{\underline{z}_1} \boldsymbol{\beta}_{02,\underline{z}_1}$ and $\boldsymbol{\beta}_{02,\underline{z}_2}^T S_{\underline{z}_2} \boldsymbol{\beta}_{02,\underline{z}_2}$. The new penalty for the tensor product of smooths is:

$$\lambda_{\underline{z}_1} \boldsymbol{\beta}_{02,\underline{z}_1}^T \mathbf{1} \otimes S_{\underline{z}_1} \boldsymbol{\beta}_{02,\underline{z}_1} + \lambda_{\underline{z}_2} \boldsymbol{\beta}_{02,\underline{z}_2}^T S_{\underline{z}_2} \otimes \mathbf{1} \boldsymbol{\beta}_{02,\underline{z}_2} \tag{2.14}$$

where $\mathbf{1}$ is a vector of 1s. We can then extend this to multiple smooths. We create a block diagonal matrix of penalties corresponding to the functions g_j: $\mathbf{S}_{\boldsymbol{\lambda}}^g$ that is block-diagonal

$$\mathbf{S}_{\boldsymbol{\lambda}}^g = \begin{bmatrix} (\lambda_1 S_1) & 0 & \cdots & 0 \\ 0 & (\lambda_2 S_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & (\lambda_m S_m) \end{bmatrix}$$

2.2.4 Re-writing the model

We now go back to the general setting where $f(\underline{\boldsymbol{x}})$ is a sum of an intercept a linear part and a smooth part:

$$f(\underline{\boldsymbol{x}}_i) = \beta_{00} + \underline{\boldsymbol{\pi}}_i^T \boldsymbol{\beta}_{01} + \sum_{j=1}^{m} g_j(\underline{\boldsymbol{z}}_i) \tag{2.15}$$

Overall, replacing $g(\underline{\boldsymbol{z}}_i)$ by its basis representation and penalty $\mathcal{P}$ by its representation, then equation 2.6:

$$\hat{f} \in \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - f\left(\underline{\boldsymbol{x}}_i\right)\right) + \mathcal{P}_{\boldsymbol{\lambda}}\left(f\right)$$

can be written in the form of a (generalized) ridge penalized linear regression

$$\hat{\boldsymbol{\beta}}_0 \in \underset{\beta_0 = (\beta_{00}, \boldsymbol{\beta}_{01}, \boldsymbol{\beta}_{02})}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - \beta_{00} - \underline{\boldsymbol{\pi}}_i^T \boldsymbol{\beta}_{01} - \mathbf{z}_i^T \boldsymbol{\beta}_{02}\right) + \frac{1}{2} \boldsymbol{\beta}_{02}^T S_{\boldsymbol{\lambda}}^g \boldsymbol{\beta}_{02} \tag{2.16}$$

$\boldsymbol{\beta}_0$ is a vector containing the intercept β_{00}, the vector of coefficients of the linear part $\boldsymbol{\beta}_{01}$ and the vector of coefficients of the smooth part $\boldsymbol{\beta}_{02}$. A problem with the model expressed in equation 2.16 is the identifiability of coefficients discussed in Wood [2017] p.211, section 5.4.1. As we have an intercept and a smooth part consisting of a sum of basis functions, the solution is not uniquely defined and we may want to add constraints to the smooth part. The constraints proposed in Wood [2017] consists in forcing the sum of each smooth over the set of observations of the predictors to be equal to 0:

$$\begin{aligned} \hat{\boldsymbol{\beta}}_0 \in & \underset{\beta_0 = (\beta_{00}, \boldsymbol{\beta}_{01}, \boldsymbol{\beta}_{02})}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - \beta_{00} - \underline{\boldsymbol{\pi}}_i^T \boldsymbol{\beta}_{01} - \mathbf{z}_i^T \boldsymbol{\beta}_{02}\right) + \frac{1}{2} \boldsymbol{\beta}_{02}^T S_{\boldsymbol{\lambda}}^g \boldsymbol{\beta}_{02} \\ & s.t. \sum_{i=1}^{n} g_j\left(\underline{\boldsymbol{z}}_i\right) = 0, \ \forall j \in \{1, ..., m\} \end{aligned}$$

We can simplify the notation by integrating the intercept, the linear component and the smooth component into a single model matrix $\mathbf{X}_0$ composed of elements $\mathbf{x}_{0,i}$. Each $\mathbf{x}_{0,i}$ is composed of a 1 for the intercept, each element of the linear part $\underline{\boldsymbol{\pi}}_i$ and the elements of model matrix $\mathbf{Z}$: $\mathbf{z}_i$: $\mathbf{x}_{0,i} = (1, \underline{\boldsymbol{\pi}}_i^T, \mathbf{z}_i^T)^T$. Furthermore, columns and rows of zeros corresponding to the intercept and the

linear part can be added to matrix $\mathbf{S}^g_{\boldsymbol{\lambda}}$ to create matrix $\mathbf{S}^0_{\boldsymbol{\lambda}}$:

$$\mathbf{S}^0_{\boldsymbol{\lambda}} = \begin{bmatrix} 0 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & 0 \\ 0 & \dots & 0 & \mathbf{S}^g_{\boldsymbol{\lambda}} \end{bmatrix} = \begin{bmatrix} 0 & \dots & \dots & \dots & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \dots & \dots & 0 \\ \vdots & \dots & 0 & 0 & \dots & \dots & \vdots \\ 0 & \dots & 0 & (\lambda_1 S_1) & 0 & \cdots & \vdots \\ \vdots & \vdots & \vdots & 0 & (\lambda_2 S_2) & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & \cdots & 0 & (\lambda_m S_m) \end{bmatrix}$$

The problem with simplified notation with constraint can then be written:

$$\hat{\boldsymbol{\beta}}_0 \in \underset{\beta_0}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - \mathbf{x}^T_{0,i} \boldsymbol{\beta}_0 \right) + \frac{1}{2} \boldsymbol{\beta}^T_0 S^0_{\boldsymbol{\lambda}} \boldsymbol{\beta}_0$$

$$s.t. \sum_{i=1}^{n} g_j \left(\underline{\boldsymbol{z}}_i \right) = 0, \ \forall j \in \{1, ..., m\}$$

Finally, Wood [2017] p.211, section 5.4.1. explains that the constraints can be integrated in the model matrix and the penalty (see Wood [2017] for more details). We call $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}^T_1)^T$ the model matrix after transformation and $\mathbf{S}_{\boldsymbol{\lambda}}$ the penalty matrix after inclusion of identifiability constraints. Our problem can then be re-written in a familiar form:

$$\hat{\boldsymbol{\beta}} \in \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - \mathbf{x}^T_i \boldsymbol{\beta} \right) + \frac{1}{2} \boldsymbol{\beta}^T S_{\boldsymbol{\lambda}} \boldsymbol{\beta} \tag{2.17}$$

In the rest of this document we will use equation 2.17 as our based model. We will call p the sum of the number of coefficients of the linear part plus the number of coefficients of all the smooths of the model. Then model matrix $\mathbf{X}$ is matrix of size $n \times p$. In the next chapter, we are going to develop a new criterion to estimate the vector of smoothing parameters.

3. DEVELOPMENT OF A CRITERION TO ESTIMATE THE VECTOR OF SMOOTHING PARAMETERS OF THE MODEL

To develop a fully automated method to estimate QAMs, we face a series of obstacles. We start by a brief discussion of cross validation and Approximate Cross Validation (ACV), followed by a discussion of surrogate losses in section 3.1 and issues with ACV and one of its variants: the Generalized ACV (GACV) in section 3.2.

Let's assume that we have a sample $(\underline{\boldsymbol{x}}_i, y_i), i = 1, ..., n$, where the $(\underline{\boldsymbol{x}}_i, y_i)$ are independent and identically distributed. The estimation of the smoothing parameters of the QAM starts from the idea that if we knew the distribution $P_{y|\underline{\boldsymbol{x}}}$ of $y|\underline{\boldsymbol{x}}$, we could determine the vector of smoothing parameters by minimizing the generalization error:

$$GE\left(\boldsymbol{\lambda}\right) = \frac{1}{n}\sum_{i=1}^{n} E_{P_{y|\underline{\boldsymbol{x}}}}\left[\mathcal{L}_\tau\left(y_i - \hat{f}_{\boldsymbol{\lambda}}\left(\underline{\boldsymbol{x}}_i\right)\right)\right]$$

(see Reiss and Huang [2012], Elisseeff and Pontil [2003][1][2][3]. As we do not have access to this distribution, it has been proposed to use the leave-one-out cross validation criterion (LOOCV) instead (Luntz and Brailovsky [1969], Stone [1974], Allen [1974], Geisser [1975]):

$$V_0(\boldsymbol{\lambda}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_\tau\left(y_i - \hat{f}_{\boldsymbol{\lambda}}^{[-i]}(\underline{\boldsymbol{x}}_i)\right) \tag{3.1}$$

where $\hat{f}_{\boldsymbol{\lambda}}^{[-i]}(\underline{\boldsymbol{x}}_i)$ is the estimated predicted value for observation i when the

[1] Elisseeff and Pontil [2003] define the generalization error by calculating the expectation over the joint distribution of $(\underline{\boldsymbol{x}}, y)$ instead of the conditional distribution $y|\underline{\boldsymbol{x}}$

[2] Gressmann et al. [2018] note that a Bayesian extension of this would consist in calculating the generalization error as an expectation over the joint distribution of y and f: $\frac{1}{n}\sum_{i=1}^{n} E_{P_{(y,f|\underline{\boldsymbol{x}})}}\left[\mathcal{L}_\tau\left(y_i - \hat{f}_{\boldsymbol{\lambda}}\left(\underline{\boldsymbol{x}}_i\right)\right)\right]$ or $\frac{1}{n}\sum_{i=1}^{n} E_{P_{(\underline{\boldsymbol{x}},y,f)}}\left[\mathcal{L}_\tau\left(y_i - \hat{f}_{\boldsymbol{\lambda}}\left(\underline{\boldsymbol{x}}_i\right)\right)\right]$

[3] Note that Yuan [2006] refers to this Generalization Error as "Generalized Comparative Kullback Leibler" (GCKL) divergence. The GCKL was defined in Xiang and Wahba [1996], Wahba [1999] as a modified version of the Kullback-Leibler (KL) divergence between the true distribution of $y|\underline{\boldsymbol{x}}$ and the model parametrized by $\boldsymbol{\lambda}$ where the terms that did not depend on $\boldsymbol{\lambda}$ are removed (as they do not affect the optimization of $\boldsymbol{\lambda}$). We will refrain from using this terminology. As the KL-divergence is defined as a distance measure of probability distributions, we would then implicitly assume a mis-specified asymmetric Laplace distribution for the response (as in Yu and Moyeed [2001])

vector of parameters $\boldsymbol{\beta}$ is estimated after removing point i from the dataset:

$$\hat{\boldsymbol{\beta}}^{[-i]} \in \underset{\beta}{\operatorname{argmin}} \frac{1}{n} \sum_{j=1, j\neq i}^{n} \mathcal{L}_\tau \left(y_j - \mathbf{x}_j^T \boldsymbol{\beta}\right) + \mathcal{P}_{\boldsymbol{\lambda}} \left(\boldsymbol{\beta}\right) \tag{3.2}$$

To justify the use of $V_0(\boldsymbol{\lambda})$, Luntz and Brailovsky [1969](see also Vapnik [1998], p.417, Cawley and Talbot [2004] for a derivation) prove that for $(\underline{\boldsymbol{x}}_i, y_i)$ i.i.d.[4]

$$E_{P_{y|\underline{\boldsymbol{x}}}}[V_0(\boldsymbol{\lambda})] = E_{P_{y|\underline{\boldsymbol{x}}}}[GE^{[-i]}\left(\boldsymbol{\lambda}\right)] \tag{3.3}$$

where $GE^{[-i]}$ is the generalization error where point i is excluded:

$$GE^{[-i]}\left(\boldsymbol{\lambda}\right) = \frac{1}{n} \sum_{j=1, j\neq i}^{n} E_{P_{y|\underline{\boldsymbol{x}}}} \left[\mathcal{L}_\tau \left(y_j - \hat{f}_{\boldsymbol{\lambda}} \left(\underline{\boldsymbol{x}}_j\right)\right)\right]$$

As the LOOCV is computationally expensive, especially when the number of datapoints increases, Nychka et al. [1995] and Yuan [2006] propose to approximate the LOOCV criterion by an Approximate Cross Validation (ACV) criterion and a Generalized Approximate Cross Validation Criterion (GACV) (that will be discussed in section 3.2) respectively. In both derivations, the exact pinball loss is replaced by a surrogate loss that is locally differentiable around 0. This type of surrogate loss has a parameter to tune the degree of rounding. In this document, we call this parameter α. The surrogate loss of Nychka et al. [1995] is:

$$\mathcal{L}_{\alpha,\tau}^{Nychka}\left(u\right) = \begin{cases} \left(\tau - 1\right) u & \left(u < -\alpha\right) \\ \left(1 - \tau\right) \frac{u^2}{\alpha} & -\alpha \leq x < 0 \\ \tau \frac{u^2}{\alpha} & 0 \leq x \leq \alpha \\ \tau u & x > \alpha \end{cases} \tag{3.4}$$

The ACV from Nychka et al. [1995] is

$$V_1 = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{\alpha,\tau} \left(\frac{y_i - \tilde{f}_{\alpha,i}}{1 - \tilde{a}_i}\right) \tag{3.5}$$

where $\tilde{f}_{\alpha,i}$ is the predicted value for a vector of observations $\underline{\boldsymbol{x}}_i$ with a rounding parameter α: $\tilde{f}_{\alpha,i} = \tilde{f}_\alpha(\underline{\boldsymbol{x}}_i) = \mathbf{x}_i^T \tilde{\boldsymbol{\beta}}_\alpha$ and

$$\tilde{\boldsymbol{\beta}}_\alpha = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{\alpha,\tau} \left(y_i - \mathbf{x}_i^T \boldsymbol{\beta}\right) + \frac{1}{2} \boldsymbol{\beta}^T S_{\boldsymbol{\lambda}} \boldsymbol{\beta} \tag{3.6}$$

where $\tilde{a}_i$ are the diagonal elements of the hat matrix $\tilde{\mathbf{A}}$:

$$\tilde{\mathbf{A}} = \mathbf{X}(\mathbf{X}^T \tilde{\mathbf{W}} \mathbf{X} + \mathbf{S}_{\boldsymbol{\lambda}})^{-1} (\mathbf{X}^T \tilde{\mathbf{W}})$$

[4] the derivation is in general for the expectation under the joint distribution $P_{\underline{\boldsymbol{x}},y}$. However, the derivation would be similar with the conditional distribution.

and $\tilde{\mathbf{W}}$ is a diagonal matrix where the diagonal elements are equal to the second derivative of the surrogate loss function evaluated at each point i=1,...,n. We will also use notation $\tilde{\mathbf{A}}_\alpha$ and $\tilde{\mathbf{W}}_\alpha$ to indicate that the loss function is rounded using parameter value α.

We are also going to use $u_i = y_i - f_i$ for the error, $\hat{u}_i = y_i - \hat{f}_i$ for the error with the exact pinball loss and $\tilde{u}_i = y_i - \tilde{f}_i$ or $\tilde{u}_{\alpha,i} = y_i - \tilde{f}_{\alpha,i}$ for the surrogate loss with rounding parameter α. In the next section we discuss the surrogate loss we use. We introduce this surrogate loss at this stage as it will be used throughout this Chapter to illustrate the different issues discussed.

3.1 Surrogate Loss

Instead of using the surrogate loss of Nychka et al. [1995], we use a centered version of a loss proposed in Zheng [2011], Wang and Ye [2016] based on the surrogate loss developed for svms in Chen and Mangasarian [1995]. As pointed out by He et al. [2020], the loss of Zheng [2011] is also similar to a surrogate loss proposed by Amemiya [1982] to approximate absolute values. Our proposal is:

$$\mathcal{L}_{\alpha,\tau}(y_i - f_i) = \tau(y_i - f_i + \gamma) + \alpha \log\left(1 + \exp\left(-\frac{y_i - f_i + \gamma}{\alpha}\right)\right) \tag{3.7}$$

where $\gamma = \alpha \log\left(\frac{1-\tau}{\tau}\right)$. He et al. [2020] point out that the loss function of Zheng [2011] (equation 3.7 where γ is replaced by 0) can be obtained by convolving the exact pinball loss function $\mathcal{L}_\tau(u) = u(\tau - I(u < 0))$ with a logistic kernel: $K_{Log}(u) = \frac{\exp(-u)}{(1+\exp(-u))^2}$ (see also Fernandes et al. [2021]). In fact, based on He et al. [2020] and Decani and Stine [1986] we show in Appendix E that surrogate loss 3.7 can be obtained by convolving the exact pinball loss $\mathcal{L}_\tau(u) = u(\tau - I(u < 0))$ with a shifted logistic kernel:

$$K_{SLog,\tau}(u) = \frac{\left(\frac{1-\tau}{\tau}\right)\exp(-u)}{\left(1 + \left(\frac{1-\tau}{\tau}\right)\exp(-u)\right)^2} \tag{3.8}$$

Our loss function is therefore a special case of the general framework to smooth the pinball loss by using convolution of He et al. [2020]. Shifted Logistic Kernels for τ varying from 0.01 to 0.99 together with their corresponding pinball loss are represented on Figure 3.1.

Further to this, another method to derive a similar surrogate loss is the entropy-prox method proposed in Nesterov [2005] and derived for absolute values in Hahn et al. [2020](see Appendix F).

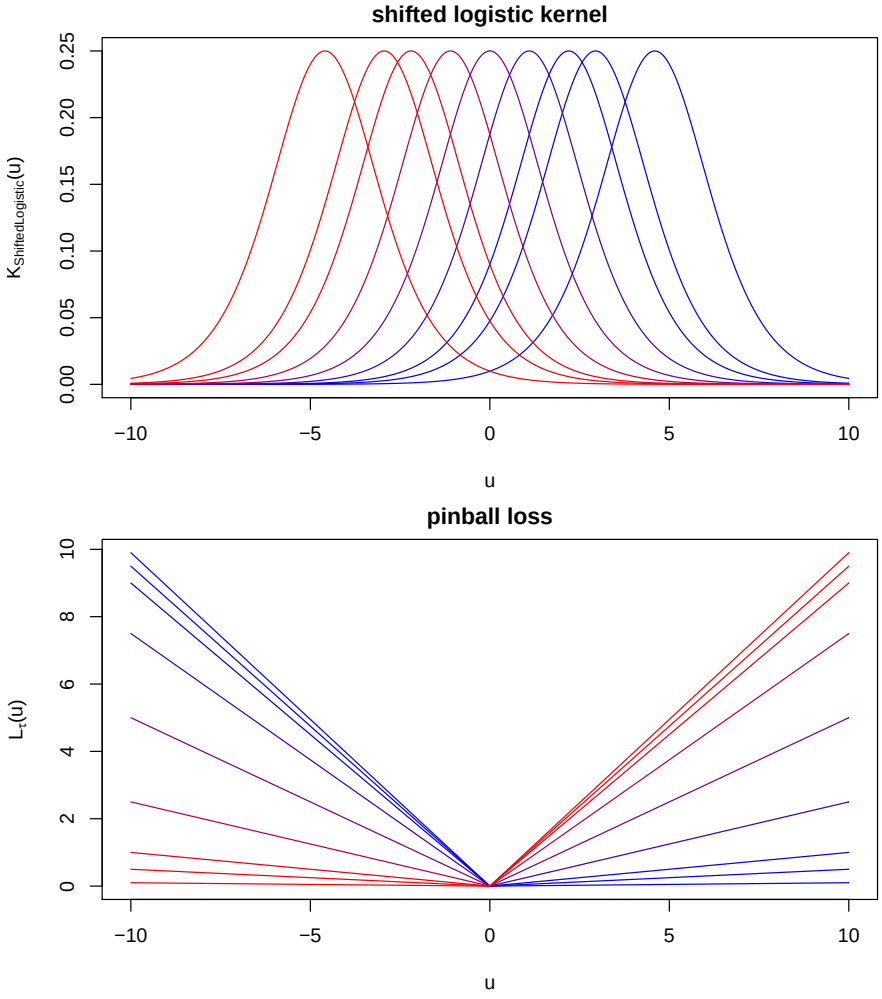


Fig. 3.1: Top chart: Shifted logistic kernel for $\tau = 0.01$ (blue), 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95 to 0.99 (red). Bottom chart: pinball loss corresponding to each shifted logistic Kernel.

There are several reasons why this loss is advantageous for our purpose. First, the loss used by Nychka et al. [1995] is continuous but not differentiable.

The first derivative of the surrogate loss is (see also Figure 3.2):

$$\left(\mathcal{L}_{\alpha,\tau}^{Nychka}\right)'(u) = \begin{cases} (\tau - 1) & (u < -\alpha) \\ undefined & (u = -\alpha) \\ \frac{2(1-\tau)u}{\alpha} & -\alpha < x < 0 \\ undefined & (u = 0) \\ \frac{2\tau u}{\alpha} & 0 < x < \alpha \\ undefined & (u = \alpha) \\ \tau u & x > \alpha \end{cases}$$

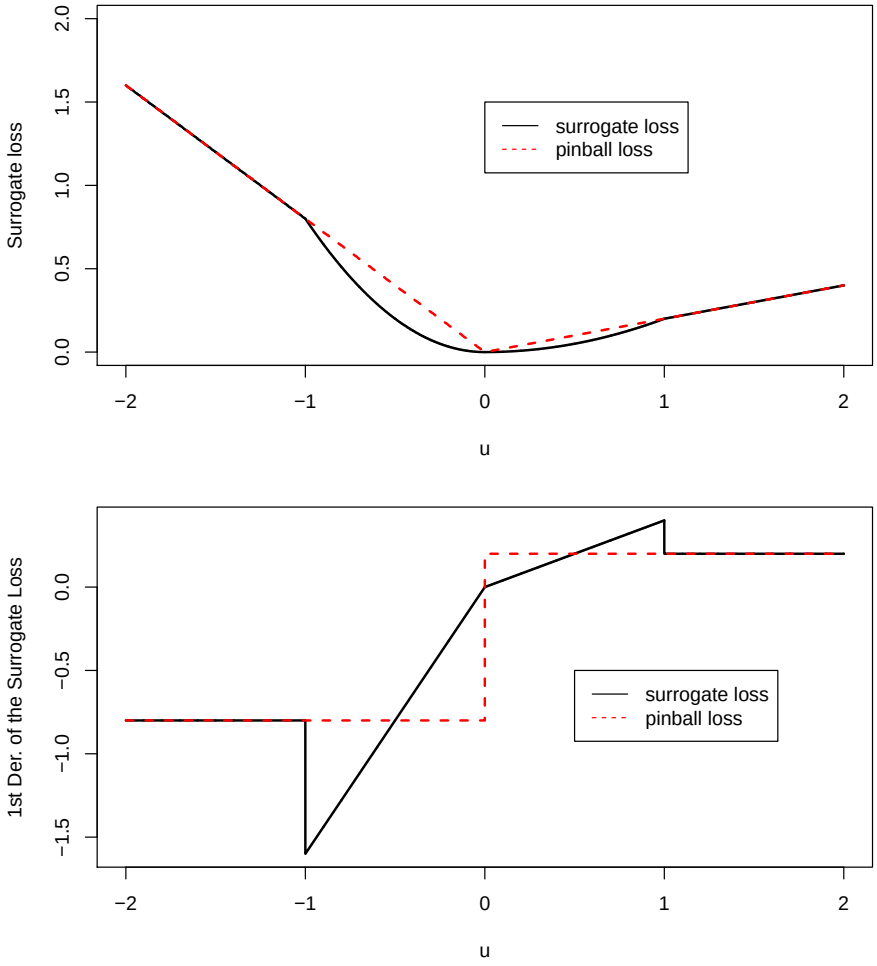


Fig. 3.2: The top chart shows the profile of the surrogate loss as used in Nychka et al. [1995]. The bottom chart is the first derivative of this loss function.

We cannot use this loss as we require a loss function that is at least 3rd order continuously differentiable. The reason is as follows. In this chapter, we derive a Quantile Generalized Approximate Cross Validation (QGACV) criterion. We will see in Chapter 5 that estimating the vector of smoothing parameters using the QGACV criterion requires the optimization of a bilevel nonsmooth and nonconvex problem. The lower level problem includes an argmin function with respect to the vector of parameters $\boldsymbol{\beta}$. Differentiating an argmin of a smooth loss function once leads to an expression that includes a second differential of the loss function (see Gould et al. [2016]). Hence, optimizing the upper level problem with respect to the vector of smoothing parameters via a hyper-gradient descent requires to calculate third order derivatives of the loss and a hyper-Newton algorithm requires fourth order derivatives.

Surrogate loss 3.7 is infinitely differentiable and can then be used to solve the bilevel optimization. Furthermore, we envisaged other potential surrogate losses. They have been proposed to 1) improve cross validation (Jung et al. [2020]), 2) calculate the effective degrees of freedom for ACV or GACV (Yuan [2006]) 3) estimate the vector of parameters given a fixed (vector of) smoothing parameter(s) (Clark and Osborne [1986], Madsen and Nielsen [1993], Chen and Wei [2005], Chen [2007], Muggeo et al. [2012]). A list of some surrogate losses that have been proposed can be found in Appendix A.

Most of these losses are not differentiable beyond the first order and cannot be used in our framework. Apart from using another infinitely smooth kernel to convolve the pinball loss (for example a Gaussian kernel) (that is proposed as a follow up to this thesis (see Conclusion (Chapter 10))), the only other alternative infinitely smooth surrogate that could have been used is the proposal of Wang et al. [2012] based on the integral of a sinc function:

$$\mathcal{L}^{\text{Wang}}_{\alpha,\tau}(u) = \begin{cases} u\left(\tau - \frac{1}{2} + \int_0^{\frac{u}{\alpha}} \frac{\sin(t)}{\pi t} dt\right) & (u > 0) \\ u\left(\tau - \frac{1}{2}\right) & (u = 0) \\ u\left(\tau - \frac{1}{2} - \int_{\frac{u}{\alpha}}^0 \frac{\sin(t)}{\pi t} dt\right) & (u < 0) \end{cases}$$

We choose loss 3.7 function over $\mathcal{L}^{Wang}$ as it has an analytic form instead of a integral form and the surrogate loss is not convex (see example on Figure 3.3 with α varying from 1 to 0.01 and $\tau = 0.3$). Note that after writing most of this document, we noticed that the work of Yilmaz and Sahiner [2019] can be applied to obtain an other type of infinitely smooth function. This is discussed in the Conclusion (Chapter 10).

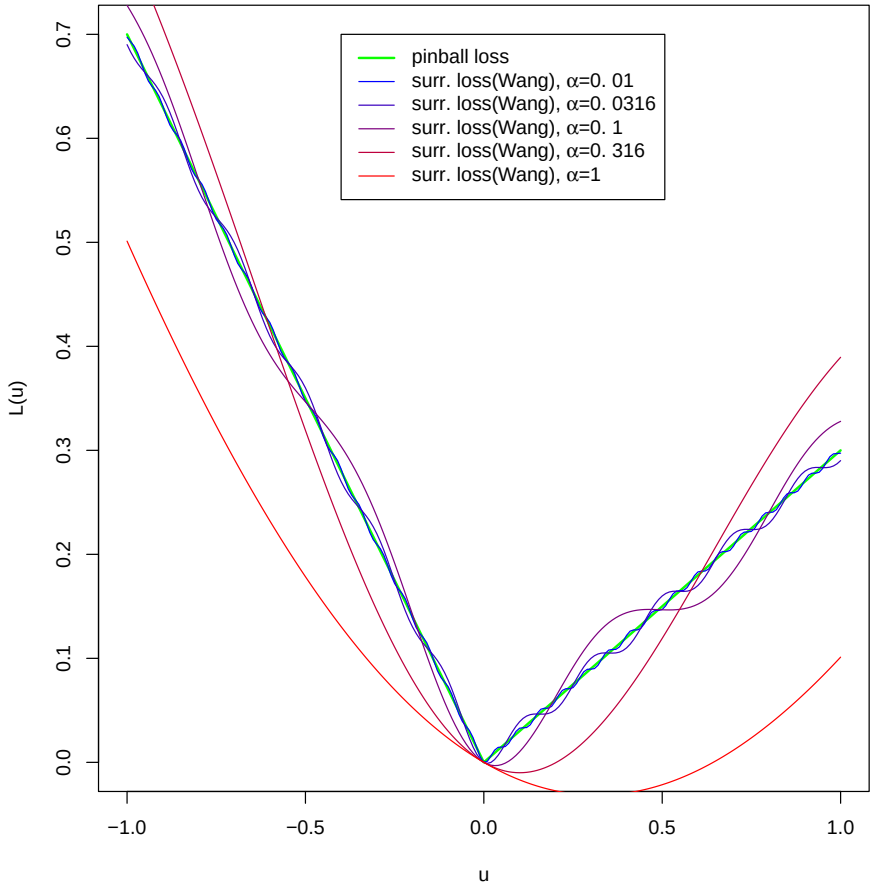


Fig. 3.3: Surrogate loss from Wang et al. [2012] for $\tau = 0.3$ and α varying from 1 to 0.01. The surrogate loss is not convex. The loss oscillates around the pinball loss. As we reduce α, the frequency of the oscillations increases.

Second, to derive an Approximate Cross Validation Criterion (ACV) in section 3.5, it is convenient to have a function that has a minimum in 0. This is because the derivation of the ACV is based on a first order approximation and the expression simplifies if it has a minimum in 0. Our version of the loss function of Zheng [2011], namely $\mathcal{L}_{\alpha,\tau}$ has always a minimum in 0 regardless of the value of τ. This is not the case with the loss function proposed in Zheng [2011] where the function has a minimum in $-\gamma$. In fact, we investigate three potential shifted versions of $\mathcal{L}_{\alpha,\tau}$ and give the reason why this version was chosen. As part of the analysis is based on the QGACV criterion, this analysis is

postponed to section 4.1.

Third, Zheng [2011] shows that for $\gamma = 0$, $\lim_{\alpha \to 0} \mathcal{L}_{\alpha,\tau}(u) = \mathcal{L}_\tau(u)$. The limit is the same for our loss function[5] In the next section we discuss issues with the ACV and GACV.

3.2 Issues with ACV and GACV

Although the ACV works well in many cases, Yuan [2006] notes that the ACV criterion of Nychka et al. [1995] overfits in some cases. Yuan [2006] then uses a method proposed in Craven and Wahba [1978], Golub et al. [1979] that consists in averaging the weights used in the calculation of the ACV, giving examples where the GACV seemed to better approximate the generalization error than the ACV.

$$V_2 = \sum_{i=1}^{n} \frac{\mathcal{L}_{\alpha,\tau}\left(y_i - \tilde{f}_{\alpha,i}\right)}{n - Tr\left(\tilde{\mathbf{A}}_\alpha\right)} \tag{3.9}$$

Yuan [2006] also uses the approximation $\frac{1}{n}\mathcal{L}_{\alpha,\tau}\left(\frac{y_i - \tilde{f}_{\alpha,i}}{1 - \frac{Tr(\tilde{\mathbf{A}})}{n}}\right) \approx \frac{\mathcal{L}_{\alpha,\tau}\left(y_i - \tilde{f}_{\alpha,i}\right)}{n - Tr\left(\tilde{\mathbf{A}}\right)}$. The validity of this approximation is discussed in Tu et al. [2019] (see section 4.6.1). However, even if the GACV seems to improve the fits that use the ACV for central quantiles, Reiss and Huang [2012] note that the GACV seems to overfit non-central quantiles. We are going to take an example to illustrate the problems encountered with the QGACV and the ACV. Let's assume that the data $(\underline{x}_i, y_i)$ is generated according to the following distribution:

$$\begin{aligned}
\underline{x}_i &\sim \mathcal{U}[0,1], \quad i = 1, ..., n \\
y_i &\sim f_1(\underline{x}_i) + \epsilon_{A,i} \\
f_1(\underline{x}_i) &= 0.2 \times (\underline{x}_i)^{11} \times (10 \times (1 - \underline{x}_i))^6 + 10 \times (10 \times \underline{x}_i)^3 \times (1 - \underline{x}_i)^{10} \\
\epsilon_{A,i} &\sim AlphaStable(\alpha_{\alpha Stable} = 1.5, \beta_{\alpha Stable} = 0, \gamma_{\alpha Stable} = 1, \delta_{\alpha Stable} = 0)
\end{aligned}$$

where $\alpha_{\alpha Stable}$, $\beta_{\alpha Stable}$, $\gamma_{\alpha Stable}$, $\delta_{\alpha Stable}$ are the parameters of the alphaStable distribution. Function f_1 can be found in Wood [2006], Wood [2019]. (in this

---

[5]

$$\begin{aligned}
\lim_{\alpha \to 0} \mathcal{L}_{\alpha,\tau}(u) &= \lim_{\alpha \to 0} \tau(u + \gamma) + \alpha \log\left(1 + \exp\left(-\frac{u + \gamma}{\alpha}\right)\right) \\
&= \lim_{\alpha \to 0} \tau u + \alpha(\tau - 1)\log(1 - \tau) - \alpha\tau \log\tau + \alpha \log\left(1 - \tau + \tau \exp\left(-\frac{u}{\alpha}\right)\right) \\
&= \begin{cases} \tau u & \text{if } u > 0 \\ (\tau - 1)u & \text{if } u < 0 \\ 0 & \text{if } u - 0 \end{cases} \\
&= \mathcal{L}_\tau(u)
\end{aligned}$$

document, this model will be referred to as Model 2 (see section 9.1)).

We generate $n = 200$ samples from this model. We then fit a smoothing spline with 20 basis functions. We take $\alpha = 0.1$ (rounding parameter of the loss function). We will then calculate the ACV and GACV criteria on a grid and select the regularization parameters that lead to the minimum of the ACV and GACV curves. We take $rhoGrid = \{-25, -24.9, ..., 1.9, 2\}$. To estimate the vector of parameters $\boldsymbol{\beta}$, we could directly optimize the penalized empirical loss with a rounding level $\alpha = 0.1$. However, there is a chance that the algorithm would not find the global minimum of the penalized empirical loss. To avoid this issue, we use a technique that we will discuss in further details in section 5.2 and 7.11: the Generalized Iteratively Reweighted Least Squares (Bissantz et al. [2009]). This consists in optimizing the penalized empirical loss first with a large level of rounding (we start at $\alpha = 1$), optimize at this level using successive quadratic approximations to obtain $\tilde{\boldsymbol{\beta}}_{\alpha=1}$, then reduce α (for example we can divide the value of α by 2), and use $\tilde{\boldsymbol{\beta}}_{\alpha=1}$ as a starting point for the optimization of the penalized empirical loss rounded at $\alpha = 1/2 = 0.5$. We then iterate until we reach $\alpha = 0.1$ (gradually reducing a parameter that determines the smoothness of the surrogate loss is also referred to as progressive smoothing in Hahn et al. [2020]).

We can see the results on Figure 3.4. We note that for this example, the ACV overfits for central quantiles but not for extreme quantiles, whilst the GACV fits correctly the central quantile but not the extreme quantile 0.925. The actual LOOCV fits the model correctly for both $\tau = 0.5$ and $\tau = 0.925$. In the next sections, we will discuss this issue in further details and propose a method to reduce the issue of overfitting for extreme quantiles with the GACV.

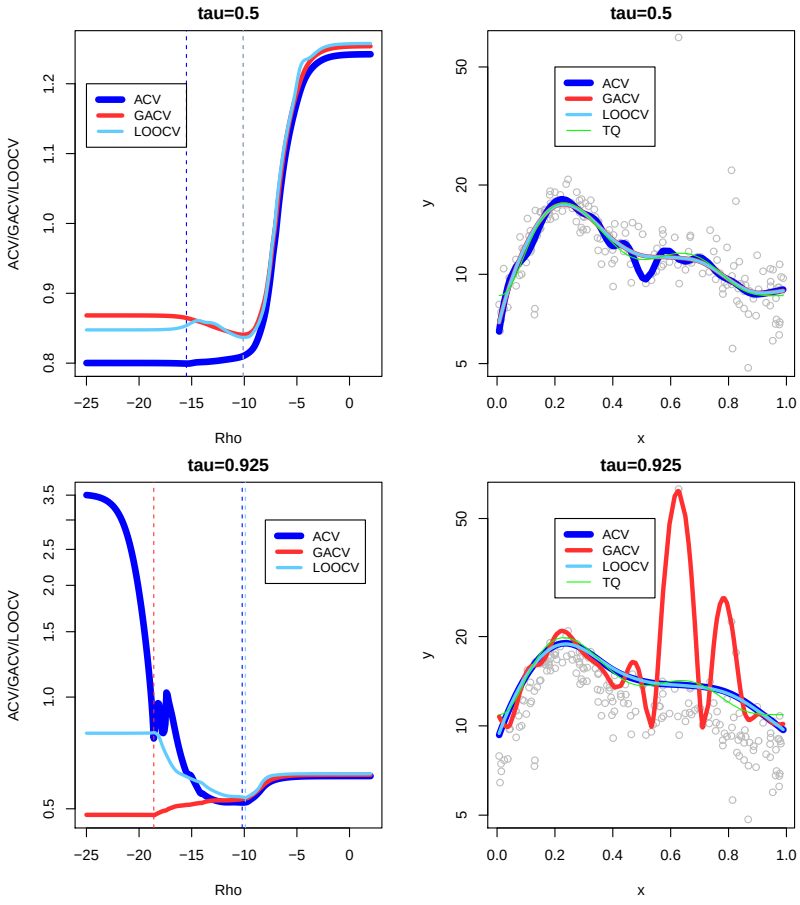


Fig. 3.4: On this Figure, we use Example 2 (see section 9.1), with $n = 200$ datapoints. We use a fixed rounding of $\alpha = 0.1$ and optimize three criteria: ACV, GACV and LOOCV on a grid of values: $rhoGrid = \{-25, -24.9, ..., 1.9, 2\}$. The vector of parameters $\boldsymbol{\beta}$ is estimated using Generalized IRLS (see sections 5.2 and 7.11). We fit a quantile smoothing spline with $m = 20$ with $\tau = 0.5$ (top row) and $\tau = 0.925$ (bottom row). On the top row for $\tau = 0.5$, on the left hand side, we can see the smoothing parameter selection criterion. We note that the LOOCV and the GACV select the same smoothing parameter whereas the ACV selects a less smooth fit. On the top row, right hand side, we can see that this results in fits with the LOOCV and GACV that are close to the true quantile (TQ in green) whilst the curve obtained using the ACV overfits. On the bottom line, for $\tau = 0.925$, on the left hand side, we can see that the ACV and the LOOCV select approximately the same degree of smoothness whilst the GACV selects a much less smooth fit. On the bottom right figure, we can see that this results in fits that approximately have the correct smoothness compared to the true quantile for ACV and LOOCV. The curve fit using GACV clearly overfits. To summarize, we here have an example where the LOOCV selects the correct degrees of smoothness for both $\tau = 0.5$ and $\tau = 0.925$, where the ACV overfits for central quantiles whilst correctly selecting the smoothness for the extreme quantile and where the GACV fits correctly the central quantile and clearly overfits the extreme quantile.

We are now going to investigate a potential explanation for the overfitting of non-central quantiles. We replicate an analysis made by Reiss and Huang [2012] with the example of Figure 3.4.

With the exact pinball loss, we can split the datapoints into three sets (see Li et al. [2007][6] and section 4.3):

- The "Below" set: $\mathcal{B} = \left\{ i \in 1, ..., n; y_i < \hat{f}_i \right\}$
- The "Above" set: $\mathcal{A} = \left\{ i \in 1, ..., n; y_i > \hat{f}_i \right\}$.
- The "Interpolated" points set: $\mathcal{I} = \left\{ i \in 1, ..., n; y_i = \hat{f}_i \right\}$.

However, if instead of using the exact pinball loss we use the surrogate loss $\mathcal{L}_{\alpha,\tau}$, no points are exactly interpolating the model. We then only have datapoints in $\mathcal{A} = \left\{ i \in 1, ..., n; y_i > \tilde{f}_{\alpha,i} \right\}$ and $\mathcal{B} = \left\{ i \in 1, ..., n; y_i < \tilde{f}_{\alpha,i} \right\}$. Reiss and Huang [2012] use a surrogate loss and argue that the relatively poor performance of the GACV for non-central quantiles is due to the fact that weights of the ACV summands ($\frac{1}{1-\tilde{a}_i}$) are very different in $\mathcal{A}$ compared to $\mathcal{B}$.

As discussed previously, as $\alpha \to 0$, some of the points will eventually interpolate the model. On Figure 3.4, we then separate the points in 3 groups. We plot the Above set $\tilde{u}_{\alpha,i} > 0$ in green, the Below set $\tilde{u}_{\alpha,i} < 0$ in blue. We also plot the points that would eventually interpolate the model as $\alpha \to 0$ in red.

We can see on the top line of Figure 3.4 that in this example, the GACV fits correctly for quantile $\tau = 0.5$. On the top line, we see that the ACV Weights $\frac{1}{1-\tilde{a}_i}$ and the ACV Summands $\frac{\sum_i \mathcal{L}_{\alpha,\tau}(\tilde{u}_{\alpha,i})}{1-\tilde{a}_i}$ seem to overlap for both sides of the model ($\tilde{u}_{\alpha,i} > 0$ in green and $\tilde{u}_{\alpha,i} < 0$ in blue). However, on the bottom line, we can see that for $\tau = 0.925$, the ACV Summands are quite different between one side of the model and the other. A lot of weight will then be given to a few observations in the Above Set compared to the Below Set. From this observation we can deduce that it is not appropriate to replace the weights by a single average as in Yuan [2006] to derive a GACV criterion for non-central quantiles. We could envisage to use a different average on both sides of the quantile. However, the location of the quantile is not known in advance. We would not know which datapoints correspond to which side. Furthermore, by using a different weight on each side, we would create a discontinuity in the criterion. In the next sections, we will derive instead a Quantile Generalized Cross Validation Criterion (QGACV) that will improve the fits for both the central quantiles and extreme quantiles. The example of Figure 3.4 is then re-fitted using the QGACV criterion on Figure 3.12. Before this, in the next section, we discuss Leave One Out Cross Validation (LOOCV) with the exact pinball loss.

[6] Li et al. [2007] refer to these sets as the left $\mathcal{L}$, right $\mathcal{R}$ and elbow $\mathcal{E}$ sets respectively.

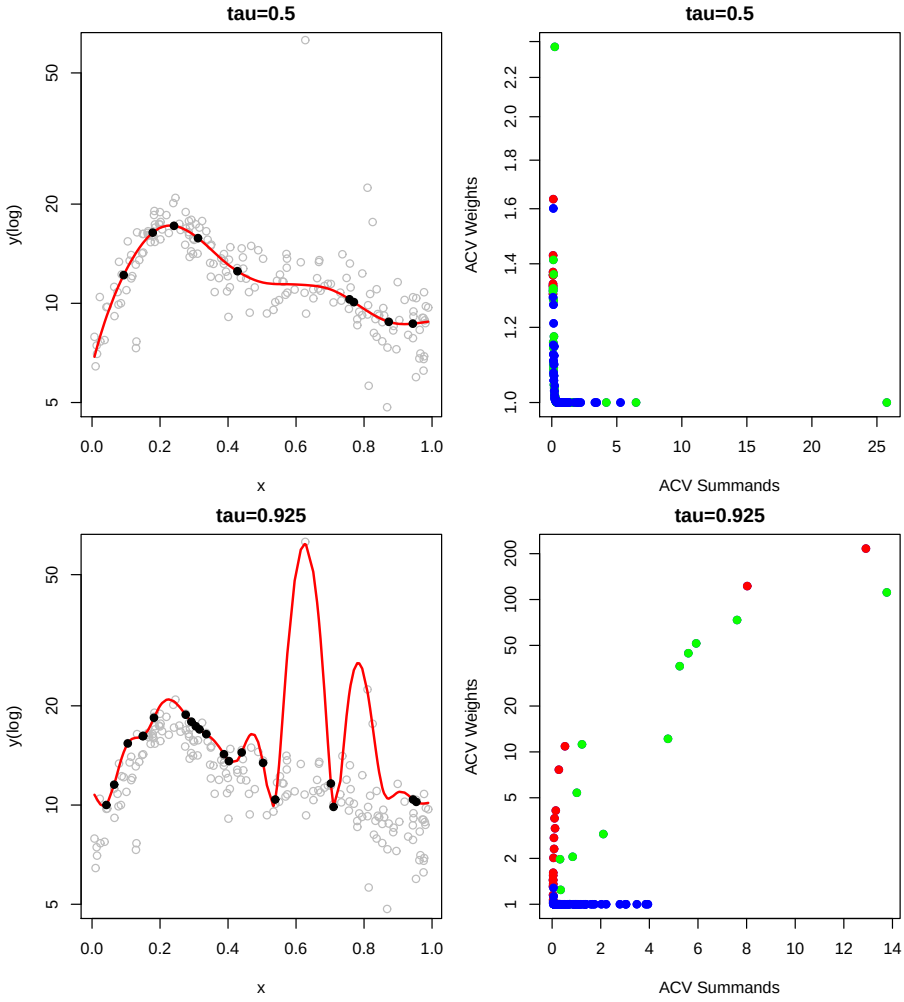


Fig. 3.5: This Figure is similar to the Figure in Reiss and Huang [2012]. We use a rounding $\alpha = 1 \times 10^{-1}$. On the top row, on the right hand side, we can see the ACV Summands $= \frac{\mathcal{L}_{\alpha,\tau}(y_i - \tilde{f}_{\alpha,i})}{1-\tilde{a}_i}$ on the x-axis vs the ACV Weights$=\frac{1}{1-\tilde{a}_i}$ on the y-axis. We can see on the top row, right that for the central quantile, the scatterplot of the ACV weights vs ACV summands overlaps for the Above set $(y_i > \tilde{f}_{\alpha,i})$ (green) and the Below set $(y_i < \tilde{f}_{\alpha,i})$ (blue). In red, we show the ACV summands and weights where the model will eventually interpolate the datapoints as $\alpha \to 0$. However, on the bottom right graph, we can see that for $\tau = 0.925$, the scatter plot of the Above set is quite different from the scatter plot of the Below set. This may be one reason why we observe a relative under-performance of the GACV for non-central quantiles.

3.3 Cross Validation with the pinball loss

Because the pinball loss is a non-differentiable function, it is relatively insensitive to small changes in the dataset. Although this property is usually seen as positive as the quantile regression estimator tends to be more robust to outliers than the quadratic loss, for the purpose of cross validation, this property is a disadvantage. Let's take an example with a linear quantile regression. We take a set of $n = 100$ data points that are generated using the following model:

$$\underline{x}_i \sim \mathcal{U}[0,1]; \quad i = 1, ..., n$$

$$y_i = \underline{x}_i + \underline{x}_i \times \epsilon_i; \quad \epsilon_i \sim \mathcal{N}(0,1)$$

We fit a linear quantile regression with package 'quantreg' (Koenker et al. [2021]). We then proceed to a leave one out scheme where we remove points one by one and refit the linear quantile regression. On Figure 3.6 the points that affect the fit are noted in blue. We can see that in this example, out of 100 datapoints, only 6 affect the fit. The removal of other datapoints do not change the fit. Furthermore, although we removed 100 points one by one alternatively, we can see that there are only 3 different fits obtained. The pinball loss function may therefore not be the most adapted for cross validation.

This issue of cross validation with the pinball loss or more generally with a non-differentiable loss has been discussed in Bengio and Grandvalet [2004], Efron and Tibshirani [1994] p.148, Jung et al. [2020]. Efron and Tibshirani [1994] pp.148-149 discuss the failure of the jackknife for the median and propose to use a delete-d jackknife procedure to reduce the issue. Jung et al. [2020] report the failure of cross validation for the pinball loss and propose to replace the loss by a rounded surrogate loss. They suggest a heuristic rule to obtain the level of rounding of the surrogate loss. By calculating the expected value of the difference between the LOOCV criterion based on the exact pinball loss and the LOOCV criterion based on the rounded surrogate loss, they show that the surrogate loss rounded where the rounding verifies certain conditions and the exact pinball loss are asymptotically equivalent.

As our aim will be to approximate the cross validation to avoid calculating it, and as it is easier to approximate the removal of one single point in the dataset than multiple points simultaneously, we will focus on obtaining a smoother loss as in Jung et al. [2020] rather using the method discussed in Efron and Tibshirani [1994] that, in our case could be expressed as using K-fold cross validation. To derive a smoother loss, in the next section, we discuss how the pinball loss can be replaced by its expected value.

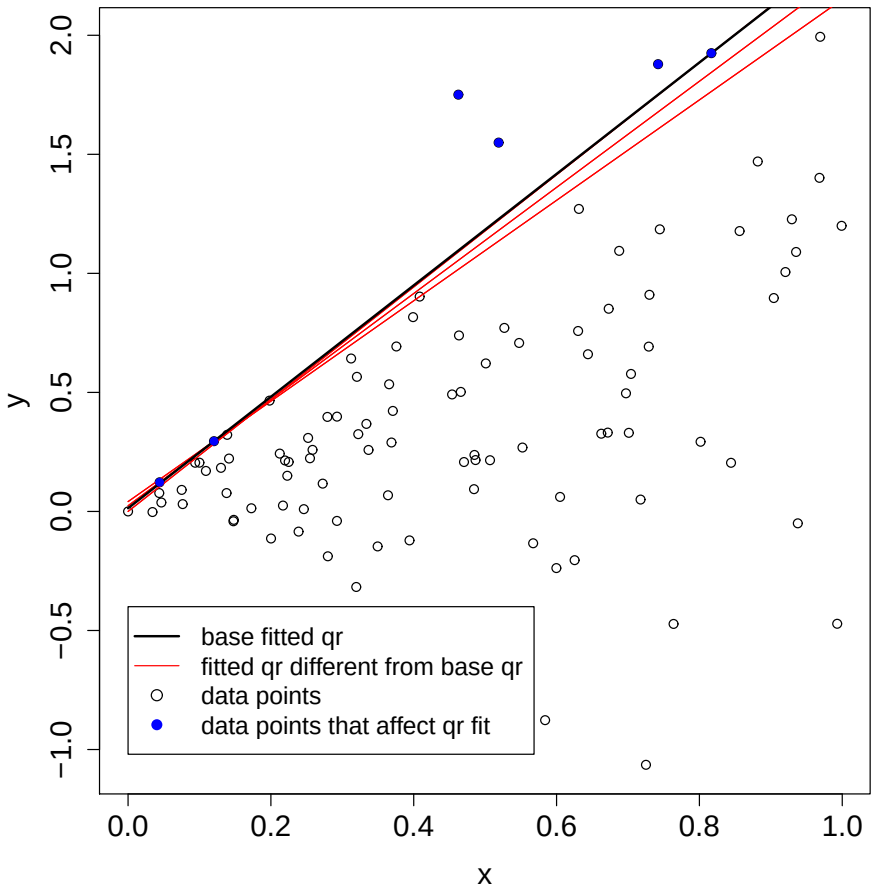


Fig. 3.6: In this example, we generate a sample from a heteroscedastic generating model $\underline{x}_i \sim \mathcal{U}[0,1]; \quad i = 1, ..., n$, $y_i = \underline{x}_i + \underline{x}_i \times \epsilon_i; \quad \epsilon_i \sim \mathcal{N}(0,1)$. We then fit a linear quantile regression at $\tau = 0.95$. We proceed to a leave one out scheme where we remove each of the $n = 100$ datapoints one by one and refit the quantile regression. We observe that 94 out of 100 datapoints do not change the fit at all. The points whose removal changes the fit are indicated in blue. We also note we only obtain 3 different fits. Note that Efron and Tibshirani [1994] p.148 take a similar example and show that calculating the jackknife for the median results in very few different values. The standard error is then undervalued.

3.4 Replacing the loss function by its expected value

In the previous section, we discussed how the exact pinball loss may not be appropriate to calculate the LOOCV criterion. In this section, we propose to replace the pinball loss function by its expected value to calculate the change in fits resulting from the removal of a datapoint. Let's go back to the LOOCV criterion from equation 3.1:

$$V_0(\boldsymbol{\lambda}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - \hat{f}_{\boldsymbol{\lambda}}^{[-i]}(\underline{\boldsymbol{x}}_i) \right)$$

First, V_0 can be rewritten:

$$V_0(\boldsymbol{\lambda}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - \hat{f}_{\boldsymbol{\lambda}}(\underline{\boldsymbol{x}}_i) + \left(\hat{f}_{\boldsymbol{\lambda}}(\underline{\boldsymbol{x}}_i) - \hat{f}_{\boldsymbol{\lambda}}^{[-i]}(\underline{\boldsymbol{x}}_i) \right) \right)$$

Where $\left(\hat{f}_{\boldsymbol{\lambda}}(\underline{\boldsymbol{x}}_i) - \hat{f}_{\boldsymbol{\lambda}}^{[-i]}(\underline{\boldsymbol{x}}_i) \right) = \mathbf{x}_i^T \left(\hat{\boldsymbol{\beta}}_{\boldsymbol{\lambda}} - \hat{\boldsymbol{\beta}}_{\boldsymbol{\lambda}}^{[-i]} \right)$ is the change in the fit by removing point i. We then propose to replace the difference $\left(\hat{\boldsymbol{\beta}}_{\boldsymbol{\lambda}} - \hat{\boldsymbol{\beta}}_{\boldsymbol{\lambda}}^{[-i]} \right)$ by $\left(\hat{\boldsymbol{\beta}}_{E,\boldsymbol{\lambda}} - \hat{\boldsymbol{\beta}}_{E,\boldsymbol{\lambda}}^{[-i]} \right)$, the difference in the vector of parameters estimated under the expected loss under the true distribution, where :

$$\hat{\boldsymbol{\beta}}_{E,\boldsymbol{\lambda}} \in \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} E_{y_i|\underline{\boldsymbol{x}}_i} \left[\mathcal{L}_\tau \left(y_i - \mathbf{x}_i^T \boldsymbol{\beta} \right) \right] + \frac{1}{2}\boldsymbol{\beta}^T S_{\boldsymbol{\lambda}} \boldsymbol{\beta}$$

The expected loss can be seen as a legitimate alternative to the loss itself. Note the link between our framework and the uncertain observation Least Absolute Deviation (LAD) framework of Liu and Yang [2020]. This framework is proposed for cases where both the observations of the predicted value and the predictors are uncertain and the loss function used is the LAD. The absolute value loss is replaced by its expected value over the joint distribution of y_i and $\underline{\boldsymbol{x}}_i$. In our case, we assume that $\underline{\boldsymbol{x}}_i$ is known and calculate the expectation over y_i. In the framework of Liu and Yang [2020] this would mean that we assume that we have an uncertainty on the predicted value y_i. We then replace the difference in predicted values between the original estimate and the leave one out estimated value $\left(\hat{f}_{\boldsymbol{\lambda}}(\boldsymbol{x}_i) - \hat{f}_{\boldsymbol{\lambda}}^{[-i]}(\underline{\boldsymbol{x}}_i) \right)$ by $\left(\hat{f}_{E,\boldsymbol{\lambda}}(\underline{\boldsymbol{x}}_i) - \hat{f}_{E,\boldsymbol{\lambda}}^{[-i]}(\underline{\boldsymbol{x}}_i) \right)$ in V_0, where $\hat{f}_{E,\boldsymbol{\lambda}}(\underline{\boldsymbol{x}}_i) = \mathbf{x}_i^T \hat{\boldsymbol{\beta}}_{E,\boldsymbol{\lambda}}$ and $\hat{f}_{E,\boldsymbol{\lambda}}^{[-i]}(\underline{\boldsymbol{x}}_i) = \mathbf{x}_i^T \hat{\boldsymbol{\beta}}_{E,\boldsymbol{\lambda}}^{[-i]}$.

$$V_{0,E}(\boldsymbol{\lambda}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - \hat{f}_{\boldsymbol{\lambda}}(\underline{\boldsymbol{x}}_i) + \left(\hat{f}_{E,\boldsymbol{\lambda}}(\underline{\boldsymbol{x}}_i) - \hat{f}_{E,\boldsymbol{\lambda}}^{[-i]}(\underline{\boldsymbol{x}}_i) \right) \right) \tag{3.10}$$

To see the effect of taking the expected value on the loss function, let's take a one dimensional example. We generate the data as follows:

$$\underline{x}_i \sim \mathcal{U}[0,1]; \quad i = 1, ..., n$$

$$y_i = \underline{x}_i + \epsilon_i; \quad \epsilon_i \sim \mathcal{N}(0,1)$$

We take $n = 25$ and $\tau = 0.3$. We then assume that we are trying to fit a univariate quantile regression model:

$$\mathbb{P}(Y \leq \tau | X = \underline{x}_i) = \beta \underline{x}_i; \quad i = 1, ..., n$$

and observe the profile of the loss function as we vary parameter β. We take values of β on a grid: $\beta_k, k = 1, ..., K$. We plot the individual functions

$$\mathcal{L}_\tau \left(y_i - \beta_k \underline{x}_i \right)$$

in dotted grey for each value of $(\underline{x}_i, y_i)$. As a surrogate to the expected loss, we are then going to plot in red the empirical loss:

$$EL \left(\beta_k \right) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - \beta_k \underline{x}_i \right)$$

We note on Figure 3.7 that the empirical loss as a function of β is piecewise linear. However, we see that the overall shape is similar to a rounded loss function and we may be able to approximate it with a smooth rounded loss. Using this type of rounded loss may then help with the LOOCV.

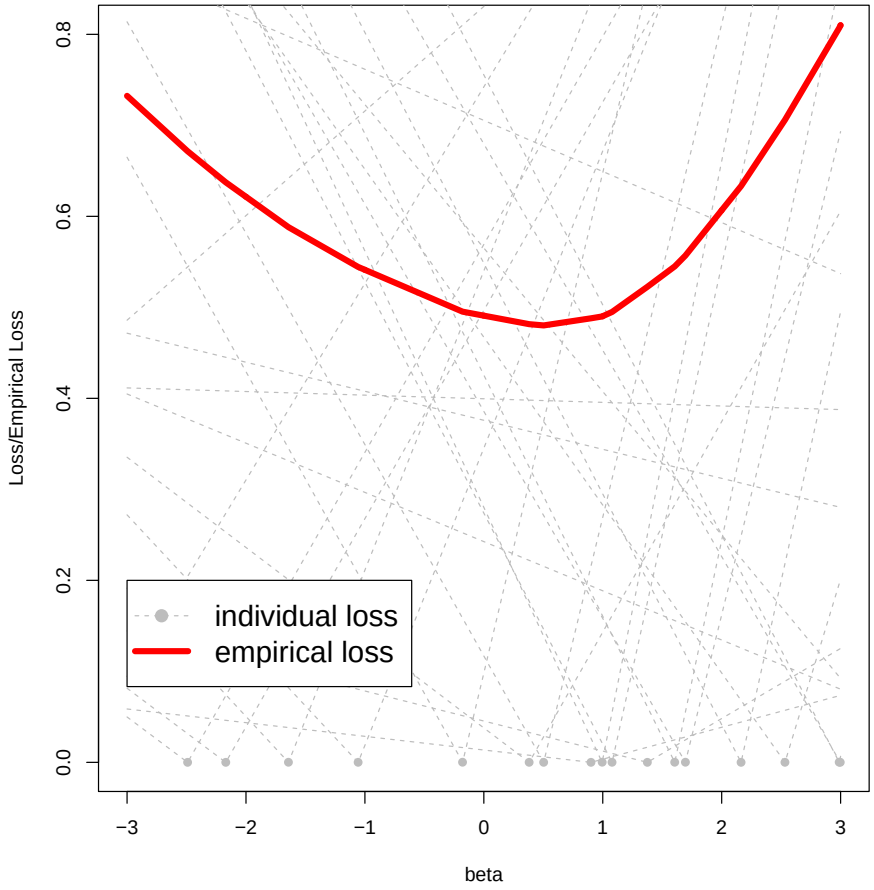


Fig. 3.7: This picture shows the empirical loss (in red) as the sum of the individual pinball losses (in grey). The empirical loss is a piecewise linear function. We use $n = 25$, $\tau = 0.3$.

If we knew the distribution of $y|\underline{x}$, we could use the expected loss. However, this distribution is unknown. We propose instead to replace the expected value with the surrogate loss of equation 3.7. This surrogate loss can be tuned with a parameter α. To find the loss function that best approximates the expected loss, we are going to find the α that leads to the minimum squared distance between the surrogate loss and the expected loss itself replaced by an empirical loss (as we do not know the underlying distribution).

Let's start with an illustrative 1 dimensional example. As we are going to

optimize on parameter β, the expected loss and the surrogate loss should match for a range of values of β. As previously, we take values of β on a grid: $\beta_1, ..., \beta_K$ (for example values between -3 and +3, spaced by 0.001). In this setting, we want to find the α that minimizes the squared error between the empirical loss and the surrogate loss over all points of the grid. This is then written:

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmin}} \frac{1}{nK} \sum_{k=1}^{K} \sum_{j=1}^{n} \left(\mathcal{L}_{\alpha,\tau} \left(y_j - \underline{x}_j \beta_k \right) - \frac{1}{n} \sum_{i=1}^{n} \left[\mathcal{L}_{\tau} \left(y_i - \underline{x}_i \beta_k \right) \right] \right)^2 \quad (3.11)$$

We optimize on a grid of values for α: $alphaGrid = \left\{ 10^{-10}, 10^{-9.9}, 10^{-9.8}, ..., 10^{0.9}, 10^{1} \right\}$. The top chart of Figure 3.8 shows the squared distance from equation 3.11 as alpha changes.

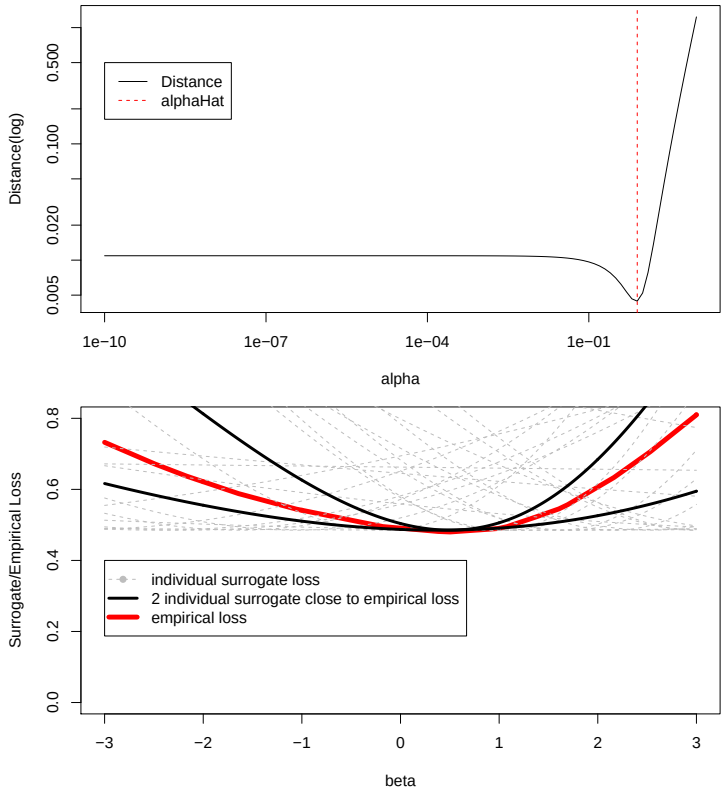


Fig. 3.8: On this picture, we can see how the expected loss can be approximated with the surrogate loss. We use $n = 25$, $\tau = 0.3$. On the bottom chart, we can see in red, the expected loss approximated by the empirical loss, in grey, we can see different surrogate losses $\mathcal{L}_{\alpha,\tau} \left(y_j - \underline{x}_j \beta_k \right)$ for all the possible values of $\underline{x}_j$. The black curves are the 2 surrogate loss curves the closest to the expected loss.

To extend this to a vector of parameters $\boldsymbol{\beta}$, we could repeat the same analysis with a multidimensional grid of values for each direction of the vector of parameters $\boldsymbol{\beta}$. However, noting that vector $\boldsymbol{\beta}$ can have a large number of coordinates (for example 100 or more), we would need to estimate the criterion over a 100 dimensional grid. This would not be very practical. Furthermore, we note that function $\mathcal{L}_\tau$ can simply be seen as a function of a single variable. The effect of each coordinate β_j on the loss function is going to be similar. We then propose the following approach. First, let's call the current true conditional quantile $f_{\tau,TQ}$. The expected value of the loss function with parameter τ estimated at the true quantile is then:

$$E_{Y|X}\left[\mathcal{L}_\tau\left(Y - f_{\tau,TQ}(X)\right)|X\right] \approx \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_\tau\left(y_i - \hat{f}\left(\underline{\boldsymbol{x}}_i\right)\right) \tag{3.12}$$

To obtain an estimate of $\hat{f}\left(\underline{\boldsymbol{x}}_i\right), i = 1, ..., n$, we obtain a pilot fit by running the algorithms described in sections 7.1, 7.2 and 7.3. To calculate the distance between the surrogate loss and the expected loss, similarly to what we have done in 1 dimension, we are going to estimate the distance at a series of functions, shifted away from true conditional quantile $f_{\tau,TQ}$:

$$\begin{aligned} E_{Y|X}\left[\mathcal{L}_\tau\left(Y - f_{\tau,TQ}(X) + \Delta_f\right)|X\right] &\approx \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_\tau\left(y_i - \hat{f}\left(\underline{\boldsymbol{x}}_i\right) + \Delta_f\right) \\ &\approx \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_\tau\left(\hat{u}_i + \Delta_f\right) \end{aligned}$$

To estimate the degrees of smoothness of the surrogate loss, we are going to find the α that minimizes the squared distance between the expected loss and the surrogate loss for a series of J shift values for $\Delta_{j,f}$: $\Delta_{1,f}, ..., \Delta_{J,f}$. The squared distance between the surrogate loss and the expected loss can be expressed as:

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmin}}\frac{1}{J}\sum_{j=1}^{J}\left(\mathcal{L}_{\alpha,\tau}\left(\Delta_{j,f}\right) - E_{Y|X}\left[\mathcal{L}_\tau\left(Y - f_{\tau,TQ}(X) + \Delta_{j,f}\right)|X\right]\right)^2 \tag{3.13}$$

replacing by the empirical loss, we obtain:

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmin}}\frac{1}{J}\sum_{j=1}^{J}\left(\mathcal{L}_{\alpha,\tau}\left(\Delta_{j,f}\right) - \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_\tau\left(\hat{u}_i + \Delta_{j,f}\right)\right)^2 \tag{3.14}$$

Noting that we are interested in the distance between the expected loss (itself replaced by an empirical loss) and the surrogate loss only at points where we are going to evaluate the surrogate loss, i.e. at points $\hat{u}_j$, $j = 1, ..., n$. we propose to use these values for the $\Delta_{j,f}$, and take $J = n$ and $\Delta_{j,f} = \hat{u}_j$. This leads to the following expression to estimate $\hat{\alpha}$:

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmin}}\frac{1}{n}\sum_{j=1}^{n}\left(\mathcal{L}_{\alpha,\tau}\left(\hat{u}_j\right) - \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_\tau\left(\hat{u}_i + \hat{u}_j\right)\right)^2 \tag{3.15}$$

On Figure 3.9, we show how Formula 3.15 can be used to find the surrogate loss that matches best the expected loss. The selected rounding is called $\hat{\alpha}$. We can then look at the effect of using the surrogate loss rounded at $\hat{\alpha}$. On the bottom chart of Figure 3.9, we note that the squared distance has a sharp minimum. This behaviour can be expected as long as the surrogate loss can match the expected loss closely (replaced by an empirical loss). In Appendix B we give two examples. In the first example, we show that if we calculate the squared distance between the surrogate loss rounded with a fixed value of parameter $\alpha = 0.5$ and the same surrogate loss rounded at α varying between 10^{-3} and 10^{1}, we also observe a sharp minimum. We could then expect this type of behaviour for formula 3.15 as long as the surrogate loss is able to match the expected loss evaluated using an empirical loss. We show a second example where the data is bimodal. In this case, the surrogate loss cannot match the expected loss replaced by an empirical loss with any level of rounding α. In this case, we still have a minimum of the squared distance, however, the minimum is less sharp.

Figure 3.10 shows the same example as on Figure 3.6 fitted using surrogate loss $\mathcal{L}_{\hat{\alpha},\tau}$ (equation 3.7), where $\hat{\alpha}$ is the rounding parameter determined using equation 3.15. We note that now all datapoints affect the fit when removed. This change helps improve the LOOCV. We are going to keep this change and derive an approximation for the cross validation criterion. Finally, in this document, we will use the following notation to designate the estimate of the vector of parameters with the surrogate loss rounded at $\hat{\alpha}$:

$$\tilde{\boldsymbol{\beta}}_{\boldsymbol{\lambda}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{\hat{\alpha},\tau} \left(y_i - \mathbf{x}_i^T \boldsymbol{\beta} \right) + \frac{1}{2} \boldsymbol{\beta}^T S_{\boldsymbol{\lambda}} \boldsymbol{\beta} \tag{3.16}$$

where the $\in$ sign has been replaced by an $=$ sign as we are now using the smooth surrogate instead of the exact pinball loss. After replacing in $V_{0,E}$ (equation 3.10), we obtain:

$$\tilde{V}_0 = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{\tau} \left(y_i - \hat{f}_i + \left(\tilde{f}_{\hat{\alpha},i} - \tilde{f}_{\hat{\alpha},i}^{[-i]} \right) \right) \tag{3.17}$$

Note that $\tilde{f}_{\hat{\alpha},i}$ is a biased estimate of the predicted value. In the next section, we will derive an approximation of $\tilde{f}_{\hat{\alpha},i}^{[-i]}$ as a function of $\tilde{f}_{\hat{\alpha},i}$. To reduce the bias, instead of using $\left(\tilde{f}_{\hat{\alpha},i} - \tilde{f}_{\hat{\alpha},i}^{[-i]} \right)$, we are going to calculate $\left(\tilde{f}_{\hat{\alpha},i} - \tilde{f}_{\hat{\alpha},i}^{[-i]} \right)_{|\tilde{f}_{\hat{\alpha},i} = \hat{f}_i}$, the difference $\left(\tilde{f}_{\hat{\alpha},i} - \tilde{f}_{\hat{\alpha},i}^{[-i]} \right)$ evaluated at $\tilde{f}_{\hat{\alpha},i} = \hat{f}_i$. This is the evaluation of the difference as if the minimum of the rounded loss function was $\hat{f}_i$ instead of $\tilde{f}_{\hat{\alpha},i}$. After this change, the LOOCV is then:

$$\tilde{\tilde{V}}_0 = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{\tau} \left(y_i - \hat{f}_i + \left(\tilde{f}_{\hat{\alpha},i} - \tilde{f}_{\hat{\alpha},i}^{[-i]} \right) |_{\tilde{f}_{\hat{\alpha},i} = \hat{f}_i} \right) \tag{3.18}$$

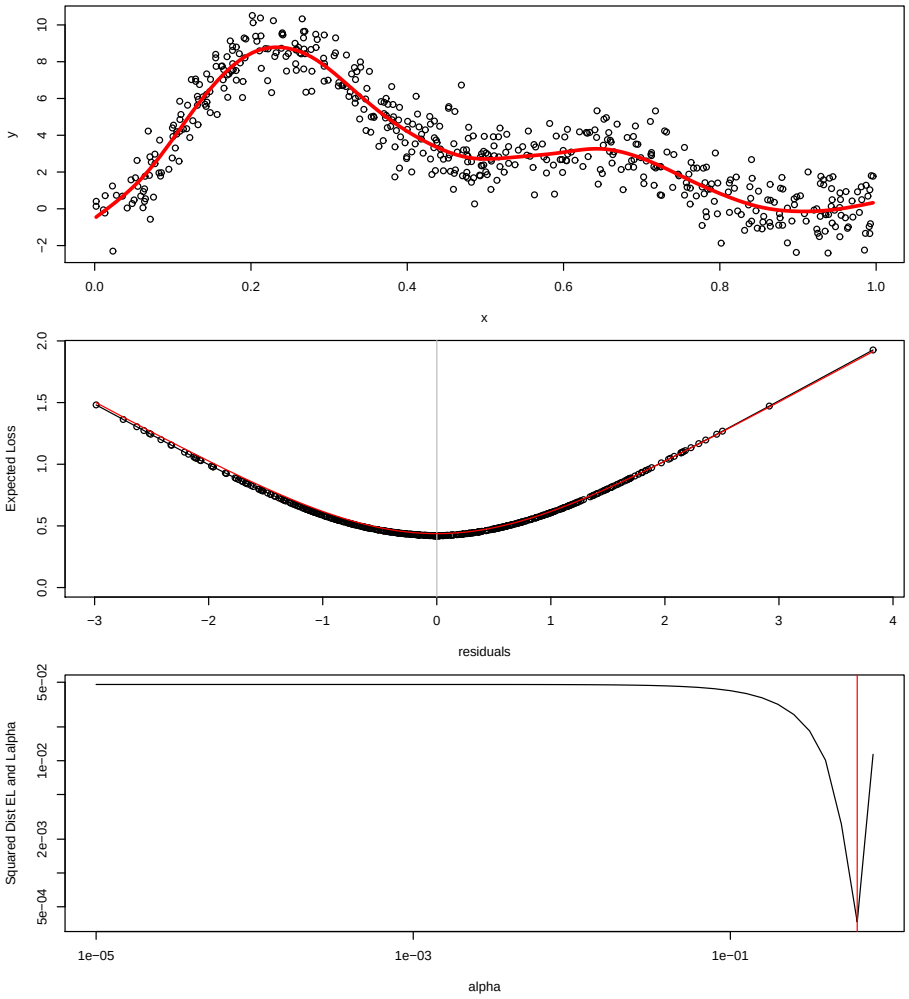


Fig. 3.9: This picture shows an example of $\hat{\alpha}$ selected using equation 3.15. The top chart shows the fit. On the middle chart, the dots are the $\hat{u}_i$. The red line is the chosen surrogate loss. The black line is the expected loss evaluated as the empirical loss. We note that in this example, the surrogate loss matches well with the expected loss. On the bottom chart, we can see the squared distance between the expected loss and the surrogate loss as α increases. The chosen $\hat{\alpha}$ is indicated in red.

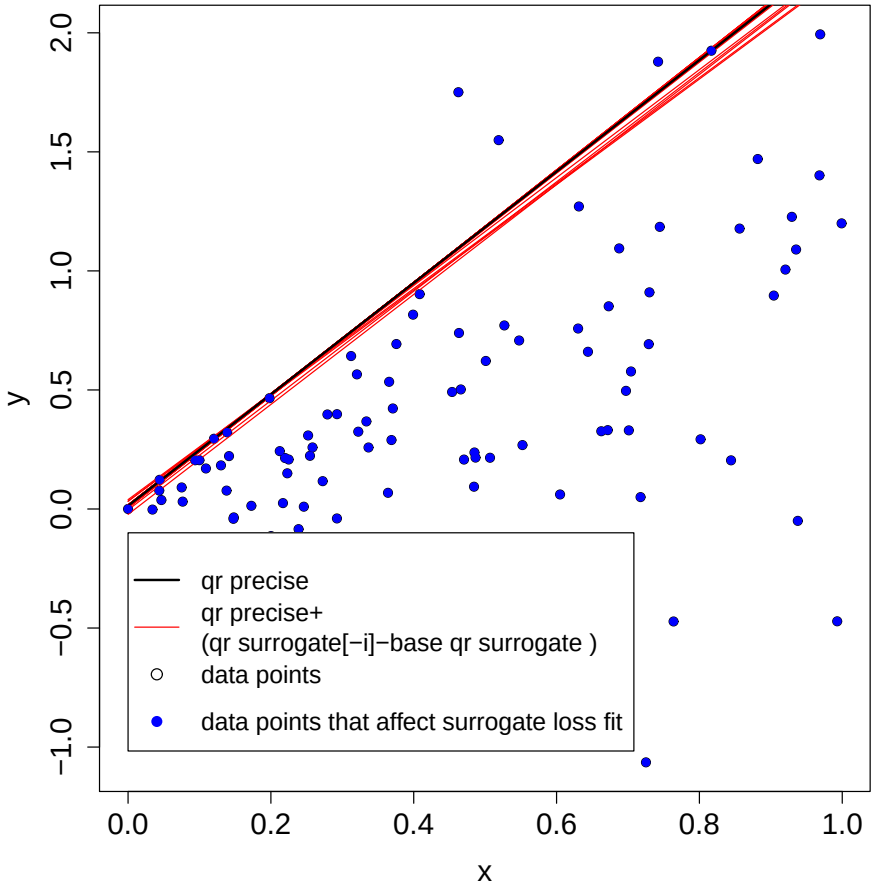


Fig. 3.10: This Figure uses the same setting as Figure 3.6. The difference is that here, we use a surrogate loss with a rounding level $\alpha = 0.2$. The rounding level chosen corresponds to $\hat{\alpha}$ calculated using Formula 3.15 for this dataset. As the fit with $\alpha = 0.2$ is biased, we show the following curves: first the black line corresponds to a precise fit with a very low level of rounding, close to the exact pinball loss. Then the red curves do not show the LOOCV curves with the surrogate loss rounded at $\alpha = 0.2$ but the original precise fit close to the exact pinball loss + (the fit with the surrogate loss rounded at $\alpha = 0.2$ where point has been removed) - (the base fit with the surrogate loss rounded at $\alpha = 0.2$ where all points are used to fit the model). Using this method, we are reducing the bias due to the rounding. The datapoints that affect the surrogate loss fit when removed are shown in blue. We can see that with the rounding of $\alpha = 0.2$, every datapoint affects the fit. This contrasts with the LOOCV calculated with the exact pinball loss where the removal of only a few datapoints were affecting the fit.

3.5 Derivation of an Approximate Cross Validation Criterion

We will now use a new notation. We will introduce a hat matrix where the rounding used to calculate the weight matrix is different from the rounding used to estimate the vector of parameters. We will then have two indices α_1 and α_2 indicating the first and second rounding. Because we will mostly use this hat matrix for the rest of this document, we drop the tilde from the notation. We will then use:

$$a_{\alpha_1,\alpha_2,i} = \left(\frac{\partial \tilde{f}_{\alpha_1,i}}{\partial y_i} \right)_{|\tilde{f}_{\alpha_1,i}=\tilde{f}_{\alpha_2,i}} \tag{3.19}$$

That is the derivative of the predicted value with the surrogate loss with a rounding of α_1 with respect to y_i calculated away from the minimum of the penalized empirical loss using the surrogate loss rounded at α_1. This derivative is evaluated at the predicted value resulting from the optimum of the penalized empirical loss using the surrogate loss rounded at α_2.

We also use the following notation:

$$a_{\alpha_1,0,i} = \lim_{\alpha \to 0} a_{\alpha_1,\alpha,i} = \left(\frac{\partial \tilde{f}_{\alpha_1,i}}{\partial y_i} \right)_{|\tilde{f}_{\alpha_1,i}=\lim\limits_{\alpha\to 0}\tilde{f}_{\alpha,i}} = \left(\frac{\partial \tilde{f}_{\alpha_1,i}}{\partial y_i} \right)_{|\tilde{f}_{\alpha_1,i}=\hat{f}_i}$$

That is the derivative of the predicted value with the surrogate loss with a rounding of α_1 with respect to y_i calculated at the predicted value at the minimum of the penalized empirical loss based on the exact pinball loss. We will also use the short hand notation:

$$a_i = a_{\hat{\alpha},0,i} = \lim_{\alpha \to 0} a_{\hat{\alpha},\alpha,i} = \left(\frac{\partial \tilde{f}_{\hat{\alpha},i}}{\partial y_i} \right)_{|\tilde{f}_{\hat{\alpha},i}=\lim\limits_{\alpha\to 0}\tilde{f}_{\alpha,i}} = \left(\frac{\partial \tilde{f}_{\hat{\alpha},i}}{\partial y_i} \right)_{|\tilde{f}_{\hat{\alpha},i}=\hat{f}_i}$$

That is the short hand notation a_i designates the sensitivity of the predicted value to y_i obtained with a penalized empirical loss using the surrogate loss with a rounding of $\hat{\alpha}$ (Formula 3.15) calculated at the predicted value estimated with the penalized empirical loss with the exact pinball loss $\hat{f}_i$.

Note then that by definition, $\tilde{a}_{\alpha,i} = a_{\alpha,\alpha,i}$. Following the same template, we also drop the tilde from hat matrix $\mathbf{A}$, diagonal weight matrix $\mathbf{W}$ and diagonal

components of the weight matrix w.

$$\begin{aligned}
\mathbf{A}_{\alpha_1,\alpha_2} &= \mathbf{X}\left(\mathbf{X}^T\mathbf{W}_{\alpha_1,\alpha_2}\mathbf{X}+\mathbf{S}_{\boldsymbol{\lambda}}\right)^{-1}\left(\mathbf{X}^T\mathbf{W}_{\alpha_1,\alpha_2}\right)\\
\mathbf{A}_{\alpha_1,0} &= \lim_{\alpha\to 0}\mathbf{A}_{\alpha_1,\alpha} = \lim_{\alpha\to 0}\mathbf{X}\left(\mathbf{X}^T\mathbf{W}_{\alpha_1,\alpha}\mathbf{X}+\mathbf{S}_{\boldsymbol{\lambda}}\right)^{-1}\left(\mathbf{X}^T\mathbf{W}_{\alpha_1,\alpha}\right)\\
\mathbf{A} = \mathbf{A}_{\hat{\alpha},0} &= \lim_{\alpha\to 0}\mathbf{A}_{\hat{\alpha},\alpha} = \lim_{\alpha\to 0}\mathbf{X}\left(\mathbf{X}^T\mathbf{W}_{\hat{\alpha},\alpha}\mathbf{X}+\mathbf{S}_{\boldsymbol{\lambda}}\right)^{-1}\left(\mathbf{X}^T\mathbf{W}_{\hat{\alpha},\alpha}\right)\\
\mathbf{W}_{\alpha_1,\alpha_2} &= diag\left(w_{\alpha_1,\alpha_2,1},...,w_{\alpha_1,\alpha_2,n}\right)\\
\mathbf{W}_{\alpha_1,0} &= \lim_{\alpha\to 0}\mathbf{W}_{\alpha_1,\alpha} = \lim_{\alpha\to 0} diag\left(w_{\alpha_1,\alpha,1},...,w_{\alpha_1,\alpha,n}\right)\\
\mathbf{W} = \mathbf{W}_{\hat{\alpha},0} &= \lim_{\alpha\to 0}\mathbf{W}_{\hat{\alpha},\alpha} = \lim_{\alpha\to 0} diag\left(w_{\hat{\alpha},\alpha,1},...,w_{\hat{\alpha},\alpha,n}\right)...\\
... &= diag\left(w_{\hat{\alpha},0,1},...,w_{\hat{\alpha},0,n}\right) = diag\left(w_1,...,w_n\right)\\
w_{\alpha_1,\alpha_2,i} &= \frac{\partial^2\mathcal{L}_{\alpha_1,\tau}(u)}{\partial u^2}|_{u=\tilde{u}_{\alpha_2,i}}\\
w_{\alpha_1,0,i} &= \lim_{\alpha\to 0} w_{\alpha_1,\alpha,i} = \frac{\partial^2\mathcal{L}_{\alpha_1,\tau}(u)}{\partial u^2}|_{u=\lim\limits_{\alpha\to 0}\tilde{u}_{\alpha,i}} = \frac{\partial^2\mathcal{L}_{\alpha_1,\tau}(u)}{\partial u^2}|_{u=\hat{u}_i}\\
w_i = w_{\hat{\alpha},0,i} &= \lim_{\alpha\to 0} w_{\hat{\alpha},\alpha,i} = \frac{\partial^2\mathcal{L}_{\hat{\alpha},\tau}(u)}{\partial u^2}|_{u=\lim\limits_{\alpha\to 0}\tilde{u}_{\alpha,i}} = \frac{\partial^2\mathcal{L}_{\hat{\alpha},\tau}(u)}{\partial u^2}|_{u=\hat{u}_i}
\end{aligned}$$

As in the case of a_i, we use a short hand notation: $\mathbf{A}$ for $\mathbf{A}_{\hat{\alpha},\mathbf{0}}$, the hat matrix evaluated with a rounding of $\hat{\alpha}$ but where the vector of parameter is estimated using the exact pinball loss, $\mathbf{W}$ for $\mathbf{W}_{\hat{\alpha},\mathbf{0}}$ and w_i for $w_{\hat{\alpha},0,i}$. After this introduction to a new notation, let's now go back to the LOOCV. The LOOCV criterion derived in equation 3.18 is:

$$\tilde{\tilde{V}}_0 = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_\tau\left(y_i-\hat{f}_i+\left(\tilde{f}_{\hat{\alpha},i}-\tilde{f}^{[-i]}_{\hat{\alpha},i}\right)_{|\tilde{f}_{\hat{\alpha},i}=\hat{f}_i}\right) \tag{3.20}$$

Evaluation of this criterion would require the re-estimation of the model n times. To avoid these multiple calculations, in this section, we derive an Approximate Cross Validation (ACV) criterion. The LOOCV estimated vector of parameters is:

$$\tilde{\boldsymbol{\beta}}^{[-i]} = \text{argmin}_{\boldsymbol{\beta}}\frac{1}{n}\sum_{j=1,j\neq i}^{n}\mathcal{L}_{\hat{\alpha},\tau}\left(y_j-\mathbf{x}_j^T\boldsymbol{\beta}\right)+\mathcal{P}_{\boldsymbol{\lambda}}(\boldsymbol{\beta}) \tag{3.21}$$

Xiang and Wahba [1996] study the LOOCV for a Generalized Linear Models. They note that the LOOCV can be calculated by equating observation i with the leave one out predicted value itself. Yuan [2006] uses this to derive an ACV criterion for QSS. Because our loss function $\mathcal{L}_{\hat{\alpha},\tau}$ has its minimum at 0, we can use the same technique. We note that do not change the minimum by adding $\mathcal{L}_{\hat{\alpha},\tau}\left(\mathbf{x}_i^T\tilde{\boldsymbol{\beta}}^{[-i]}-\mathbf{x}_i^T\boldsymbol{\beta}\right)$ to the function. Hence:

$$\tilde{\boldsymbol{\beta}}^{[-i]} = \text{argmin}_{\boldsymbol{\beta}}\frac{1}{n}\sum_{j=1,j\neq i}^{n}\mathcal{L}_{\hat{\alpha},\tau}\left(y_j-\mathbf{x}_j^T\boldsymbol{\beta}\right)+\mathcal{L}_{\hat{\alpha},\tau}\left(\mathbf{x}_i^T\tilde{\boldsymbol{\beta}}^{[-i]}-\mathbf{x}_i^T\boldsymbol{\beta}\right)+\mathcal{P}_{\boldsymbol{\lambda}}(\boldsymbol{\beta}) \tag{3.22}$$

This means that the LOOCV can be obtained by an implicit equation, replacing y_i by $\mathbf{x}_i^T \tilde{\boldsymbol{\beta}}^{[-i]}$. We can use this to our advantage. If function $\tilde{f}_{\hat{\alpha},i}(y_i)$ is approximately linear around the current fit and $\tilde{f}_{\hat{\alpha},i}^{[-i]}(y_i)$ is on the same side of y_i as $\tilde{f}_{\hat{\alpha},i}(y_i)$[7], we have :

$$\begin{aligned}
\tilde{a}_{\hat{\alpha},i} =& \frac{\partial \tilde{f}_{\hat{\alpha},i}}{\partial y_i} \approx \frac{\left(\tilde{f}_{\hat{\alpha},i} - \tilde{f}_{\hat{\alpha},i}^{[-i]}\right)}{\left(y_i - \tilde{f}_{\hat{\alpha},i}^{[-i]}\right)} \\
&\Rightarrow \left(\tilde{f}_{\hat{\alpha},i} - \tilde{f}_{\hat{\alpha},i}^{[-i]}\right) \approx \tilde{a}_{\hat{\alpha},i}\left(y_i - \tilde{f}_{\hat{\alpha},i}^{[-i]}\right) = \tilde{a}_{\hat{\alpha},i}\left(y_i - \tilde{f}_{\hat{\alpha},i} + \tilde{f}_{\hat{\alpha},i} - \tilde{f}_{\hat{\alpha},i}^{[-i]}\right) \\
&\Rightarrow \left(\tilde{f}_{\hat{\alpha},i} - \tilde{f}_{\hat{\alpha},i}^{[-i]}\right) \approx \frac{\tilde{a}_{\hat{\alpha},i}\left(y_i - \tilde{f}_{\hat{\alpha},i}\right)}{1 - \tilde{a}_{\hat{\alpha},i}}
\end{aligned} \tag{3.23}$$

Then to reduce the bias due to the rounding of the loss function, we need to estimate $\left(\tilde{f}_{\hat{\alpha},i} - \tilde{f}_{\hat{\alpha},i}^{[-i]}\right)_{|\tilde{f}_{\hat{\alpha},i}=\hat{f}_i}$. We propose to estimate this as:

$$\left(\tilde{f}_{\hat{\alpha},i} - \tilde{f}_{\hat{\alpha},i}^{[-i]}\right)_{|\tilde{f}_{\hat{\alpha},i}=\hat{f}_i} \approx \frac{a_i\left(y_i - \hat{f}_i\right)}{1 - a_i};$$
$$\text{where: } a_i = \left(\frac{\partial \tilde{f}_{\hat{\alpha},i}}{\partial y_i}\right)_{|\tilde{f}_{\hat{\alpha},i}=\hat{f}_i}$$

As discussed at the beginning of this section, we removed the tilde for the diagonal elements of the a_i. a_i is the derivative of the predicted value calculated with the surrogate loss rounded at $\hat{\alpha}$ but the location where it is calculated is the estimate of the predicted value with the exact pinball loss $\hat{f}_i$ instead of $\tilde{f}_{\hat{\alpha},i}$. Interestingly, using this first order approximation creates a link between approximate cross validation and one of the formula for the effective degrees of freedom (edf). In section 4.2 two formulae for the edf and the conditions for their equivalence is discussed. The second formula based on Stein [1981] estimates the edf as the sum of a_i (this sum will then appear in the GACV and QGACV criteria). Replacing in $\tilde{\tilde{V}}_0$, we obtain:

$$\tilde{V}_1 = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_\tau\left(\frac{y_i - \hat{f}_i}{1 - a_i}\right) = \frac{1}{n}\sum_{i=1}^{n} \frac{\mathcal{L}_\tau\left(y_i - \hat{f}_i\right)}{1 - a_i} \tag{3.24}$$

We here obtain a similar criterion to the ACV in Nychka et al. [1995]. The two differences are that first, the exact pinball loss is used to calculate the criterion. The rounded loss function used as a surrogate to the expected loss is only used to calculate the diagonal of the hat matrix $\mathbf{A}$. Second, we developed a method to obtain the rounding the surrogate loss by finding parameter $\hat{\alpha}$ that

[7] This is the same assumption as made by Yuan [2006], Reiss and Huang [2012].

leads to the smallest squared distance with the empirical loss and the surrogate loss. One advantage of our ACV formula compared to the ACV derived in Nychka et al. [1995], Yuan [2006], Reiss and Huang [2012] is that we are using the exact pinball to calculate most elements of the ACV criterion. We can then obtain an accurate estimate of the vector of parameters with the exact pinball loss or a close approximation. Nychka et al. [1995] and Yuan [2006] use a single rounding to estimate the vector of parameters and estimate the edf. The accuracy of the estimate of the vector of parameters $\boldsymbol{\beta}$ is then limited by the computability of the ACV criterion.

3.6 QGACV

As discussed previously, Yuan [2006] notes that using the ACV can sometimes lead to overfitting. To improve the fit, Yuan [2006] uses a method proposed in Craven and Wahba [1978], Golub et al. [1979] that consists in averaging the weights a_i: each weight at the denominator of the ACV is replaced by the average weight: $\frac{tr(\mathbf{A})}{n}$. We could do the same with the ACV criterion $\tilde{V}_1$ that we derived in equation 3.24:

$$\tilde{V}_2 = \sum_{i=1}^{n} \frac{\mathcal{L}_\tau\left(y_i - \hat{f}_i\right)}{n - tr\left(\mathbf{A}\right)} \tag{3.25}$$

This leads to a GACV-type criterion. As discussed previously, Reiss and Huang [2012] note that the GACV sometimes overfits non-central quantiles. They explain this by very different ACV weights on both sides of the quantiles for extreme quantiles and we showed an example of this on Figure 3.5. We could use this analysis to improve the fits of the GACV criterion. As the GACV seems to have a good performance for central quantiles, where we have an equal number of datapoints on both sides of the model, we could try to replicate this setting for non-central quantiles. We are going to take again the example of Figure 3.4 where we had the GACV failing for an extreme quantile. If we knew where quantiles were, quantile $\tau = 0.925$ could also be obtained by picking the same number of datapoints below the quantile as we have above the quantile and fitting a central quantile model ($\tau = 0.5$)[8].

[8] The type of technique consisting in sampling less points on the side with the highest number of points for imbalanced datasets could be referred to as under-sampling (see for example He and Garcia [2009]). Under-sampling techniques are used for classification using classifiers such as support vector machines (svms) with imbalanced datasets. As pointed out by He and Garcia [2009], some under-sampling techniques could be "informed", i.e. the points are picked in a certain way to improve the classification with the subsampled dataset (for example one-sided sampling as in Kubat et al. [1997] or Yen and Lee [2006], Liu et al. [2008]) or random. In this section, we will use random under-sampling. There are two reasons for this. First, the under-sampling will solely be used to derive a criterion (QGACV). In the QGACV, the under-sampling will not appear any longer the QGACV criterion will be calculated with the full original dataset. Second, using an informed method to under-sample may bias our derivation depending on the method used.

With this approach, we would then have similar weights on both side of the quantile. This in turn could improve the estimation of the smoothing parameters. Although we do not know the location of the quantile, we could run a first estimation of a quantile model and use this reference to pick the points that are above the quantile, below the quantile and points that will eventually interpolate the quantile model as $\alpha \to 0$. Let's try this approach with the example of Figure 3.4.

We first fit a model at quantile $\tau = 0.925$ using the GACV to obtain the smoothing parameter on a grid of values for smoothing parameter ρ: $rhoGrid = \{-25, -24.9, ..., 1.9, 2\}$. We then identify the points that would eventually interpolate the model as $\alpha \to 0$ and the points that are above the quantile and we subsample from below the quantile model a number of points equal to the number of points above the quantile. In the example of Figure 3.11, there are $n = 200$ datapoints. We fit a smoothing spline using the GACV criterion with $\alpha = 1 \times 10^{-1}$, $m = 20$ basis functions. Based on the overfitted pilot fit, there are 20 datapoints that will eventually be interpolated as $\alpha \to 0$ (indicated in black on the Figure) and 14 datapoints above the quantile model (this is 93% of the number of datapoints, close to $\tau = 0.925$). We then subsample 14 datapoints below the model. We have in total 20+14+14=48 datapoints. Now that we have a subsampled dataset, we fit a second smoothing spline, still using the GACV to estimate the smoothing parameter on a grid. However, we now fit a central quantile ($\tau = 0.5$). We simulate different subsamples and chose one subsample where there is a clear improvement in the fit. Note that the purpose is to illustrate the concept of subsampling. Given that we have a relatively small subsampled dataset, this technique does not necessarily work for every possible random subsample of the datapoints below the quantile pilot fit. [9][10]

We can see the results on Figure 3.11. On the second row of the Figure, with the chosen subsampled dataset, after fitting the model using GACV with $\tau = 0.5$ (left hand side chart), we can see on the second row, right hand side chart that the ACV weights are much more equal above and below the quantile model. The edf of the fitted curve is 9, much lower than the original edf of 20. On the third row of Figure 3.11, we refit the original quantile $\tau = 0.925$ with the original dataset (n=200) with the smoothing parameter ρ selected with the subsampled dataset rescaled to take into account the difference between the value of the empirical loss with the subsampled dataset and the central quantile

[9] In this example, it does result in a reduction of the edf on average. The edf for the starting fit is close to 20 (that is equal to the number of basis functions). Out of 100 subsample simulations, we obtain an average edf for the fitted median regression of 16.53 with a standard deviation of 2.93. Of course, because the edf of the started overfitted curve is equal to the number of basis functions, this simulation is somewhat truncated because the edf of the fitted curve for the subsampled dataset cannot exceed 20. However, out of 100 simulations, only 4 subsampled dataset fits had an edf of 20.

[10] As mentioned in Footnote 8, we could have improved this, possibly by using informed undersampling techniques. However, as mentioned, the subsampling will be used in this section to derive a criterion and we then use a standard random under-sampling technique.

and the original dataset with quantile 0.925:

$$\rho_{rescale} = \log\left(\exp\left(\rho_{sub,\tau=0.5}\right) \times \frac{\frac{1}{200}\sum_{j=1}^{200}\mathcal{L}_{\alpha,\tau=0.925}\left(y_j - \tilde{f}_j^{\tau=0.925}\right)}{\frac{1}{48}\sum_{j\in SubsampledSet}\mathcal{L}_{\alpha,\tau=0.5}\left(y_j - \tilde{f}_j^{\tau=0.5}\right)}\right)$$

where $\tilde{f}_j^{\tau=0.925}$ is the original fit with the dataset at quantile $\tau = 0.925$ and $\tilde{f}_j^{\tau=0.5}$ is the fit with the subsampled dataset at $\tau = 0.5$. This method is only an approximation and may not be usable in every case but can give us some idea about the potential applicability of such an idea. We obtain a much better fit for quantile 0.925. The edf of the final curve is 6.65.

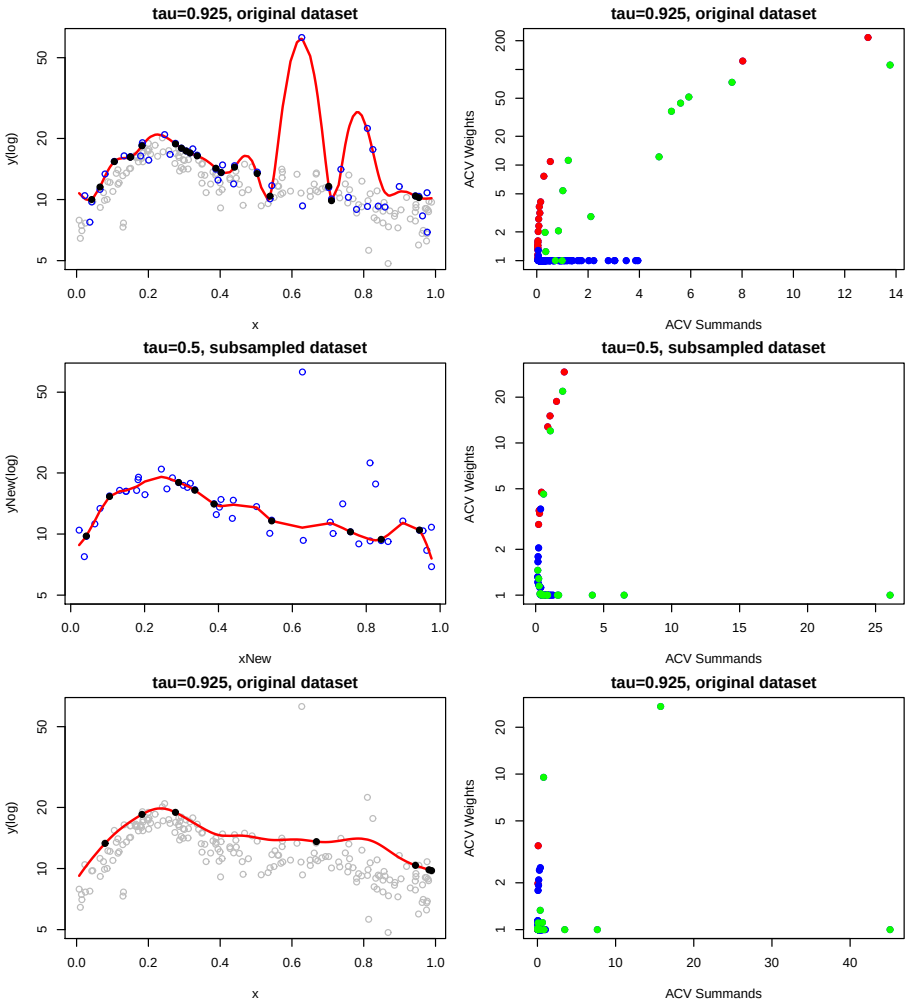


Fig. 3.11: We take again the example of Figure 3.4, $n = 200, \tau = 0.925$. On the top line, left, we can see that with the GACV, the model overfits. On the top right hand side, as in Reiss and Huang [2012], we show that the ACV Weights ($\frac{1}{1-a_i}$) (y-axis) vs the ACV Summands ($\frac{\sum_i \mathcal{L}_{\alpha,\tau}(\hat{u}_i)}{1-a_i}$) (x-axis). Unlike Reiss and Huang [2012], we here separate the points in 3 groups. In red, the points that would be interpolated by the model as $\alpha \to 0$, in green the points that are above the quantile model, in blue the points that are below the quantile model. We can see that the weights for summands above the quantile and below the quantile are quite different. In the middle row, after excluding points that will be interpolated as $\alpha \to 0$, we pick the same number of points below the quantile as there are points above the quantile. To do this, we run draw several randomly subsampled datasets, fit the curve the median curve ($\tau = 0.5$). We then pick one of the random draws that clearly improve the fit. On the chart in the middle row on the right, we can see that now the weights on both sides of the quantile are much more similar. Finally on the bottom row, we refit the quantile regression based on the smoothing parameter selected by using the subsampled dataset on the middle row rescaled to take into account the difference in the mean empirical losses. We see that using this method, there is no overfitting any longer.

We propose to use a similar reasoning to correct the averaging of the ACV weights. Although the GACV still overfits sometimes for central quantiles (see section 4.7), in many cases the fit using GACV is closer to the true quantile for central quantiles than for extreme quantiles. We may then try to replicate the performance of the GACV used to fit central quantiles for extreme quantiles. Our derivation will also take into account the number of datapoints interpolated by the model. Our new criterion will then reduce the overfitting for central quantiles as well as the overfitting for extreme quantiles. Let's assume that $\tau < 0.5$. This means that the "Below" set $\mathcal{B} = \left\{ i \in 1, ..., n; y_i < \hat{f}_i \right\}$ should contain less points than the "Above" set: $\mathcal{A} = \left\{ i \in 1, ..., n; y_i > \hat{f}_i \right\}$. With the exact pinball loss, a third dataset is the dataset of the "interpolated" points: $\mathcal{I} = \left\{ i \in 1, ..., n; y_i = \hat{f}_i \right\}$. Note that $\#\left(\mathcal{A} \bigcup \mathcal{B} \bigcup \mathcal{I}\right) = n$. Where $\#$ designates the cardinal of a set.

Similarly to our example, we will now try to rebalance the dataset by subsampling as many datapoints above the model as there are below. There are approximately $Tr(\mathbf{A})$ datapoints that are interpolated by the model: $\#\mathcal{I} \approx Tr(\mathbf{A})$ (see section 4.3). There are then approximately $\tau \times (n - Tr(\mathbf{A}))$ datapoints in $\mathcal{B}$. We will subsample $\tau \times (n - Tr(\mathbf{A}))$ above the quantile. Let's call this new dataset $\mathcal{S}$. Our new dataset becomes $\mathcal{A} \bigcup \mathcal{I} \bigcup \mathcal{S}$. and $\#(\mathcal{A} \bigcup \mathcal{I} \bigcup \mathcal{S}) = 2\tau \times (n - Tr(\mathbf{A})) + Tr(\mathbf{A})$. To obtain the same quantile from the subsampled data, we modify the pinball loss function:

$$\mathcal{L}_\tau^{Sub}(u) = I(u < 0)\,\mathcal{L}_\tau(u) + \frac{1-\tau}{\tau} I(u > 0)\,\mathcal{L}_\tau(u) + I(u = 0)\mathcal{L}_\tau(u) \quad (3.26)$$

Note that we do not need to add the interpolated points as $\mathcal{L}_\tau(u) = 0$ for the interpolated points. With this new weighting, we now have a replicated loss for a central quantile ($\tau = 0.5$) as:

$$\begin{aligned}
\mathcal{L}_\tau^{Sub}(u) &= I(u < 0)(\tau - 1)u + I(u > 0)\frac{1-\tau}{\tau}\tau u + 0 \\
&= \frac{1-\tau}{0.5}\left(-I(u < 0) \times 0.5 \times u + I(u > 0) \times 0.5 \times u\right) \\
&= \frac{1-\tau}{0.5}\mathcal{L}_{0.5}(u)
\end{aligned}$$

The ACV Criterion can now be re-written:

$$\tilde{\tilde{V}}_1 = \frac{1}{2\tau\left(n - Tr\left(\mathbf{A}_{Sub}\right)\right)} \sum_{s \in \mathcal{B}} \frac{\mathcal{L}_\tau\left(y_i - \hat{f}_i\right)}{1 - a_i} + \frac{1}{2\tau\left(n - Tr\left(\mathbf{A}_{Sub}\right)\right)} \frac{1-\tau}{\tau} \sum_{s \in \mathcal{S}} \frac{\mathcal{L}_\tau\left(y_i - \hat{f}_i\right)}{1 - a_i} \quad (3.27)$$

Where $\mathbf{A}_{Sub}$ is the hat matrix for the model estimated using the subsampled dataset. As discussed previously, we removed the points that are exactly interpolated (points that are in $\mathcal{I}$) from the expression because they do not add anything to the sum. Now that we rebalanced the dataset, we can take the

average of the more even weights. We replace a_i by its average over the new dataset: $\frac{Tr(\mathbf{A}_{Sub})}{2\tau(n-Tr(\mathbf{A}_{Sub}))}$:

$$\tilde{\tilde{V}}_2 = \sum_{s\in\mathcal{B}} \frac{\mathcal{L}_\tau\left(y_i - \hat{f}_i\right)}{(2n\tau - (1+2\tau)\,Tr\,(\mathbf{A}_{Sub}))} + \frac{1-\tau}{\tau}\sum_{s\in\mathcal{S}} \frac{\mathcal{L}_\tau\left(y_i - \hat{f}_i\right)}{(2n\tau - (1+2\tau)\,Tr\,(\mathbf{A}_{Sub}))} \tag{3.28}$$

We then propose to estimate: $\frac{1-\tau}{\tau}\sum_{s\in\mathcal{S}} \frac{\mathcal{L}_\tau\left(y_i-\hat{f}_i\right)}{(2n\tau-(1+2\tau)Tr(\mathbf{A_{Sub}}))}$ by $\sum_{s\in\mathcal{A}} \frac{\mathcal{L}_\tau\left(y_i-\hat{f}_i\right)}{(2n\tau-(1+2\tau)Tr(\mathbf{A_{Sub}}))}$. We can also have the interpolated points appear in the sum. As we are now dividing by a constant, if we add the interpolated points (in $\mathcal{I}$), $\tilde{\tilde{V}}_2$ remains unchanged. Finally, as $Tr\,(\mathbf{A}_{Sub})$ is the degrees of freedom of the model, we propose to estimate it using $Tr\,(\mathbf{A})$. We then obtain:

$$\begin{aligned} V_3 =& \sum_{s\in\mathcal{B}} \frac{\mathcal{L}_\tau\left(y_i - \hat{f}_i\right)}{(2n\tau - (1+2\tau)\,Tr\,(\mathbf{A}))} + \sum_{s\in A} \frac{\mathcal{L}_\tau\left(y_i - \hat{f}_i\right)}{(2n\tau - (1+2\tau)\,Tr\,(\mathbf{A}))}... \\ &... + \sum_{s\in\mathcal{I}} \frac{\mathcal{L}_\tau\left(y_i - \hat{f}_i\right)}{(2n\tau - (1+2\tau)\,Tr\,(\mathbf{A}))} \\ =& \sum_{i=1}^{n} \frac{\mathcal{L}_\tau\left(y_i - \hat{f}_i\right)}{(2n\tau - (1+2\tau)\,Tr\,(\mathbf{A}))} \end{aligned}$$

Generalizing to the case $\tau \geq 0.5$, we obtain:

$$\begin{aligned} QGACV = V_3 &= \sum_{i=1}^{n} \frac{\mathcal{L}_\tau\left(y_i - \hat{f}_i\right)}{(2n\phi - (1+2\phi)\,Tr\,(\mathbf{A}))} \\ \phi &= \min\left(\tau, 1-\tau\right) \end{aligned}$$

Alternatively, we will write the QGACV:

$$\begin{aligned} QGACV &= \sum_{i=1}^{n} \frac{\mathcal{L}_\tau\left(y_i - \hat{f}_i\right)}{n.eff - cTr\,(\mathbf{A})} \\ n.eff &= 2n\phi \\ c &= (1+2\phi) \end{aligned}$$

Figure 3.12 shows how the QGACV can be used to improve the fits of Figure 3.4. We can see that the overfit observed for extreme quantile using both the ACV and the GACV on Figure 3.4 has been reduced. The fitted extreme quantile has a similar degree of smoothness to the true quantile. The methodology used to fit the curves on Figure 3.4 will be developed in details in chapters 5, 6, 7.

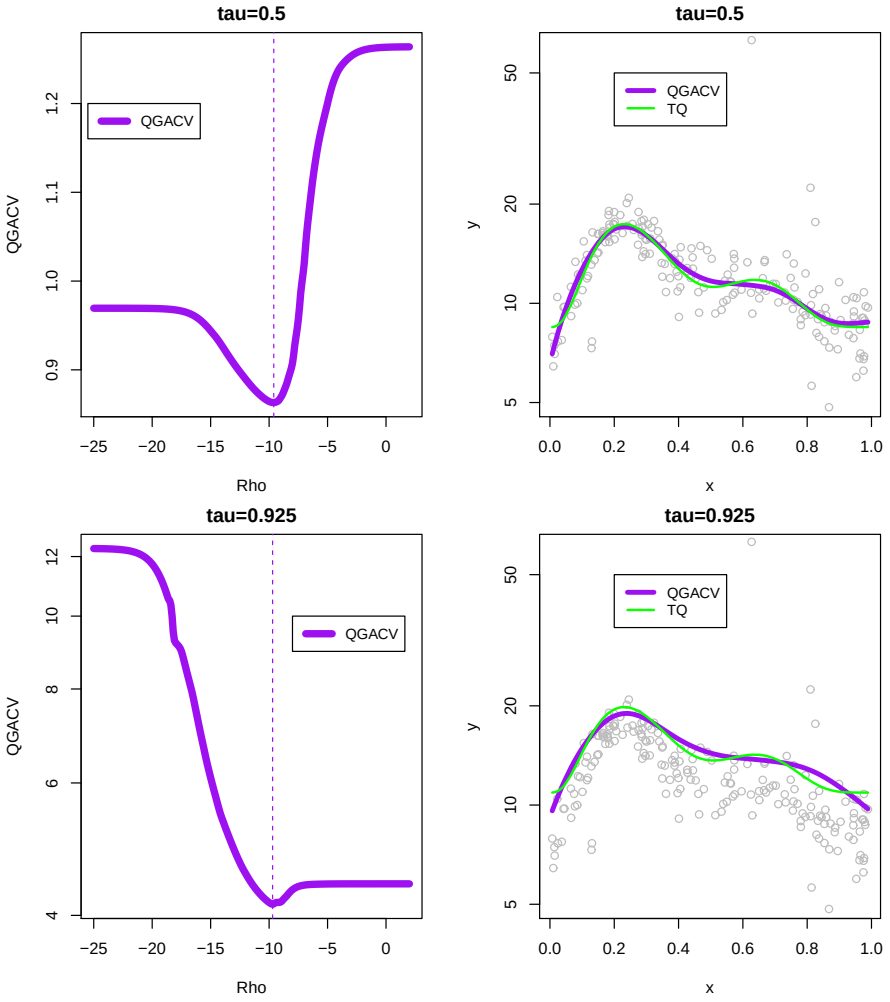


Fig. 3.12: This figure shows how the QGACV can be used to improve fits. We take again the example of Figure 3.4 (Example 2, n=200 datapoints). The QGACV formula requires two levels of rounding. A first level of rounding to calculate the edf ($\hat{\alpha}$), calculated using formula 3.15 (we obtain $\hat{\alpha} = 1.25$ for $\tau = 0.5$ and $\hat{\alpha} = 1.58$ for $\tau = 0.925$. We use the methodology described in sections 7.2, 7.3, 7.5) and a second level of rounding that we call α_L to approximate the exact pinball loss (we use the method described in sections 5.2 and 7.11). We here use $\alpha_L = 0.1$ to match with the rounding levels we have used previously (in the rest of this document, a smaller α_L will be used to obtain a more precise estimation of the quantile model). α_L is also used to estimate the vector of parameters that is used for all elements of the QGACV formula including the vector of parameters used to calculate the edf ($\hat{\alpha}$ is only used for the rounding of the loss function to calculate the second derivative of the loss function). The QGACV is then optimized on a grid of values for ρ: $rhoGrid = \{-25, -24.9, ..., 1.9, 2\}$ (in section 5.2, we develop a method to optimize directly the QGACV surface). We can see that the overfit observed for extreme quantile using both the ACV and the GACV on Figure 3.4 has been reduced.

4. ON ALTERNATIVE LOSSES, CROSS VALIDATION, EDF AND QGACV

In this section, we discuss several points related to the choice of loss, cross validation, edf and QGACV. First, in section 4.1, we compare three potential shifted versions of the surrogate loss of Zheng [2011] and indicate what is the advantage of the version chosen. Then in section 4.2 we discuss two different formulae for the effective degrees of freedom of the model and how they are related. In section 4.3, we then discuss how the effective degrees of freedom of the model equates the number of interpolated points. In section 4.4, we discuss the bias in the edf due to the rounding of the loss function. In section 4.5, we then discuss one issue with the derivation of the GACV. In section 4.6, we look into possible explanations for the overfitting observed with the GACV. In section 4.7, we discuss overfitting for central quantiles. In section 4.8 we note that the QGACV could be seen as equivalent to a GACV formula where the multiplier of the edf varies with τ. In section 4.9, we discuss an alternative to the QGACV, the SGACV. Finally, in section 4.10, we explain how the QGACV can be used to fit Total Variation penalized Quantile Additive Models.

4.1 Three possible loss functions

Zheng [2011] proposed a smooth surrogate loss for the pinball loss based on an idea for svms (Chen and Mangasarian [1995]) and that is similar to the function developed for absolute values in Amemiya [1982]:

$$\boldsymbol{\mathcal{L}^{A}_{\alpha,\tau}(u) = \tau u + \alpha \log\left(1 + \exp\left(-\frac{u}{\alpha}\right)\right)} \tag{4.1}$$

However, $\mathcal{L}^{A}_{\alpha,\tau}$ does not have a minimum at 0. The derivation of the ACV (see section 3.5) relies on the loss function minimizing in 0. The derivation could be made with $\mathcal{L}^{A}_{\alpha,\tau}$, however, we note that a small modification of $\mathcal{L}^{A}_{\alpha,\tau}$ is sufficient to obtain a similar loss function that has a minimum at 0. We note that loss $\mathcal{L}^{A}_{\alpha,\tau}$ has a minimum that does not depend on u:

$$\frac{d\mathcal{L}^{A}_{\alpha,\tau}(u^*)}{du} = 0 \Leftrightarrow u^* = \alpha \log\frac{(1-\tau)}{\tau}$$

If instead of $\mathcal{L}^{A}_{\alpha,\tau}$, we shift u by the minimum of $\mathcal{L}^{A}_{\alpha,\tau}$, we obtain $\mathcal{L}^{B}_{\alpha,\tau}$, a

function that has always a minimum at 0:

$$\boldsymbol{\mathcal{L}^{B}_{\alpha,\tau}(u) = \tau(u+\gamma) + \alpha \log\left(1 + \exp\left(-\frac{u+\gamma}{\alpha}\right)\right)}$$

where $\gamma = \alpha \log\left(\frac{1-\tau}{\tau}\right)$, then:

$$\frac{d\mathcal{L}^{B}_{\alpha,\tau}(u^*)}{du} = 0 \Leftrightarrow u^* = 0$$

Last, we note that $\mathcal{L}^{B}$ is not always above or below the exact pinball loss. Let's calculate the value of $\mathcal{L}^{B}_{\alpha,\tau}(0)$:

$$\mathcal{L}^{B}_{\alpha,\tau}(0) = -\alpha\left(\tau \log(\tau) + (1-\tau)\log(1-\tau)\right)$$

We could then subtract $\mathcal{L}^{B}_{\alpha,\tau}(0)$ from $\mathcal{L}^{B}_{\alpha,\tau}(u)$. We would then obtain loss function $\mathcal{L}^{C}_{\alpha,\tau}$ that has a minimum at zero that is equal to zero:

$$\begin{aligned}\boldsymbol{\mathcal{L}^{C}_{\alpha,\tau}(u)} &= \boldsymbol{\mathcal{L}^{B}_{\alpha,\tau}(u) + \alpha\left(\tau \log(\tau) + (1-\tau)\log(1-\tau)\right)} \\ &= \boldsymbol{\tau u + \alpha \log\left((1-\tau) + \tau \exp\left(-\frac{u}{\alpha}\right)\right)}\end{aligned}$$

We note that $\mathcal{L}^{C}_{\alpha,\tau}(0) = \alpha \log((1-\tau)+\tau) = 0$ and $\frac{d\mathcal{L}^{C}_{\alpha,\tau}(u^*)}{du} = 0 \Leftrightarrow u^* = 0$. The three loss functions are represented on Figure 4.1. Note that for a different purpose, Mukhoty et al. [2019] make similar remarks about the Huber loss. They derive a scaled Huber loss that is a majorizer of the absolute value loss whereas the Huber loss is a minimizer and show that the IRLS is equivalent to an MM algorithm on this scaled Huber loss.

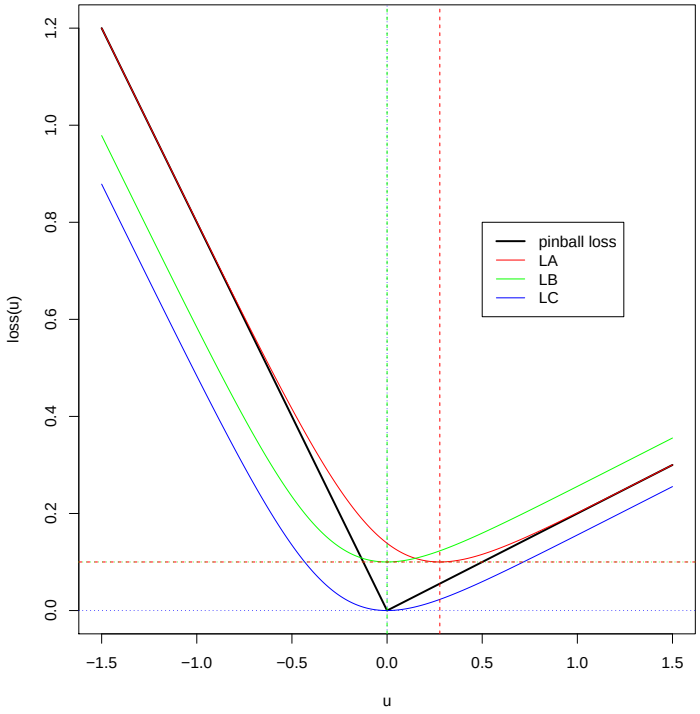


Fig. 4.1: This Figure shows the exact pinball loss (black), the approximate loss function $\mathcal{L}^A$ Zheng [2011] in red, $\mathcal{L}^B(u) = \mathcal{L}^A(u+\gamma)$, where $\gamma = \alpha \log\left(\frac{1-\tau}{\tau}\right)$ that is a loss function with a minimum at 0. We also have $\mathcal{L}^B_{\alpha,\tau}(0) = -\alpha\left(\tau\log(\tau) + (1-\tau)\log(1-\tau)\right)$ and $\mathcal{L}^C_{\alpha,\tau}(u) = \mathcal{L}^B_{\alpha,\tau}(u) - \mathcal{L}^B_{\alpha,\tau}(0)$ (blue). We note that $\mathcal{L}^A$ is always above the exact pinball loss, $\mathcal{L}^C$ is always below the pinball loss. Note that for another purpose, similar remarks are made in Mukhoty et al. [2019] for the Huber loss.

In deciding what loss to use, we took the following considerations into account:

- $\mathcal{L}^A$ does not have a minimum at 0 whereas $\mathcal{L}^B$ and $\mathcal{L}^C$ do.
- For levels of α that we typically use to estimate the degrees of freedom ($\hat{\alpha}$ estimated using equation 3.15), the bias with $\mathcal{L}^A$ is lower that the bias with $\mathcal{L}^B$ or $\mathcal{L}^C$ (see Figure 4.2). Although this could be considered as an advantage of $\mathcal{L}^A$, we estimate the vector of parameters with a low level of rounding and $\hat{\alpha}$ is only used to calculate the hat matrix. In an example on Figure 4.4, we can see the edf curve calculated using the loss $\mathcal{L}^A$ is identical to the edf curve calculated using $\mathcal{L}^B$ or $\mathcal{L}^C$.

- For larger values of α, $\mathcal{L}^A$ bias continuously increases whereas bias with $\mathcal{L}^B$ or $\mathcal{L}^C$ reaches a plateau (see Figures 4.2 and 4.3). However, we do not take this into account as we will not use very large values of α in our algorithm.

- Last, given that $\mathcal{L}^C$ has a minimum of 0 in 0, it is always below the exact pinball loss we may think that this may be the most appropriate loss to use, as an infinitely differentiable alternative to the Huberized pinball loss. This would then be similar to the use of the infinitely differentiable pseudo-Huber penalty instead of the Huber penalty as in Fountoulakis and Gondzio [2016]. However, there are cases where switching the loss function from $\mathcal{L}^B$ to $\mathcal{L}^C$ may result in overfitting. This is due to a local minimum that sometimes appears in the QGACV criterion. With a low level of rounding α, the local minimum appears for both $\mathcal{L}^B$ and $\mathcal{L}^C$. However, when we increase the level of rounding, the local minimum remains with $\mathcal{L}^C$ but is smoothed out with $\mathcal{L}^B$. In section 5.2, we discuss a method to optimize the QGACV consisting in starting the optimization with a large level of rounding (we use $\hat{\alpha}$) and gradually reducing α whilst keeping the rounding level at $\hat{\alpha}$ to calculate the edf. With $\mathcal{L}^C$, the algorithm finds the local minimum at the beginning and stays on this local minimum, leading to overfitting. With $\mathcal{L}^B$, the QGACV is smoothed out at $\hat{\alpha}$. An example is shown on Figure 4.4.

To summarize, loss $\mathcal{L}^A$ does not have a minimum in 0. It would then be more difficult to use $\mathcal{L}^A$ to derive an Approximate Cross Validation criterion compared to using $\mathcal{L}^B$ or $\mathcal{L}^C$. Then, using loss $\mathcal{L}^C$ can lead to overfitting in some cases. We then choose loss $\mathcal{L}^B$ as the most appropriate for our purpose. (in this document, $\mathcal{L}^B_{\alpha,\tau}$ is referred to as $\mathcal{L}_{\alpha,\tau}$)

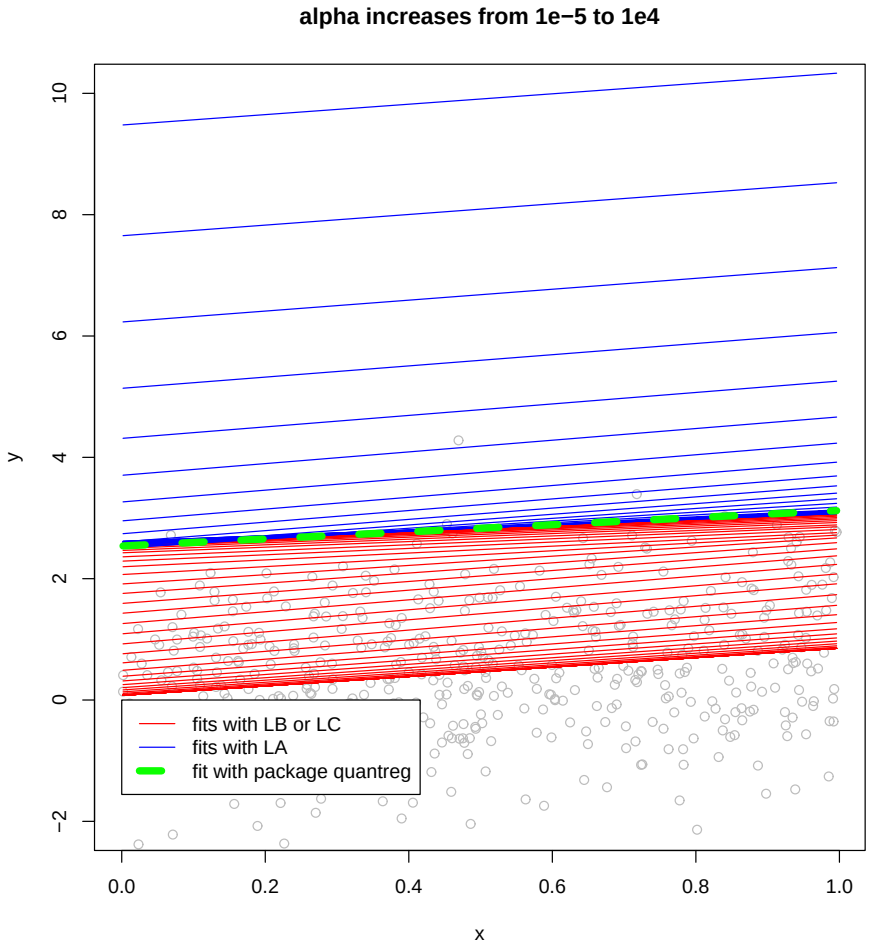


Fig. 4.2: On this picture, we can see how the different fits of a linear quantile regression using surrogate losses $\mathcal{L}^A$, $\mathcal{L}^B$ and $\mathcal{L}^C$. We start with a very small rounding $\alpha = 10^{-5}$. We then gradually increase it to $\alpha = 10^4$. The data is generated as: $n = 500$, $\underline{x}_i \sim \mathcal{U}[0,1]$, $y_i \sim \underline{x}_i + \epsilon_i$,$\epsilon_i \sim \mathcal{N}(0,1)$. With a very small value of $\alpha = 10^{-5}$, the fits with all loss functions match the linear quantile regression fit of package 'quantreg'. As we increase α, the bias increases for the fits with $\mathcal{L}^A$, $\mathcal{L}^B$ and $\mathcal{L}^C$. The fits with $\mathcal{L}^A$ move up and the fits with $\mathcal{L}^C$ move down towards the central quantile ($\tau = 0.5$).

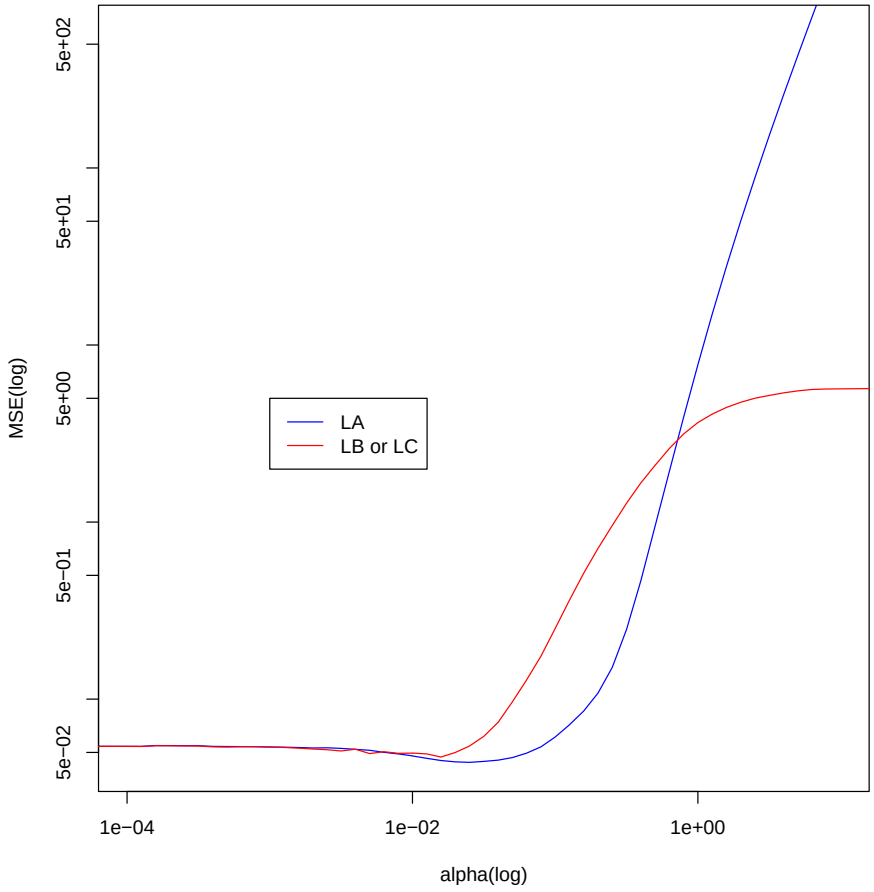


Fig. 4.3: In this picture, we calculate the mean squared error between the fit with Loss A and the precise linear quantile regression fit with package quantreg (in blue), and between the fit with Loss B (or C) and the fit with package quantreg (in red). We can see that for large values of α, the bias with Loss A is larger than with Loss B or C. The bias then decreases faster with Loss A than with loss B or C. Between $\alpha = 10^{-2}$ and around 0.6, the MSE with Loss A is lower than with Loss B or C. Then for alpha below 10^{-2}, the MSE is similar for the different loss functions. We use Example 5 (see section 9.1), $n = 500, \tau = 0.99$. We use 20 basis functions.

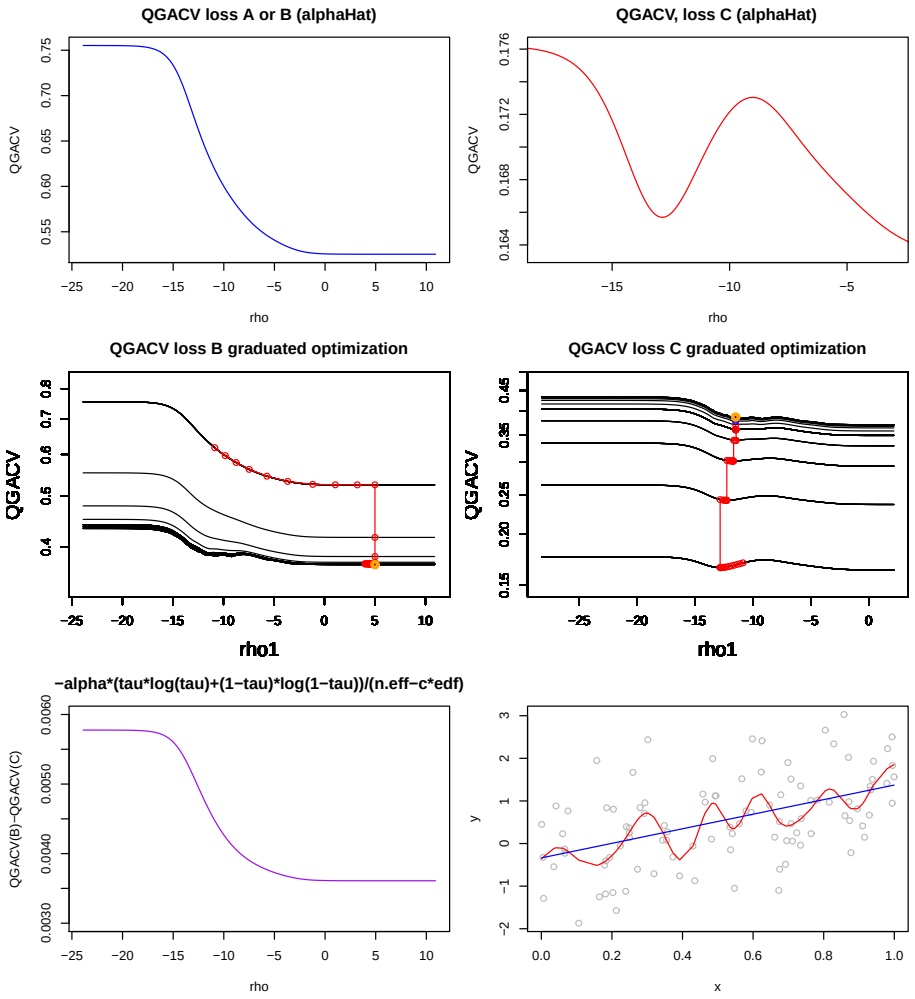


Fig. 4.4: This Figure shows QGACV charts related to the fits shown on the bottom right chart (the blue straight line corresponds to a fit using graduated optimization as described in section 5.2 and chapter 7 using $\mathcal{L}^A$ or $\mathcal{L}^B$. The red curve corresponds to the same fit using $\mathcal{L}^C$. We can see that the curve overfits). On the top left chart, we show the QGACV criterion calculated using $\mathcal{L}^A$ or $\mathcal{L}^B$ when $\alpha = \hat{\alpha}$ (note that $\alpha = \hat{\alpha}$ is determined using $\mathcal{L}^A$ with the methods of sections 7.2, 7.6). On the top right chart, we can see the QGACV using $\mathcal{L}^C$ using the same rounding $\alpha = \hat{\alpha}$. The middle charts show the optimization paths using the graduated optimization method as described in section 5.2 and chapter 7. This consists in starting with a value of the rounding parameter $\alpha = \hat{\alpha}$, then reduce it down to a lower value α_U, using the minimum of the previous level as a starting point. On the left hand side, we can see that with a high level of rounding, with loss $\mathcal{L}^B$ ($\mathcal{L}^A$ would give the same chart), the local minimum between $\rho = -10$ and $\rho = -15$ has been smoothed out. The optimization leads to a value of the smoothing parameter $\rho = 5$ (this is the maximum value we allow in our algorithm). On the middle right chart, we can see that the local minimum has not been smoothed out and the optimization stays on this local minimum leading to overfitting. Note that with $\mathcal{L}^A$ or $\mathcal{L}^B$ the QGACV curve goes down as we reduce α. With $\mathcal{L}^C$, the QGACV curve goes up. The bottom left chart shows $-\alpha\frac{\tau\log(\tau)+(1-\tau)\log(1-\tau)}{n.eff-c\times edf}$. This is the difference between the QGACV(LossA or B)-QGACV(LossC). This function explains why we obtain a local minimum with Loss C.

4.2 Effective Degrees of Freedom

In this section, we discuss the equivalence between two formulae for the effective/generalized degrees of freedom, one based on the covariance between the observations and the predicted value, the other based on the sensitivity of the predicted value to the observations. As pointed out in Vaiter et al. [2017], Efron [1986] defines the edf as the covariance between the observations and the predicted values divided by the variance of the error: if $\mathbf{Y}$ is a random vector of observations with $\mathbf{Y} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma \mathbf{Id}_n)$, where $\mathbf{Id}_n$ is the identity matrix of size n then the edf is calculated as:

$$edf = \sum_{i=1}^{n} \frac{cov\left(Y_i, \hat{f}_i(\mathbf{Y})\right)}{\sigma^2} \tag{4.2}$$

where $\hat{f}_i$ is the i^{th} predicted value. Then, under the same normality assumptions for $\mathbf{Y}$ as previously, Stein [1981] shows that if a function $f : \mathbb{R}^n \to \mathbb{R}$ is almost differentiable[1] and if $E\|\nabla f(\mathbf{Y})\| < \infty$ (see also Meyer and Woodroofe [2000]) then the expected value of the derivative of the function of $\mathbf{Y}$ is linked to the covariance of $\mathbf{Y}$ and $f(\mathbf{Y})$ via the following formula:

$$E\left[(\mathbf{Y} - \boldsymbol{\mu}) f(\mathbf{Y})\right] = \sigma^2 E\left[\nabla f(\mathbf{Y})\right] \tag{4.3}$$

The condition of almost differentiability can also be replaced by an equivalent condition of absolute continuity in each direction $Y_1, ..., Y_n$ (see for example Dossal et al. [2013], Kato [2009], Fourdrinier et al. [2018] pp.34-40). Stein's lemma has also been shown to hold for the slightly more general condition of weak differentiability[2] (see Fourdrinier et al. [2018] pp.34-40, Hansen [2018], supplementary material). Under the assumption of weak differentiability (or almost differentiability/ directional absolute continuity) and finiteness of the expected gradient, from Stein's lemma (see Ye [1998][3], Kato [2009], Dossal et al. [2013], Vaiter et al. [2017], Tibshirani and Rosset [2019], Gao and Fang [2011]), we then have:

$$edf = E\left[\sum_{i=1}^{n} \frac{\partial \hat{f}_i(\mathbf{Y})}{\partial Y_i}\right] \tag{4.4}$$

[1] From Stein [1981] (see also Meyer and Woodroofe [2000]): a function $f : \mathbb{R}^n \to \mathbb{R}$ is almost differentiable if there exists a function $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ such that $\forall \mathbf{z} \in \mathbb{R}^n$, for almost all $\mathbf{x} \in \mathbb{R}^n$, $f(\mathbf{x} + \mathbf{z}) - f(\mathbf{x}) = \int_0^1 \mathbf{z}^T \nabla f(\mathbf{x} + t\mathbf{z})\, dt$

[2] Fourdrinier et al. [2018], definition 2.1, Hansen [2018], supplementary material: A locally integrable function $f : \mathbb{R}^n \to \mathbb{R}$ is deemed weakly differentiable if for every function $\psi : \mathbb{R}^n \to \mathbb{R}$ of class C^∞ with compact support, there exists n locally integrable functions $\partial_i f : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., n$ such that:

$$\int \partial_i f(\mathbf{x}) \psi(\mathbf{x})\, d\mathbf{x} = -\int f(\mathbf{x}) \frac{\partial \psi}{\partial x_i}(\mathbf{x})\, d\mathbf{x}$$

.

[3] Ye [1998] calls this expression the generalized degrees of freedom

this can be estimated as:

$$\widehat{edf} = \sum_{i=1}^{n} \widehat{edf}_i = \sum_{i=1}^{n} \frac{\partial \hat{f}_i}{\partial y_i} \tag{4.5}$$

Equation 4.4 can then be seen as an alternative definition of the edf. As pointed out by Liu et al. [2020], the definition of the edf as in equation 4.2 and in equation 4.4 are equivalent for $\mathbf{Y} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma \mathbf{Id}_n)$ and under the weak differentiability/finiteness of the expected gradient conditions but may not always be equivalent, for example if $\mathbf{Y}$ is not normally distributed (see Efron [2004], Janson et al. [2015], Eilers [2018] for discussions on these two formulae). Extensions to discontinuous/piecewise Lipschitz functions/lower semi-continuous (for example like the ℓ_0 penalized regression) have been derived in Vaiter et al. [2013], Tibshirani [2015], Deledalle et al. [2014], Vaiter et al. [2017], Tibshirani and Rosset [2019], Mikkelsen and Hansen [2018]. For discontinuous functions, the correction term for the edf is a function of the discontinuities of function f.

In the case of the linear regression constrained to a convex set, Kato [2009], Lemma 2.2 shows that 4.2 and 4.4 are equivalent by showing directional absolute continuity. This includes variants of the lasso penalized linear regression. As mentioned in Koenker and Mizera [2014] a (generalized) ridge penalized quantile regression is a similar problem to a linear regression with (generalized) lasso-type penalty where the role of the empirical risk and the penalty have been reversed (ridge penalized quantile regression could be formulated as a semi-definite quadratic program with linear inequality constraints). Hence this should apply to our case. To see this, we can see that in our case, we also have (locally) directional absolute continuity. This is because first, function $\mathbf{y} \to \hat{\boldsymbol{\beta}}(\mathbf{y})$ where $\hat{\boldsymbol{\beta}}$ is estimated as per equation 2.17 is a continuous function of $\mathbf{y}$ (see Li et al. [2007] in the case of a penalized quantile regression in a reproducing kernel Hilbert space). Then, in section 4.3, we will see that the partial derivative $\frac{\partial \hat{f}_i}{\partial y_i}$ is defined and bounded (equal to 0 or 1, see Li et al. [2007]). Hence applying Ó Searcóid [2006], theorem 9.5.1, to function $y_i \to \hat{f}_i(y_i)$, function $\mathbf{y} \to \hat{\boldsymbol{\beta}}(\mathbf{y})$ is directionally locally Lipschitz in direction y_i and therefore locally directionally absolutely continuous (see Hunter [2014], 3.A.2).

4.3 Partial derivative of the predicted value with respect to the observations

As described in equation 2.17, the problem with the exact pinball loss is:

$$\hat{\boldsymbol{\beta}} \in \operatorname{argmin}_{\boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - \mathbf{x}_i^T \boldsymbol{\beta} \right) + \frac{1}{2} \boldsymbol{\beta}^T \mathbf{S}_{\boldsymbol{\lambda}} \boldsymbol{\beta}$$

In the case of a Reproducing Kernel Hilbert Space (RKHS) ($dim(\beta) = n+1$), Li et al. [2007] shows that the edf is equal to the number of interpolated points.

In our case, we have $dim(\boldsymbol{\beta}) \leq n-1$. To calculate $\frac{\partial \hat{f}_i}{\partial y_i}$, we start by finding the optimum of this regularized empirical loss (REL). Taking the REL as a function of $\boldsymbol{\beta}$, the subdifferential of the REL is:

$$\begin{aligned}
&\partial_{\boldsymbol{\beta}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - \mathbf{x}_i^T \boldsymbol{\beta} \right) + \frac{1}{2} \boldsymbol{\beta}^T \mathbf{S}_{\boldsymbol{\lambda}} \boldsymbol{\beta} \right\} |_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}} \\
&= \{ \boldsymbol{\beta} \in \mathbb{R}^p \text{ such that } \forall \boldsymbol{\delta} \in \mathbb{R}^p \ ... \\
&... \ \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - \mathbf{x}_i^T \boldsymbol{\beta} \right) \geq \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - \mathbf{x}_i^T \hat{\boldsymbol{\beta}} \right) + \boldsymbol{\delta}^T \left(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}} \right) \Bigg\} + \mathbf{S}_{\boldsymbol{\lambda}} \hat{\boldsymbol{\beta}}
\end{aligned}$$

We can simplify this expression by using the following two theorems:

Theorem 1 (Mordukhovich and Nam [2017], Theorem 6.2): If $f_j, j = 1, ..., p$ are convex function, subject to the intersection of the relative interiors of the domains of each f_j being non empty, for x in the intersection of the domains of the f_j:

$$\partial \left(\sum_{j=1}^{p} f_j \right) (x) = \sum_{j=1}^{p} \partial f_j(x)$$

See also Mordukhovich and Nam [2013], Corollary 2.45, p.63.

Theorem 2 (Mordukhovich and Nam [2017], Theorem 7.2): If C affine : $C : \mathbb{R}^n \to \mathbb{R}^p$, such that $C(x) := Ax + b$ and f is a convex function: $f : \mathbb{R}^p \to (-\infty, +\infty]$. Then, if $x \in \mathbb{R}^n$ and $y = C(x)$, assuming that the range of C contains a point of the relative interior of the domain of f, then:

$$\partial (f \circ C)(x) = A^T (\partial f(y)) = \left\{ A^T v | v \in \partial f(y) \right\}$$

See also Mordukhovich and Nam [2013], Theorem 2.51, p.66.

Applying these 2 theorems to our problem, the subdifferential can be rewritten:

$$\begin{aligned}
&\partial_{\boldsymbol{\beta}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - \mathbf{x}_i^T \boldsymbol{\beta} \right) + \frac{1}{2} \boldsymbol{\beta}^T \mathbf{S}_{\boldsymbol{\lambda}} \boldsymbol{\beta} \right\} |_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}} \\
&= \frac{1}{n} \sum_{i=1}^{n} \left(\partial_{\boldsymbol{\beta}} \mathcal{L}_\tau \left(y_i - \mathbf{x}_i^T \boldsymbol{\beta} \right) \right) |_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}} + \mathbf{S}_{\boldsymbol{\lambda}} \hat{\boldsymbol{\beta}} \\
&= \frac{1}{n} \sum_{i=1}^{n} -\mathbf{x}_i \left(\partial_{u_i} \mathcal{L}_\tau (u_i) \right) |_{u_i = \left(y_i - \hat{f}_i \right)} + \mathbf{S}_{\boldsymbol{\lambda}} \hat{\boldsymbol{\beta}}
\end{aligned}$$

where:

$$\left(\partial_{u_i} \mathcal{L}_\tau (u_i) \right) |_{u_i = \left(y_i - \hat{f}_i \right)} = \begin{cases} \tau & \text{if } y_i > \hat{f}_i \\ [\tau - 1, \tau] & \text{if } y_i = \hat{f}_i \\ \tau - 1 & \text{if } y_i < \hat{f}_i \end{cases}$$

The optimality condition at $\hat{\boldsymbol{\beta}}$ is then:

$$\mathbf{0}_p \in \frac{1}{n}\sum_{i=1}^{n} -\mathbf{x}_i \left(\partial_{u_i}\mathcal{L}_\tau\left(u_i\right)\right)|_{u_i=\left(y_i-\hat{f}_i\right)} + \mathbf{S}_{\boldsymbol{\lambda}}\hat{\boldsymbol{\beta}}$$

Let's now assume that we change y_i by an infinitesimal amount $\epsilon \to 0$ and calculate edf_i. Let's call $\hat{\boldsymbol{\beta}}^{New}$ the new estimate of the vector of parameters after the shift in point y_i and $\hat{f}_i^{New} = \mathbf{x}_i^T\hat{\boldsymbol{\beta}}^{New}$ the new predicted value after the shift of y_i. Then the new optimality condition is:

$$\mathbf{0}_p \in \frac{1}{n}\sum_{j=1,j\neq i}^{n} -\mathbf{x}_j \left(\partial_{u_j}\mathcal{L}_\tau\left(u_j\right)\right)|_{u_j=\left(y_j-\hat{f}_j^{New}\right)} - \frac{1}{n}\mathbf{x}_i \left(\partial_{u_i}\mathcal{L}_\tau\left(u_i\right)\right)|_{u_i=\left(y_i+\epsilon-\hat{f}_i^{New}\right)} \cdots$$
$$\ldots + \mathbf{S}_{\boldsymbol{\lambda}}\hat{\boldsymbol{\beta}}^{New} \tag{4.6}$$

There are 3 cases:

1. Case 1: $y_i > \hat{f}_i$. In this case, we see that if ϵ is sufficiently small, we have also have $y_i + \epsilon > \hat{f}_i$. This means that if we take $\hat{f}_i^{New} = \hat{f}_i$ (as the only change we made was to increase y_i, we also take $\hat{f}_j^{New} = \hat{f}_j, \quad j = 1, ..., n, j \neq i$ and, therefore $\hat{\boldsymbol{\beta}}^{New} = \hat{\boldsymbol{\beta}}$), the subgradient condition is verified for any value of ϵ sufficiently small. Because the REL is convex, it has a single minimum and therefore, as we found one value of $\hat{f}_i^{New}$ that verifies the subgradient condition, it is the minimum. Hence, $df_i = \frac{\partial \hat{f}_i}{\partial y_i} = \lim_{\epsilon\to 0}\frac{f_i^{New}(y_i+\epsilon)-\hat{f}_i(y_i)}{\epsilon} = \lim_{\epsilon\to 0}\frac{0}{\epsilon} = 0$

2. Case 2: $y_i < \hat{f}_i$. Taking the same reasoning as in Case 1, we also have $df_i = 0$

3. Case 3: $y_i = \hat{f}_i$. We defined the following in chapter 3:

   - The "Below" set: $\mathcal{B} = \left\{i \in 1, ..., n; y_i < \hat{f}_i\right\}$
   - The "Above" set: $\mathcal{A} = \left\{i \in 1, ..., n; y_i > \hat{f}_i\right\}$.
   - The "Interpolated" points set: $\mathcal{I} = \left\{i \in 1, ..., n; y_i = \hat{f}_i\right\}$.

   As $y_i = \hat{f}_i$, point i is in the "Interpolated" set. The model goes directly through the point. If we change y_i to $y_i^{New} = y_i + \epsilon$, as long there are no "interfering points" that will replace y_i as a support point of the model, y_i^{New} will remain the support point for the quantile. If ϵ was a large amount, at some point, a new point would enter the "Interpolated" set and y_i^{New} would move out of the "Interpolated" set. An illustration can be found on Figure 4.5. On this Figure, we can see that y_3 is a support point of the regression. Other points are very far. As we move point y_3 by

an amount sufficiently small to avoid interference from other points, the quantile regression model follows point y_3 1 to 1.

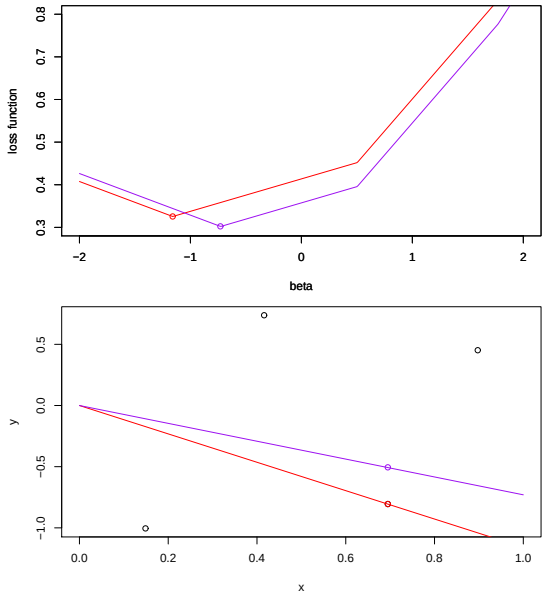


Fig. 4.5: The figure shows a fit of n=4 datapoints. The data is generated as $\underline{x}_i \sim \mathcal{U}[0,1]$, $y_i = \underline{x}_i + \varepsilon_i$, $\varepsilon_i \sim \mathcal{N}(0,1)$. We fit a model with a single parameter and no intercept ($y_i = \beta \times \underline{x}_i$). We will then shift a point where $y_i = \hat{f}_i$. In this case, we have $y_3 = \hat{f}_3$. On the top chart, we can see the empirical loss $\frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_\tau(y_i - \underline{x}_i\beta)$, $\tau = 0.25$. The minimum before a shift of y_3 is $\beta = -1.16$. We then shift $y_3^{New} = y_3 + 0.3$. We note that the new minimum of the empirical loss has shifted to $\beta = -0.73$. We note that point $(\underline{x}_3, y_3)$ is still a support point of the model (in purple). In this case, because the data is sparse, there is no "interfering point" that would take the place for point y_3 for quantile $\tau = 0.25$. $\hat{f}_3$ moves then 1 to 1 with y_3.

This means that as long as ϵ is small enough, we will have $\hat{f}_i^{New} = \hat{f}_i + \epsilon$, then the point remains in the "Interpolated" set, i.e. $y_i + \epsilon - f_i^{New} = 0$ and

$$\frac{\partial \hat{f}_i}{\partial y_i} = \lim_{\epsilon \to 0} \frac{\hat{f}_i^{New} - \hat{f}_i}{\epsilon} = \lim_{\epsilon \to 0} \frac{\hat{f}_i + \epsilon - \hat{f}_i}{\epsilon} = 1.$$

Note that we can write $y_i + \epsilon - f_i^{New}$ as: $\mathbf{x}_i^T \left(\hat{\boldsymbol{\beta}} + \boldsymbol{\xi}\epsilon - \hat{\boldsymbol{\beta}}^{New}\right)$ where $\boldsymbol{\xi}$ is such that $\mathbf{x}_i^T \boldsymbol{\xi} = 1$. Then $\hat{f}_i^{New} = \hat{f}_i + \epsilon$ corresponds to $\hat{\boldsymbol{\beta}}^{New} = \hat{\boldsymbol{\beta}} + \boldsymbol{\xi}\epsilon$.

This means that after fixing the value of $\hat{f}_i^{New}$ to $\hat{f}_i + \epsilon$, the determination

of $\boldsymbol{\beta}^{New}$ becomes a new equality constrained optimization:

$$\hat{\boldsymbol{\xi}} \in \text{argmin}_{\boldsymbol{\xi}} \frac{1}{n} \sum_{j=1, j\neq i}^{n} \mathcal{L}_\tau \left(u_j\right) |_{u_j=\left(y_j - \hat{f}_j - \mathbf{x}_j^T \boldsymbol{\xi}\epsilon\right)} + \frac{1}{2} \left(\hat{\boldsymbol{\beta}} + \boldsymbol{\xi}\epsilon\right)^T \mathbf{S}_{\boldsymbol{\lambda}} \left(\hat{\boldsymbol{\beta}} + \boldsymbol{\xi}\epsilon\right)$$
$$s.t. \mathbf{x}_i^T \boldsymbol{\xi} = 1$$

Overall, we see that for points that are interpolated, $\widehat{edf}_i = \frac{\partial \hat{f}_i}{\partial y_i} = 1$ and for points that are not interpolated $\widehat{edf}_i = \frac{\partial \hat{f}_i}{\partial y_i} = 0$, then:

$$\widehat{edf} = \sum_{i=1}^{n} \frac{\partial \hat{f}_i}{\partial y_i} = \#\mathcal{I} = \text{Number of Interpolated Points}$$

4.4 Bias in edf

We proposed to replace the pinball loss function by its expectation, itself approximated by a surrogate loss. This surrogate loss rounded at level $\hat{\alpha}$ is used only to calculate the edf (see section 4.3). In this section, we take an example to show the impact of replacing the exact pinball loss by the surrogate loss rounded at $\hat{\alpha}$ on the edf and GACV. We assume that we have a single covariate $\underline{x}$ that is generated using a uniform distribution and that y is bimodal:

$$\begin{aligned}
\underline{x}_i &\sim \mathcal{U}\left[0,1\right] \\
y_i &\sim \underline{x}_i + 10 \times (\underline{z}_i - 0.5) + \epsilon_i \\
\underline{z}_i &\sim Bernoulli(0.5) \\
\epsilon_i &\sim \mathcal{N}\left(0,1\right)
\end{aligned}$$

We then simulate a sample of size $n = 150$ and fit a smoothing spline with 15 basis functions. We will then compare the EDF and GACV curves with a small rounding and a loss function rounded at $\hat{\alpha}$ estimated using formula 3.15. To calculate the GACV and EDF curves, the value of penalty parameter ρ is taken on a grid with values $rhoArr = \{-20, 19.99, 19.98, ..., 1.99, 2\}$. The first level of rounding $\hat{\alpha}$ is estimated using formula 3.15. We obtain $\hat{\alpha} = 3.16$. The second level of rounding, is the small value that we fix at $\alpha = 10^{-4}$ to obtain values that are sufficiently precise. We call this value α_L. We will continue to use this terminology for a precise estimation of the model in section 5.2, and chapter 7. The formula used are for the precise estimate:

$$GACV_{\alpha_L} = \frac{\sum_{i=1}^{n} \mathcal{L}_{\alpha_L,\tau} \left(y_i - \tilde{f}_{\alpha_L,i}\right)}{n - Tr\left(\tilde{\mathbf{A}}_{\alpha_L}\right)}$$
$$edf_{\alpha_L} = Tr\left(\tilde{\mathbf{A}}_{\alpha_L}\right)$$

For the estimate using a large rounding $\hat{\alpha}$, we use the method introduced in the previous chapter where the rounding used to calculate the edf is $\hat{\alpha}$ but all the other elements of the formula are estimated using α_L:

$$GACV_{\hat{\alpha},\alpha_L} = \frac{\sum_{i=1}^{n} \mathcal{L}_{\alpha_L,\tau}\left(y_i - \tilde{f}_{\alpha_L,i}\right)}{n - Tr\left(\mathbf{A}_{\hat{\alpha},\alpha_L}\right)}$$
$$edf_{\hat{\alpha},\alpha_L} = Tr\left(\mathbf{A}_{\hat{\alpha},\alpha_L}\right)$$

We then estimate the model for quantile $\tau = 0.75$ and calculate the GACV and EDF curves. On Figure 4.6, we plot the effective degrees of freedom (top chart), the GACV curve (middle) and the resulting curve with the selected value for the smoothing parameter ρ. In this example, replacing the exact loss with the expected loss itself replaced by a surrogate loss rounded at $\hat{\alpha}$ leads to a small bias in the estimation compared to using a very small $\alpha = \alpha_L$ to estimate the edf. Then we can see that the selected ρ is slightly higher with the surrogate loss than with a loss function close to the exact pinball loss. By replacing the loss function by the expected loss itself replaced by a surrogate loss rounded at $\hat{\alpha}$, we obtain a smoother edf and gacv curves with a small bias. In this example, using the precise GACV criterion would lead to a slightly less smooth fit.

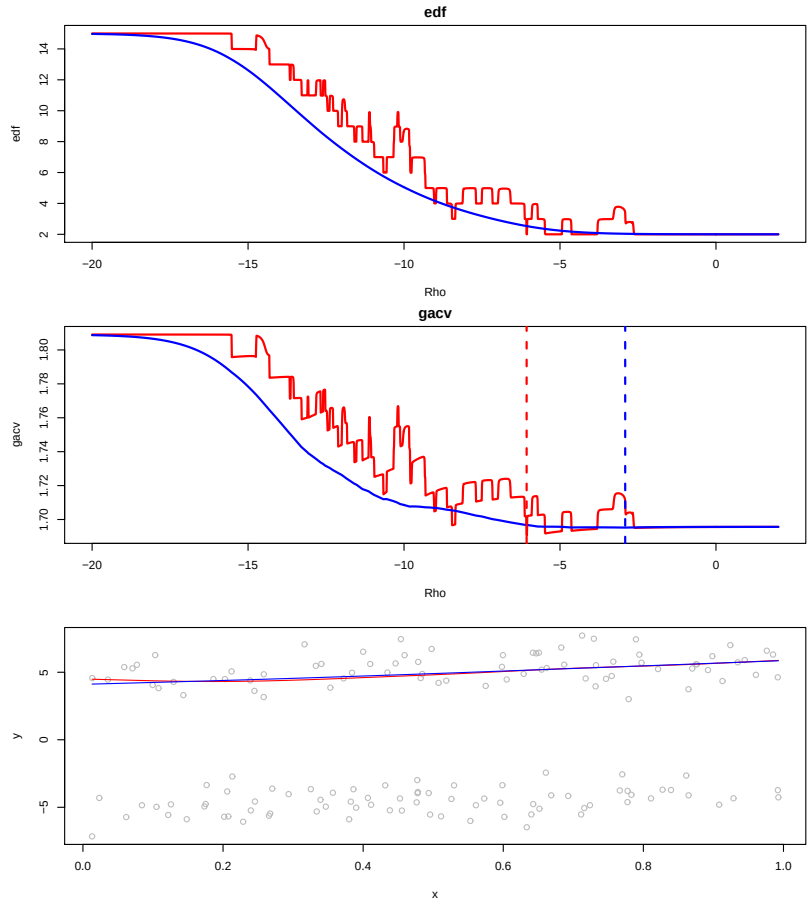


Fig. 4.6: On the top chart, we can see the edf curve with a pinball loss close to the exact pinball loss in red and the edf with the surrogate loss with parameter $\hat{\alpha}$ in blue. On the middle chart, we can see the GACV criterion with a small rounding in red and with $\hat{\alpha}$ in blue. The minimum of the gacv rounded at $\hat{\alpha}$ and at a very small rounding are indicated by a blue and a red vertical dotted vertical lines respectively. In this example, using the smoothed edf leads to a slightly smoother fit. On the bottom chart we can see the resulting fits are quite close (red to the surrogate loss close to the exact pinball loss and blue with the smoothed loss at $\hat{\alpha}$).

4.5 *Issue with the derivation of GACV for quantile regression*

The derivation of the QGACV solves an issue in the method used to derive a GACV criterion for the pinball loss. In this section, we detail the issue that occurs if we were trying to derive a GACV criterion directly using the exact

pinball loss function. The LOOCV criterion as in equation 3.1 is:

$$V_0 = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_\tau(y_i - \hat{f}_i^{[-i]})$$

where:

$$\hat{f}_i^{[-i]} = \mathbf{x}_i^T\boldsymbol{\beta}^{[-i]}$$

and

$$\hat{\boldsymbol{\beta}}^{[-i]} \in \text{argmin}_{\boldsymbol{\beta}}\frac{1}{n}\sum_{j=1,j\neq i}^{n}\mathcal{L}_\tau(y_j - \mathbf{x}_j^T\boldsymbol{\beta}) + \mathcal{P}(\boldsymbol{\lambda})$$

The Approximate Cross Validation (ACV) as proposed in Nychka et al. [1995] is a first order approximation of $\hat{f}_i^{[-i]}$. To derive this first order approximation, in section 3.4, we replaced the exact pinball loss by the expected loss, itself approximated by a smooth surrogate with a rounding parameter equal to $\hat{\alpha}$. However, as we have seen in section 4.3 that $y_i \to \hat{f}_i(y_i)$ is continuous and that $\frac{\partial f_i}{\partial y_i} = 1$, we may want to try to derive the ACV criterion with the exact pinball loss. Li et al. [2007] makes the remark that if in the GACV formula derived in Yuan [2006], we replace the $Tr(\mathbf{A})$ by the number of interpolated points then we could obtain a formula for the GACV with the exact pinball loss:

$$GACV = \frac{\sum_{i=1}^{n}\mathcal{L}_\tau(y_i - \hat{f}_i)}{n - edf} \tag{4.7}$$

We are now going to call $\hat{a}_{PL,i} = \frac{\partial \hat{f}_i}{\partial y_i}$, the sensitivity of the predicted value calculated with the exact pinball loss with respect to y_i. Let's try to derive this from first principles. With the exact pinball loss, the first order approximation would be written:

$$\begin{aligned}
\hat{a}_{PL,i} =& \frac{\partial \hat{f}_i}{\partial y_i} \approx \frac{\left(\hat{f}_i - \hat{f}_i^{[-i]}\right)}{\left(y_i - \hat{f}_i^{[-i]}\right)} \\
&\Rightarrow \left(\hat{f}_i - \hat{f}_i^{[-i]}\right) \approx \hat{a}_{PL,i}\left(y_i - \hat{f}_i^{[-i]}\right) \\
&= \hat{a}_{PL,i}\left(y_i - \hat{f}_i + \hat{f}_i - \hat{f}_i^{[-i]}\right) \\
&\Rightarrow \left(\hat{f}_i - \hat{f}_i^{[-i]}\right) \approx \frac{\hat{a}_{PL,i}\left(y_i - \hat{f}_i\right)}{1 - \hat{a}_{PL,i}}
\end{aligned}$$

However, here, the main issue is that with the exact pinball loss:

$$\hat{a}_{PL,i} = \frac{\partial \hat{f}_i}{\partial y_i} = \begin{cases} 0 & \text{if } y_i \neq \hat{f}_i \\ 1 & \text{if } y_i = \hat{f}_i \end{cases}$$

This means that

$$\left(\hat{f}_i - \hat{f}_i^{[-i]}\right) \approx \begin{cases} 0 & \text{if } y_i \neq \hat{f}_i \\ \text{not determined} & \text{if } y_i = \hat{f}_i \\ \left(\text{because we have } \left(y_i - \hat{f}_i\right)\text{=0 and } (1 - \hat{a}_{PL,i} = 0)\right) & \end{cases}$$

It could be argued that given the way we derived the ACV criterion $\tilde{V}_1$ (formula 3.24), the value of $\left(\hat{f}_i - \hat{f}_i^{[-i]}\right)$ for $(y_i - \hat{f}_i) = 0$ could be calculated as a limit. This is because in $\tilde{V}_1 = \frac{1}{n}\sum_{i=1}^{n} \frac{\mathcal{L}_\tau\left(y_i - \hat{f}_i\right)}{1-a_i}$, the rounding of the loss function only appears in hat matrix $\mathbf{A}$ as we are using the exact pinball loss for all other components of the formula. Therefore, if $a_i \underset{\alpha\to 0}{\to} 1$, as $y_i - \hat{f}_i = 0$, $\forall\alpha$, we have:

$$\left(\hat{f}_i - \hat{f}_i^{[-i]}\right)|_{y_i=\hat{f}_i} \approx \frac{a_i\left(y_i - \hat{f}_i\right)}{1 - a_i}|_{y_i=\hat{f}_i} \underset{\alpha\searrow 0}{\to} 0$$

This would mean that for our ACV with the exact pinball loss, we would have $\left(\hat{f}_i = \hat{f}_i^{[-i]}\right) \quad \forall i$. The ACV criterion would then be simply $\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_\tau(y_i - \hat{f}_i)$. Regarding the convergence of a_i to 1, although we do not have a general proof we show in Appendix G that for a linear quantile regression with one covariate, $a_i \underset{\alpha\to 0}{\to} 1$. This then means that a GACV-type criterion as in Li et al. [2007] where the edf is replaced by the number of interpolated points could not be derived.

4.6 Other possible explanations for underperformance of ACV and GACV

In section 3.6, we use an explanation from Reiss and Huang [2012] for the relative poor performance of the GACV for non-central quantiles. However, some other reasons for this issue have been proposed. In this section, we review some of the other possible explanations.

4.6.1 Flipped Points

The derivation of the ACV in Yuan [2006] relies on the following approximation:

$$\mathcal{L}_{\alpha,\tau}\left(y_i - \tilde{f}_i\right) - \mathcal{L}_{\alpha,\tau}\left(y_i - \tilde{f}_i^{[-i]}\right) \approx \left(\tilde{f}_i^{[-i]} - \tilde{f}_i\right) \times \mathcal{L}'_{\alpha,\tau}\left(y_i - \tilde{f}_i\right) \tag{4.8}$$

i.e. as long as $\tilde{f}_i$ and $\tilde{f}_i^{[-i]}$ are on the same side of y_i, if α is sufficiently small, function $\mathcal{L}_{\alpha,\tau}$ is approximately linear and the slope is approximately $\mathcal{L}'_{\alpha,\tau}$. Tu et al. [2019] argue that a potential reason for the relative poor performance of the GACV for non-central quantile is the observation that there are "flipped points". The "flipped points" are points where $\tilde{f}_i$ is on one side of y_i and $\tilde{f}_i^{[-i]}$

is on the other side of y_i. In this case, the approximation 4.8 is not very accurate.

We can try to apply this to the example of Figure 3.4. In this example, the central quantile was fitting correctly with the GACV but quantile $\tau = 0.925$ was not fitting correctly. We propose to test it the following way. After having calculated the LOOCV fits: $\tilde{f}_i^{[-i]}$, we test if $\tilde{f}_i^{[-i]}$ is on the same side of y_i as $\tilde{f}_i$. Then we can first calculate a corrected ACV. To do this, we replace the "flipped points" in the ACV by their actual LOOCV estimates, i.e.

$$\begin{gathered}\textbf{If } \left(\text{sign}\left(y_i - \tilde{f}_i^{[-i]}\right) \neq \text{sign}\left(y_i - \tilde{f}_i\right)\right) \\ \textbf{Then replace } \frac{\mathcal{L}_{\alpha,\tau}\left(y_i - \tilde{f}_i\right)}{1-\tilde{a}_i} \text{ by } \mathcal{L}_{\alpha,\tau}\left(y_i - \tilde{f}_i^{[-i]}\right) \text{ in the ACV criterion}\end{gathered}$$

We can also calculate a corrected GACV. To calculate the edf of the GACV, we average the weights $\tilde{a}_i$. We could correct the $\tilde{a}_i$ for which if sign $\left(y_i - \tilde{f}_i^{[-i]}\right) \neq$ sign $\left(y_i - \tilde{f}_i\right)$. We correct "flipped points" by replacing $\tilde{a}_i$ by $\tilde{a}_i^{corrected}$, the value of $\tilde{a}_i$ we would need to have to obtain the correct LOOCV value. Note that this means that the $\tilde{a}_i^{corrected}$ could be greater than 1. Then the GACV is calculated as before, with the only difference that the $\tilde{a}_i$ are replaced by $\tilde{a}_i^{corrected}$ in the averaging. To derive $\tilde{a}_i^{corrected}$ we use the following:

$$\begin{aligned}\mathcal{L}_{\alpha,\tau}\left(y_i - \tilde{f}_i^{[-i]}\right) &= \frac{\mathcal{L}_{\alpha,\tau}\left(y_i - \tilde{f}_i\right)}{1 - \tilde{a}_i^{corrected}} \\ \Rightarrow \tilde{a}_i^{corrected} &= \frac{\mathcal{L}_{\alpha,\tau}\left(y_i - \tilde{f}_i^{[-i]}\right) - \mathcal{L}_{\alpha,\tau}\left(y_i - \tilde{f}_i\right)}{\mathcal{L}_{\alpha,\tau}\left(y_i - \tilde{f}_i^{[-i]}\right)}\end{aligned}$$

The results of this experiment can be found on Figure 4.7. As we can see over the whole regularization path over the 271 grid points of $rhoGrid = \{-25, -24.9, ..., 1.9, 2\}$, there are only 4 points where there is 1 datapoint where $\tilde{f}_i$ is on one side of y_i and $\tilde{f}_i^{[-i]}$ is on the other side of y_i (top left chart). We can also see that the corrected GACV has the same minimum as the uncorrected GACV, resulting in exactly the same fit as the original GACV. Therefore, we can conclude that although the remark of Tu et al. [2019] is correct, this may not be the only reason why there is overfit in extreme quantiles and at least in this example, correcting for the flipped points did not seem sufficient to correct the overfitting.

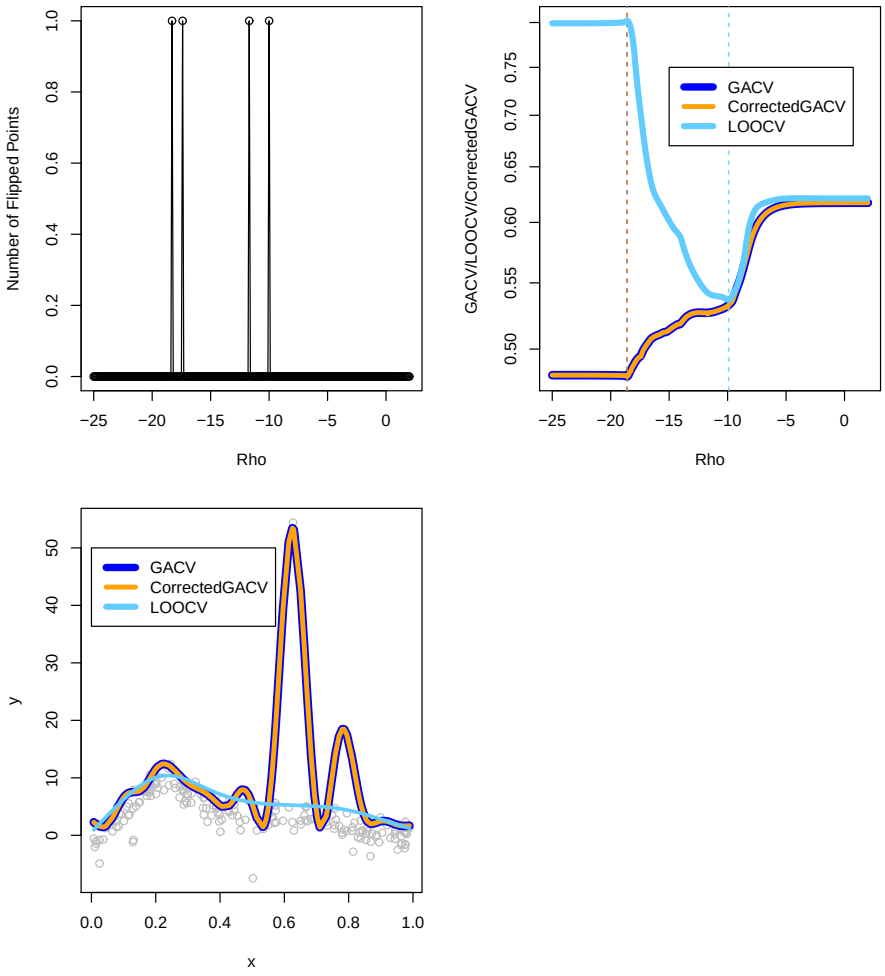


Fig. 4.7: On this Figure, we try to correct for the flipped points as described in Tu et al. [2019]. The chart on the top left represents the "Flipped Points" (1) vs non "Flipped Points" (0) for each point of the grid $rhoGrid = \{-25, -24.9, ..., 1.9, 2\}$ with $\alpha = 0.1$. The top right chart shows the GACV, LOOCV, and CorrectedGACV, where we correct to take into account the flipped points. We note that there is little difference between the corrected and non-corrected GACV. The chart on the bottom left shows the resulting fits. As the GACV ad CorrectedGACV curves have the same minimum, the fits using the GACV and CorrectedGACV are the same.

4.6.2 Approximation for the GACV formula

Yuan [2006] derives GACV formula 3.9. This derivation relies on the following approximation for the ACV:

$$V_1 = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_{\alpha,\tau}\left(\frac{y_i - \tilde{f}_{\alpha,i}}{1-\tilde{a}_i}\right) \approx \frac{1}{n}\sum_{i=1}^{n}\frac{\mathcal{L}_{\alpha,\tau}\left(y_i - \tilde{f}_{\alpha,i}\right)}{1-\tilde{a}_i}$$

If we averaged the weights of the ACV without this approximation, then the GACV formula obtained would be:

$$CorrectedGACV2 = \breve{V}_2 = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_{\alpha,\tau}\left(\frac{y_i - \tilde{f}_{\alpha,i}}{1-\frac{Tr(\tilde{\mathbf{A}})}{n}}\right) \tag{4.9}$$

Although we have not tested this approximation thoroughly, given a relatively small α, it seems that V_2 and $\breve{V}_2$ give relatively similar results. However, for a large α, this approximation may not be appropriate any longer. As an illustration, we generate the following example:

$$\begin{aligned}
\underline{x}_i &\sim \mathcal{U}[0,1] \quad i = 1,...,n\\
y_i &= f_1(\underline{x}_i) + \epsilon_i\\
\epsilon_i &\sim \mathcal{N}(0,1)\\
f_1(\underline{x}_i) &= 0.2 \times \underline{x}_i^{11} \times (10 \times (1-\underline{x}_i))^6 + 10 \times (10 \times \underline{x}_i)^3 \times (1-\underline{x}_i)^{10}
\end{aligned}$$

Function f_1 is taken from package 'mgcv' function 'gamSim', function f_1 plus a noise following a standard normal distribution is referred to as Example 1 in this document (section 9.1). We then run the following test. We calculate a GACV and the CorrectedGACV2 using two values of α: $\alpha = 0.05$ and $\alpha = 1$ on a grid of values for ρ: $rhoGrid = \{-30, -29.5, ..., 4.5, 5\}$. We then fit a smoothing spline based on the smoothing parameter selected using the Generalized IRLS algorithm that will be introduced in section 5.2. Specifically, we use Algorithm 11, section 7.11. This algorithm has a stopping condition and gives a precise estimate of the vector of parameter. The final level of rounding α_L is selected based on a stopping condition. We obtain $\alpha_L = 8.22 \times 10^{-6}$ or $\alpha_L = 4.11 \times 10^{-6}$. The results are displayed on Figure 4.8. For a small rounding parameter $\alpha = 0.05$, the correction is really small. For a large rounding parameter $\alpha = 1$, the difference is much larger. However, in this example, the resulting fit is similar to the fit without correction. Even with an extremely large rounding, there is little difference. It then seems that with the usual small rounding used in Yuan [2006] or Reiss and Huang [2012], this approximation is not sufficient to explain the overfit observed for GACV with non-central quantiles.

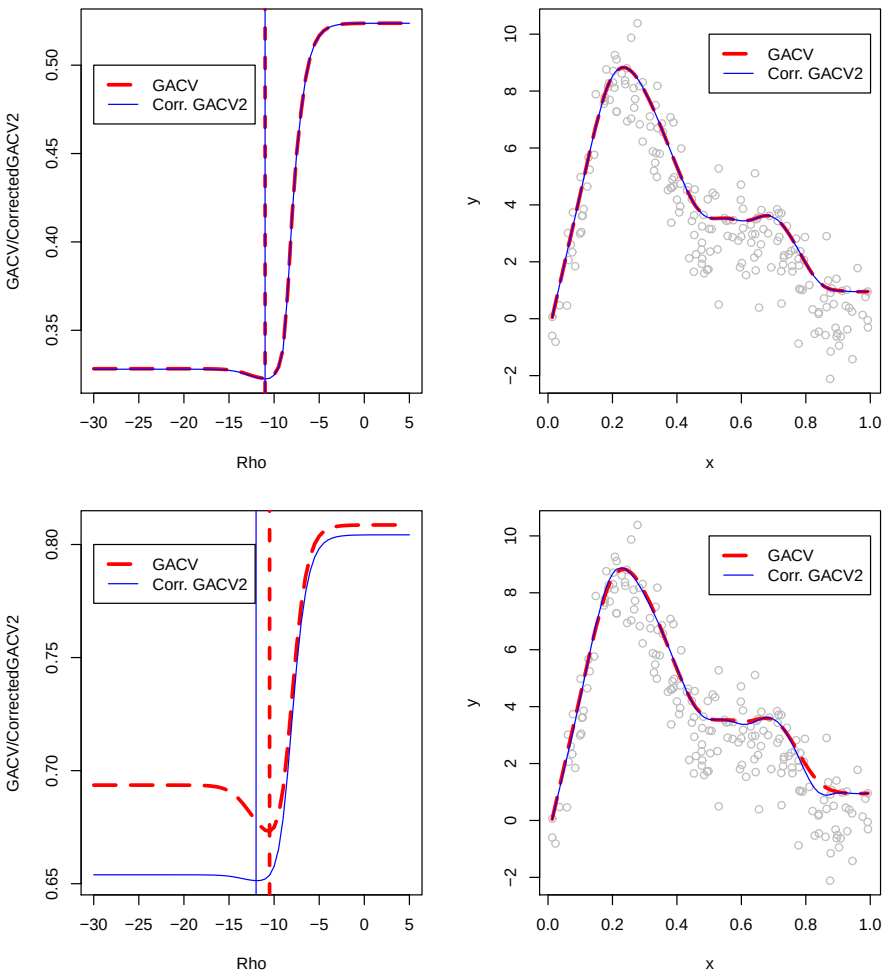


Fig. 4.8: On this picture, we can see the effect of using a corrected GACV formula where we take into account the fact that for a non-zero α, $\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_{\alpha,\tau}\left(\frac{y_i-\tilde{f}_{\alpha,i}}{1-\tilde{a}_i}\right) \neq \frac{1}{n}\sum_{i=1}^{n}\frac{\mathcal{L}_{\alpha,\tau}\left(y_i-\tilde{f}_{\alpha,i}\right)}{1-\tilde{a}_i}$. On the charts, the (uncorrected) GACV $\left(GACV = \frac{\sum_i \mathcal{L}_{\alpha,\tau}(\tilde{u}_i)}{n-Tr\left(\tilde{\mathbf{A}}\right)}\right)$ is shown in red whereas the corrected GACV $\left(= \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_{\alpha,\tau}\left(\frac{y_i-\tilde{f}_{\alpha,i}}{1-\frac{Tr\left(\tilde{\mathbf{A}}\right)}{n}}\right)\right)$ is shown in blue. On the top row, we can see that for a small value of α, there is little difference between the GACV and the corrected GACV. On the bottom row, for a larger α, the difference may become non-negligible.

4.6.3 Higher derivatives

Our derivation of the ACV in section 3.5 as well as Yuan [2006]'s are based on the remark that if we replace y_i by $\tilde{f}_i^{[-i]}$, then $\tilde{f}_i$ changes to $\tilde{f}_i^{[-i]}$. This remark gives us a way to estimate the derivative of $\tilde{f}_i$ with respect to y_i:

$$\frac{\partial \tilde{f}_i}{\partial y_i} \approx \frac{\left(\tilde{f}_i^{[-i]} - \tilde{f}_i\right)}{\left(\tilde{f}_i^{[-i]} - y_i\right)} \Rightarrow \left(\tilde{f}_i^{[-i]} - \tilde{f}_i\right) \approx \frac{\partial \tilde{f}_i}{\partial y_i}\left(\tilde{f}_i^{[-i]} - y_i\right) \tag{4.10}$$

Yuan [2006] indicates that one issue with approximation 4.10 is that the first order approximation neglects higher order terms. Let's take an example to see how good the approximation is (this example is shown on Figure 4.9). We fit a generated example with the GACV criterion. We fit quantiles $\tau = 0.5$ and $\tau = 0.95$ and use $\alpha = 0.1$. We then take a point (for example here, we ranked the residuals and took the point the closest to the model fit). We then plot how $\tilde{f}_i$ changes as y_i changes by a given amount. We can see that for the central quantile, the relationship between y_i and $\tilde{f}_i$ is approximately linear over a large interval of values of y_i whereas it does not seem linear for the extreme quantile.

We could have investigated higher order expansions in the derivation of the approximation of the LOOCV (see Giordano et al. [2019]) by calculating the derivative $\frac{\partial^2 \tilde{f}_i}{\partial y_i^2}$. We did not explore this possibility in this thesis because first, with the exact pinball loss, these second order terms are equal to 0. The second order terms are then introduced artificially and change depending on the rounding of the surrogate loss α. Second, deriving the ACV criterion relies on a implicit equation that can then be simplified. Adding these second order terms may not lead to an expression that can be simplified and that is amenable to optimization to obtain the smoothing parameters automatically. Last, as seen on Figure 3.4, the ACV does not necessarily oversmooth extreme quantiles and may have a better performance than the GACV. The first order expansion may therefore not be the main reason for the overfit observed for non-central quantiles with the GACV. However, even if the observation made here was not used, studying the reasons for the non-linear behaviour of Figure 4.9 may be interesting and is given as a potential extension in the conclusion (Chapter 10).

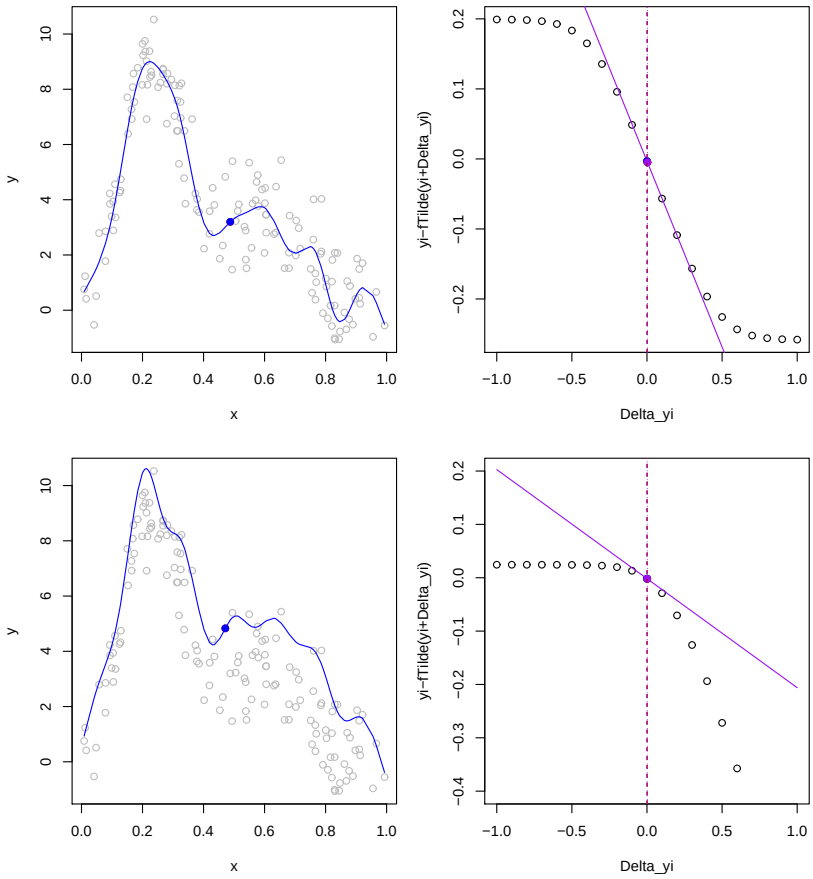


Fig. 4.9: On the top row, we can see a fit for $\tau = 0.5$. The selected point is represented by a blue dot. The point (x_i, y_i) selected is the closest point to the model. On the top right chart, we can see that if we move y_i up or down by Δ_{y_i}, the difference $\tilde{f}_i(y_i + \Delta_{y_i}) - \tilde{f}_i(y_i)$ changes linearly with Δ_{y_i}. The purple line represents the approximation of the relationship $\tilde{a}_i = \frac{\partial \tilde{f}_i}{\partial y_i}$. We use $\alpha = 0.1$. Then the purple line equation is $(y_i - \tilde{f}_i - \Delta_{y_i} \times \tilde{a}_i)$. On the bottom line, we fit a model y_i with the GACV for quantile $\tau = 0.95$. We take the closest point (x_j, y_j) to the model (blue dot). We repeat the same analysis as previously for quantile $\tau = 0.95$. We note that in this case, the change in $\tilde{f}_i$ when we change y_i by Δ_{y_i} is non linear. If y_i is increased by a positive $+\Delta_{y_i}$, the model tends to follow y_i. However, if we decrease y_i by an amount $-\Delta_{y_i}$, the support point changes. Then as y_i is reduced, $\tilde{f}_i$ does not change any longer. Using the slope a_i to predict the new value of $\tilde{f}_i$ for a large change in y_i would lead to a non-negligible error.

4.7 Overfitting Central Quantiles

Overfitting for non-parametric mean regression with a Generalized Cross Validation criterion (GCV) has been extensively discussed in the literature (see for example Lukas [2006], Lukas [2008], Lukas et al. [2010], Lukas et al. [2016]). Although it does not seem to have been reported yet, we can observe a similar type of overfit for the GACV criterion for quantiles close to the median. On Figure 4.10, we can see two examples of overfit. For these two examples, the ACV and GACV give similar fits. For central quantiles, the QGACV criterion has a multiplier of 2 for the edf. This is similar to what is proposed in Lukas [2006], Lukas et al. [2016] to correct the GCV criterion for mean GAM regression for $n \geq 100$. Owing to this, fitting with the QGACV suffers less from overfitting for central quantiles.

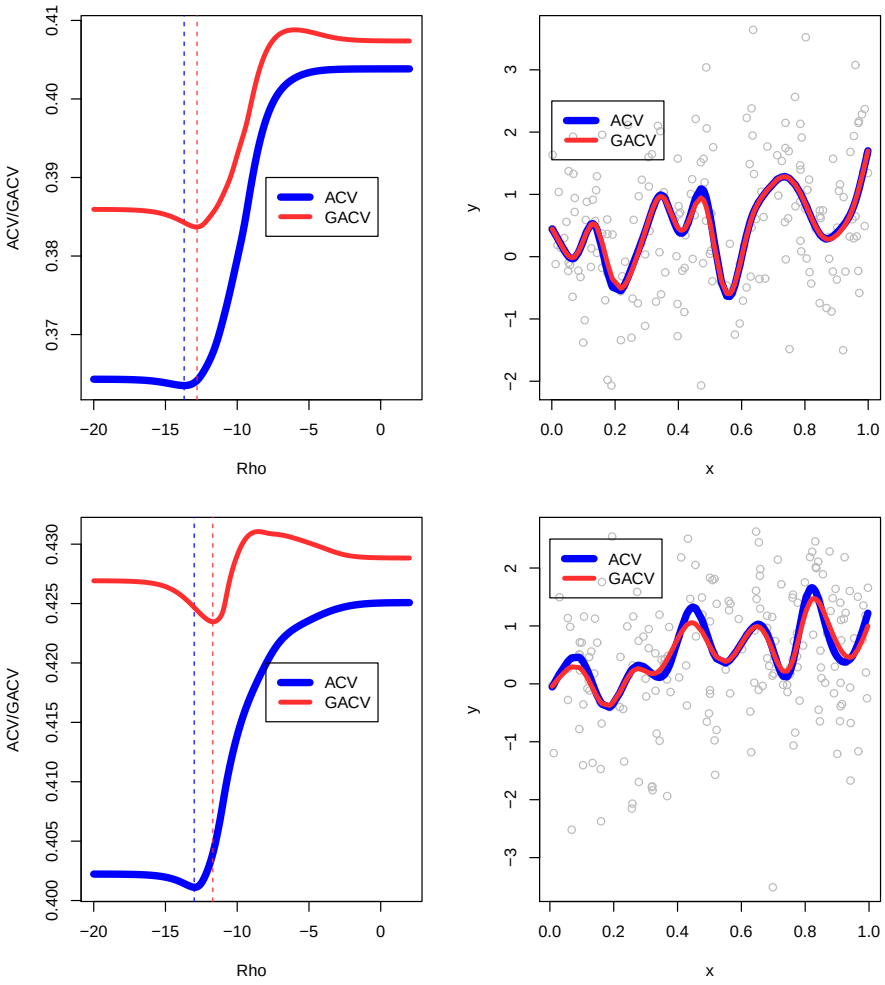


Fig. 4.10: This Figure shows two examples of overfits for central quantile ($\tau = 0.5$). To generate these charts, we take values of the smoothing parameter ρ on a grid of values $rhoGrid = \{-20, -19.9, ..., 1.9, 2\}$. We fix $\alpha = 0.05$. In both cases, the GACV and ACV give similar fits. We also note a behaviour that can sometimes make the GACV hard to optimize if we do not use the correct starting point. The GACV in both cases is not quasi-convex. Starting from large values of the smoothing parameter λ, the GACV rises, reaches a peak before falling again to a global minimum. If we start at high values of λ, we would then miss the global minimum. Of course, in this specific example, as the true quantile is a straight line, this would be beneficial. However, in the general case, we would want to find the global minimum of the GACV. This observation has already been made in the case of the Generalized Cross Validation criterion for GAMs in the reference manual of package 'mgcv' Wood [2020].

4.8 Equivalent multiplier

The QGACV formula is:

$$QGACV = \sum_{i=1}^{n} \frac{\mathcal{L}_\tau\left(y_i - \hat{f}_i\right)}{n.eff - cTr\left(\mathbf{A}\right)}$$

where $n.eff = 2n\phi$, $c = (1 + 2\phi)$ and $\phi = \min(\tau, 1 - \tau)$. Given that the QGACV is used for a fixed τ, we can multiply at the numerator and denominator by $\frac{n}{n.eff}$ without changing the minimum. Hence, the QGACV is equivalent to:

$$QGACV \equiv \sum_{i=1}^{n} \frac{\mathcal{L}_\tau\left(y_i - \hat{f}_i\right)}{n - \left(\frac{1+2\phi}{2\phi}\right) Tr\left(\mathbf{A}\right)} \tag{4.11}$$

Our formula could be seen as equivalent to a type of GACV formula where the multiplier of the edf changes with the quantile. On Figure 4.11, we can see a graphical representation of how the multiplier $\frac{1+2\phi}{2\phi}$ changes as τ changes. As discussed in section 4.7, we can see that the multiplier for $\tau = 0.5$ is 2 (as $\phi = \min(\tau, 1 - \tau) = 0.5$, then $\frac{1+2\phi}{2\phi} = 2$), corresponding to what has been proposed to mean spline regression fitted using GCV in Lukas et al. [2016] for the case $n \geq 100$.

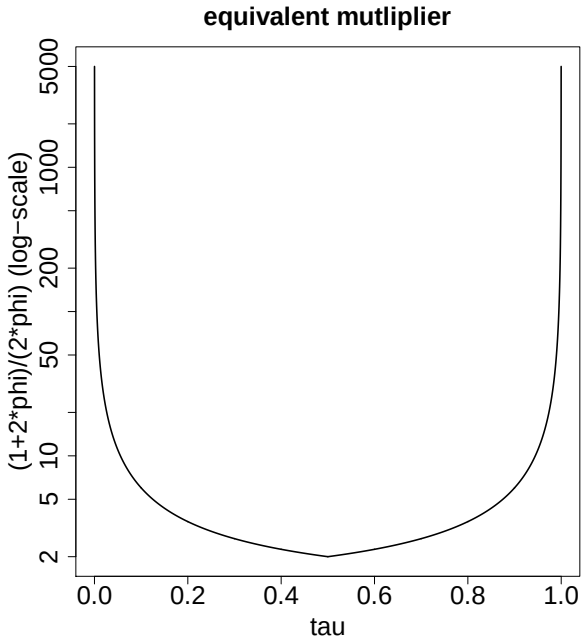


Fig. 4.11: change in the multiplier $\frac{1+2\phi}{2\phi}$ as τ changes. We note that the weight goes to infinity as we get closer to 0 or 1.

4.9 An alternative to the QGACV: the Smooth Loss GACV (SGACV)

So far, when we used ACV or GACV, apart from the calculation of the edf where we used a relatively high level of rounding $\hat{\alpha}$, we were using a small rounding parameter α. However, we note that as we increase the rounding, models obtained becomes smoother and smoother. On Figure 4.12, we calculate the same example as in Figure 3.4, but we use $\alpha = 1$ for all parts of the GACV instead. We can see that if we smooth all parts of the GACV criterion, we can avoid overfits.

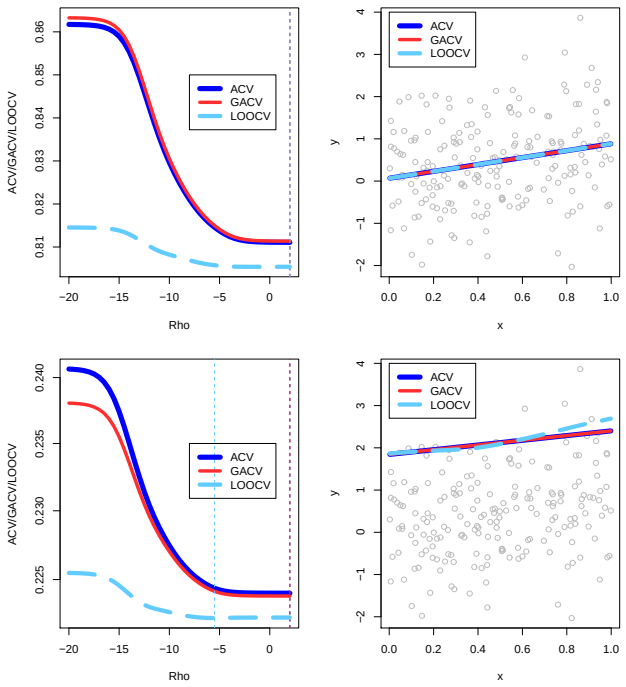


Fig. 4.12: This picture shows the same example as on Figure 4.10. The difference with Figure 4.10 is that we use a rounding $\alpha = 1$ to calculate the ACV and the GACV. All elements of the ACV, GACV and LOOCV are rounded at this level. We therefore use a biased estimate of the vector of parameters $\tilde{\beta}_{\alpha=1}$ in the calculation of the criteria. To have a precise estimate of the regression model, once the smoothing parameter has been chosen, we use Generalized IRLS with $\alpha_L = 10^{-4}$ to estimate the curve (see Algorithm 10, section 7.11). We optimize the criteria on a grid of values for ρ: $rhoGrid = \{-20, -19.5, ..., 1.5, 2\}$. We can see that with a more rounded loss function, the GACV, ACV and LOOCV all select a similar smooth fit.

To understand why using a higher degree of smoothness for the loss function may lead to a smoother fit selected by the GACV criterion, we can look at the

GACV optimization path. In sections 5.1 and 7.9, we will develop a method to optimize the QGACV. This method is implemented in package 'qgacv' introduced in chapter 8. Function 'qam' that is the function used to estimate a QAM fit using QGACV can also be used to fit a GACV-like formula where the edf is calculated using the trace of the hat matrix rounded at a fixed parameter $\alpha = \hat{\alpha}$. We can then visualize the optimization path of this GACV-like surface. Figure 4.13 shows an example path of an overfitted model, optimized using Generalized Fellner-Schall (see section 5.2). The path shows GACV curves at different levels of rounding. For a high level of rounding, the smoothing parameter selection seems appropriate. As we lower the rounding level α, the optimization leads us to smaller smoothing parameter values.

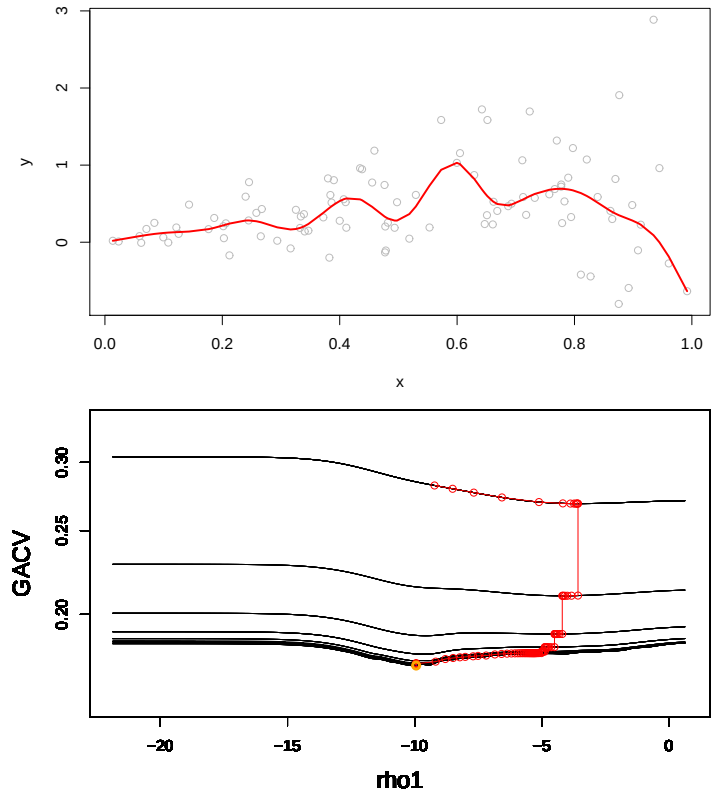


Fig. 4.13: This picture shows that with a high level of rounding, the minimum of the GACV is around -5. as we decrease the rounding level, a new minimum appears. The global minimum with a low level of rounding is around -10. This value of the smoothing parameter leads to an overfitted model.

In section 3.4, we replaced the change in the predicted value with and without observation i by its value under the expected loss. However, other parts of the LOOCV criterion were still calculated using the exact pinball loss. The

experiments point to the idea of using a rounded loss function for all parts of the GACV criterion (as in Jung et al. [2020]). We could follow our previous methodology by replacing the pinball loss by its expected value not only to estimate the LOOCV but also the function used to calculate the criterion and estimate the vector of parameters.

$$\begin{aligned}\overset{\bullet}{V}_{0,E} &= \frac{1}{n}\sum_{i=1}^{n} E_{y_i|\underline{\boldsymbol{x}}_i}\left[\mathcal{L}_\tau\left(y_i - \hat{f}_{E,\boldsymbol{\lambda}}\left(\underline{\boldsymbol{x}}_i\right) + \left(\hat{f}_{E,\boldsymbol{\lambda}}\left(\underline{\boldsymbol{x}}_i\right) - \hat{f}^{[-i]}_{E,\boldsymbol{\lambda}}\left(\underline{\boldsymbol{x}}_i\right)\right)\right)\right] \\ &= \frac{1}{n}\sum_{i=1}^{n} E_{y_i|\underline{\boldsymbol{x}}_i}\left[\mathcal{L}_\tau\left(y_i - \hat{f}^{[-i]}_{E,\boldsymbol{\lambda}}\left(\underline{\boldsymbol{x}}_i\right)\right)\right]\end{aligned}$$

Note that $\hat{f}^{[-i]}_{E,\boldsymbol{\lambda}}\left(\underline{\boldsymbol{x}}_i\right)$ does not depend on y_i. We then replace the outside expectation by the surrogate loss rounded at $\hat{\alpha}$:

$$\overset{\bullet}{V}_0 = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_{\hat{\alpha},\tau}\left(y_i - \tilde{f}^{[-i]}_{\hat{\alpha},i}\right) \tag{4.12}$$

In equation 3.23, we derived the following approximation:

$$\left(\tilde{f}_{\hat{\alpha},i} - \tilde{f}^{[-i]}_{\hat{\alpha},i}\right) \approx \frac{\tilde{a}_{\hat{\alpha},i}\left(y_i - \tilde{f}_{\hat{\alpha},i}\right)}{1 - \tilde{a}_{\hat{\alpha},i}}$$

Replacing in 4.12, we obtain the ACV criterion from Nychka et al. [1995] but where loss function $\mathcal{L}^{Nychka}_{\alpha,\tau}\left(u\right)$ was replaced by our surrogate loss (equation 3.7. Furthermore, Nychka et al. [1995] propose to have a small rounding whereas here we have a relatively high rounding $\hat{\alpha}$ in all part of the formula:

$$\overset{\bullet}{V}_1 = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_{\hat{\alpha},\tau}\left(\frac{y_i - \tilde{f}_{\hat{\alpha},i}}{1 - \tilde{a}_{\hat{\alpha},i}}\right) \tag{4.13}$$

We can then calculate the average of the weights to obtain the smoothed loss GACV criterion:

$$V_4 = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_{\hat{\alpha},\tau}\left(\frac{y_i - \tilde{f}_{\hat{\alpha},i}}{1 - \frac{Tr\left(\tilde{\mathbf{A}}_{\hat{\alpha}}\right)}{n}}\right) \tag{4.14}$$

Formula V_4 has not yet been implemented in package 'qgacv' (see Chapter 8). However, to test this idea, we can use the following approximation that can be used in package 'qgacv' (by replacing parameters n.eff by n and c by 1 in the QGACV formula (see Section 3.6) and use $\alpha_U = \hat{\alpha}$ (see section 7.6)):

$$SGACV = V_4 \approx \sum_{i=1}^{n} \frac{\mathcal{L}_{\hat{\alpha},\tau}\left(y_i - \tilde{f}_{\hat{\alpha},i}\right)}{n - Tr\left(\tilde{\mathbf{A}}_{\hat{\alpha}}\right)} \tag{4.15}$$

This approximation may not be accurate but still gives us an idea of the pertinence of using the SGACV criterion. The SGACV is similar to the GACV derived in Yuan [2006] but with the loss function that has been replaced by $\mathcal{L}_{\alpha,\tau}$ (equation 3.7) and with a high level of rounding $\hat{\alpha}$ derived by taking the expectation of the loss. Example fits with the SGACV approximation can be found on Figure 4.14. The SGACV approximation seems to give fits close to the true quantiles for examples with no outliers. However, we can see for example on Example 12 that extreme datapoints seem to lead to overfits for extreme quantiles. This is not the case with the QGACV. We can see on Figure 4.15 the optimization path for an extreme quantile ($\tau = 0.99$) with Example 12. The minimum of the SGACV is almost identical to the minimum of the GACV. This shows a limitation of the SGACV approach for extreme quantiles.

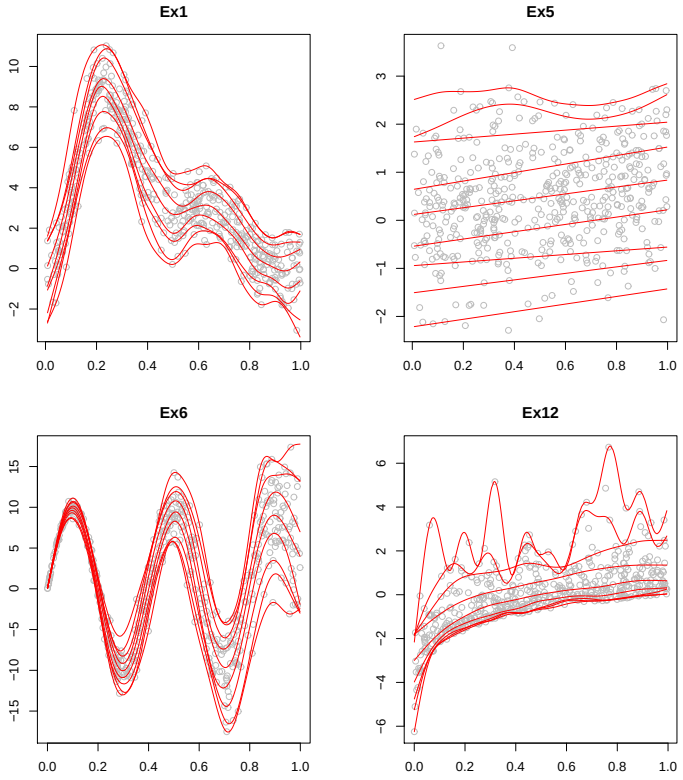


Fig. 4.14: Example fits with the SGACV criterion approximation. We note that the fits seem to be close to the true quantiles in these examples apart from the case where we fit extreme quantiles with data that has outliers (Ex12). We can look at the optimization path of the SGACV approximation.

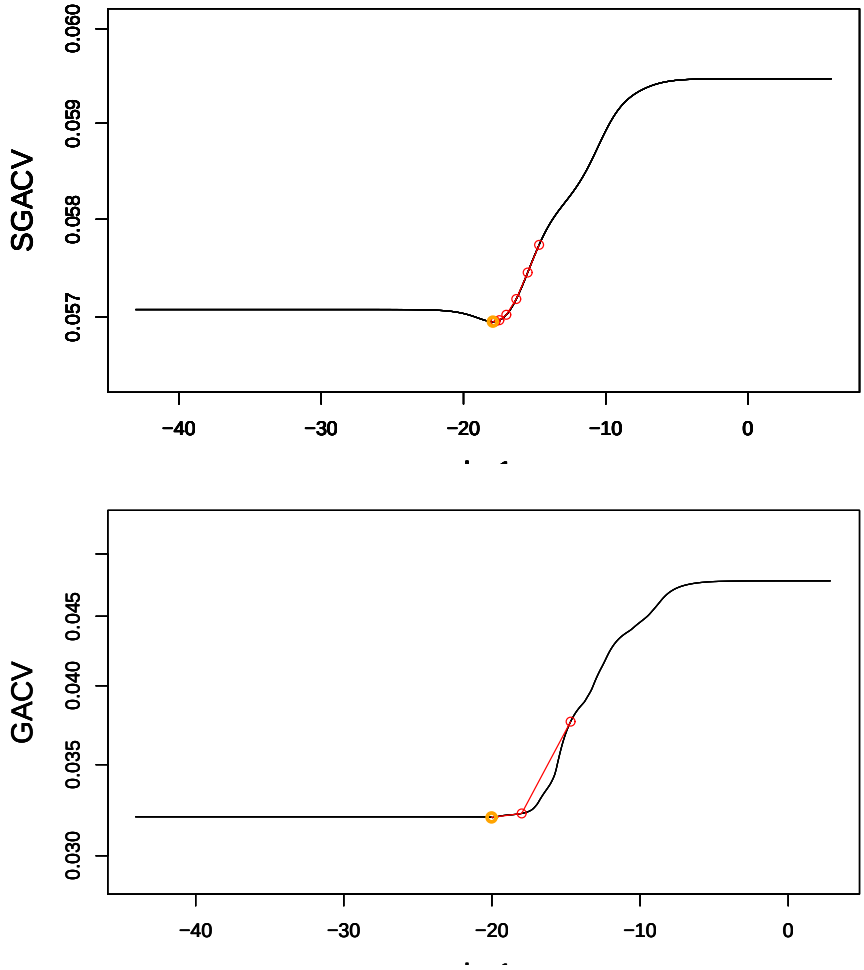


Fig. 4.15: Optimization path of the SGACV approximation and GACV, Example 12, tau=0.99. For the GACV, we also use function qgacv, replacing parameter $n.eff$ by n and parameter c by 1 and use $\hat{\alpha} = \alpha_U = \alpha_L = 0.05$ (see section 7 for further details). In this case, the GACV is sufficiently smooth to be optimized. We note that the minimum of the SGACV approximation is roughly the same as the minimum of the GACV. Therefore, in this case, the SGACV does not improve the fit compared to the GACV.

4.10 Application of the QGACV to QSS with exact pinball loss and QSS with total variation penalty

In this section, we would like to show the applicability of the QGACV criterion beyond the ridge-type penalized additive model. We show that it can also be used for a Total Variation penalized quantile smoothing spline model. In section 4.5 we noted that we could not derive a QGACV criterion with the exact pinball loss in a principled way to obtain formula:

$$QGACV = \frac{\sum_{i=1}^{n} \mathcal{L}_\tau(y_i - \hat{f}_i)}{n.eff - c \times edf} \tag{4.16}$$

where edf is the number of interpolated points. However, we could use the number of interpolated points as an approximate for $tr(\mathbf{A})$ to test the applicability of the QGACV formula to other types of regression with a pinball loss and a penalty. We could then optimize the QGACV on a grid to select the smoothing parameter. For example, Koenker [2019] shows how a quantile smoothing spline (qss) can be fitted using package 'quantreg' together with package 'splines'. To fit the qss, the model matrix is created using function

```
X <- model.matrix(y ~ bs(x, df = df),tau=tau,data=data)
fit <- rq(y ~ bs(x, df = df), tau = tau,data = data)
plot(data$x,data$y)
lines(data$x,X%*%fit$coef)
```

where *df* is the degrees of freedom of the smoothing spline. We can then fit multiple splines and calculate the QGACV for a range of degrees of freedom (from df=2,...,40)(for df=2, we are fitting a linear quantile regression) and choose the fit with the lowest qgacv. The resulting fits can be seen on Figure 4.16. We can see that in this example, the QGACV can be used to obtain fits that are close to true quantiles.

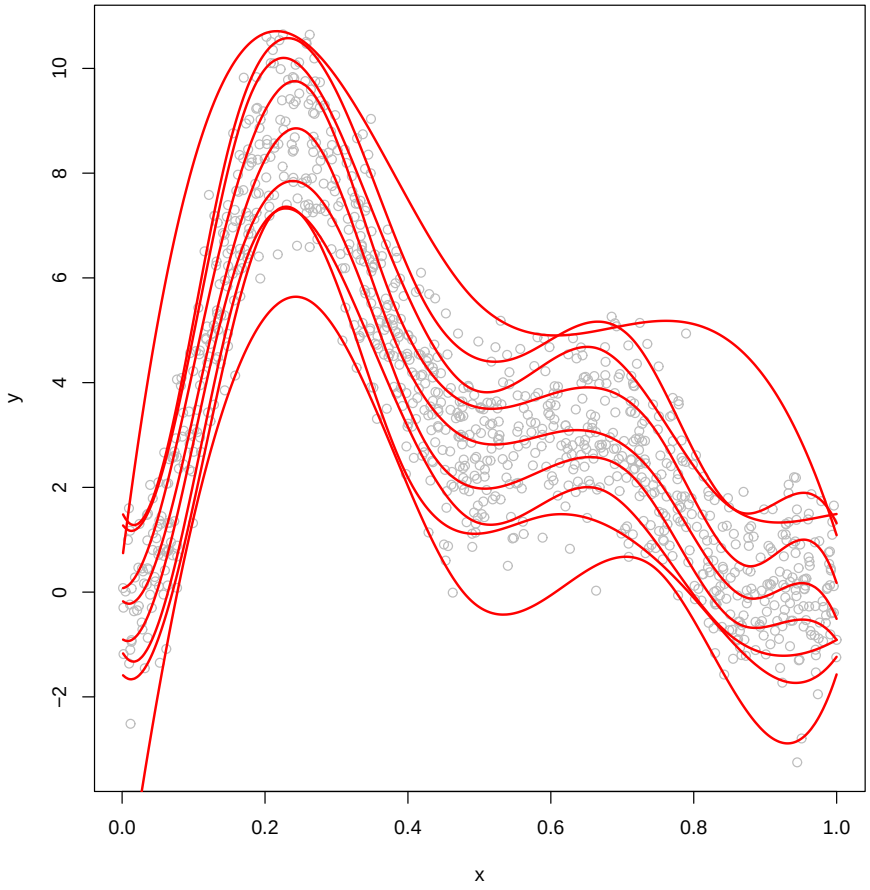


Fig. 4.16: On this picture, we show an example of quantile smoothing splines fitted using the QGACV. To calculate the QGACV, instead of using the edf calculated as the trace of the hat matrix $\mathbf{A}$ rounded at $\hat{\alpha}$, we use the exact pinball loss and count the number of datapoints that interpolate the model. The QGACV is optimized on a grid. We note that with this setting, we can still use the QGACV.

We could apply this reasoning to total variation penalized smoothing. Koenker et al. [1994] propose to use a total variation penalty instead of a ridge-type penalty for a penalized smoothing spline:

$$\hat{f} \in \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - f \left(\underline{\boldsymbol{x}}_i \right) \right) + \lambda \int \left| f'' \left(z \right) \right| dz \tag{4.17}$$

This type of quantile smoothing spline is solved using a Frisch-Newton in-

terior point method (Koenker and Ng [2005]) in package quantreg, function rqss.

```
fit<-rqss(y~qss(x,lambda=lambda),tau=tau,data=data)
```

we fit different models with the regularization parameter λ varying from 10^{-5} to 10. We then calculate the QGACV and pick the λ that gives the lowest QGACV. An example can be found on Figure 4.17. To determine the number of interpolated points, we note that for $\lambda \leq 10$, the residuals of the fitted model are of order between 10^{-14} and 10^{-12} for the interpolated points. To determine if a point is interpolated, we then count if the residual is smaller than 10^{-10}. We see that the QGACV criterion could be applied to the total variation penalized Quantile Additive Model. This direction of research should be explored further but this preliminary test seems to indicate that the QGACV can also be used to select correctly the degrees of freedom of piecewise linear quantile models. This could be compared to the approach presented in Muggeo et al. [2020].

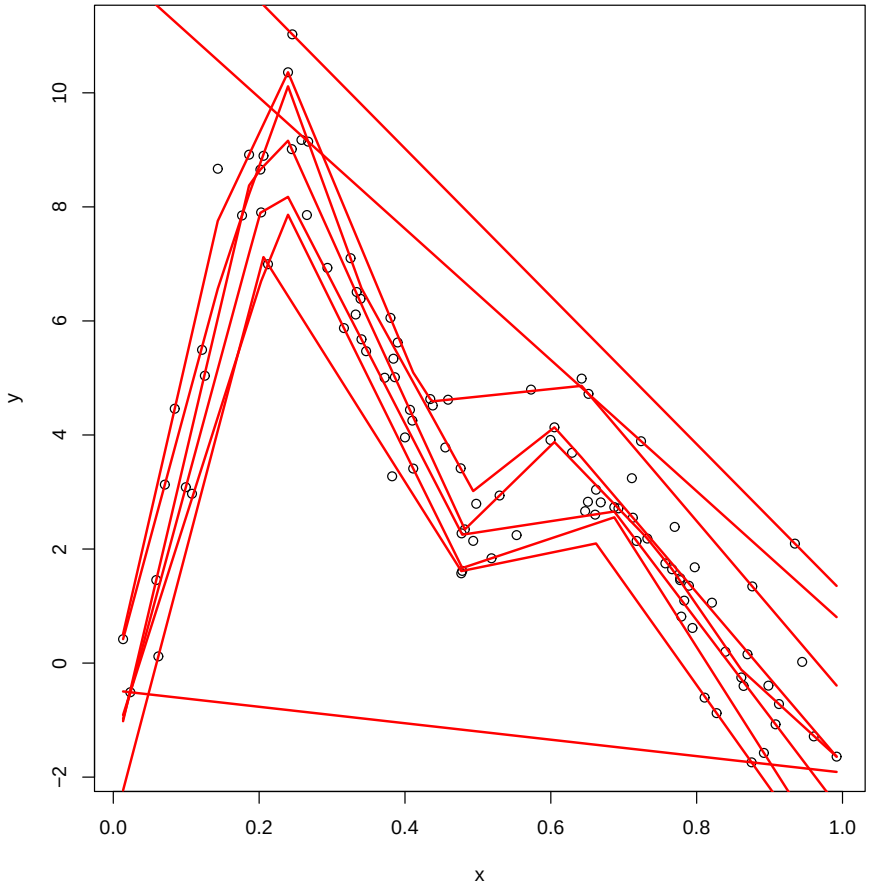


Fig. 4.17: Different fits using quantile smoothing spline with total variation penalty solved using Frisch-Newton interior point method as implemented in function quantreg::rqss. The regularization weight λ is chosen on a grid, based on the QGACV criterion where the edf is estimated by counting the number of interpolated points.

5. NUMERICAL IMPLEMENTATION

The estimation of the vector of smoothing parameters $\boldsymbol{\lambda}$ and the vector of parameters $\boldsymbol{\beta}$ can be framed as a bi-level optimization problem with an "Upper Level" (the estimation of the smoothing parameters) and a "Lower Level" (the estimation of the vector of parameters).

UpperLevel:

$$\hat{\boldsymbol{\lambda}} \in \operatorname{argmin}_{\boldsymbol{\lambda}} QGACV\left(\boldsymbol{\lambda}\right) := \frac{\sum_{i=1}^{n} \mathcal{L}_{\tau}\left(y_i - \mathbf{x}_i^T \hat{\boldsymbol{\beta}}\right)}{n.eff - cTr\left(\mathbf{A}\right)}$$

LowerLevel :

$$s.t. \hat{\boldsymbol{\beta}} \in \operatorname{argmin}_{\boldsymbol{\beta}} \left(\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{\tau}\left(y_i - \mathbf{x}_i^T \boldsymbol{\beta}\right) + \frac{1}{2} \boldsymbol{\beta}^T \left(\sum_{k} \lambda_k \mathbf{S}_k \right) \boldsymbol{\beta} \right)$$

where: $\mathbf{A} = \mathbf{X}\left(\mathbf{X}^T \mathbf{W} \mathbf{X} + \mathbf{S}_{\boldsymbol{\lambda}}\right)^{-1}\left(\mathbf{X}^T \mathbf{W}\right)$; $\quad \mathbf{W} = diag\left(w_1, ..., w_n\right)$
$w_i = \mathcal{L}''_{\hat{\alpha},\tau}\left(y_i - \mathbf{x}_i^T \hat{\boldsymbol{\beta}}\right)$; $n.eff = 2n\phi$; $c = (1 + 2\phi))$; $\phi = \min\left(\tau, 1 - \tau\right)$

The issues we are facing to solve this problem are the following:

1. The lower level problem is a non-smooth and convex optimization problem.
2. The upper level is a non-smooth and non-convex optimization problem.
3. As we update the vector of smoothing parameters $\boldsymbol{\lambda}$, the estimated vector of parameters $\boldsymbol{\beta}$ must also be updated.

For more information on nonsmooth bilevel optimization problems, see Rosset [2009], Ochs et al. [2015], Gould et al. [2016],Ochs et al. [2016], Feng and Simon [2018]. Furthermore, we have the following requirements:

R.1 **Scalable Method without Parallelization**: The method should handle as large a number of observations as possible: some methods work well for a relatively small number of observations but are not scalable (we preclude the use of parallel processing).

R.2 **Accurate Convergence**: we must be able to estimate the model to an arbitrary level of error relatively fast.

R.3 **Convergence speed not linked to smoothing parameters**: The speed of convergence must be insensitive to the value of the vector of smoothing parameters $\boldsymbol{\lambda}$: the convergence speed of many methods slows down exponentially as the smoothing parameter goes to 0.

R.4 **Take into account that we need to optimize the upper and lower level problems simultaneously**

R.5 **Fully automated method**: The user should not have to modify parameters for the method to converge.

In the next section, we review methods that could be considered to solve the bilevel optimization.

5.1 Optimization of the lower and upper level problems

As solving the upper level problem requires the solution of the lower level problem, we first discuss the solution of the lower level problem given a fixed vector of smoothing parameters $\boldsymbol{\lambda}$.

We start by noting that for a fixed vector of smoothing parameters, the lower level problem fits in the classical paradigm of optimization for statistical learning: **"Regularized (Empirical) Risk Minimization(RRM or RERM)"**(see Steinwart and Christmann [2011], Takeuchi et al. [2006], Torossian et al. [2020], Lei et al. [2017]). It also fits in the mathematical optimization framework of a **"separable problem"** (see Stella and Patrinos [2016], Stella et al. [2017]).

We also note that the lower level problem is related to other optimization problems, namely **Support Vector Machines (SVMs)** (see Hsu and Lin [2002], Takeuchi and Furuhashi [2004], Hwang and Shim [2005], Takeuchi et al. [2006], Bottou and Lin [2007], Christmann and Steinwart [2008], Chapelle [2007], Menon [2009], Shim et al. [2009], Zhu et al. [2009]), $\boldsymbol{\ell^p}$ **(or** $\boldsymbol{\ell^q}$**) regression** (Kloft et al. [2011], Lyu et al. [2013], Lai et al. [2013], Liang et al. [2013], Qinghui et al. [2016]) and **Generalized Lasso** (Zhu [2017]). We will now discuss specific methods that could possibly be used to solve the lower level problem.

First, for an unpenalized linear quantile regression, Koenker and Bassett Jr [1978] uses linear programming. However, as the lower problem is the sum of a piecewise linear function and a quadratic function, this method cannot be used here.

Then, we could envisage the use of methods for smooth convex functions such as Newton-Raphson. However, as noted by Yu [2009], Yu et al. [2010], Hiriart-Urruty and Lemaréchal [2013], Section VIII, Remark 2.1.3, although

we could think that because the non-differentiability of the function to optimize is of Lebesgue measure 0, the function is almost smooth and, therefore, we can just use standard optimization methods. However, Hiriart-Urruty and Lemaréchal [2013] point out that often, the optimal point or set of the function to optimize is on the non-differentiable set. Yu [2009] gives an example function $f(x, y) = 10|x|+y$ and shows that the BFGS algorithm heavily oscillates around the optimum as the optimum is a non-differentiable line. These oscillations do not appear when using a subgradient descent. This led to the development of many methods to deal with the non-differentiability. Mankau and Schuricht [2019] gives a non-exhaustive list of methods that include bundle methods (Teo et al. [2010], Mäkelä [2002]), proximal point/splitting methods (Combettes and Pesquet [2011]), smoothing methods, gradient sampling (Burke et al. [2020]) and derivative-free methods (see Mankau and Schuricht [2019] Fiege [2017] for further references). We could add to this envelope methods (see Giselsson and Fält [2018]) that are generalizations of proximal and splitting methods.

The methods described in Pietrosanu et al. [2021], Gao [2015] and implemented in package 'cqrReg' Gao and Kong [2015] are 4 methods that can be used for composite quantile regression. These methods may also be appropriate for our problem:

- **(Greedy) primal coordinate descent:** Although primal coordinate descent cannot directly be used for our problem as it is a non-separable problem[1] (i.e. the absolute value applies to a linear combination of the coordinates of the vector of parameters rather than each coordinate separately), greedy coordinate descent can be used to optimize in the primal (see Wu and Lange [2008], Mkhadri et al. [2017], Mkhadri et al. [2015]).
- **Interior Point (IP) /Frisch Newton IP:** Koenker and Park [1996] and Koenker and Ng [2005] use an Interior point method for a linear quantile regression whilst Bosch et al. [1995] formulates a Quantile Smoothing Spline model with quadratic penalty as a quadratic program and solve it using an IP method. One advantage of IP methods is that they can be use both for linear or non-linear programming. Hence, they could also be used for the optimization of a piecewise linear function penalized by a quadratic function.
- **Minorize-Maximization (MM):** is a method where a smooth surrogate that majorizes the piecewise linear function is minimized (Hunter and Lange [2000]). This method could be used for our problem. Note that the EM algorithm used in Zhou et al. [2014] is a type of MM algorithm (see also Lange [2016]).

[1] see Shi et al. [2016], Wright [2015]. Warga [1963], Cevher [2016], Shi et al. [2016] give the following examples of non-separable problems where CD stops on a point that is not the global minimum: $f(x_1, x_2) = |x_1 - x_2| - \min(x_1, x_2)$, $f(x_1, x_2) = x_1^2 + x_2^2 + |x_1 - x_2|$ and $f(x_1, x_2) = \frac{\sqrt{2}}{2}(|x_1 + x_2| + 2|x_1 - x_2|)$ respectively.

- **Alternating Direction Method of Multipliers (ADMM)**: It seems that ADMM methods have grown in popularity recently to estimate quantile regression (see Kong et al. [2015], Shenoy and Gorinevsky [2015], Shenoy et al. [2015], Juban et al. [2016], Ali et al. [2016], Yu and Lin [2017], Yu et al. [2017], Gu et al. [2018], Yu et al. [2019], Gu and Zou [2017]). The ADMM method consists in splitting the optimization in 2 parts and optimize each part alternatively. To make sure we stay in the feasible set, a projection is used after each optimization.

We could add to this (non-exhaustive) list :

- **Dual solved using quadratic programming:** The primal problem without identifiability constraints can first be transformed into a problem with a diagonal penalty using a technique described in the case of SVMs by Li et al. [2016], Maji [2012], Zhang et al. [2012] and in the case of GAMs in Wood [2017], 5.4.2. A null space penalty may also be added to obtain a diagonal penalty that is full rank (Wood [2017], section 5.4.3). Then the dual of the diagonalized penalty problem can be transformed in a constrained quadratic program (as in Koenker and Mizera [2014], Hofmeister [2017], the dual can be solved using quadprog Turlach and Weingessel [2015]).

- **Cutting plane/bundle method:** Teo et al. [2010], Prados [2017].

- **Dual or primal/dual coordinate descent**: Hsieh et al. [2008], Wright [2015], Sangnier et al. [2016], Yang et al. [2015], Yi and Huang [2017].

- **Envelope methods**: as in Stella et al. [2017], Stella and Patrinos [2016], Giselsson and Fält [2018]. Kong et al. [2015] notes that an ADMM problem can be fit into a 'separable problem' framework.

- **Gradient sampling**: Boldi and Chavez-Demoulin [2017], Boldi et al. [2019].

- **Smoothed loss gradient descent with Barzilai-Borwein step**: He et al. [2020]

- **Generalized Newton**: Zheng [2014]

- **Subgradient BFGS**:Yu et al. [2010], Yu [2009].

Note that although we did not mention it in the list of potential methods, BFGS/L-BFGS have been shown to be directly applicable to convex nonsmooth optimization problems in certain settings (see Lewis and Overton [2009], Skajaa [2010], Lewis and Overton [2013], Guo and Lewis [2017], Guo [2018], Asl and Overton [2020], Asl and Overton [2021], Guo and Lewis [2018a], Guo and Lewis [2018b], Xie and Waechter [2017]). However, Yu [2009] cites Lukšan and Vlček [1999], Lewis and Overton [2008], Haarala [2004] who report some failures of BFGS/L-BFGS. Yu [2009] also notes that these algorithms may not be

adapted to typical large scale regularized empirical risk minimization problems. Similarly, Asl and Overton [2020] conclude that for large scale nonsmooth optimization, using L-BFGS on a smoothed version of the problem seems to be more appropriate than using BFGS directly on the nonsmooth problem. The main drawbacks of the methods listed above for our purpose are:

1. **Speed for accurate convergence**: Although the ADMM method gives an approximate solution quickly it is slow to obtain an accurate solution (see Boyd et al. [2011] p.17).

2. **The method is not scalable without parallelization** For example the dual solved using quadratic programming or coordinate descent can be used to estimate the lower level problem but the dual is a problem of size n. This means that we cannot use this method for a large number of observations (for example in the case of an svm, Osuna et al. [1997] notes that QP cannot be used beyond a few thousand points). Dual CD could be made faster by using parallelization but we specifically precluded it so that our package could be used on a standard computer.

3. **The method is not adapted to solve the upper level problem**: Many of the methods described above could be used to solve the lower level problem. However, they could not be used to find the global minimum of the upper level problem. For example, the Gradient Sampling algorithm Boldi and Chavez-Demoulin [2017] converges to a local minimum of a non-convex function. The starting point could therefore significantly change the estimated value.

4. **The method has a convergence speed linked to the vector of smoothing parameters $\boldsymbol{\lambda}$**: This is the case for the vast majority of methods above, for example, the forward backward method (Stella et al. [2017]) or cutting plane/bundle method (Teo et al. [2010]). The following articles give the order of convergence of several different methods: Shalev-Shwartz and Zhang [2013], Palaniappan and Bach [2016], Lacoste-Julien et al. [2013], Chapelle [2007], Menon [2009]. We note that many of the methods described have a convergence in $O(\frac{1}{\lambda})$.[2]

5. **The method may require to manual change some parameters** Xu et al. [2017a], Xu et al. [2017b] note that ADMM may require manual adjustment of parameters to converge.

5.2 Method selected: smooth surrogate, Generalized Iteratively Reweighted Least Squares and Graduated Optimization

In chapter 3, we introduced a shifted version of a smooth surrogate function function that can be found in Zheng [2011] and Wang and Ye [2016] based

[2]Some methods in Menon [2009] do not seem to have convergence in $O(\frac{1}{\lambda})$. However, the SMO and SVM^{light} converge in $O(n^2)$, OCAS does converge in $O(min(\frac{1}{\lambda}, \frac{1}{\lambda^3})$ and SGD methods are slow to converge accurately.

on Chen and Mangasarian [1995] and similar to the function introduced in Amemiya [1982]:

$$\mathcal{L}_{\alpha,\tau}\left(y_i - f_i\right) = \tau\left(y_i - f_i + \gamma\right) + \alpha \log\left(1 + \exp\left(-\frac{y_i - f_i + \gamma}{\alpha}\right)\right) \tag{5.1}$$

where $\gamma = \alpha \log \frac{1-\tau}{\tau}$. Apart from Amemiya [1982], some other early articles discussing the smoothing method consisting in replacing the non-differentiable loss with a smooth surrogate include Clark and Osborne [1986] and Madsen and Nielsen [1993]. Extensions to quantile smoothing splines were discussed in Bosch et al. [1995], O'Connell and Nychka [1995](see also Chen and Wei [2005], Chen [2007], Muggeo et al. [2012]). The reasons to use surrogate loss 5.1 were discussed in section 3.1.

By replacing the loss function by a smooth surrogate, we can solve the lower level problem as long as α is not too small. However, two problems remain. First, how do we choose the level of rounding α and second, can this method be used to solve the upper level problem. For the first question, if we directly take a very small α, although we still have a smooth function, the optimization may be as problematic as if we had to optimize the exact pinball loss. However, if for a small level of rounding α, we had a good starting point, we may still be able to optimize the function. This is the idea proposed in Bissantz et al. [2009]: the Generalized Iteratively Reweighted Least Squares.

For a non-smooth convex function, we start at a high level of rounding $\alpha_{(1)}$ and gradually reduce the level of rounding α, using the minimum of the function at the previous rounding level as a starting point. For $\alpha_{(1)}$, we choose a level of rounding that is sufficiently smooth to optimize relatively easily. We propose to choose $\alpha_{(1)} = \hat{\alpha}$, the level chosen to match the expected loss (equation 3.15). For the first step of the algorithm, we use $\hat{\alpha}$ used for all parts of the algorithm. Then α is reduced (note that for the calculation of the edf for the upper level problem we will use $\hat{\alpha}$ for all levels of the algorithm). Instead of having one optimum, we then have a sequence of α: $\alpha_{(1)} = \hat{\alpha}, \alpha_{(2)}, ..., \alpha_{(Final)} = \alpha_L$ with their associated optimal values: $\boldsymbol{\beta}_{(1)}, ..., \boldsymbol{\beta}_{(Final)}$. We call α_L the final rounding level that is used to approximate the exact pinball loss. The advantage is that even if a rounding level $\alpha_{(k)}$ is small, we are able to find the optimal value $\boldsymbol{\beta}_{(k)}$ starting from the previous optimal $\boldsymbol{\beta}_{(k-1)}$. Note that a similar type of algorithm is also proposed in Hahn et al. [2020] (discussed briefly in section 3.1 and Appendix F). Hahn et al. [2020] replace a lasso penalty by a smooth surrogate and, similarly to Bissantz et al. [2009] use a method (that they call progressive smoothing) consisting in gradually reducing the parameter controlling the degree of approximation of the surrogate loss. Also note that IRLS can be seen as a type of MM algorithm (Mukhoty et al. [2019], Nguyen and McLachlan [2017], Sun et al. [2016]). However, in our case, we could say that our method is closely related to MM algorithms because our surrogate loss is neither a majorizer nor a minorizer of the exact pinball loss (see section 4.1).

Second, as α increases, the QGACV becomes closer to a quasi-convex function. Similarly to the method to solve the lower level problem, we can start at a high level of rounding $\alpha_{(1)} = \hat{\alpha}$ and reduce it gradually, using the previous vector of smoothing parameters as a start. We then obtain a sequence of vectors of smoothing parameters $\boldsymbol{\rho}_{\alpha_{(1)}}, ..., \boldsymbol{\rho}_{\alpha_U}$ associated with a series of values for α: $\alpha_{(1)} = \hat{\alpha}, \alpha_{(2)}, ..., \alpha_U$. α_U is the level where we stop the optimization of the QGACV. We assume that $\alpha_U \geq \alpha_L$. As the QGACV function is non-smooth and non-convex, this type of algorithm can be referred to as graduated optimization/graduated Non-convexity as proposed in Blake and Zisserman [1987] (see also Hazan et al. [2016]). As explained previously, the level of rounding $\hat{\alpha}$ is kept constant to calculate the effective degrees of freedom. The gradient of the QGACV depends on the third derivative of the loss function whereas the gradient of the penalized empirical loss depends on the first derivative of the loss function. For the Upper Level, as we have $\lim_{\alpha \to 0, u \to 0^-} \mathcal{L}'''_{\alpha,\tau}(u) = +\infty$, $\lim_{\alpha \to 0, u \to 0^+} \mathcal{L}'''_{\alpha,\tau}(u) = -\infty$, the algorithm used to optimize the QGACV may become unstable if α becomes too small. Even if ideally we would like to reduce α as much as possible, we will stop the optimization of the Upper Level problem at a relatively high level α_U. We can now re-write the problem as:

Upper Level:

$$\hat{\boldsymbol{\lambda}} = \text{argmin}_{\boldsymbol{\lambda}} \lim_{\alpha \to 0} QGACV_{\hat{\alpha},\alpha}(\boldsymbol{\lambda}) := \lim_{\alpha \to 0} \frac{\sum_{i=1}^{n} \mathcal{L}_{\alpha,\tau}\left(y_i - \mathbf{x}_i^T \tilde{\boldsymbol{\beta}}_\alpha\right)}{n.eff - cTr\left(\mathbf{A}_{\hat{\alpha},\alpha}\right)}$$

Lower Level:

$$\text{s.t.: } \tilde{\boldsymbol{\beta}}_\alpha = \text{argmin}_{\boldsymbol{\beta}} \left(\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{\alpha,\tau}\left(y_i - \mathbf{x}_i^T \boldsymbol{\beta}\right) + \frac{1}{2} \boldsymbol{\beta}^T \left(\sum_k \lambda_k \mathbf{S}_k \right) \boldsymbol{\beta} \right)$$

where: $\mathbf{A}_{\hat{\alpha},\alpha} = \mathbf{X}\left(\mathbf{X}^T \mathbf{W}_{\hat{\alpha},\alpha} \mathbf{X} + \mathbf{S}_{\boldsymbol{\lambda}}\right)^{-1} (\mathbf{X}\mathbf{W}_{\hat{\alpha},\alpha})$; $\mathbf{W}_{\hat{\alpha},\alpha} = diag\left(w_{\hat{\alpha},\alpha,1}, ..., w_{\hat{\alpha},\alpha,n}\right)$ $w_{\hat{\alpha},\alpha,i} = \mathcal{L}''_{\hat{\alpha},\tau}\left(y_i - \mathbf{x}_i^T \tilde{\boldsymbol{\beta}}_\alpha\right)$; $n.eff = 2n\phi$; $c = (1 + 2\phi)$); $\phi = \min(\tau, 1 - \tau)$

Note that for a non-zero α, $\in$ can be replaced by an $=$ as we removed the non-differentiability. Also note that the estimated rounding level $\hat{\alpha}$ used to approximate the expected loss with the surrogate loss appears only in the calculation of the second derivative of the loss function. For the exact pinball loss, the second derivative would be zero almost everywhere apart from a series of hyperplanes (measure 0) where it is not defined.

The first part of the algorithm will then consist in alternating between the Upper Level problem and the Lower Level problem between $\alpha_{start} = \hat{\alpha}$ and $\alpha = \alpha_U$. We make one step for the Upper Level problem then find the optimum of the Lower Level problem with the new value of $\boldsymbol{\lambda}$ and iterate. Once we reached the optimum of the Upper Level problem and of the Lower Level problem given the vector of smoothing parameters, we fix $\boldsymbol{\lambda}$ at the value we obtained for $\alpha = \alpha_U$. We can continue to reduce α for the Lower Level problem only.

This is because to solve the Lower Level problem, we only need a stable second derivative of the loss function. The first derivative of the loss function is bounded between -1 and +1. The second derivative of the loss function can explode close to a non-differential point as $\alpha \to 0$. We can mitigate this using half stepping and making the Hessian positive definite (see section 7.11). On the other hand, the optimization of the QGACV requires derivatives of the surrogate loss up to 3rd or 4th order. 3rd or 4th order derivatives of the surrogate loss explode faster than 2nd order derivatives. Furthermore, given that we are using graduated optimization for find the minimum of the QGACV, we may not find the global minimum of the function. This is because the function is not convex. With graduated optimization, we may find a local minimum. Under these conditions, having an extremely precise estimate of the location of the minimum may not be necessary. Hence, by precaution, we stop the optimization of the QGACV before the optimization of the penalized empirical loss. The methodology used to estimate α_U and α_L is given in Appendix 7.6. To optimize the lower level problem, a Newton-Raphson approach with half stepping is implemented:

$$\boldsymbol{\beta}^{k+1} = \boldsymbol{\beta}^{k} - \delta_{\boldsymbol{\beta}} \left(\frac{\partial^2 l^{Pen}_{\alpha,\alpha}}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} \right)^{-1} \frac{\partial l^{Pen}_{\alpha,\alpha}}{\partial \boldsymbol{\beta}}$$

where δ_β is equal to 1 and is successively divided by 2 if the step does not lead to a reduction in the penalized empirical loss. If the Hessian is not positive definite, we use Algorithm 2, section 7.4 to make it positive finite. We have:

$$\tilde{u}_{\alpha,i} = \left(y_i - \mathbf{x}_i^T \tilde{\boldsymbol{\beta}}_\alpha \right)$$

$$l^{Pen}_{\alpha_1,\alpha_2} = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{\alpha_1,\tau} \left(\tilde{u}_{\alpha_2,i} \right) + \frac{1}{2} \boldsymbol{\beta}^T \left(\sum_{k=1}^{q} e^{\rho_k} \mathbf{S}_k \right) \boldsymbol{\beta}$$

The derivatives needed can be found in Appendix C. Note that to optimize the QGACV we use the vector of log-smoothing parameters $\boldsymbol{\rho} = \log(\boldsymbol{\lambda})$. To optimize the upper level problem, two methods have been implemented. The first method is (hyper) Newton-Raphson with half stepping:

$$\boldsymbol{\rho}^{k+1} = \boldsymbol{\rho}^{k} - \delta_{\boldsymbol{\rho}} \left(\frac{d^2 QGACV_{\hat{\alpha},\alpha}}{d\boldsymbol{\rho} d\boldsymbol{\rho}^T} \right)^{-1} \left(\frac{dQGACV_{\hat{\alpha},\alpha}}{d\boldsymbol{\rho}} \right)$$

where $\boldsymbol{\rho} = (\rho_1, ..., \rho_m)^T$ and $\rho_k = \log(\lambda_k)$ $k = 1, ..., m$ and $\delta_{\boldsymbol{\rho}}$ is equal to 1 and is successively divided by 2 is the step does not lead to a reduction in the QGACV. If the Hessian is not positive definite, we use Algorithm 2 to make it positive finite. The second method is the Generalized Fellner-Schall of Wood and Fasiolo [2017]. First, we have:

$$\frac{dQGACV_{\hat{\alpha},\alpha}(\boldsymbol{\rho})}{d\lambda_k} = \frac{dQGACV_{\hat{\alpha},\alpha}(\boldsymbol{\rho})}{d\rho_k} \frac{d\rho_k}{d\lambda_k} = \frac{1}{\lambda_k} \frac{dQGACV_{\hat{\alpha},\alpha}(\rho)}{d\rho_k} \tag{5.2}$$

Then, in Appendix C, we define a new matrix:

$$\tilde{\mathbf{F}}_{\hat{\alpha},\alpha} = \left(\mathbf{X}^T\tilde{\mathbf{W}}_{\hat{\alpha},\alpha}\mathbf{X} + \mathbf{S}_{\boldsymbol{\lambda}}\right)^{-1}\left(\mathbf{X}^T\tilde{\mathbf{W}}_{\hat{\alpha},\alpha}\mathbf{X}\right) \tag{5.3}$$

in replacement of matrix

$$\mathbf{A}_{\hat{\alpha},\alpha} = \mathbf{X}\left(\mathbf{X}^T\mathbf{W}_{\hat{\alpha},\alpha}\mathbf{X} + \mathbf{S}_{\boldsymbol{\rho}}\right)^{-1}\mathbf{X}^T\mathbf{W}_{\hat{\alpha},\alpha}$$

In the QGACV formula, only the trace of matrix $\mathbf{A}_{\hat{\alpha},\alpha}$ is used and we have the equality $tr(\mathbf{A}_{\hat{\alpha},\alpha}) = tr(\mathbf{F}_{\hat{\alpha},\alpha})$ (see Appendix C). As $\lambda_k > 0$ and following Wood [2017], pp.269-272, p.322, exercise 6, p.445-446, the Fellner Schall update in our case is:

$$\lambda_k^{New} = -\frac{c\left(\sum_{i=1}^n \mathcal{L}_{\alpha,\tau}\left(\tilde{u}_{\alpha,i}\right)\right)\frac{dtr(\mathbf{F}_{\hat{\alpha},\alpha})}{d\rho_k}}{\left(n.eff - ctr\left(\mathbf{F}_{\hat{\alpha},\alpha}\right)\right)}\left(\sum_{i=1}^n \mathcal{L}'_{\alpha,\tau}\left(\tilde{u}_{\alpha,i}\right)\frac{d\tilde{u}_{\alpha,i}}{d\rho_k}\right)^{-1}\lambda_k \tag{5.4}$$

Full details of the optimization algorithms as well as other procedures including initialization procedures are given in chapter 7.

6. NON-PARAMETRIC BOOTSTRAP, CONFIDENCE INTERVALS AND SMOOTHING PARAMETER RELAXATION

Numerous methods have been proposed to build confidence intervals for quantile regression. Some of these methods make assumptions about the data distribution or some properties of the data. In this document, to develop a methodology that is as general as possible, we propose to use non-parametric bootstrap. The advantage is that we can test the coverage of our confidence interval in the most general setting. An under or an over-coverage may not be due to the assumptions made to obtain the confidence interval. In section 6.1, we discuss different methods that can be used to obtain the confidence intervals.

Koenker [1994], Kocherginsky et al. [2005], Tarr [2012] refer to the non-parametric bootstrap-based confidence interval as the xy method (for alternative bootstrap methods, see Tarr [2012], Koenker [1994], Kocherginsky et al. [2005], Zhang et al. [2020b]). The non-parametric bootstrap consists in resampling with replacement the data:

$$\mathcal{X} = (\underline{\boldsymbol{x}}_i, y_i)_{i=1,...,n} \tag{6.1}$$

We then obtain B_0 datasets:

$$\mathcal{X}^{*1}, ..., \mathcal{X}^{*B_0} \tag{6.2}$$

Given a fixed estimated vector of smoothing parameters $\hat{\boldsymbol{\lambda}}$ using the original dataset, we can then estimate B_0 vectors of parameters:

$$\hat{\boldsymbol{\beta}}^{*1}, ..., \hat{\boldsymbol{\beta}}^{*B_0} \tag{6.3}$$

where:

$$\hat{\boldsymbol{\beta}}^{*b} = \left(\hat{\beta}_1^{*b}, ..., \hat{\beta}_p^{*b}\right)^T$$

From these bootstrapped vector of parameters, we can calculate the bootstrapped predicted values:

$$\hat{f}_i^{*b} = \mathbf{x}_i^T \hat{\boldsymbol{\beta}}^{*b} \quad i = 1, ...n, b = 1, ..., B_0 \tag{6.4}$$

6.1 Methods to estimate the confidence interval using the non-parametric bootstrap samples

In this section, we discuss two methods that could be used to estimate the confidence interval: the percentile method and the estimate of a covariance

matrix based on the bootstrap samples that can then be used to estimate the standard error.

Percentile Method

Efron and Hastie [2016] note that if the number of replicates is not large enough, the distribution of the bootstrapped parameters may not yet be close to a normal distribution. It may then be more appropriate to directly calculate the percentiles of the bootstrapped replicates to calculate confidence intervals. Let's call $G_{\hat{\beta}_j^*}$ the empirical CDF of the bootstrap replicates of $\hat{\beta}_j^*$ and $G_{\hat{f}_i^*}$ the empirical CDF of the predicted value $\hat{f}_i^*$. Then the CI are:

$$(G^{-1}_{\hat{\beta}_j^*}[0.025]; G^{-1}_{\beta_j^*}[0.975])$$

and

$$(G^{-1}_{\hat{f}_i^*}[0.025]; G^{-1}_{\hat{f}_i^*}[0.975])$$

Covariance Matrix

The covariance matrix is calculated as (Davino et al. [2013], section 4.3.2, Tarr [2012], Efron and Tibshirani [1994] p.47, Kharrat et al. [2019])[1]:

$$\hat{V}_{boot} = \frac{1}{B_0 - 1} \sum_{b=1}^{B_0} \left(\hat{\boldsymbol{\beta}}^{*b} - \overline{\boldsymbol{\beta}}^*\right) \left(\hat{\boldsymbol{\beta}}^{*b} - \overline{\boldsymbol{\beta}}^*\right)^T \tag{6.5}$$

where:

$$\overline{\boldsymbol{\beta}}^* = \frac{1}{B_0} \sum_{b=1}^{B_0} \hat{\boldsymbol{\beta}}^{*b}$$

We also have:

$$\hat{se}_{\hat{\boldsymbol{\beta}}^*} = \sqrt{\left(\hat{V}_{boot}\right)}$$

Then the standard error of the predicted values is calculated as (see Ruppert et al. [2003] p.135):

$$\hat{se}_{\hat{f}_i^*} = \sqrt{\left(\mathbf{x}_i \hat{V}_{boot} \mathbf{x}_i^T\right)}$$

The full standard error for the full prediction including the intercept is then:

$$\hat{se}_{\hat{f}} = \sqrt{diag\left(\mathbf{X}^T \hat{V}_{boot} \mathbf{X}\right)}$$

To obtain the standard error for a specific smooth (with or without intercept), we restrict the standard error calculation to the column/rows of the

[1]Note that Tarr [2012] calculates the variance matrix by subtracting $\hat{\boldsymbol{\beta}}$ instead of $\overline{\boldsymbol{\beta}}^*$. However, the non-parametric bootstrap samples may not be centred around $\hat{\boldsymbol{\beta}}$.

smooth. If for example Smooth 1 starts at column $S_{1,start}$ and ends at $S_{1,end}$, assuming that the intercept is located at column 1, Smooth 1 standard error is then:

$$\hat{se}_{\hat{f}_{1,i}}^{Intercept} = \left\{ \left(\mathbf{x}_i \left[1, S_{1,start} : S_{1,end}\right]\right) \left(\hat{V}_{boot} \left[(1,1), (S_{1,start} : S_{1,end}, S_{1,start} : S_{1,end})\right]\right) \left(\mathbf{x}_i \left[1, S_{1,start} : S_{1,end}\right]\right)^T \right\}^{\frac{1}{2}}$$

and the standard error for the smooth without intercept variance:

$$\hat{se}_{\hat{f}_{1,i}} = \left\{ \left(\mathbf{x}_i \left[S_{1,start} : S_{1,end}\right]\right) \left(\hat{V}_{boot} \left[S_{1,start} : S_{1,end}, S_{1,start} : S_{1,end}\right]\right) \left(\mathbf{x}_i \left[S_{1,start} : S_{1,end}\right]\right)^T \right\}^{\frac{1}{2}}$$

In this case, the CI for the vector of parameters is:

$$CI^*_{\hat{\boldsymbol{\beta}}} = \hat{\boldsymbol{\beta}} \pm 1.96 \times \hat{se}_{\hat{\boldsymbol{\beta}}^*}$$

The CI for the predicted values:

$$CI^*_{\hat{f}_i} = \hat{f}_i \pm 1.96 \times \hat{se}_{\hat{f}_i^*}$$

6.2 Smoothing parameter relaxation

Preliminary coverage tests for percentile or covariance-based confidence intervals seemed to indicate under-coverage, especially for non-central quantiles. This seemed to be mainly due to two factors. First, using the QGACV to find the vector of smoothing parameters tends to lead to over-smoothed models for non-central quantiles with a relatively low number of observations. Over-smoothing tends to reduce as the number of observations increases. The second factor is the method we use to obtain the non-parametric bootstrap samples. The method used consists in fixing the vector of smoothing parameters to $\hat{\boldsymbol{\rho}}$, resample the dataset with replacement, then refit the model, re-estimating only the vector of parameters. This approach does not take into account the variance of the vector of smoothing parameters.

Let's take an example. As in 6.1, we re-sample with replacement $\mathcal{X} = (\underline{\boldsymbol{x}}_i, y_i)$ to obtain B_0 datasets of size n: $\mathcal{X}^{*1}, ..., \mathcal{X}^{*B_0}$. However, instead of fixing the vector of smoothing parameters $\hat{\boldsymbol{\rho}}$, we refit the model entirely using datasets $\mathcal{X}^{*1}, ..., \mathcal{X}^{*B_0}$. This gives us a series of bootstrapped vector of smoothing parameters and vector of parameters:

$$\left(\left(\hat{\boldsymbol{\rho}}^{*1}, \hat{\boldsymbol{\beta}}^{*1}\left(\hat{\boldsymbol{\rho}}^{*1}\right)\right), ..., \left(\hat{\boldsymbol{\rho}}^{*B_0}, \hat{\boldsymbol{\beta}}^{*B_0}\left(\hat{\boldsymbol{\rho}}^{*B_0}\right)\right)\right)$$

We use the notation $\hat{\boldsymbol{\beta}}^{*b}\left(\hat{\boldsymbol{\rho}}^{*b}\right)$ to indicate that $\hat{\boldsymbol{\beta}}^{*b}$ is calculated using dataset $\mathcal{X}^{*b}$ and smoothing parameter $\hat{\boldsymbol{\rho}}^{*b}$. The top of Figure 6.1 shows a histogram of

$\hat{\boldsymbol{\rho}}^{*b} = \log(\hat{\boldsymbol{\lambda}}^{*b})$ for $B_0 = 200$ with an example where $n = 100$. We note that the estimated $\hat{\boldsymbol{\rho}}^{*b}$ range from -11.96 to $+5$. (Note that $+5$ is the upper limit that we fixed for each log smoothing parameter ρ_j, $j = 1, .., m$ (see Algorithm 5)). The bottom chart shows in red the model fit, in dotted purple the percentile bootstrap CI with the vector of smoothing parameters fixed at $\hat{\boldsymbol{\rho}}$ and the percentile bootstrap CI where the vector of smoothing parameters is re-estimated together with the vector of parameters for each bootstrap sample in orange. We note that the latter is significantly larger than the former. This explains in part why we have CI undercoverage. Note that the method consisting in refitting the model entirely (both the vector of smoothing parameters and the vector of parameters) for each bootstrap sample is only used as an illustration in this section. It is not implemented in the R package nor used anywhere else in this document. Similarly, the notation B_0 elsewhere in this document is used to mean the number of bootstrap samples used to calculate the CI given a fixed vector of smoothing parameters as described in section 6.1.

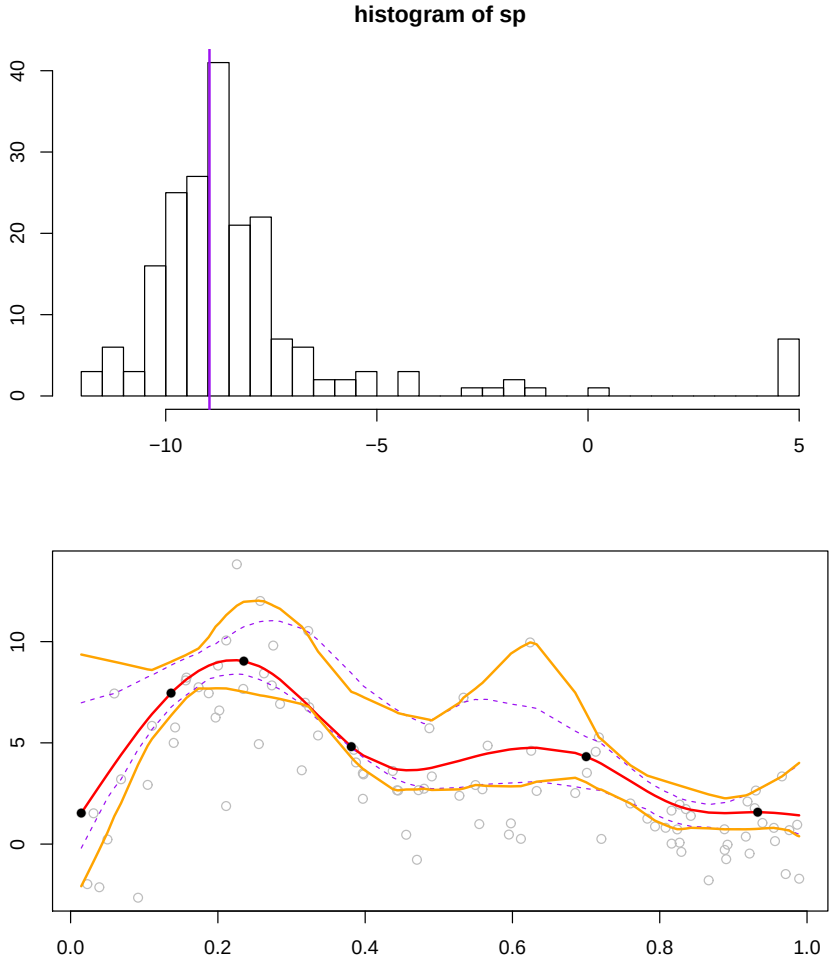


Fig. 6.1: In this example, we take a sample with $n = 100$ from Model 2 (as in section 9.1). We then resample the original dataset with replacement $B_0 = 200$ times: $\mathcal{X}^{*1}, ..., \mathcal{X}^{*B_0}$. We then refit both the vector of smoothing parameters and vector of parameters. We obtain a series of bootstrapped vector of smoothing parameters and vector of parameters $\left(\left(\hat{\boldsymbol{\rho}}^{*1}, \hat{\boldsymbol{\beta}}^{*1}\left(\hat{\boldsymbol{\rho}}^{*1}\right)\right), ..., \left(\hat{\boldsymbol{\rho}}^{*B_0}, \hat{\boldsymbol{\beta}}^{*B_0}\left(\hat{\boldsymbol{\rho}}^{*B_0}\right)\right)\right)$. The top chart shows the histogram of $\hat{\boldsymbol{\rho}}^{*b} = \log(\hat{\boldsymbol{\lambda}}^{*b})$, $b = 1, ..., B_0$. The purple vertical line shows the location of the estimated vector of smoothing parameters $\hat{\boldsymbol{\rho}} = \log(\hat{\boldsymbol{\lambda}})$. The bottom chart shows the curve fit in red, the original non-parametric percentile bootstrap CI in dotted purple (the vector of smoothing parameters is fixed at $\hat{\boldsymbol{\rho}}$), and the non-parametric percentile bootstrap CI based on the estimated curves where both the vector of smoothing parameters and the vector of parameters are re-estimated in orange. We can see that the latter is larger than the former. This could partly explain the under-coverage we observe with the confidence interval calculated using a fixed $\hat{\boldsymbol{\rho}}$.

To reduce the under-coverage, re-estimating the vector of smoothing parameters for each bootstrap sample would be too computationally expensive. To improve the coverage of the CI, we propose to use the 1 s.e. rule as described in Hastie et al. [2009] p.244, Wood [2017], section 2.6.9, pp268-269. As there is uncertainty in the estimation of $\hat{\boldsymbol{\rho}}$, instead of using the $\hat{\boldsymbol{\rho}}$ that leads to the lower cross validation criterion, Hastie et al. [2009] propose to choose $\hat{\boldsymbol{\rho}}$ such that the cross validation is at most 1 s.e. higher than its minimum. Hastie et al. [2009] choose a $\hat{\boldsymbol{\rho}}$ that leads to a more parsimonious model. In the case of Hastie et al. [2009], the cross validation is calculated via a simulation. The standard error for each point of the cross validation path is therefore obtained automatically without any further calculation.

We propose to use a similar method with a different aim. The aim of Hastie et al. [2009] is to obtain a parsimonious model. In our case, we want a less parsimonious model. However, we can still use the idea that there is uncertainty in the choice of the smoothing parameters and purposefully choose a model such that the QGACV is at most 1 s.e. away from its minimum. As we do not have access to the standard error of the QGACV, we propose to estimate it using non-parametric bootstrap. To differentiate the number of bootstrap samples for the CI calculation and for calculation of the se of the QGACV, we will call the number of bootstrap samples B instead of B_0:

$$se(QGACV\left(\hat{\boldsymbol{\rho}}\right)) \approx \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left(QGACV\left(\hat{\boldsymbol{\rho}},\boldsymbol{\beta}^{*b}\right)-\overline{QGACV}\left(\hat{\boldsymbol{\rho}},\boldsymbol{\beta}^{*}\right)\right)^2}$$

$$\overline{QGACV}\left(\hat{\boldsymbol{\rho}},\boldsymbol{\beta}^{*}\right)=\frac{1}{B}\sum_{b=1}^{B}QGACV\left(\hat{\boldsymbol{\rho}},\boldsymbol{\beta}^{*b}\right)$$

Note that by calculating the standard error of the QGACV using bootstrapping, we have somewhat less risk of bias than if we calculated the LOOCV without approximation and used a scheme similar to what is described in Hastie et al. [2009] to estimate the standard error of the cross validation. Bengio and Grandvalet [2004] note that even with i.i.d. data, calculating the standard error of the cross validation based on K-fold cross validation results in a biased estimate.

As noted in Hastie et al. [2009], Tibshirani [2014], Bengio and Grandvalet [2004], this is due to the fact that the calculation of the standard error of K-fold cross validation relies on overlapping blocks that are correlated. [2]. Tibshirani [2014] note that the bias may be larger as we increase the number of folds. This would result in a large bias for the estimation of the LOOCV cross validation if we were to estimate the LOOCV directly. We will then seek to find a vector $\grave{\boldsymbol{\rho}} = \log(\grave{\boldsymbol{\lambda}})$ on the QGACV surface such that

$$QGACV\left(\grave{\boldsymbol{\rho}}\right) > QGACV\left(\hat{\boldsymbol{\rho}}\right) + sd\left(QGACV\left(\hat{\boldsymbol{\rho}}\right)\right)$$

To find this value, we could use a local quadratic approximation of the QGACV surface.

$$QGACV\left(\boldsymbol{\rho}\right) \approx QGACV\left(\hat{\boldsymbol{\rho}}\right) + \left(\boldsymbol{\rho} - \hat{\boldsymbol{\rho}}\right)^T \mathcal{G}_{\hat{\boldsymbol{\rho}}}^{QGACV} + \frac{1}{2}\left(\boldsymbol{\rho} - \hat{\boldsymbol{\rho}}\right)^T \mathcal{H}_{\hat{\boldsymbol{\rho}}}^{QGACV}\left(\boldsymbol{\rho} - \hat{\boldsymbol{\rho}}\right)$$

Where $\mathcal{G}_{\hat{\boldsymbol{\rho}}}^{QGACV} = \frac{dQGACV(\boldsymbol{\rho})}{d\boldsymbol{\rho}}|_{\boldsymbol{\rho}=\hat{\boldsymbol{\rho}}}$ is the total derivative of the QGACV with respect to the vector of smoothing parameters and $\mathcal{H}_{\hat{\boldsymbol{\rho}}}^{QGACV} = \frac{d^2QGACV(\boldsymbol{\rho})}{d\boldsymbol{\rho}d\boldsymbol{\rho}^T}|_{\boldsymbol{\rho}=\hat{\boldsymbol{\rho}}}$ is the total derivative of the Hessian of the QGACV with respect to the vector of smoothing parameters. We are then trying to find $\grave{\mathbf{d}}$ such that:

$$QGACV\left(\hat{\boldsymbol{\rho}}\right) + sd\left(QGACV\left(\hat{\boldsymbol{\rho}}\right)\right) = QGACV\left(\hat{\boldsymbol{\rho}}\right) + \left(\grave{\mathbf{d}}^T \mathcal{G}_{\hat{\boldsymbol{\rho}}}^{QGACV} + \frac{1}{2}\grave{\mathbf{d}}^T \mathcal{H}_{\hat{\boldsymbol{\rho}}}^{QGACV}\grave{\mathbf{d}}\right)$$

The issue is now to know how to determine vector $\grave{\mathbf{d}}$. In a similar situation, Wood [2017], section 6.2.9, pp.268-269 suggests to weight $\grave{\mathbf{d}}$ in each direction by diagonal the precision matrix $\left(\frac{\partial^2 l}{\partial\boldsymbol{\rho}\partial\boldsymbol{\rho}^T}\right)^{-1}$ where l is the penalized log likelihood of a GAM. We can use a similar idea by using a weight for each direction that is proportional to the diagonal of the inverse Hessian of the QGACV with respect to $\boldsymbol{\rho}$:

$$\grave{\mathbf{d}} = -\grave{\zeta}\sqrt{diag\left(\left(\mathcal{H}_{\hat{\boldsymbol{\rho}}}^{QGACV}\right)^{-1}\right)} = -\grave{\zeta}\boldsymbol{v}$$

[2]The K-fold cross validation standard error is calculated as (see Tibshirani [2014], Bengio and Grandvalet [2004]):

$$se\left(K-fold-cv\right) = \sqrt{\frac{1}{K\left(K-1\right)}\sum_{k=1}^{k}\left(cv_{\mathcal{S}_k}^{[-\mathcal{S}_k]} - \overline{cv}\right)^2}$$

$$cv_{\mathcal{S}_k}^{[-\mathcal{S}_k]} = \frac{1}{n_k}\sum_{(i\in\mathcal{S}_k)}\mathcal{L}\left(y_i - \hat{f}_i^{[-\mathcal{S}_k]}\right)$$

$$\overline{cv} = \frac{1}{K}\sum_{k=1}^{k} cv_{\mathcal{S}_k}^{[-\mathcal{S}_k]}$$

$$\hat{f}_i^{[-\mathcal{S}_k]} = \mathbf{x}_i^T\hat{\boldsymbol{\beta}}^{[-\mathcal{S}_k]}$$

the dataset is split in K non-overlapping blocks $\mathcal{S}_1, ..., \mathcal{S}_K$ of sizes $n_1, ..., n_K$. $\hat{\boldsymbol{\beta}}^{[-\mathcal{S}_k]}$ is the estimate of the vector of parameters with the original dataset with block $\mathcal{S}_k$ removed. LOOCV is a K-fold cross validation scheme where the size of each block is 1 and the number of blocks is n.

where ζ is a parameter to determine. ζ can then be determined by solving the following equation:

$$\begin{aligned}
&\frac{\grave{\zeta}^2}{2}\boldsymbol{v}^T\mathcal{H}_{\hat{\boldsymbol{\rho}}}^{QGACV}\boldsymbol{v} - \grave{\zeta}\boldsymbol{v}^T\mathcal{G}_{\hat{\boldsymbol{\rho}}}^{QGACV} - sd\left(QGACV\left(\hat{\boldsymbol{\rho}}\right)\right) = 0 \\
&\qquad\Rightarrow \grave{\zeta} = \frac{\boldsymbol{v}^T\mathcal{G}_{\hat{\boldsymbol{\rho}}}^{QGACV}}{\boldsymbol{v}^T\mathcal{H}_{\hat{\boldsymbol{\rho}}}^{QGACV}\boldsymbol{v}} \ldots \\
&\qquad\qquad \ldots \pm \frac{\sqrt{\left(\boldsymbol{v}^T\mathcal{G}_{\hat{\boldsymbol{\rho}}}^{QGACV}\right)^2 + 2\left(\boldsymbol{v}^T\mathcal{H}_{\hat{\boldsymbol{\rho}}}^{QGACV}\boldsymbol{v}\right) sd\left(QGACV\left(\hat{\boldsymbol{\rho}}\right)\right)}}{\boldsymbol{v}^T\mathcal{H}_{\hat{\boldsymbol{\rho}}}^{QGACV}\boldsymbol{v}}
\end{aligned} \tag{6.6}$$

as we want $\grave{\zeta}$ positive, we take the positive sign.

Choice of rounding level

Let's now take a look at the quadratic approximation of the QGACV in one dimension. As can be seen on Figures 6.2, 6.3, the quadratic approximation can change significantly as the rounding α changes. On Figure 6.3, we can see that with the first and second rounding levels ($\alpha = 2$ and $\alpha = 1$), the stationary points reached are in a flat zone. The minimum of the QGACV has been smoothed out completely by the rounding. The quadratic approximation for these rounding levels would lead to a step that is too large. For the third stopping point (for $\alpha = 0.5$), the curvature is negative. This curvature could not be used either. We could then use the curvature at $\alpha = 0.25$. Then if we decrease α further ($\alpha = 0.12$ and $\alpha = 0.062$), we are still at a locally convex point but the curvature does not seem appropriate to approximate the overall shape of the function. Last, we can see that for $\alpha = 0.031$ and lower, the stationary point reached is on a concave part of the surface. The quadratic approximation would then not be usable. Taking all this into account, we propose the following. When running the graduated optimization algorithm, at each stationary point reached for each level α, we record the Gradient and Hessian of the QGACV. For each level α, we then calculate the number of elements of the diagonal of the Hessian that are higher than a threshold level (for example $\frac{\mathcal{H}_{\alpha_k,ii}}{QGACV(\alpha_k)} > 0.1\%$). For the rounding level, we choose the level α_k that is the highest possible where the maximum number of directions verify the threshold condition.

The algorithm is described in more details in section 7.12. As the QGACV function may be quite far from a locally quadratic function, even at the chosen level of rounding α, we propose to iterate the algorithm, recalculating the

Hessian and Gradient of the QGACV at each new point.

$$\zeta^{(k+1)} = \frac{\left(\boldsymbol{v}^{(k)}\right)^T \mathcal{G}^{QGACV}_{\boldsymbol{\rho}^{(k)}}}{\left(\boldsymbol{v}^{(k)}\right)^T \mathcal{H}^{QGACV}_{\boldsymbol{\rho}^{(k)}} \left(\boldsymbol{v}^{(k)}\right)} ... \\ ... + \frac{\sqrt{\left(\left(\boldsymbol{v}^{(k)}\right)^T \mathcal{G}^{QGACV}_{\boldsymbol{\rho}^{(k)}}\right)^2 + 2\left(\left(\boldsymbol{v}^{(k)}\right)^T \mathcal{H}^{QGACV}_{\boldsymbol{\rho}^{(k)}} \left(\boldsymbol{v}^{(k)}\right)\right) sd\left(QGACV\left(\boldsymbol{\rho}^{(k)}\right)\right)}}{\left(\boldsymbol{v}^{(k)}\right)^T \mathcal{H}^{QGACV}_{\boldsymbol{\rho}^{(k)}} \left(\boldsymbol{v}^{(k)}\right)}$$

In the algorithm, we restrict the Hessian and gradient only to the directions that are sufficiently curved (see section 7.12). On Figure 6.4, we show the example of Figure 6.1 where we add the percentile bootstrap confidence interval calculated using the relaxed smoothing parameter. We can see that in this example, the CI calculated using the covariance matrix calculated using the relaxed smoothing parameter improves the confidence interval that is now closer to the confidence interval calculated using the non-parametric bootstrap where both the vector of smoothing parameters and the vector of parameters are re-estimated.

Overall, the method described in section 7.12 consists in fitting the quantile additive model to obtain $\hat{\boldsymbol{\rho}}$, relax the smoothing parameter to obtain $\grave{\boldsymbol{\rho}}$, estimate the covariance matrix using non-parametric bootstrap based on $\grave{\boldsymbol{\rho}}$. Then the CI is calculated by taking the original fit $\hat{\boldsymbol{f}}$ based on $\hat{\boldsymbol{\rho}}$ and calculate the confidence interval around this based on the covariance matrix calculated using $\grave{\boldsymbol{\rho}}$.

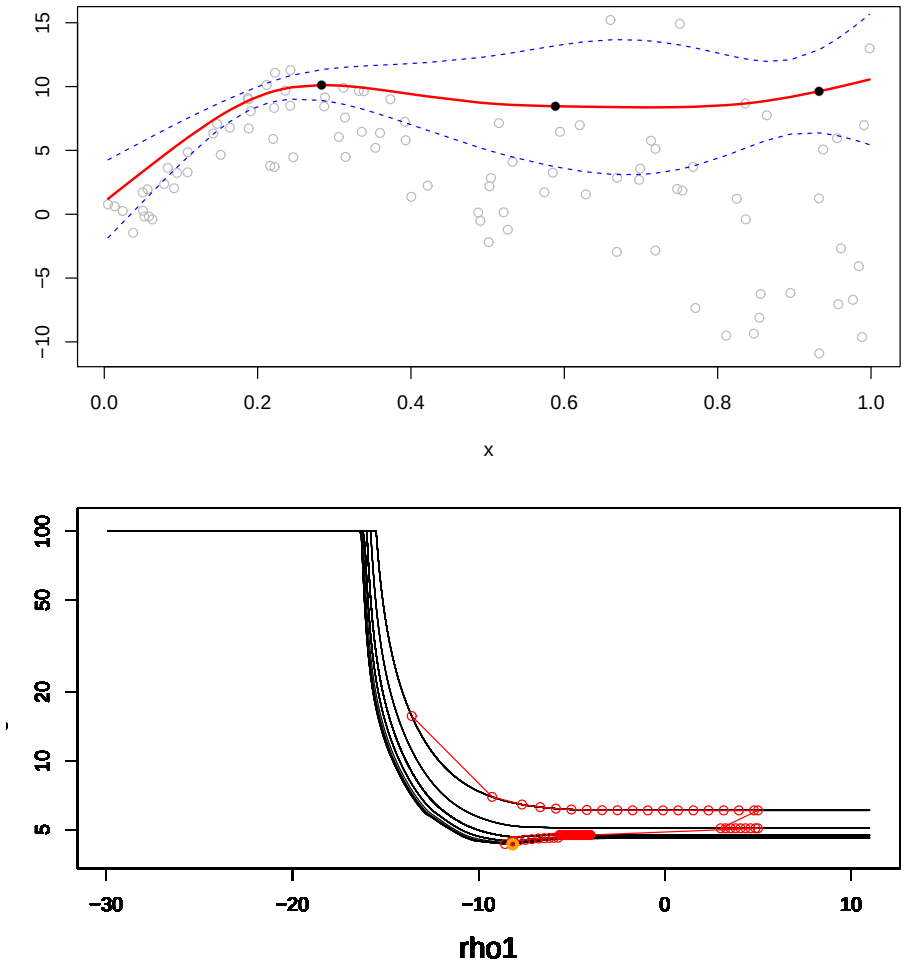


Fig. 6.2: The top picture shows a fitted quantile smoothing spline. The bottom chart shows the optimization path of the QGACV. We note that for the largest smoothing level ($\alpha = 1.58$), the minimum of the QGACV has been smoothed out completely and the minimum of the QGACV is at the maximum possible value of ρ (=5). The details of each level of the optimization together with a quadratic approximation at the optimum is shown on Figure 6.3. The QGACV values can be negative if $n.eff < c$. On this chart, this happens for values of ρ approximately below -15. These negative values are replaced on the chart by the maximum positive value calculated (here around 100).

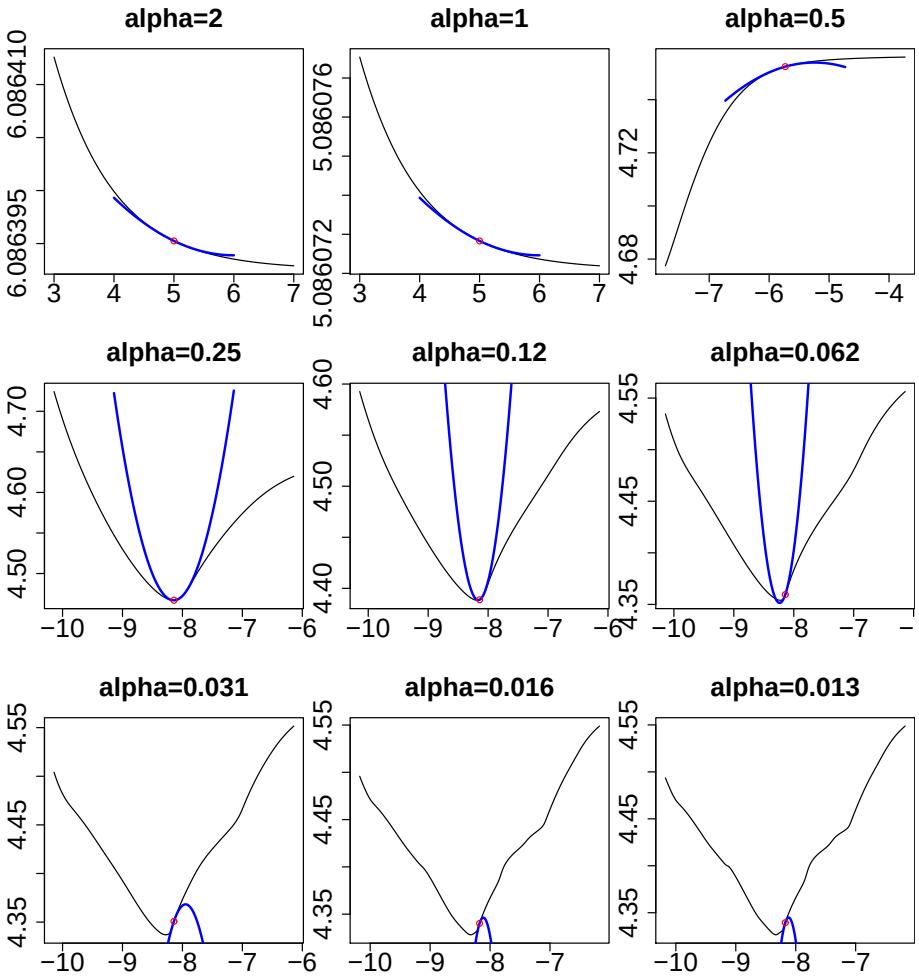


Fig. 6.3: On this chart, we can see the details of the optimization at each level of rounding α from the graduated optimization shown on the bottom chart of Figure 6.2. The blue curves represent a quadratic approximation $QGACV\left(\hat{\boldsymbol{\rho}}\right) + \left(\mathbf{d}^T \mathcal{G}_{\hat{\boldsymbol{\rho}}}^{QGACV} + \frac{1}{2}\mathbf{d}^T \mathcal{H}_{\hat{\boldsymbol{\rho}}}^{QGACV}\mathbf{d}\right)$ around the stopping point of the function. We note that for the largest rounding and second largest ($\alpha = 2$ and $\alpha = 1$), the (global) minimum of the QGACV function has been smoothed out. For the third rounding level, the curvature around the stopping point of the optimization is negative. For these three levels, the quadratic approximations are then a very poor approximation of the overall shape of the function at the optimum at a lower rounding level. As α is reduced, at some point ($\alpha = 0.12$ and below) the local quadratic approximation does not approximate correctly the overall shape of the function. The steps made using this approximation would be too small. For even smaller values of α, the optimization stopped on a concave part of the function, the quadratic approximation cannot be used. Using the method described in this section, the rounding level chosen for the quadratic approximation would then be $\alpha = 0.25$, level where the Hessian is sufficiently large and where α is the highest possible. Note that it could be argued that even at $\alpha = 0.25$, the Hessian does not approximate correctly the overall shape of the function. However, as discussed, we will run an iterative algorithm and also we could improve this by reducing the size of the steps between one level of the graduated optimization and the next (for example by increasing the multiplier of parameter α between each level of the graduated optimization that we call MultAlpha (see section 7.9) from 0.5 to 0.9. Using this, we would chose an optimal α at $\alpha = 0.46$).

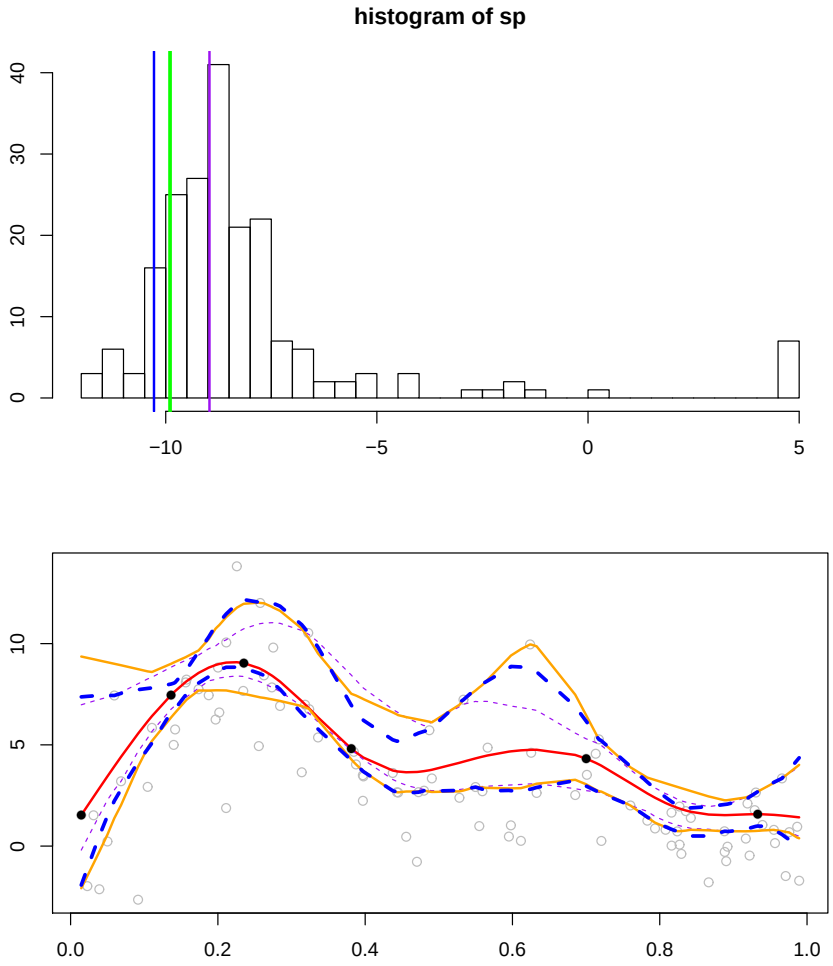


Fig. 6.4: This figure uses the same setting as Figure 6.1. The difference is that here, we add the percentile bootstrap confidence interval based on the relaxed smoothing parameter over the graph of Figure 6.1. On the top chart, the vertical blue line shows the relaxed smoothing parameter to compare to the vertical purple line that is the original non-relaxed smoothing parameter. With a Gaussian distribution, minus 1 standard deviation corresponds approximately to percentile 0.159. Because the distribution of the smoothing parameter is not Gaussian, instead of subtracting one standard deviation away from $\hat{\rho}$, we calculate percentile 0.159 of the distribution of the smoothing parameter. This is shown as a vertical green line on the chart as a reference point. We see that even if the relaxed sp was estimated by using the standard error of the QGACV, in this example, the relaxed smoothing parameter is not far from the location of the 0.159 quantile of the distribution of the histogram, that would correspond to 1 standard deviation away from the mean if the bootstrap samples of the smoothing parameter was normally distributed. Of course, the relaxed sp may not always be so close to this quantile of the distribution of the sp but at least using a relaxed sp may improve the coverage. On the bottom chart, as previously, the red line is the original fit with the non-relaxed smoothing parameter, the dotted purple lines show the CI with the non-relaxed smoothing parameter. The orange lines show the confidence interval with non-parametric bootstrap where both the smoothing parameter and the vector of parameters are re-estimated for each bootstrap sample. The dotted blue line shows the percentile bootstrap CI using the relaxed smoothing parameter.

7. GENERAL METHODOLOGY

In the previous chapters, we described the most important parts of the method: the method used to obtain the rounding level $\hat{\alpha}$, the graduated optimization used to optimize the QGACV, the optimization of the penalized empirical loss to obtain the smoothing parameters, the non-parametric bootstrap to relax the smoothing parameters and obtain the confidence interval. Although these are core to our method, some peripheral algorithms are necessary for the methodology to be implemented and were left out. In this chapter, we fill this gap by describing in details all the algorithms necessary to implement our methodology and how they fit together. To facilitate the description, we start with a flowchart of the complete methodology. It can be described as follows:

First, we fit a mean GAM as reference (section 7.1). This gives us a starting point for smoothing parameters and a fitted model to the mean of the data. Then as the Generalized IRLS consists in rounding the loss function and optimizing at successively lower rounding levels, we need a starting value for $\alpha = \alpha_{start}$ and an stopping value α_{end}. We determine a preliminary value for these using the quantiles of the residuals of the mean GAM fit. These starting and ending values will then be used in a second step to obtain $\hat{\alpha}$ and stopping values that are adapted to the correct quantile. This is described in section 7.2.

We can then use the smoothing parameters of the mean GAM to obtain pilot values for the smoothing parameters (see section 7.3). Thanks to the first approximate values of α_{start} and α_{end} and our pilot values for the vector of smoothing parameters, we can estimate a pilot fit for the QAM model. This pilot fit will then be used to obtain $\hat{\alpha}$ (section 7.5) and more precise values for the stopping values of α for the graduated optimization (α_U) and the Generalized IRLS (α_L) (section 7.6).

The next step that has not been described previously consists in finding a point where to start the graduated optimization algorithm. Given that the penalized loss is a convex function, the starting point for the vector of parameters is not crucial (although a good starting point can make the algorithm converge in less steps). However, for the graduated optimization of the QGACV, as we have a non-smooth and non-convex function, the starting point is more important. Of course, if the level of rounding $\hat{\alpha}$ is sufficiently high, we may start our optimization with a quasi-convex function. However, an incorrect starting point could lead to missing a global optimum of the QGACV. The algorithm

we use to determine our starting point is based on the extremes values for the effective degrees of freedom. When smoothing parameters are large, the edf is at its lowest and when the smoothing parameters are small, the edf is at its maximum. For each smoothing parameter, we choose a starting point for the optimization such that the edf is at mid-point between its smallest and largest possible values. This initialization procedure is described in section 7.7.

Once we have our initial $\boldsymbol{\rho}_{init}$, we can start the graduated optimization (section 7.9). Optimizing the QGACV for a fixed α requires to alternate between a step for the Upper Level (along $\boldsymbol{\rho}$) and an optimization at the Lower Level problem (described in section 7.10). Parameter α is reduced from $\hat{\alpha}$ down to α_U (the stopping criterion is described in section 7.9). Note that as α is increased, the minimum of the QGACV could be smoothed out completely. At the beginning of each level of the graduated optimization, we use a procedure where we reduce each smoothing parameter in each direction to find a "non-flat" zone of the QGACV if we are in a flat zone. A "non-flat" zone for a given direction is characterized by either a non-zero Hessian or non-zero gradient (section 7.8). Once we reached the optimum at rounding level α_U, we fix the smoothing parameters $\hat{\boldsymbol{\rho}}$ and continue the optimization of the penalized empirical loss to obtain a precise model estimate (7.11).

Last, we run the smoothing parameter relaxation procedure (7.12) followed by the estimation of the non-parametric bootstrap fitted curves (7.13). At the end of each section we will also indicate the computational cost of each part of the algorithm. The serialization of these algorithms will then give us the overall computational cost. We assume that:

- n is the number of observations
- m is the number of smooths, also the length of vector $\boldsymbol{\lambda}$.
- p is the length of vector $\boldsymbol{\beta}$. Model matrix $\mathbf{X}$ is then $n \times p$.
- B_0 is the number of non-parametric bootstrap samples for calculating the confidence interval. (see chapter 6)
- B is the number of non-parametric bootstrap samples for smoothing parameter relaxation (section 6.2).

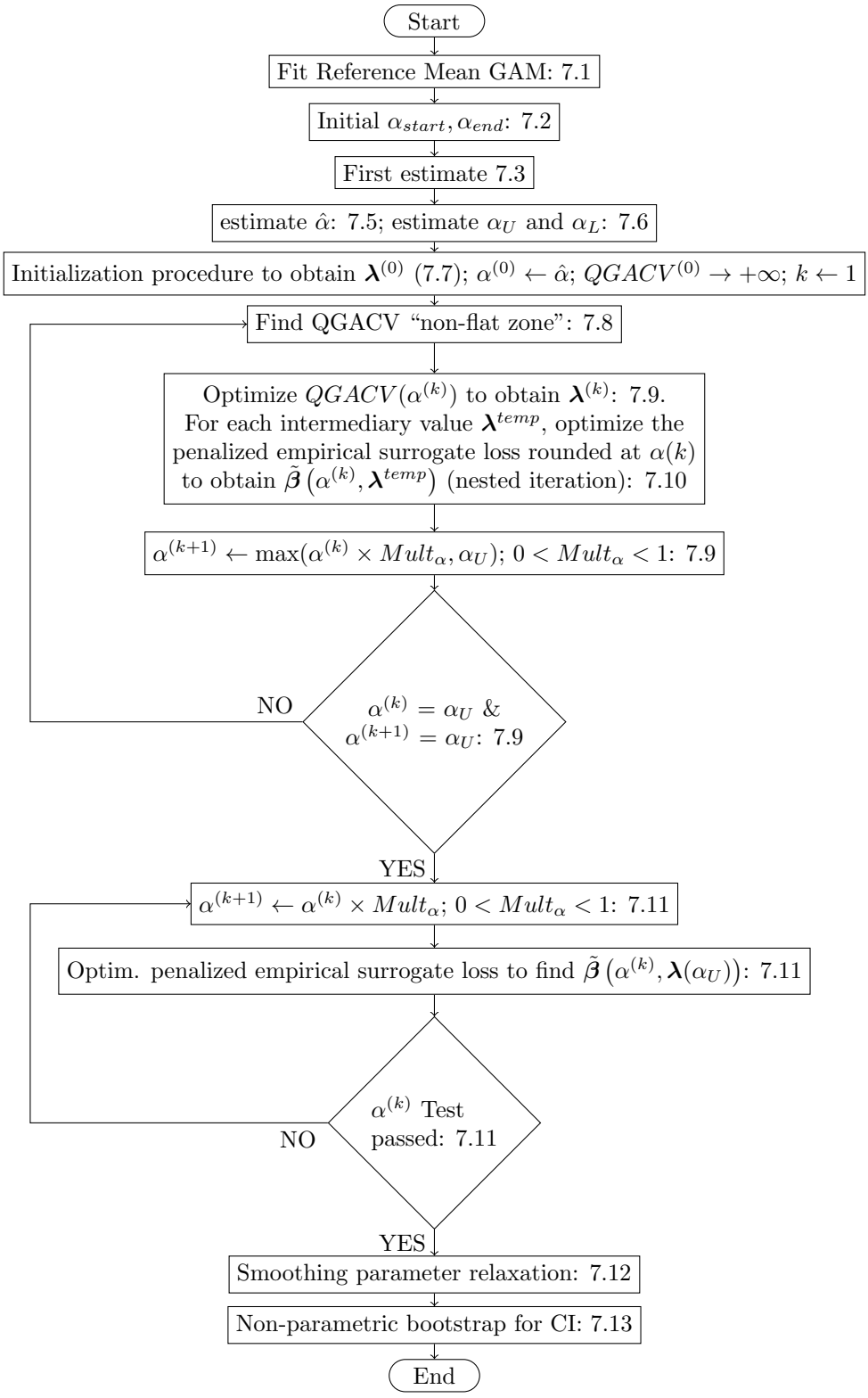
The computational cost for the whole algorithm is

$$\mathcal{O}\left(\left(\frac{m^2}{2} + (B + B_0 + m)\, n\right) p^2 + (B + B_0)\, p^3 + n^{\frac{4}{3}}\right)$$

.

This can be decomposed in $\mathcal{O}\left(\left(\frac{m(m+1)}{2} + n\right) p^2 + p^3\right)$ that is the computational cost of the QGACV Hessian, $\mathcal{O}\left((B + B_0)\left(np^2 + p^3\right)\right)$ that is the computational cost of the non-parametric bootstrap samples for the smoothing parameter relaxation and the calculation of the confidence interval and $\mathcal{O}\left(n^{\frac{4}{3}}\right)$ that is the computational cost of sorting algorithms.

Flowchart of the algorithms



The previous chart was created using the 'flowchart' latex package Robson [2015] and package 'TikZ'.

7.1 Fit Reference mean GAM

To have a first reference point for our fit, we start by estimating a mean GAM model with function $mgcv :: gam$. Function $mgcv :: gam$ interprets the formula and gives values of y, $\mathbf{X}$, $\mathbf{S}$, $\boldsymbol{\beta}_{GAM}$, $\boldsymbol{\lambda}_{GAM}$ that will then be used in the next steps. If the mean GAM fit fails, we add a small perturbation to the data and try to fit again. Wood [2017], p.282 and Wood and Fasiolo [2017] gives the lead order cost of $\mathcal{O}(np^2)$ per smoothing parameter, mainly due to the QR decomposition, for GCV, REML and Generalized Fellner Schall. The order of computation at this stage of the algorithm is then $\mathcal{O}(mnp^2)$ (Wood [2017], p.286).

7.2 Initial Values α_{start} and α_{end}

The estimation of a pilot fit for a quantile additive model requires the determination of a starting value α_{start} and stopping value α_{end} for rounding parameter α to estimate the vector of parameters using Generalized IRLS (Section 7.11). We propose to estimate α_{start} and α_{end} based on the quantiles of the residual of the initial GAM fit (section 7.1). From the mean GAM fit, we calculate a vector of residuals:

$$\mathbf{u}^{GAM} = (\mathbf{y} - \hat{\boldsymbol{f}}^{GAM})$$

Quantiles of this vector of errors can be used to obtain two extreme points, one on the left hand side, the other on the right hand side of the model fit (for example {2.5%, 97.5%}). α_{start} is chosen sufficiently large to obtain a detectable distance (for example 5%) between the surrogate loss and the exact pinball loss at the two extreme quantile points. Note that because we start from a GAM fit, we compare the exact pinball loss to the surrogate loss at central quantile $\tau = 0.5$.

Starting from α_{start}, α_{end} is then obtained by reducing α such that for two central quantiles (for example {49.995%, 50.005%}) we obtain a surrogate loss no further away than a certain percentage (for example 0.1%) from the exact pinball loss at these quantile points. As the 2 quantiles may be on the same side of 0, instead of directly taking the quantiles, we pick the positive and negative errors such that their quantile is the closest to the target quantiles. For example, let's imagine that we have 100 datapoints and that the errors quantiles are as follows:

Error	-0.16	-0.076	-0.065	-0.03	-0.00084	0.002	0.011	0.043
Quantile	0.47	0.48	0.49	0.5	0.51	0.52	0.53	0.54

If we were to take the points at quantiles $\leq 49.995\%$ (49%) and $\geq 50.005\%$ (51%), we would have two points that are on the same side of the origin (-0.065 and -0.00084). In this case, the two points are relatively far apart but there could be other cases where they are very close. This would lead to a very low estimated α_{end}. To reduce this problem, instead of using -0.00084 as the second point, we will use the point on the positive side that is the closest to quantile 50.005%. In this case, this is quantile 0.52 (0.002). This procedure is described in Algorithm 1 and a visual explanation is given on Figure 7.1.

In terms of computational costs, the calculation of the error vector is $\mathcal{O}(n)$ (as the predicted value is already calculated by function mgcv::gam). The sort function currently uses the default method in R that is using a primitive "xtfrm". The order of computation is not indicated for this default method. In Appendix H, we take an example to compare the default method to other methods available in function sort where the order of computation is indicated. For this example, the default method seems to have an order of computation $\mathcal{O}(n^{\frac{4}{3}})$. We could improve this by using quick sort that has an order of computation of $\mathcal{O}(n \times \log(n))$. However, Algorithm 1 uses the stats::ecdf function that also starts with a sort algorithm that uses the default sort algorithm. We then use $\mathcal{O}(n^{\frac{4}{3}})$ as an estimate of the order of computation for the sort algorithms. All in all at this stage of the Algorithm, the order of computation is then $\mathcal{O}(mnp^2 + n^{\frac{4}{3}})$

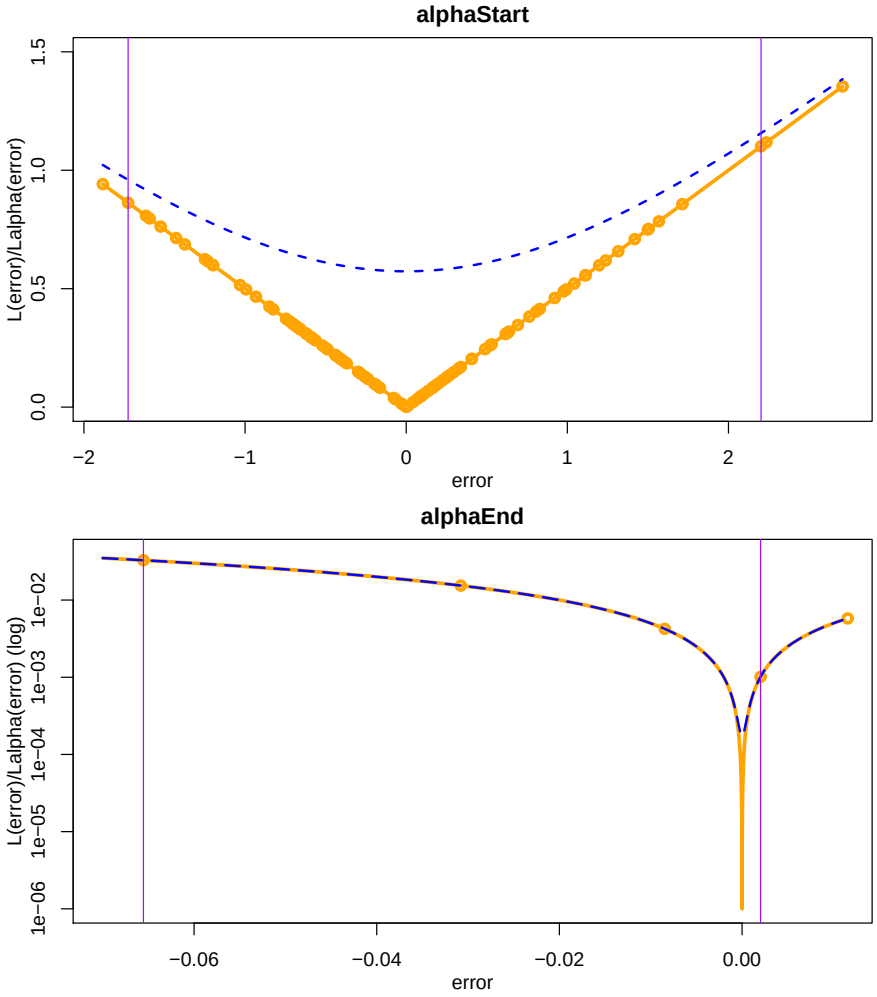


Fig. 7.1: On this Figure, we can see how α_{Start} and α_{end} are selected. We first fit a mean GAM using function mgcv:gam. The error is then calculated as: $u_i = (y_i - \hat{f}_i^{GAM}), i = 1, ..., n$. We then use the quantiles of the error as base points to calculate the distance between the surrogate loss and the exact pinball loss. On the top chart, we see how α_{Start} is selected. In orange, we can see the exact pinball loss. Each orange dot represents the value of the error for our example that has $n = 100$ datapoints. The vertical purple lines show the selected values using quantiles of the data. We calculate the distance at these vertical lines between the surrogate loss in dotted blue and the exact pinball loss. We choose α_{Start} such that on both sides, the surrogate loss rounded at α_{Start} is greater than or equal to 1.05 times the value of the exact pinball loss. On the bottom chart, we show how α_{end} is selected. We first select quantile points where we are going to compare the surrogate loss to the exact pinball loss. As both points are on one side of the origin, we force the values selected to be on both sides of 0. The points selected are -0.0655 and +0.0020. We then try to find the value α_{end} such that the surrogate loss rounded at α_{end} is at most 0.1% above the exact pinball loss. The y axis is on a log scale. We floor the values at 10^{-6} to be able to use the log scale.

Algorithm 1 Obtain α_{Start}, α_{end}
This algorithm is mostly the same as Algorithm 4.
As both algorithms are similar, they are implemented as a single function in package qgacv.

Calculate the error vector $\mathbf{u^{GAM}} = \left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}^{\mathbf{GAM}}\right)$.
$\mathbf{u^{GAM}} \leftarrow \mathbf{sort}(\mathbf{u^{GAM}})$
Set $Interval_{\alpha_{Start}}$ and $Interval_{\alpha_{end}}$ ▷ *e.g.* 0.95 *and* 0.0001
Set $Dist_{Start}$ *and* $Dist_{end}$ ▷ *e.g. 5% and 0.1%*
Set m_α ▷ *e.g.* $m_\alpha = 1.01$
calculate the empirical cdf of $\mathbf{u^{GAM}}$*:* $ecdf_{\mathbf{u^{GAM}}}$
##
The two tests are not necessary for this algorithm
but we leave them here as we use the same code
as for algorithm 4.
if $(\forall j = 1, .., n, u_j < 0)$ ***then***
    $\mathbf{u}^{Greater} = \{u_n^{GAM}\}$; $\mathbf{u}^{Lower} = \{u_1^{GAM}, ..., u_{n-1}^{GAM}\}$
else if $(\forall j = 1, .., n, u_j > 0)$ ***then***
    $\mathbf{u}^{Lower} = \{u_1^{GAM}\}$; $\mathbf{u}^{Greater} = \{u_2^{GAM}, ..., u_n^{GAM}\}$
##
else
    $\mathbf{u}^{Greater} = \{u_j^{GAM}, j = 1, ..., n,\ s.t. u_j^{GAM} \geq 0\}$
    $\mathbf{u}^{Lower} = \{u_j^{GAM}, j = 1, ..., n,\ s.t. u_j^{GAM} < 0\}$
end if
$L_0 \leftarrow length(\mathbf{u}^{Lower})$
$G_0 \leftarrow length(\mathbf{u}^{Greater})$
Calculate
$DistToCenter_{\alpha_{Start}} = \frac{Interval_{\alpha_{Start}}}{2}$ ▷ *e.g.* $= \frac{0.95}{2} = 0.475$
$UpperSide \leftarrow \min(\tau + DistToCenter_{\alpha_{Start}}, ecdf_{\mathbf{u}^{GAM}}(u_{G_0}^{Greater}))$
▷ *e.g.* $\min(0.5 + 0.475, ecdf_{\mathbf{u}^{GAM}}(2.70)) = \min(0.5 + 0.475, 1) = 0.975$
$LowerSide \leftarrow \max(\tau - DistToCenter_{\alpha_{Start}}, ecdf_{\mathbf{u}^{GAM}}(u_1^{Lower}))$
▷ *e.g.* $\max(0.5 - 0.475, ecdf_{\mathbf{u}^{GAM}}(-1.88)) = \max(0.5 - 0.475, 0.01) = 0.025$
$UpperLimit \leftarrow \underset{u_j^{Greater}, j=1,...,G_0}{\operatorname{argmin}} \left\{ecdf_{\mathbf{u}^{GAM}}\left(u_j^{Greater}\right) \geq UpperSide\right\}$
▷ *e.g* 2.20
$LowerLimit \leftarrow \underset{u_j^{Lower}, j=1,...,L_0}{\operatorname{argmax}} \left\{ecdf_{\mathbf{u}^{GAM}}\left(u_j^{Lower}\right) \leq LowerSide\right\}$
▷ *e.g* -1.72
$z_1 \leftarrow (\mathcal{L}_\tau(LowerLimit), \mathcal{L}_\tau(UpperLimit))^T$
$\alpha \leftarrow \alpha_{AbsoluteMin} \leftarrow 10^{-10}$
$\gamma \leftarrow \alpha \log\left(\frac{1-\tau}{\tau}\right)$ ▷ *in this Alg.,* $\gamma = 0$ *(not necessarily for Alg. 4)*
$z_2 \leftarrow (\mathcal{L}_{\alpha,\tau}(LowerLimit - \gamma), \mathcal{L}_{\alpha,\tau}(UpperLimit - \gamma))^T$
$RatioVec \leftarrow \frac{z_2}{z_1}$
while $RatioVec[1] \leq Dist_{Start}$ *Or* $RatioVec[2] \leq Dist_{Start}$ ***do***
    $\alpha \leftarrow \alpha \times m_\alpha$
    $\gamma \leftarrow \alpha \log\left(\frac{1-\tau}{\tau}\right)$ ▷ *in this Alg.,* $\gamma = 0$ *(not necessarily for Alg. 4)*
    $z_2 \leftarrow (\mathcal{L}_{\alpha,\tau}(LowerLimit - \gamma), \mathcal{L}_{\alpha,\tau}(UpperLimit - \gamma))^T$
    $RatioVec \leftarrow \frac{z_2}{z_1}$
end while
$\alpha_{Start} \leftarrow \alpha$

34: Calculate
$DistToCenter_{\alpha_{end}} = \frac{Interval_{\alpha_{end}}}{2}$ ▷ *e.g.* $= \frac{0.0001}{2} = 0.00005$
35: $UpperSide \leftarrow \min(\tau + DistToCenter_{\alpha_{end}}, ecdf_{\mathbf{u}^{GAM}}(u_{G_0}^{Greater}))$
▷ *e.g.* $\min(0.5 + 0.00005, ecdf_{\mathbf{u}^{GAM}}(2.7)) = 0.50005$
36: $LowerSide \leftarrow \max(\tau - DistToCenter_{\alpha_{Start}}, ecdf_{\mathbf{u}^{GAM}}(u_1^{Lower}))$
▷ *e.g.* $\max(0.5 - 0.00005, ecdf_{\mathbf{u}^{GAM}}(-1.88)) = 0.49995$
37: $UpperLimit \leftarrow \underset{u_j^{Greater}, j=1,\ldots,G_0}{\operatorname{argmin}} \left\{ecdf_{\mathbf{u}^{GAM}}\left(u_j^{Greater}\right) \geq UpperSide\right\}$
▷ *e.g. 0.0020*
38: $LowerLimit \leftarrow \underset{u_j^{Lower}, j=1,\ldots,L_0}{\operatorname{argmax}} \left\{ecdf_{\mathbf{u}^{GAM}}\left(u_j^{Lower}\right) \leq LowerSide\right\}$
▷ *e.g. -0.0655*
39: $z_1 \leftarrow (\mathcal{L}_\tau(LowerLimit), \mathcal{L}_\tau(UpperLimit))^T$
40: $\gamma \leftarrow \alpha \log\left(\frac{1-\tau}{\tau}\right)$ ▷ *in this Alg.,* $\gamma = 0$ *(not necessarily for Alg. 4)*
41: $z_2 \leftarrow (\mathcal{L}_{\alpha,\tau}(LowerLimit - \gamma), \mathcal{L}_{\alpha,\tau}(UpperLimit - \gamma))^T$
42: $RatioVec \leftarrow \frac{z_2}{z_1}$
43: ***while*** $RatioVec[1] \geq Dist_{end}$ *Or* $RatioVec[2] \geq Dist_{end}$ ***do***
44: $\alpha \leftarrow \frac{\alpha}{m_\alpha}$
45: $\gamma \leftarrow \alpha \log\left(\frac{1-\tau}{\tau}\right)$ ▷ *in this Alg.,* $\gamma = 0$ *(not necessarily for Alg. 4)*
46: $z_2 \leftarrow (\mathcal{L}_{\alpha,\tau}(LowerLimit - \gamma), \mathcal{L}_{\alpha,\tau}(UpperLimit - \gamma))^T$
47: $RatioVec \leftarrow \frac{z_2}{z_1}$
48: ***end while***
49: $\alpha_{end} \leftarrow \alpha$

7.3 Obtaining a pilot fit

To obtain a pilot fit for the QAM at the correct quantile, we must first obtain pilot values for the vector of smoothing parameters. We propose to use the vector of smoothing parameters of the mean GAM fit as a reference. We note that as mean and median are closely related, we could obtain an approximate value for the vector of smoothing parameters $\boldsymbol{\lambda}_{\tau=0.5,pilot}$ by calculating a simple ratio. Assuming that a median GAM fit will have the same degrees of smoothness as a mean GAM fit, we only have to take the ratio between the residuals of the two models to obtain an approximate value for $\boldsymbol{\lambda}_{\tau=0.5,pilot}$:

$$\boldsymbol{\lambda}_{j,\tau=0.5,pilot} = \frac{\lambda_{j,GAM}}{n} \frac{\sum_{i=1}^{n} |u_i|}{\sum_{i=1}^{n} u_i^2}$$

where: $u_i = y_i - \mathbf{x}_i^T \hat{\beta}_{GAM}$.[1] Some alternative methods:

- First, we calculate $y_i^{New} = \hat{f}_i^{GAM} + sign\,(u_i)\sqrt{|u_i|}$, then we fit a second $GAM^{(2)}$ where the y_i are replaced by y_i^{New}. Then $\lambda_{j,\tau=0.5,pilot} = \frac{\lambda_{j,GAM^{(2)}}}{n}$.
- First, we calculate $y_i^{New} = \hat{f}_i^{GAM} + sign\,(u_i)\sqrt{\mathcal{L}_{\overline{\alpha},\tau}\,(u_i)}$, then we fit a second $GAM^{(2)}$ where the y_i are replaced by y_i^{New}. Then $\lambda_{j,\tau=0.5,pilot} = \frac{\lambda_{j,GAM^{(2)}}}{n}$, where $\overline{\alpha}=$: $\hat{\alpha}$ or α_L or $\frac{\hat{\alpha}+\alpha_L}{2}$

Now that we have pilot values for the vector of smoothing parameters, we propose to use these smoothing pilot values to fit a pilot QAM at the correct quantile using Iteratively Reweighted Least Squares (Algorithms 10 or 11) using α_{Start} as a starting value and α_{end} as a stopping value (for Algorithm 10, using α_{end} instead of α_L). The computational cost of this fit is $\mathcal{O}(np^2 + p^3)$ (see section 7.11). At this stage we are then at a total of $\mathcal{O}(mnp^2 + p^3 + n^{\frac{4}{3}})$

7.4 Positive definite Hessian

Both for the Hessian of the QGACV and the Hessian of the penalized empirical loss, we may have a non-positive definite matrix. First for the Hessian of the QGACV, as the QGACV surface is not convex, the matrix could be negative definite or the function could be convex in some directions and concave in other directions. Note that the Hessian of the QGACV is used if we optimize the QGACV surface using Newton-Raphson but not Generalized Fellner-Schall. However, to calculate the gradient of the QGACV surface, the Hessian of the penalized empirical loss must be inverted for implicit differentiation. The Hessian of the penalized empirical loss may be non invertible. As we reduce α, the second order derivatives of the loss function goes to infinity for the interpolated points and to zero for the points that are not interpolated. The Hessian then

[1] Note that in package 'mgcv', the loss function is not divided by the number of observations n and we need to adjust for this (as the loss function is divided by n in package 'qgacv').

has a very large condition number and cannot be inverted. We propose to use a procedure to make the Hessian of the QGACV or the penalized empirical loss positive definite. We start with a Cholesky decomposition of the Hessian. If the Cholesky decomposition fails, we add an element ϵ to the diagonal of the matrix and increase this element until the Cholesky decomposition succeeds (for example, we start at $\epsilon = 1e-9$ and multiply successively by 10). To avoid making to large steps, after having found the element that makes the matrix positive definite, we multiply it by 10,000 (10×1000), test again if the Cholesky decomposition succeeds and stop.

Algorithm 2 Make positive definite

given a Hessian $\mathcal{H}$
set $DiagElement = 1e-9$
$Chol = try(Cholesky(\mathcal{H}))$
if Chol does not return an error **then**
    invert from Cholesky decomposition $\mathcal{H}$
else
    $exit \leftarrow False$
    while $exit$ is $False$ **do**
        while Chol returns an error **do**
            $Chol \leftarrow try(Cholesky(\mathcal{H} + diag(DiagElement)))$
            $DiagElement \leftarrow DiagElement \times 10$
        end while
        $DiagElement \leftarrow DiagElement \times 1000$
        $Chol = try(Cholesky(\mathcal{H} + diag(DiagElement)))$
        if Chol does not return an error **then**
            $exit \leftarrow True$
        end if
    end while
    invert from Cholesky decomposition $(\mathcal{H} + diag(DiagElement))$
end if

7.5 estimation of $\hat{\alpha}$

The estimation of $\hat{\alpha}$ follows the method described in section 3.15. The algorithm is detailed below. In terms of computational cost, as discussed previously, the sorting algorithm is assumed to be $\mathcal{O}(n^{\frac{4}{3}})$. At this stage of the algorithm, we are still at a computational cost of $\mathcal{O}(mnp^2 + p^3 + n^{\frac{4}{3}})$

Algorithm 3 Estimation of $\hat{\alpha}$

set a grid of values of $\alpha_{Grid} = (\alpha_1, ..., \alpha_A)^T$
$Dist \leftarrow (0, ..., 0)$
set StopLevel
Calculate residuals from the pilot fit: $u_i = y_i - \hat{f}_i^{Pilot}$
Sort the residuals $s_i = sort(u_i)$ (This is to help display the results on a chart).
for i in 1:n **do**
    $EL_i \leftarrow \frac{1}{n} \sum_{j=1}^{n} \mathcal{L}_\tau(u_j + s_i)$
end for
for a in 1:A **do**
    $\alpha_{temp} \leftarrow \alpha_{Grid}[a]$
    $Dist_a \leftarrow \frac{1}{n} \sum_{i=1}^{n} (EL_i - \mathcal{L}_{\alpha_{temp},\tau}(u_i))^2$
    if $Dist_a > StopLevel \times \min(Dist_1, ..., Dist_{a-1})$ **then** exit for loop
    end if
end for
$\hat{\alpha} \leftarrow \alpha_{Grid}[\text{argmin}_{a=1,...,A}\, Dist_a]$

7.6 estimate α_U and α_L

We obtained an estimate of the vector of smoothing parameters $\boldsymbol{\rho}$ based on a pilot fit in 7.3. We then estimated $\hat{\alpha}$ in 7.5. We now propose to estimate α_U the rounding level at which we stop the graduated optimization and α_L the level where we finish the precise estimation of the vector of parameters $\boldsymbol{\beta}$. The algorithm is similar to the algorithm used to estimate α_{end}. In section 7.2, we used the residuals from a GAM fit and estimated α_{start} and α_{end} with the symmetric loss function $\mathcal{L}_{\alpha,\tau=0.5}$. We will now run a similar algorithm with the error of the pilot fit at the correct quantile. For example, if $\tau = 0.05$, the error associated with each quantile is:

value	-0.077	-0.053	-0.0012	-0.00188	-0.00096	-0.00094
quantile	0.01	0.02	0.03	0.04	0.05	0.06
value	0.000074	0.31	0.40	0.497	0.517	0.557
quantile	0.07	0.08	0.09	0.1	0.11	0.12

To estimate α_U, we first need to determine the points on each side of quantile 0.05 that we will be seeking to find in the residuals as our base points. Assuming that we are looking for an interval of 0.1 between the upper and lower limit, we would then use quantiles $0.05 - \frac{0.1}{2} = 0$ and $0.05 + \frac{0.1}{2} = 0.1$. To avoid issues, as described in Algorithm 4, the first quantile is floored at 0.01. We then try to seek the values closest to quantiles 0.01 and 0.1. We then use the corresponding values -0.077 and 0.497 as the points where to compare the exact pinball loss to the surrogate loss (see Figure 7.2).

For α_L, we have a similar issue to what we had in section 7.2. To determine α_L, we are trying to find points that are for example 0.0001 away from quantile

0.05. Taking the closest quantiles, we would then obtain 0.04 and 0.06. We note that in this case, the values associated are -0.00188 and -0.0094. These two values are extremely close to each other and both on the same side of 0. If we were to use these two values as the points where to compare the exact pinball loss with the surrogate loss to find rounding parameter α_L, we would obtain a very small value. To avoid this issue, as in section 7.2, we find the value of the residual that is positive and that is closest to quantile 0.06. In this case, this is quantile 0.07 with error value 0.000074. We will then use these two values to determine α_L. This example is illustrated on Figure 7.2 and the Algorithm implementation is given in 4.

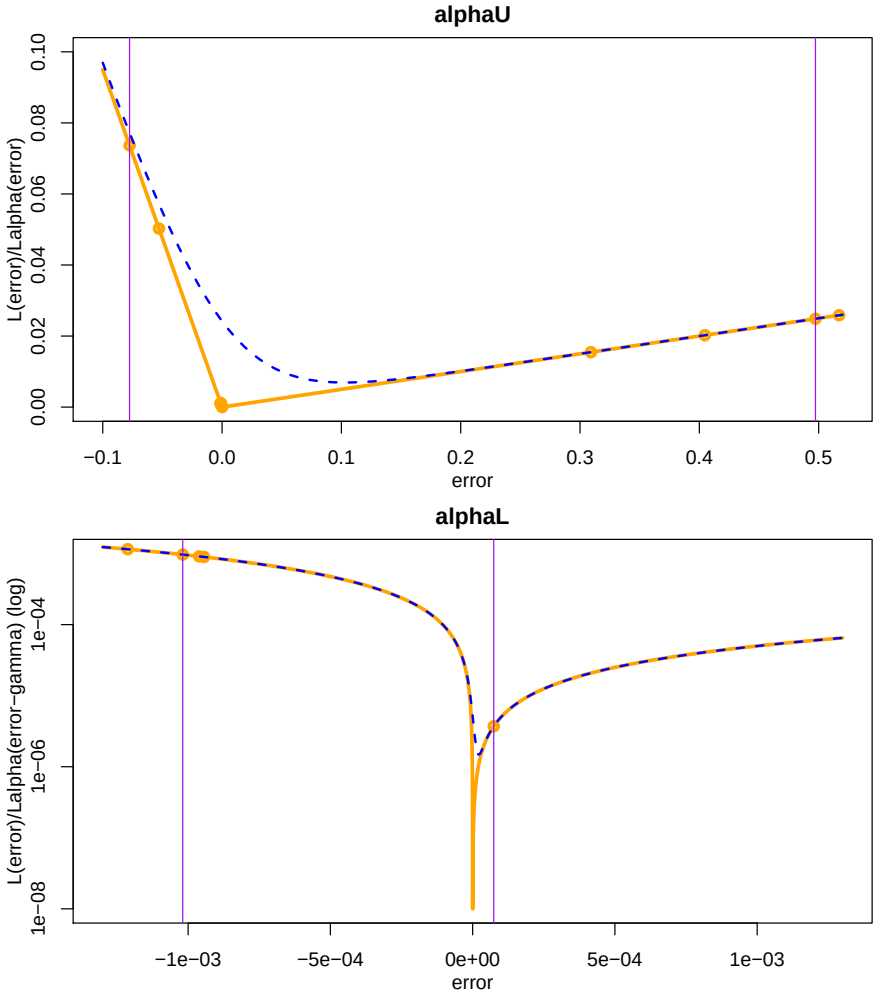


Fig. 7.2: On this Figure, we can see how α_U and α_L are selected. We fitted a pilot QAM fit in section 7.3. The error is then calculated as: $u_i = (y_i - \hat{f}_i^{QAM,Pilot}), i = 1, ..., n$. We then use the quantiles of the error as base points to calculate the distance between the surrogate loss and the exact pinball loss. On the top chart, we see how α_U is selected. In orange, we can see the exact pinball loss. Each orange dot represents the value of the error for our example that has $n = 100$ datapoints. The vertical purple lines show the selected values using quantiles of the data. We calculate the distance at these vertical lines between the surrogate loss in dotted blue and the exact pinball loss. We choose α_U such that on both sides, the surrogate loss rounded at α_U is greater than or equal to 1.05 times the value of the exact pinball loss. On the bottom chart, we show how α_L is selected. We first select quantile points where we are going to compare the surrogate loss to the exact pinball loss. As both points are on one side of the origin, we force the values selected to be on both sides of 0. The points selected are -1.02×10^{-3} and 7.42×10^{-5}. We then try to find the value α_L such that the surrogate loss rounded at α_L is at most 0.1% above the exact pinball loss. The y axis is on a log scale. We floor the values at 10^{-8} to be able to use the log scale.

Algorithm 4 Obtain α_U, α_L
This algorithm is similar to Algorithm 1.
As both algorithms are similar, they are implemented as a single function in package qgacv.

for a given n, τ ▷ e.g. $n = 100, \tau = 0.05$
estim. $\hat{\boldsymbol{\beta}}_{\tau}^{QAM,Pilot}$ using Alg. 10 ▷ from α_{Start} and α_{end} (Alg. 1)
or Algorithm 11 ▷ starting at α_{Start}
Calculate the error vector $\mathbf{u}^{\mathbf{QAM}} = \left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}_{\tau}^{\mathbf{QAM}}\right)$
Set $Interval_{\alpha_U}$ and $Interval_{\alpha_L}$ ▷ *e.g.* 0.1 *and* 0.0001
Set $Dist_U$ *and* $Dist_L$ ▷ *e.g. 5% and 0.1%*
Set m_α ▷ *e.g.* $m_\alpha = 1.01$
calculate the empirical cdf of $\mathbf{u}^{\mathbf{QAM}}$*:* $ecdf_{\mathbf{uQAM}}$
if $(\forall j = 1, .., n, u_j < 0)$ ***then***
    $\mathbf{u}^{Greater} = \{u_n^{QAM}\}$; $\mathbf{u}^{Lower} = \{u_1^{QAM}, ..., u_{n-1}^{QAM}\}$
else if $(\forall j = 1, .., n, u_j > 0)$ ***then***
    $\mathbf{u}^{Lower} = \{u_1^{QAM}\}$; $\mathbf{u}^{Greater} = \{u_2^{QAM}, ..., u_n^{QAM}\}$
else
    $\mathbf{u}^{Greater} = \{u_j^{QAM}, j = 1, ..., n,\ s.t. u_j^{QAM} \geq 0\}$
    $\mathbf{u}^{Lower} = \{u_j^{QAM}, j = 1, ..., n,\ s.t. u_j^{QAM} < 0\}$
end if
$L_0 \leftarrow length(\mathbf{u}^{Lower})$ ▷ *e.g.* $L_0 = 6$
$G_0 \leftarrow length(\mathbf{u}^{Greater})$ ▷ *e.g.* $G_0 = 94$
Calculate
$DistToCenter_{\alpha_U} = \frac{Interval_{\alpha_U}}{2}$ ▷ *e.g.* $= \frac{0.1}{2} = 0.05$
$UpperSide \leftarrow \min(\tau + DistToCenter_{\alpha_U}, ecdf_{\mathbf{u}^{QAM}}(u_{G_0}^{Greater}))$
▷ *e.g.* $\min(0.05 + 0.05, ecdf_{\mathbf{u}^{QAM}}(5.77)) = \min(0.05 + 0.05, 1) = 0.1$
$LowerSide \leftarrow \max(\tau - DistToCenter_{\alpha_U}, ecdf_{\mathbf{u}^{QAM}}(u_1^{Lower}))$
▷ *e.g.* $\max(0.05 - 0.05, ecdf_{\mathbf{u}^{QAM}}(-0.077)) = \max(0, 0.01) = 0.01$
$UpperLimit \leftarrow \underset{u_j^{Greater}, j=1,...,G_0}{\operatorname{argmin}} \left\{ecdf_{\mathbf{u}^{GAM}}\left(u_j^{Greater}\right) \geq UpperSide\right\}$
▷ *e.g* 0.4973
$LowerLimit \leftarrow \underset{u_j^{Lower}, j=1,...,L_0}{\operatorname{argmax}} \left\{ecdf_{\mathbf{u}^{GAM}}\left(u_j^{Lower}\right) \leq LowerSide\right\}$
▷ *e.g* -0.077
$z_1 \leftarrow (\mathcal{L}_\tau(LowerLimit), \mathcal{L}_\tau(UpperLimit))^T$
$\alpha \leftarrow \hat{\alpha}$
$\gamma \leftarrow \alpha \log\left(\frac{1-\tau}{\tau}\right)$
$z_2 \leftarrow (\mathcal{L}_{\alpha,\tau}(LowerLimit - \gamma), \mathcal{L}_{\alpha,\tau}(UpperLimit - \gamma))^T$
$RatioVec \leftarrow \frac{z_2}{z_1}$
while $RatioVec[1] \geq Dist_U$ *Or* $RatioVec[2] \geq Dist_U$ ***do***
    $\alpha \leftarrow \frac{\alpha}{m_\alpha}$
    $\gamma \leftarrow \alpha \log\left(\frac{1-\tau}{\tau}\right)$
    $z_2 \leftarrow (\mathcal{L}_{\alpha,\tau}(LowerLimit - \gamma), \mathcal{L}_{\alpha,\tau}(UpperLimit - \gamma))^T$
    $RatioVec \leftarrow \frac{z_2}{z_1}$
end while
$\alpha_U \leftarrow \alpha$

Calculate
$DistToCenter_{\alpha_L} = \frac{Interval_{\alpha_L}}{2}$ ▷ *e.g.* $= \frac{0.0001}{2} = 0.00005$
$UpperSide \leftarrow \min(\tau + DistToCenter_{\alpha_L}, ecdf_{\mathbf{u}^{QAM}}(u_{G_0}^{Greater}))$
▷ *e.g.* $\min(0.05 + 0.00005, ecdf_{\mathbf{u}^{QAM}}(5.77)) = 0.05005$
$LowerSide \leftarrow \max(\tau - DistToCenter_{\alpha_L}, ecdf_{\mathbf{u}^{QAM}}(u_1^{Lower}))$
▷ *e.g.* $\max(0.05 - 0.00005, ecdf_{\mathbf{u}^{QAM}}(-0.077)) = 0.04995$
$UpperLimit \leftarrow \underset{u_j^{Greater}, j=1,\ldots,G_0}{\operatorname{argmin}} \left\{ecdf_{\mathbf{u}^{GAM}}\left(u_j^{Greater}\right) \geq UpperSide\right\}$
$LowerLimit \leftarrow \underset{u_j^{Lower}, j=1,\ldots,L_0}{\operatorname{argmax}} \left\{ecdf_{\mathbf{u}^{GAM}}\left(u_j^{Lower}\right) \leq LowerSide\right\}$
$z_1 \leftarrow (\mathcal{L}_\tau(LowerLimit), \mathcal{L}_\tau(UpperLimit))^T$
$\gamma \leftarrow \alpha \log\left(\frac{1-\tau}{\tau}\right)$
$z_2 \leftarrow (\mathcal{L}_{\alpha,\tau}(LowerLimit - \gamma), \mathcal{L}_{\alpha,\tau}(UpperLimit - \gamma))^T$
$RatioVec \leftarrow \frac{z_2}{z_1}$
while $RatioVec[1] \geq Dist_L$ *Or* $RatioVec[2] \geq Dist_L$ ***do***
    $\alpha \leftarrow \frac{\alpha}{m_\alpha}$
    $\gamma \leftarrow \alpha \log\left(\frac{1-\tau}{\tau}\right)$
    $z_2 \leftarrow (\mathcal{L}_{\alpha,\tau}(LowerLimit - \gamma), \mathcal{L}_{\alpha,\tau}(UpperLimit - \gamma))^T$
    $RatioVec \leftarrow \frac{z_2}{z_1}$
end while
$\alpha_L \leftarrow \alpha$

7.7 Smoothing parameter initialization procedure

To find initial values for the smoothing parameters, we propose starting values that lead to edf values that are at mid-point between the edf extremes. Let's assume we have a regression with m smooths. We start with the 1st smooth. We set the value of the smoothing parameters for all the other smooths at the maximum possible values ($\rho_2 = ... = \rho_m = MaxRho$). If we take $\rho_1 = MaxRho$, we then obtain the smallest possible edf of the model. For example for a semi-parametric regression with an unpenalized linear regression part, the edf would be the number of predictors +1. If then ρ_1 takes its smallest possible value $\rho_1 = MinRho$, we obtain the maximum possible edf change that can result from a change of ρ_1. The algorithm will then consist in finding ρ_{Mid} such that the $edf(\rho_{Mid}) \approx \frac{edf(MinRho)+edf(MaxRho)}{2}$. We use this method because we could expect that if we are in the mid-point of the edf curve, we are at one of the points of highest steepness and potentially largest change in QGACV. This procedure is described in the algorithm below. Note that in all the algorithms implemented, we use the vector of log smoothing parameters $\boldsymbol{\rho} = \log(\boldsymbol{\lambda})$. To match what is implemented in package 'qgacv', we use the log smoothing parameters in the description of the algorithm.

Algorithm 5 Smoothing parameter initialization procedure

set tol
set $AddElement$
set $\boldsymbol{\rho}_{max}$ and $\boldsymbol{\rho}_{min}$, vectors containing the maximum and the minimum values of vector $\boldsymbol{\rho}$, respectively. ▷ e.g. $\rho_{min,j} = -30$ $\rho_{max,j} = +5$
$edfMax \leftarrow$ edf calculated with all the smoothing parameters equal to the minimum possible value
for j in 1:m **do**
    $edfHigh_j \leftarrow edfMax$
    $\boldsymbol{\rho}_{LowEdf} \leftarrow \boldsymbol{\rho}_{min}$
    $\boldsymbol{\rho}_{HighEdf} \leftarrow \boldsymbol{\rho}_{min}$
    $\rho_{LowEdf,j} \leftarrow \rho_{max,j}$
    Calculate $edfLow_j$ using $\boldsymbol{\rho}_{LowEdf}$, where all entries are equal to maximum values and entry j is set to the minimum.
    $edfDiff_j = edfHigh_j - edfLow_j$.
    We then calculate an upper or lower level target. We stop when we reach a level between the upper and lower level:
    $edfTargetUpper_j = edfHigh_j - edfDiff/2 \times (1 - tol)$
    $edfTargetLower_j = edfHigh_j - edfDiff/2 \times (1 + tol)$
    $ExitLoop \leftarrow False$
    while (($edfHigh_j > edfTargetUpper_j$) **And**
        ($edfLow_j < edfTargetLower_j$) **And**
        ($ExitLoop$ is $False$)) **do**
        $\boldsymbol{\rho}_{MidEdf} \leftarrow \frac{\boldsymbol{\rho}_{LowEdf} + \boldsymbol{\rho}_{HighEdf}}{2}$
        Calculate $edfMid_j$ using $\boldsymbol{\rho}_{MidEdf}$
        if ($edfMid_j > edfTargetLower_j$) **then**
            $\boldsymbol{\rho}_{HighEdf} \leftarrow \boldsymbol{\rho}_{MidEdf}$
            $edfHigh_j \leftarrow edfMid_j$
        else
            $\boldsymbol{\rho}_{LowEdf} \leftarrow \boldsymbol{\rho}_{MidEdf}$
            $edfLow_j \leftarrow edfMid_j$
        end if
        if $|\rho_{LowEdf,j} - \rho_{HighEdf,j}| <$
          $(MinDistHighLow \times I(\rho_{min,j} <> \rho_{min,j})$ **then**
            $ExitLoop \leftarrow True$
        end if
        if $ExitLoop$ is $False$ **then**
            if (($edfHigh_j \geq edfTargetLower_j$) **And**
                ($edfHigh_j \leq edfTargetUpper_j$)) **then**
                $\rho_j \leftarrow \rho_{HighEdf,j}$
            end if
            if (($edfLow_j \geq edfTargetLower_j$) **And**
                ($edfLow_j \leq edfTargetUpper_j$)) **then**
                $\boldsymbol{\rho}_j \leftarrow \rho_{LowEdf,j}$
            end if
        else
            $\rho_j \leftarrow \rho_{HighEdf,j}$
        end if
    end while
end for

45: $NotNegative \leftarrow False$
46: $AllRhoAtMax \leftarrow False$
47: **while** $((NotNegative = False)$**And** $(AllRhoAtMax = False))$ **do**
48: Calculate $edf(\boldsymbol{\rho})$
49: **if** $n.eff > c \times edf$ **then**
50: $NotNegative \leftarrow True$
51: **else**
52: $AllRhoAtMax \leftarrow True$
53: **for** j in 1:m **do**
54: **if** $\rho_j < \rho_{max,j}$ **then**
55: $AllRhoAtMax \leftarrow False$
56: **end if**
57: **end for**
58: **if** AllRhoAtMax=False **then**
59: **for** j in 1:m **do**
60: $\rho_j \leftarrow \min(\rho_j + AddElement, \rho_{max,j})$
61: **end for**
62: **end if**
63: **end if**
64: **end while**

7.8 Find a non-flat zone

After rounding parameter α has been reduced, we check that we are either in a zone of the QGACV with a gradient or in a zone where the Hessian is non-zero. If the gradient and Hessian are both small, this means that we are in a flat zone of the QGACV surface. In each direction j, where the QGACV surface is flat, we are going to reduce the smoothing parameter until either we reach a zone where $n.eff < c \times tr(\mathbf{A})$ or a zone where either the Hessian or the gradient are non-zero. In terms of computational cost, calculating the QGACV Hessian has a computational cost of $\mathcal{O}((\frac{m(m+1)}{2} + n)p^2 + p^3)$ (see Appendix C). In total at this stage of the algorithm, we are then at $\mathcal{O}(m(n + \frac{m}{2})p^2 + p^3 + n^{\frac{4}{3}})$

Algorithm 6 Find Non-flat zone

set $GradientMin2$
set $HessianMin2$
set $AddElement$ ▷ e.g. $AddElement = -1$
set $\boldsymbol{\rho}_{max}$ and $\boldsymbol{\rho}_{min}$, vectors containing the maximum and the minimum values of vector $\boldsymbol{\rho}$, respectively.
Calculate the QGACV Hessian: $\mathcal{H}_{QGACV} = \frac{\partial^2 QGACV}{\partial \boldsymbol{\rho} \partial \boldsymbol{\rho}^T}$
Calculate the QGACV gradient: $\mathcal{G}_{QGACV} = \frac{\partial QGACV}{\partial \boldsymbol{\rho}}$
$StopLoop \leftarrow False$
for j in 1:m **do**
    while $\left(\left(\left|\frac{\mathcal{G}_{QGACV,j}}{QGACV}\right| < GradientMin2\right)$ **And** $(\rho_j > \rho_{min,j})$ **And**
    $(StopLoop = False)$ **And** $\left(\frac{\mathcal{H}_{QGACV,(j,j)}}{QGACV} < HessianMin2\right)\right)$ **do**
        $\rho_{temp,j} \leftarrow \max(\rho_j + AddElement, \rho_{min,j})$
        Calculate edf
        if $((c \times edf < n.eff)$ **And** $(\rho_{temp,j} <> \rho_{min,j}))$ **then**
            calculate $\tilde{\boldsymbol{\beta}}$
            $\rho_j \leftarrow \rho_{temp,j}$
            recalculate QGACV value, gradient, Hessian.
        else
            $StopLoop \leftarrow True$
        end if
    end while
end for

7.9 Optimize QGACV

At this stage of the algorithm, we calculated the values of $\hat{\alpha}$, α_U, α_L, as well as the initial values of the vector of smoothing parameters $\boldsymbol{\rho}$. The QGACV optimization procedure then follows the description given in section 5.2. The two possible algorithms are a Generalized Fellner-Schall and a Newton-Raphson hyper-optimization.

Algorithm 7 Graduated Optimization with Generalized Fellner-Schall

Obtain $\boldsymbol{\beta}_{initial}$ and $\boldsymbol{\rho}_{initial}$ (see Algorithm 5).
set MultAlpha ▷ e.g. MultAlpha=0.5
set $MaxIterInner$
$\alpha \leftarrow \hat{\alpha}$; $\boldsymbol{\rho} \leftarrow \boldsymbol{\rho}_{Initial}$
$LastLoop \leftarrow False$
while $LastLoop$ is $False$ **do**
    $CountInner \leftarrow 0$
    $ErrInner \leftarrow \infty$
    set ϵ_{Inner}
    while $((CountInner < MaxIterInner)$ AND
     $(ErrInner > \epsilon_{inner}))$ **do**
        $\boldsymbol{\rho}_{old} \leftarrow \boldsymbol{\rho}$
        if $(CountInner = 0)$ **then**
            Find Non-flat Zone (Algorithm 6)
        end if
        Obtain $\boldsymbol{\rho}$ using Generalized Fellner Schall Step (see 5.2). Each step
        for $\boldsymbol{\rho}$ is followed by an update of $\tilde{\boldsymbol{\beta}}$ using Alg. 9 (nested iteration).
        $ErrInner \leftarrow \sum(\boldsymbol{\rho} - \boldsymbol{\rho}_{old})^2$
        $CountInner \leftarrow CountInner + 1$
    end while
    $\alpha_{Old} \leftarrow \alpha$
    $\alpha \leftarrow MultAlpha \times \alpha$
    if $\alpha \leq \alpha_{Old}$ **then**
        $\alpha \leftarrow \alpha_{Old}$
    end if
    if $(\alpha_{Old} = \alpha_U)$ And $(\alpha = \alpha_U)$ **then**
        $LastLoop \leftarrow True$
        $\alpha_{Old} \leftarrow \alpha$
        $\alpha \leftarrow MultAlpha \times \alpha$
        if $(\alpha \leq \alpha_L)$ **then**
            $\alpha \leftarrow \alpha_L$
        end if
    end if
end while

Algorithm 8 Graduated Optimization with Hyper-Newton Raphson

Obtain $\boldsymbol{\beta}_{initial}$ and $\boldsymbol{\rho}_{initial}$ (see Algorithm 5).
$\alpha \leftarrow \hat{\alpha}$
$\boldsymbol{\rho} \leftarrow \boldsymbol{\rho}_{Initial}$
set $MaxIterInner$
Set $MaxIterationsHalfStep$
$LastLoop \leftarrow False$
while $LastLoop$ is $False$ **do**
    $CountInner \leftarrow 0$
    $ErrInner \leftarrow \infty$
    set ϵ_{Inner}
    while $((CountInner < MaxIterInner)$ AND
     $(ErrInner > \epsilon_{inner}))$ **do**
        $\boldsymbol{\rho}_{old} \leftarrow \boldsymbol{\rho}$
        if $(CountInner = 0)$ **then**
            Find Non-flat Zone (Algorithm 6)
        end if
        Calculate the hessian $\mathcal{H}_{qgacv}$
        Calculate the gradient $\mathcal{G}_{qgacv}$
        $\mathcal{H}_{qgacv} \leftarrow MakePositiveDefinite(\mathcal{H}_{qgacv})$ (algorithm 2)
        $Step \leftarrow \mathcal{H}^{-1}_{qgacv}\mathcal{G}_{qgacv}$
        $qgacv_{old} \leftarrow qgacv$
        $NbIterationsHalfStep \leftarrow 1$
        $\delta \leftarrow 1$
        while $(qgacv_{old} < qgacv)$ And
     $(NbIterationsHalfStep < MaxIterationsHalfStep)$ **do**
            $\boldsymbol{\rho} \leftarrow \boldsymbol{\rho}_{old} - \delta \times Step$
            calculate $\tilde{\boldsymbol{\beta}}$ (algorithm 9)
            calculate $qgacv$
            $\delta \leftarrow \frac{\delta}{2}$
            $NbIterationsHalfStep \leftarrow NbIterationsHalfStep + 1$
        end while
        if $(qgacv_{old} > qgacv)$ And $(n.eff > c \times edf)$ **then**
            update $\boldsymbol{\rho}$
        end if
        $ErrInner \leftarrow \sum(\boldsymbol{\rho} - \boldsymbol{\rho}_{old})^2$
        $CountInner \leftarrow CountInner + 1$
    end while
    $\alpha_{Old} \leftarrow \alpha$
    $\alpha \leftarrow MultAlpha \times \alpha$
    if $\alpha \leq \alpha_{Old}$ **then**
        $\alpha \leftarrow \alpha_{Old}$
    end if
    if $(\alpha_{Old} = \alpha_U)$ And $(\alpha = \alpha_U)$ **then**
        $LastLoop \leftarrow True$
        $\alpha_{Old} \leftarrow \alpha$
        $\alpha \leftarrow MultAlpha \times \alpha$
        if $(\alpha \leq \alpha_L)$ **then**
            $\alpha \leftarrow \alpha_L$
        end if
    end if
end while

7.10 Optimize the penalized empirical loss with a fixed α and fixed ρ

For a fixed α and $\boldsymbol{\rho}$, the penalized empirical loss is a convex function. We use Newton-Raphson with half-stepping to optimize it. Note that as discussed in section 7.4, the Hessian of the penalized empirical loss may not be positive definite. We then apply Algorithm 2 to render it positive definite before stepping. In terms of computational costs, calculating the Hessian is $\mathcal{O}(np^2)$. Then the algorithm to make the matrix positive definite is $\mathcal{O}(p^3)$.

Algorithm 9 Optimize the penalized empirical loss with fixed α and $\boldsymbol{\rho}$

set $\epsilon_{gradient}$
set $MaxIter$
set ϵ_β
set $PenalizedLoss = 0$
set $Grad = 1000$
set $NbIter = 0$
while $\sum(\mathcal{G}^2_{PenalizedLoss}) > (|(PenalizedLoss)| \times \epsilon_{Gradient})$ **And**
    $(NbIter < MaxIter)$ **And**
    ($StopCondition$ is False) **And**
    $(\sum(\tilde{\boldsymbol{\beta}} - \tilde{\boldsymbol{\beta}}_{Old})^2 > \epsilon_\beta)$ **do**
    Calculate the Hessian $\mathcal{H}_{PenalizedLoss}$
    Calculate the gradient $\mathcal{G}_{PenalizedLoss}$
    $\mathcal{H}_{PenalizedLoss} \leftarrow MakePositiveDefinite(\mathcal{H}_{PenalizedLoss})$ (algorithm 2)
    $Step \leftarrow \mathcal{H}^{-1}_{PenalizedLoss}\mathcal{G}_{PenalizedLoss}$
    $PenalizedLoss_{old} \leftarrow PenalizedLoss$
    $NbIterationsHalfStep \leftarrow 1$
    $\delta \leftarrow 1$
    while $(PenalizedLoss_{old} < PenalizedLoss)$ And
    $(NbIterationsHalfStep < MaxIterationsHalfStep)$ **do**
        $\tilde{\boldsymbol{\beta}} \leftarrow \tilde{\boldsymbol{\beta}}_{old} - \delta \times Step$
        calculate $PenalizedLL$
        $\delta \leftarrow \frac{\delta}{2}$
        $NbIterationsHalfStep \leftarrow NbIterationsHalfStep + 1$
    end while
    if $(PenalizedLoss_{old} > PenalizedLoss)$ **And**
        $IterHalfStep < MaxIterHalfStep)$ **then**
        update $\tilde{\boldsymbol{\beta}}_{Old} \leftarrow \tilde{\boldsymbol{\beta}}$
        update $\tilde{\boldsymbol{\beta}}$
    else
        $StopCondition \leftarrow True$
    end if
end while

7.11 Generalized IRLS to optimize the penalized empirical loss

Once we have determined the vector of smoothing parameters, there are 2 possibilities to obtain a precise estimate for the vector of parameters. The first possibility consists in simply fixing the vector of smoothing parameters at the value estimated at α_U and continue to reduce α down to α_L (Algorithm 10) and successively estimate the vector of parameters. The second possibility is to reduce α until a stopping criterion is verified (Algorithm 11). The computational costs are determined by the Algorithm 9: $\mathcal{O}(np^2 + p^3)$.

Algorithm 10 Optimize penalized empirical loss until we reach α_L

given a starting value $\alpha = \alpha_U$ or $\alpha = \alpha_{start}$ or $\alpha = \hat{\alpha}$
set ϵ_{rel}
$LastLoop \leftarrow False$
$\tilde{\boldsymbol{\beta}}_{Old} \leftarrow \tilde{\boldsymbol{\beta}}$
while LastLoop is False **do**
    estimate $\tilde{\boldsymbol{\beta}}$ with rounding level α (algorithm 9)
    $\alpha_{Old} \leftarrow \alpha$
    $\alpha \leftarrow MultAlpha \times \alpha$
    if $\alpha \leq \alpha_L$ **then**
        $\alpha \leftarrow \alpha_L$
        $LastLoop \leftarrow False$
    end if
    if $(\alpha_{Old} = \alpha_L)$ And $(\alpha = \alpha_L)$ **then**
        $LastLoop \leftarrow True$
    end if
end while

Algorithm 11 Optimize penalized empirical loss with stopping criterion

given a starting value $\alpha = \alpha_U$
set ϵ_{rel}
$LastLoop \leftarrow False$
$\tilde{\boldsymbol{\beta}}_{Old} \leftarrow \tilde{\boldsymbol{\beta}}$
while LastLoop is False **do**
    estimate $\tilde{\boldsymbol{\beta}}$ with rounding level α (algorithm 9)
    if $\frac{\sum(\tilde{\boldsymbol{\beta}}-\tilde{\boldsymbol{\beta}}_{Old})^2}{\sum \tilde{\boldsymbol{\beta}}^2_{Old}} > \epsilon_{rel}$ **then**
        $\alpha_{Old} \leftarrow \alpha$
        $\alpha \leftarrow MultAlpha \times \alpha$
        $\tilde{\boldsymbol{\beta}}_{Old} \leftarrow \tilde{\boldsymbol{\beta}}$
    else
        $LastLoop \leftarrow True$
    end if
end while
$\alpha_L \leftarrow \alpha$ ▷ We replace α_L with stopping level.

7.12 Smoothing parameter relaxation

The smoothing parameter relaxation follows the methodology described in section 6.2. We give below more details about the algorithm used. Regarding the computational costs, the smoothing parameter relaxation starts with the estimation of B bootstrap estimates for the vector of parameters. The computational cost for one estimate is $\mathcal{O}(np^2 + p^3)$. For our B bootstrap estimates this is then $\mathcal{O}(B(np^2 + p^3))$. Then calculating the QGACV Hessian is $\mathcal{O}(m(n + \frac{m}{2})p^2 + p^3 + n^{\frac{4}{3}})$. This results in a computational cost of $\mathcal{O}\left(\left(\frac{m^2}{2} + (B + m)\, n\right) p^2 + Bp^3 + n^{\frac{4}{3}}\right)$.

7.13 Non-parametric bootstrap for confidence interval

The last step in the algorithm is to estimate non-parametric bootstrapped models as described in 6.1. Last, the computational cost of the B_0 non-parametric bootstrap samples is $\mathcal{O}\left(B_0\left(np^2 + \frac{p^3}{3}\right)\right)$. Adding to this the computational costs from the previous sections, we obtain a total computational cost of $\mathcal{O}\left(\left(\frac{m^2}{2} + (B + B_0 + m)\, n\right) p^2 + (B + B_0)\, p^3 + n^{\frac{4}{3}}\right)$

Algorithm 12 Smoothing parameter relaxation

Obtain the Hessian, Gradients for each level of the graduated optimization and the qgacv at the optimum of each level $(\mathcal{H}^{qgacv}_{\alpha_1}(\hat{\rho}_{\alpha_1}), ..., \mathcal{H}^{qgacv}_{\alpha_A}(\hat{\rho}_{\alpha_A}))$ and $(\mathcal{G}^{qgacv}_{\alpha_1}(\hat{\rho}_{\alpha_1}), ..., \mathcal{G}^{qgacv}_{\alpha_A}(\hat{\rho}_{\alpha_A}))$ and $(qgacv_{\alpha_1}(\hat{\rho}_{\alpha_1}), ..., qgacv_{\alpha_A}(\hat{\rho}_{\alpha_A}))$. These are obtained automatically from the graduated optimization.
Obtain $B = 10$ non-parametric bootstrap samples $\boldsymbol{\beta}^*_1, ..., \boldsymbol{\beta}^*_B$.
Calculate the B QGACV estimates using fixed smoothing parameter $\hat{\boldsymbol{\rho}}$: $qgacv(\hat{\boldsymbol{\rho}}, \boldsymbol{\beta}^*_1), ..., qgacv(\hat{\boldsymbol{\rho}}, \boldsymbol{\beta}^*_B)$.
Estimate $se(qgacv)$
Limit$\leftarrow$0.001
for $\alpha \in (\alpha_1, ..., \alpha_A)$ **do**
    $NumberPositiveDiagonalValues(\alpha) \leftarrow Count(\mathcal{H}^{qgacv}_{\alpha}(\hat{\rho}_{\alpha}) > Limit)$
end for
$\alpha_{selected} \leftarrow \max_{\alpha \in (\alpha_1,...,\alpha_A)}(NumberPositiveDiagonalValues(\alpha) = max(NumberPositiveDiagonalValues))$
$ChosenRowCols$ are the row/columns where $\hat{\rho}_{\alpha} > Limit$. We will then only update these selected columns/rows.
$k \leftarrow 1$
$Stop \leftarrow False$
$qgacv(1) \leftarrow qgacv(\hat{\boldsymbol{\rho}})$
while $\left(\frac{|(qgacv(k) - (qgacv(\hat{\boldsymbol{\rho}}) + se(qgacv)))|}{qgacv(k)} > 0.001\right)$**And**
(!$Stop$) **And** $(NumberPositiveDiagonalValues(\alpha_{selected})! = 0)$ **do**
    Update $\dot{\zeta}(k)$ using formula 6.6.
    $\rho(k)[ChosenRowCols] \leftarrow \rho(k-1)[ChosenRowCols]$ ...
    $$... - \zeta(k) \times \sqrt{diag\left(\left(\mathcal{H}^{QGACV}_{\hat{\boldsymbol{\rho}}}[ChosenRowCols]\right)^{-1}\right)}$$
    $ExitLoop \leftarrow False$
    while $(ExitLoop) = False$ **do**
        if $((n.eff(k) < c(Tr(A(k))))$ **Or**
        $(qgacv(k) > (qgacv(\hat{\boldsymbol{\rho}}) + se(qgacv))))$ **then**
            $\zeta(k) \leftarrow \frac{\zeta(k)}{1.05}$
            $Stop \leftarrow True$
        else $ExitLoop \leftarrow True$
        end if
    end while
    Update $NumberPositiveDiagonalValues(\alpha_{selected})$
    Update $ChosenRowCols$
end while

8. INTRODUCTION TO R PACKAGE QGACV

The methodology described in this thesis has been implemented in R package 'qgacv'. In this section, we give an introduction to the package. The main function of the package is function 'qam'. This function can be used similarly to function 'gam' in package 'mgcv'. Let's take an example. We can use function 'qgacvExamples'. This function contains 13 one-dimensional examples, two 2-dimensional examples and one 5-dimensional example. Example 1 is based on a function that can be found in Wood [2006], Wood [2019].

```
n<-200
tau<-0.7
ExampleNumber<-1
Ex<-qgacvExamples(ExampleNumber,n,tauArr=tau,seed=1,SetSeedYesNo=T)
```

Function 'qgacvExamples' returns a list containing 'data' an another vector/array containing the true quantiles. Let's print the curve:

```
plot(Ex$data$x,Ex$data$y,col="darkgrey",xlab="x",ylab="y",
main="True Quantile")
lines(Ex$data$x,Ex$TrueQuantiles,col="green",lwd=3)
```

The true quantile is displayed on Figure 8.1.

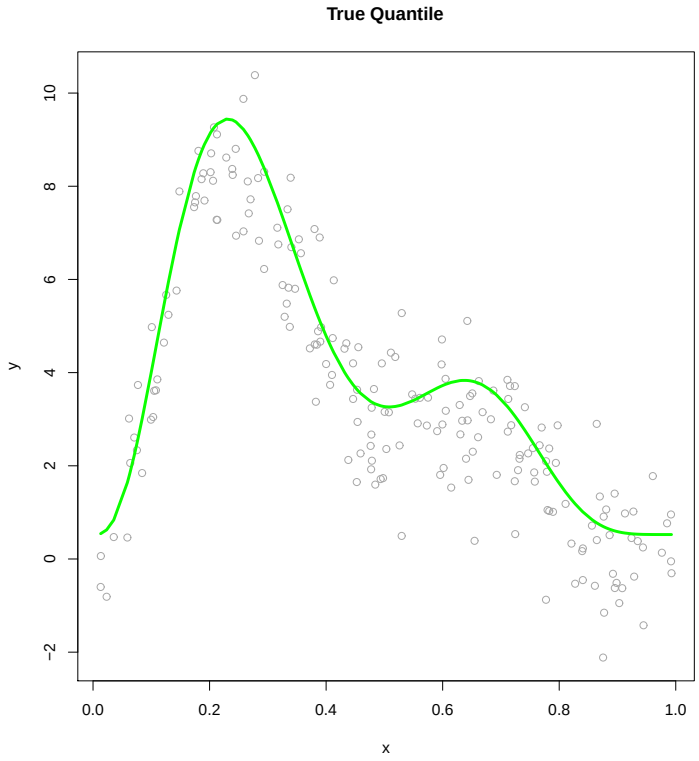


Fig. 8.1: This Figure shows the result of
plot(Ex$data$x,Ex$data$y,col="darkgrey",xlab="x",
ylab="y",main="True Quantile") followed by
lines(Ex$data$x,Ex$TrueQuantiles,col="green",lwd=3)
In green, we can see the true quantile.

We now use function qgacv::qam to fit a quantile smoothing spline with quantile parameter $\tau = 0.7$. $Nbboot_{CI} = 100$ indicates the number of bootstrap samples for the calculation of the confidence interval B_0. By switching $alphaHatDisplayEstimation$ to TRUE, a Figure showing how $\hat{\alpha}$ was chosen is displayed (see Figure 8.2). By default, the hat matrix is not returned by the function. We can return it by switch field 'ReturnHatMatrix' to TRUE. Two hat matrices are returned: The hat matrix $\mathbf{A}_{\hat{\alpha},\alpha_L}$ and $\mathbf{A}_{\alpha_L,\alpha_L} = \tilde{\mathbf{A}}_{\alpha_L}$

```
fit<-qam(y~s(x,k=20),data=Ex$data,tau=tau,CI_Nbboot=100,
 alphaHatDisplayEstimation=T, ReturnHatMatrix=TRUE)
```

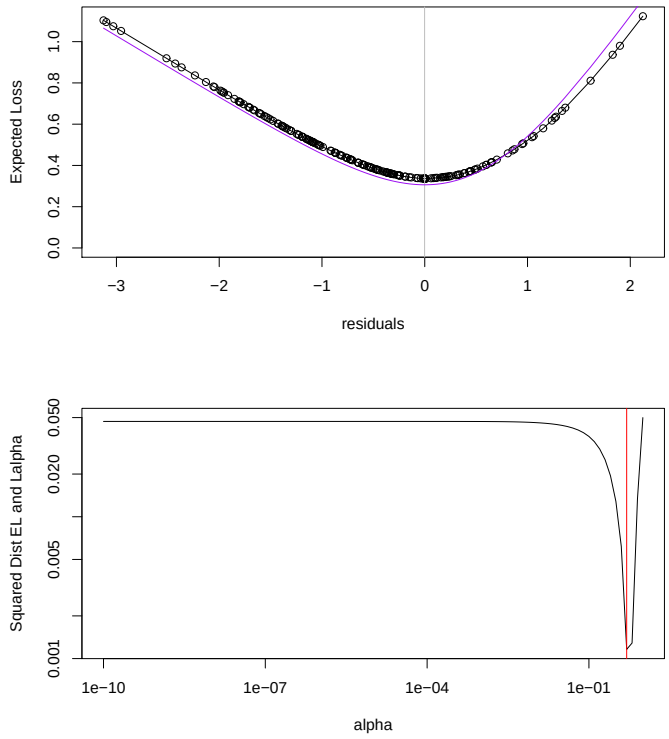


Fig. 8.2: The following charts show the selection of parameter $\hat{\alpha}$. On the top chart, the black line shows the empirical expected loss. The purple loss corresponds to the surrogate loss rounded at $\hat{\alpha}$. The bottom chart shows the selection of $\hat{\alpha}$

The fit for QSS can be displayed with function 'plot1d'. 'CIType' indicates the type of confidence interval (here 'covar' indicates covariance matrix calculated using the non-parametric bootstrap samples using relaxed fit). 'covar' is the default type for the confidence interval. We can also display the bootstrap samples themselves. They are contained in the list created by function 'qam'. To obtain the curves, we multiply model matrix $\mathbf{X}$ (fit$X) by each np-bootstrap estimated vector of parameter estimated using the relaxed vector of smoothing parameter $\grave{\boldsymbol{\rho}}$, that is $\grave{\boldsymbol{\beta}}^{*b}$ with $b = 1, ..., B_0$ contained in matrix fit$boot$bootstrapBeta. The result of this code is shown on Figure 8.3.

```
par(mfrow=c(2,1))
par(mar=c(3,3,3,3))
plot1d(fit,CIType="covar")
title(main="fit with CIs")
plot(Ex$data$x,Ex$data$y,col="darkgrey")
```

```
for(ii in 1:fit$NBBOOT){
lines(Ex$data$x,fit$X%*%fit$boot$bootstrapBeta[,ii],col="blue")}
title(main="Relaxed sp bootstrap samples")
```

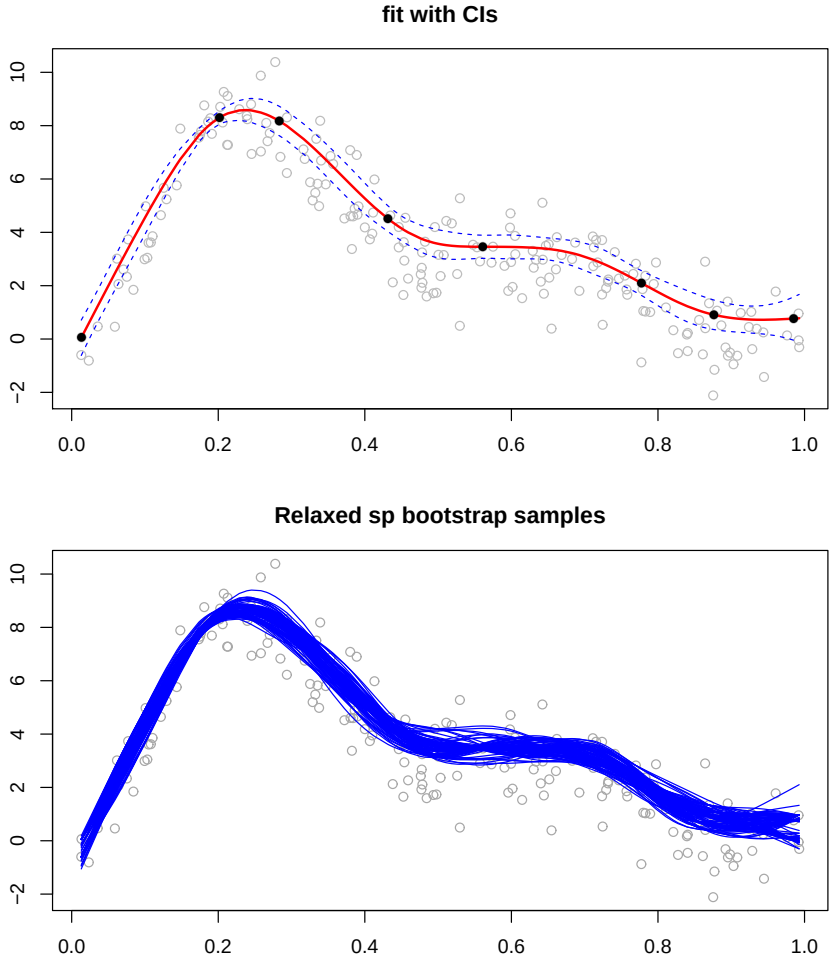


Fig. 8.3: The top chart shows a fit in red and the confidence interval obtained in dotted blue. The black dots are the interpolated points. Note the top chart is similar to charts that can be obtained using package 'mgcViz' Fasiolo and Nedellec [2020], Fasiolo et al. [2020a] together with package 'qgam' Fasiolo et al. [2020c]. The bottom chart shows the non-parametric bootstrap fitted curves.

We can then use function 'print' to view some details on the fitted object.

```
> print(fit)
```

```
Estimated degrees of freedom:
7  total = 8

QGACV score: 0.6271852

Quantile Parameter:  0.7

Percentage of points below the model:  0.695

Adjusted Percentage of points below the model:  0.703125
```

We can then print a summary of the object.

```
> summary(fit)
Formula:
y ~ s(x, k = 20)

Parametric coefficients:
Estimate Std. Error t value Pr(>|t|)
[1,]  4.01932    0.08688   46.26   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
edf Ref.df     F p-value
s(x) 6.998  6.998 76.42  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1



alphaValues:
alphaHat: 5e-01
alphaU: 3.4e-02
alphaL: 2.1e-06

QGACV: 0.6271852

Quantile Parameter:  0.7
Percentage of points below the model:  0.695
```

We can then visualize the whole optimization path of the QGACV.

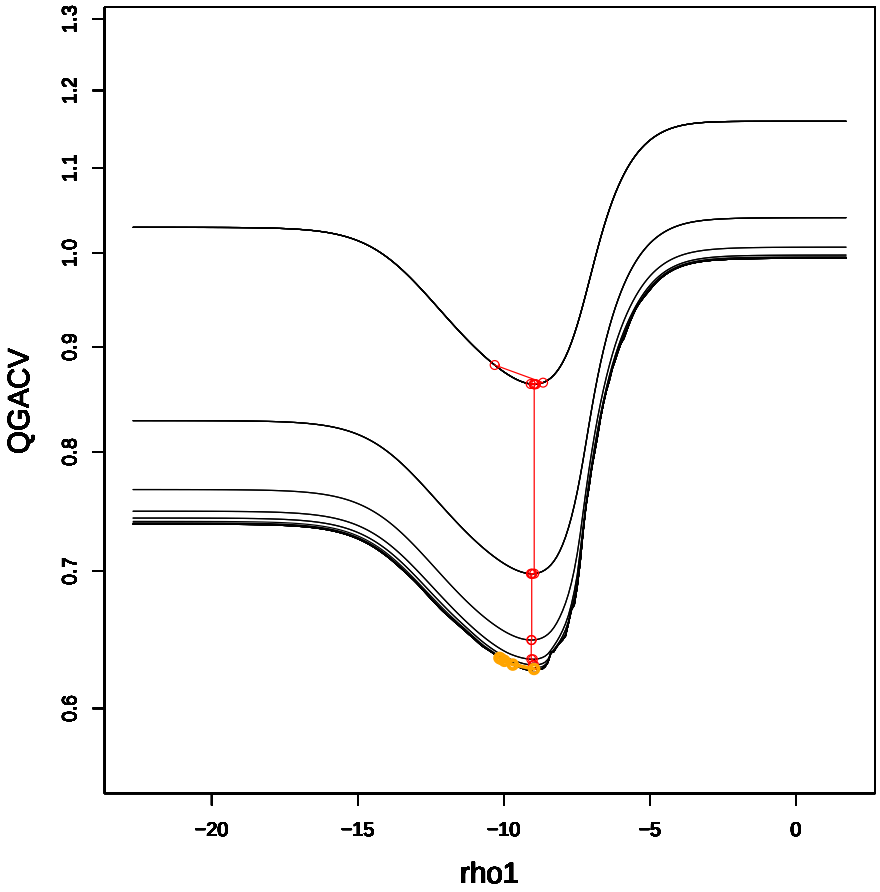


Fig. 8.4: This picture shows an example of graduated optimization of the QGACV. The red path is the optimization using Fellner-Schall. The orange path shows the relaxation of the smoothing parameter.

Finally, we may want to print the diagonal of the hat matrix $\tilde{\mathbf{A}}_{\alpha_L}$ where all components of the matrix are calculated at a very small level of rounding α_L (here $\alpha_L = 2.1 \times 10^{-6}$). This can give us an indication of the location of the points that would be eventually interpolated by the model as $\alpha \to 0$.

```
round(diag(fit$HatMatrix_L),digits=2)
```

```
[1] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
[38] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[75] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
```

```
[112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[149] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
[186] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
```

We have a 1 for each point that will be eventually interpolated by the model. We can also fit using different basis functions. We here show how the choice of basis function can lead to different fits for the model. We here fit models with 3 different basis functions (see Wood [retrieved 2020b]): "P-splines" from Eilers and Marx [1996], "cyclic cubic splines" and "adaptive splines". On Figure 8.5, we can see that with a p-spline, the curve seems to be too smooth compared to the true quantile, the cyclic cubic spline seems to fit incorrectly to the true quantile whilst the adaptive spline (by default, the adaptive spline uses 4 smoothing parameters) seems to be fitting the true quantile the best.

```
n<-200
tau<-0.2
ExampleNumber<-12
ExDiffBasis<-qgacvExamples(ExampleNumber,n,tauArr=tau,seed=1,SetSeedYesNo=T)
fitps<-qam(y~s(x,k=20,bs="ps"),data=ExDiffBasis$data,tau=tau)
fitcc<-qam(y~s(x,k=20,bs="cc"),data=ExDiffBasis$data,tau=tau)
fitad<-qam(y~s(x,k=20,bs="ad"),data=ExDiffBasis$data,tau=tau)
par(mfrow=c(2,2))
par(mar=c(3,3,3,3))
plot1d(fitps);plot1d(fitcc);plot1d(fitad);
```

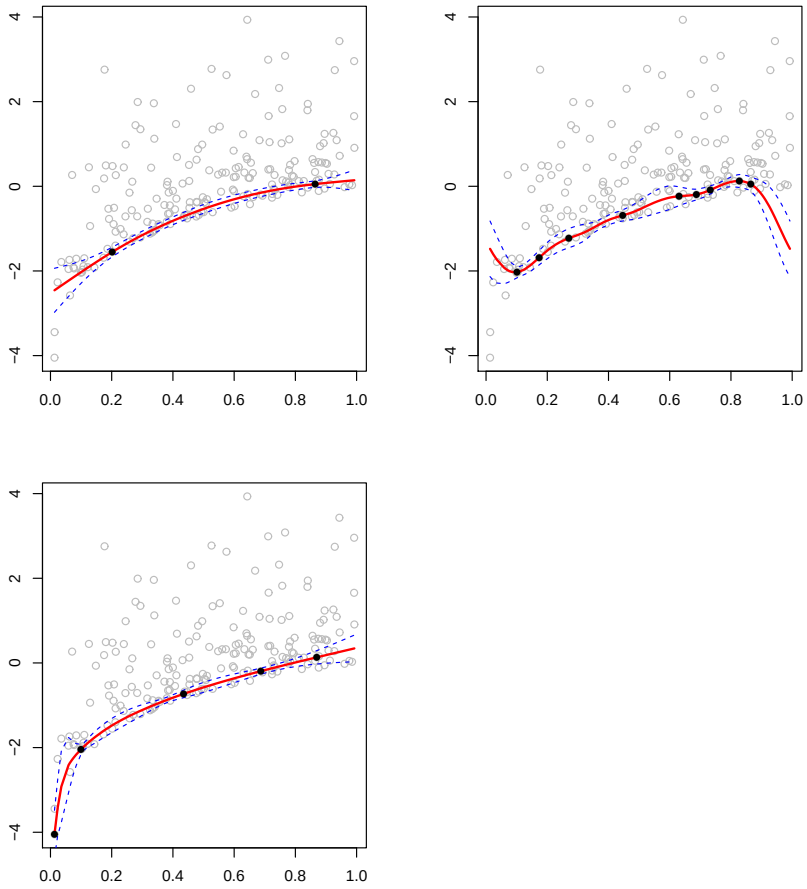


Fig. 8.5: The charts are three fits with different basis functions. The top left uses a P-spline (bs="ps"), the top right is a cyclic cubic regression spline (bs="cc"). The bottom left chart uses an adaptive spline (bs="ad"). The adaptive spline has 4 smoothing parameters. We see that the P-spline oversmooths the quantile and the cyclic spline does not fit the quantile correctly. The adaptive spline fit is close to the true quantile. Note that these charts are similar to charts that can be obtained using package 'mgcViz' Fasiolo and Nedellec [2020], Fasiolo et al. [2020a] together with package 'qgam' Fasiolo et al. [2020c].

We can continue with a 2d chart. Example 15 in function 'qgacvExamples' is generated as the sum of two univariate functions of two different variables (see section 9.1). Function 'plot2d' plots the fitted surface with the datapoints using package 'rgl'. Parameter 'transparency' is a number between 0 and 1 to indicate the transparency of the surface. Function 'plot2dSurf' adds a surface to the existing plot. We add the true quantile surface with transparency=0.3.

```
n<-200
tau<-0.05
ExampleNumber<-15
Ex2<-qgacvExamples(ExampleNumber,n,tauArr=tau,seed=1,SetSeedYesNo=T)
fit2d<-qam(y~s(x1,k=20)+s(x2,k=20),data=Ex2$data,tau=tau)
plot2d(fit2d,transparency=1)
plot2dSurf(Ex2$TrueQuantileSurf[[1]],transparency=0.3)
```

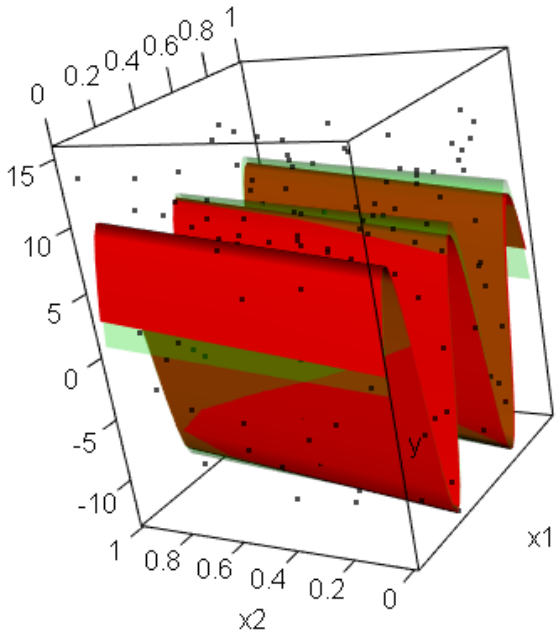


Fig. 8.6: On this figure, we can see the fit:
"qam(y s(x1,k=20)+s(x2,k=20),data=Ex2$data,tau=tau)"
with an instance of Example 15, $\tau = 0.05$, $n = 200$. The fitted model is the red surface. The true quantile surface is the green surface. We can see that given the relatively extreme quantile and the low number of datapoints, the fit is relatively close to the true quantile.

Another example with a Gaussian Process. This example is based on Wood [retrieved 2020a]. Example 16 (see section 9.1) is a 2d surface example. We are going to fit a low rank smooth Gaussian Process (GP) as in Kammann and Wand [2003]. The covariance function of the GP is $1 - \frac{1.5 \times d}{0.4} + 0.5(\frac{d}{0.4})^3$ if $d \leq 0.4$. We use 100 basis functions. With function 'plot2dCI', we plot the surface together

with the confidence interval based on the covariance matrix estimated using the np bootstrap samples calculated with the relaxed smoothing parameter:

```
n<-1000
tau<-0.8
ExampleNumber<-16
Ex3<-qgacvExamples(ExampleNumber,n,tauArr=tau,seed=1,SetSeedYesNo=T)
GPrangeParam<-0.4
fit<-qam(y~s(x1,x2,k=100,bs="gp",m=c(1,GPrangeParam)),data=Ex3$data,tau=tau)
plot2dCI(fit)
```

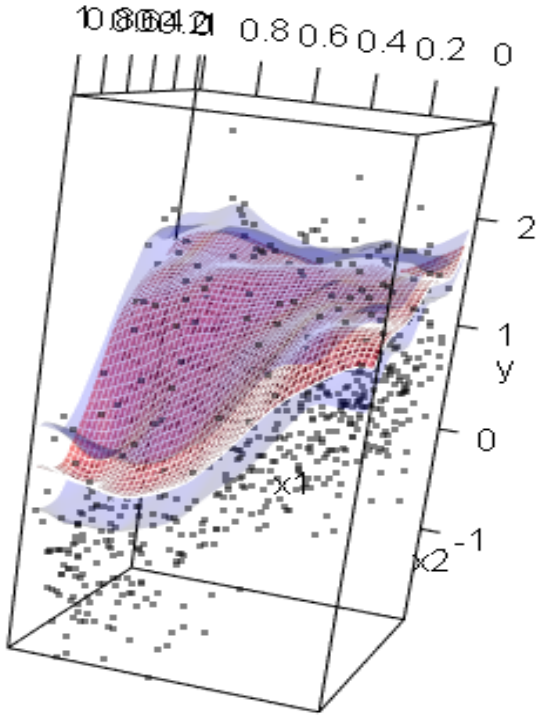


Fig. 8.7: On this example, we can see a fit to a draw of Example 16. The model is a 2 dimensional Low Rank Gaussian Process smooth with correlation function $1 - \frac{1.5\times d}{0.4} + 0.5(\frac{d}{0.4})^3$ if $d \leq 0.4$ and 0 otherwise (see description of in Wood [retrieved 2020a]). The fit is displayed with function qgacv::plot2dCI. We can see the fitted surface in red and the CI in purple. We use $n = 1000, \tau = 0.8$. Note this chart is similar to charts that can be obtained using package 'mgcViz' Fasiolo and Nedellec [2020], Fasiolo et al. [2020a] together with package 'qgam' Fasiolo et al. [2020c]

Finally, we can try a 5 (2+1+1+1+1) covariate example (Example 17, see section 9.1). Because we have one 2d smooth and three 1d smooths, we use function plot that is a pass-through to function 'plot.gam' in package 'mgcv':

```
n<-5000
tau<-0.7
ExampleNumber<-17
Ex4<-qgacvExamples(ExampleNumber,n,tauArr=tau,seed=1,SetSeedYesNo=T)
fit<-qam(y~s(x1,x2,k=40)+s(x3)+s(x4)+s(x5),data=Ex4$data,tau=tau)
plot(fit,pages=1,scale=0)
```

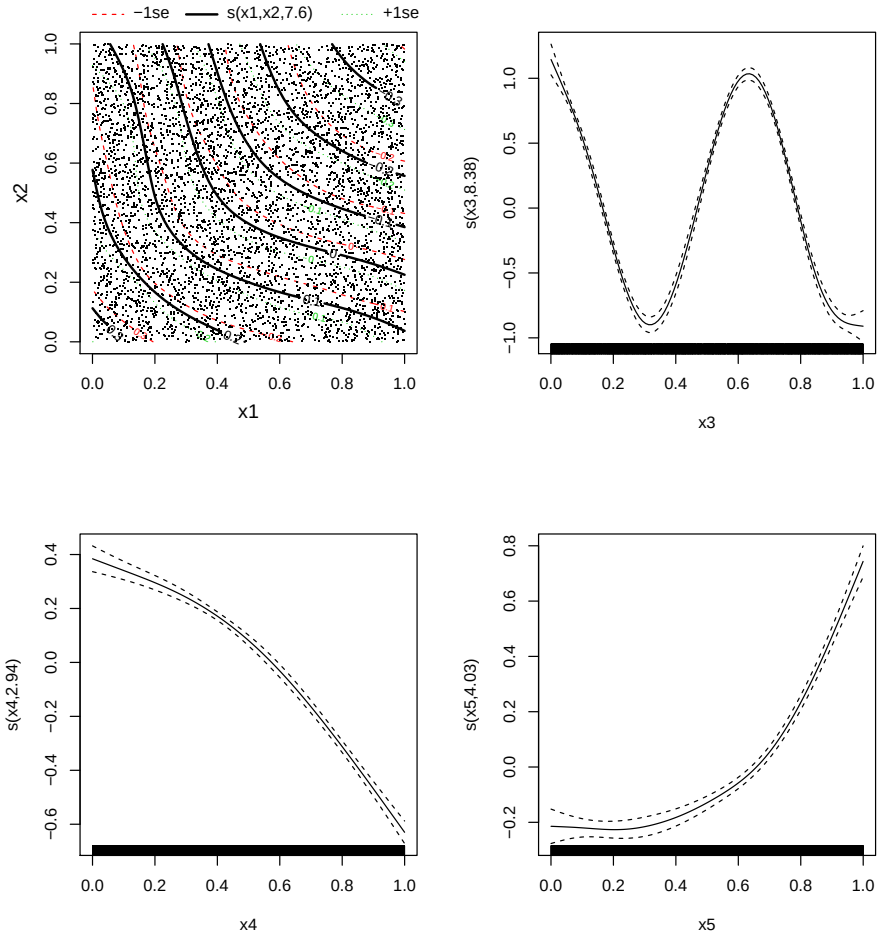


Fig. 8.8: This Figure shows the result of fit
qam(y s(x1,x2,k=40)+s(x3)+s(x4)+s(x5),data=Ex4$data,tau=tau)
using function plot. We see that the fit is close to the true quantiles.

9. COMPARISON AND CONFIDENCE INTERVAL COVERAGE

Our methodology and R package comes after a series of proposals that have recently been made to fit QAMs. Contrary to the methods introduced recently (Fasiolo et al. [2020b], Geraci [2019b], Muggeo et al. [2020]), our method is fully frequentist.

The Bayesian approach for additive (mixed) models is based on the remark that minimizing the pinball loss function is equivalent to assuming a misspecified asymmetric Laplace distributed error (Yu and Moyeed [2001]. The likelihood is then referred to as a working likelihood (see for example Wu and Narisetty [2020]). The Bayesian inference is then performed via Markov Chains(MCMC)/ Hamiltonian Monte Carlo (HMC) or Integrated Nested Laplace Approximation (INLA). Articles that discuss this approach include Yue and Rue [2011], Waldmann et al. [2013] and packages available include 'bamlss'/'R2bayesX'(Umlauf et al. [2019], Umlauf et al. [2018]) and 'brms'/'stan' (Bürkner [2017]).

Geraci [2019b] also uses Bayesian inference based on the asymmetric Laplace distribution (ALD) for quantile additive mixed models. A Laplace approximation is used to obtain a working marginal likelihood. Obtaining the marginal likelihood from the joint likelihood requires the estimation of an intractable integral. The smoothing parameters are then determined by maximizing the approximate marginal likelihood. The method is implemented in package 'aqmm' (Geraci [2019a]). The method developed in Geraci [2019b] has advantages, especially as it allows to fit quantile additive mixed whereas package 'qgacv' current does not allow the inclusion for random effects for quantile additive models. However, if used only using fixed effects, it has some drawbacks that are described in this section.

Fasiolo et al. [2020b] also use Bayesian modelling to estimate QAMs. However, to avoid using a mis-specified asymmetric Laplace likelihood, Fasiolo et al. use the updating belief distributions of Bissiri et al. [2016]. Unlike the traditional Bayesian update, the assumption of this framework is that the parameters and the prior are related via a loss function. This approach has been implemented in package 'qgam' Fasiolo et al. [2020c]. Although this method has many advantages, if also has some drawbacks that we described in this section.

Torretta [2016], Torretta et al. [2015] develop a method to estimate the

smoothing parameter of a Quantile Smoothing Spline that follows mixed modelling approach where the likelihood follows an asymmetric Laplace distribution and the random component is normally distributed with mean zero and a homoscedastic variance that is proportional to the pseudo-inverse of the smoothing penalty. The regularization parameter is then estimated using the Schall algorithm. Although Torretta [2016] focuses on the one dimensional case, this method could possibly be extended to an additive model setting. Note that the method was adapted to total variation penalized QAMs in Muggeo et al. [2020]. Our methodology is fully frequentist and does not make distributional assumptions. Compared to the methods proposed previously, our method has the following advantages:

- **Standard methodology**: As the asymmetric Laplace used for Bayesian and mixed methods is mis-specified, credible intervals may have poor coverage[1], especially for extreme quantiles. (see a study of coverage for credible intervals Waldmann [2018]). Because of this unreliability, to obtain good coverage, Geraci [2019b] must depart from the classical Bayesian framework, where credible intervals are obtained as part of the inference (confidence intervals are calculated using bootstrapping). Our method remains within the classical frequentist approach (see for example Efron and Hastie [2016]). We have one algorithm to estimate the vector of smoothing parameters and vector of parameters and another algorithm to estimate the confidence interval.

- **No other distributional assumptions:** Apart form the mis-specified Asymmetric Laplace distribution, some other distributional assumptions are sometimes made. For example, Shim et al. [2009] assumes a heteroscedastic non-parametric location scale model for the response. Our approach does not make such assumptions.

- **No reliance on the central quantile for the estimation of the smoothing parameters**: The smoothing parameters are determined locally where the quantile is. Our methodology relies on a mean GAM fit only to obtain a first estimate of the smoothing parameters used to obtain the surrogate loss parameter $\hat{\alpha}$ (see section 7.2). We then run the algorithm a second time at the correct quantile (see section 7.6). Everything else after this is determined at the correct quantile. Package 'aqmm' relies partly on the central quantile to estimate the smoothing parameters. If the central quantile degree of smoothness is the same as non-central quantiles, then the fits should work correctly. However, if the degrees of smoothness is different, the fit may not have the optimal degree of smoothness. An example of a dataset where the central quantile degree of smoothness is different from non-central quantiles along with fitted curves with 'qgacv:qam' and 'aqmm:aqmm' is shown in Appendix D.1.

[1] Note that although Sriram et al. [2013], Sriram and Ramamoorthi [2018] show consistency of the posterior under ALD priors under certain conditions, it is still a subject of debate and a correction is necessary to obtain correct credible intervals.

- **The method can be used with multimodal data:** Transformation methods (Box-Cox, Yeo-Johnson,...) are used in packages 'gamlss' (Rigby et al. [2013]) and 'vgam' (Yee [2008]) to fit centile/quantile regression curves. The transformation methods are fast and give good results as long as the data is unimodal. If the data is multimodal, the quantiles may not be recovered. Package 'qgam' Fasiolo et al. [2020c] also partially fails for multimodal data for a different reason. The fitting algorithm of 'qgam' starts from the determination of an "optimal" rounding. This optimal rounding relies on the calculation of residuals from a mean GAM fit. If for example the data has two modes with a large gap, then the optimal rounding becomes large, leading to a large bias for estimated quantiles close on each side of the gap. As our package does not make any distributional assumptions, multimodal data can be fitted. An example of multimodal data fit can be found in Appendix D.2.

- **Less biased estimation for extreme quantiles than package 'qgam'**: Package 'qgam' calculates an optimal rounding parameter and uses a distribution with a rounded peak for all parts of the algorithm. This can sometimes leads to a large bias in the estimate, for example for extreme quantiles. An example can be found in Appendix D.3. An extreme quantile is fit with 'qgacv' and 'qgam' and is compared to the linear quantile regression fit of 'quantreg'.

- **Robustness to extreme datapoints:** Fits using packages 'qgam' and 'aqmm' can sometimes be quite sensitive to extreme datapoints. Fits using 'qgacv' tends to be more robust to extreme points. An example for 'qgam' can be found on Figure D.4 (Appendix D.4). An example for 'aqmm' can be found on Figure D.5 (Appendix D.4). Note that this robustness of qgacv:qam comes with the disadvantage of sometimes oversmoothing for extreme quantiles resulting in sometimes poor coverage for confidence intervals for extreme quantiles (see section 6.2).

- **A fit that matches empirical quantiles**: We use the Generalized IRLS to estimate the vector of parameters $\hat{\boldsymbol{\beta}}$. We can then theoretically obtain an arbitrarily precise estimation of the empirical quantile model (assuming we estimated correctly the degree of smoothness of the quantile). We can test this, for example, with a linear quantile regression. If the underlying link between the observed variable and the predictors is linear and we estimate correctly the degrees of smoothness, we should then recover the linear quantile regression fit from package 'quantreg'. This may not be the case with other packages. On Figures D.2, D.3 and D.4, we show examples where the underlying generating model is linear and where the fit of package 'qgacv' is close to the fit obtained with a linear quantile regression fitted using package 'quantreg'.

- **Fit invariance beyond certain number of basis functions**: Kato [2011] proposed to use a Euclidean norm penalty instead of a ridge-type

penalty together with a method to determine the smoothing parameter. In a recent note, Koenker [2020c] gives an example where the fit based on the method developed by Kato [2011] changes significantly if the number of basis functions is increased. Our methodology is robust to the number of basis functions used for central quantiles and extreme quantiles. As long as the number of basis functions is sufficiently large, the fits are largely unaffected by an increase in the number of basis functions. We give an example of this relative invariance in Appendix D.5.

- **Fitting independently multiple smoothing parameters**: Reproducing Kernel Hilbert Space methods such as the one used in package 'cosso' Lin et al. [2013], or 'Infinite Task Learning' Brault et al. [2019] have one (or two) regularization parameters in total regardless of the dimension of the vector of predictors. The regularization parameters for each dimension is then determined adaptively. It can then prove difficult to fit models with several covariates were the fit smoothness in different directions are very different. An illustration of this problem can be found in Appendix D.6. We use two sinusoids with different periods. The fit with function 'cosso:cosso' overfits in one direction. Other articles/packages describing quantile regression in RKHS include Takeuchi et al. [2006], Li et al. [2007], Koo et al. [2013].

- **Reduced chance of overfitting central quantiles as compared to mean GAM**: As discussed in section 4.7, it has been observed that for mean GAM regression that Generalized Cross Validation was sometimes leading to overfits. Lukas [2010] indicates that this phenomenon occurs especially with a small number of observations and with correlated observations. Although not mentioned in the literature, it seems that the GACV applied to quantile regression as described in Yuan [2006], Reiss and Huang [2012] leads sometimes to such overfit. In Lukas et al. [2016], a rule consisting in multiplying the edf by a factor of 2 (for $n \geq 100$) to reduce the overfitting issue is discussed. In the QGACV formula derived, the for central quantiles ($\tau = 0.5$), the multiplier is identical. In Appendix D.7, we give an example where we generate 25 samples of a linear autoregressive model of order 2. We see that the quantile additive fit tends to overfit less than a mean gam fit.

In the next sections, we are going to compare package 'qgacv' with packages 'aqmm' and 'qgam'. Note that although the method developed by Torretta [2016], Torretta et al. [2015] does not seem to be available, the method of Muggeo et al. [2020] that is closely related is available as part of R package 'quantRegGrowth' (Muggeo [2021]). However, it has not been tested in this document as it seems that it was not available at the time of running the comparisons of this chapter. In section 9.1, we discuss the setting of our tests.

9.1 Settings for the tests

This section is focusing on testing and comparison of our methodology. To show the performance of our algorithm, we run three studies. These studies rely on 18 different data generating models. The first study consists in generating 50 different instances with each of the 18 models and fit quantile models to the data. The quantile models are then compared to the true quantiles and we plot the best out of 50, worst out of 50 and 25th out of 50 fits (section 9.2).

The second and third studies are comparison studies with other packages. In the second study, we fit the same generated data as in the first study at the same quantiles with three different packages: "qgam" (version 1.3.1), "aqmm" (version 1.0) and "qgacv". For each model, we fit 50 different samples generated with the model. Then for each package, each fit and each sample, we calculate the mean squared error compared to the true quantile. We then report average MSE across all samples.

The third study is a comparison of the confidence interval coverage between packages "qgacv" and "qgam" (package 'aqmm' version 1.0 was used to conduct the experiments. This version did not contain confidence intervals or credible intervals. Version 1.1 allows to estimate confidence intervals and could be included in future comparisons). Our three studies are based on a series of 1, 2, 3 and 5 dimensional models. These models are constructed the following way. First, we generate a series of covariates all independent that follow a uniform distribution:

$$\underline{x}_{1i}, ..., \underline{x}_{5i} \sim Uniform[0,1]$$

$$cov(\underline{x}_{ji}, \underline{x}_{ki}) = 0,\ j \neq k$$

The examples we take are then based on the following non-linear functions:

$$f_1(\underline{x}) = 0.2 \times \underline{x}^{11} \times (10 \times (1-\underline{x}))^6 + 10 \times (10 \times \underline{x})^4 \times (1-\underline{x})^{10}$$

$$f_2(\underline{x}) = 10 \times \sin(5 \times \pi \times \underline{x})$$

$$f_3(\underline{x}_1, \underline{x}_2) = cos(5 \times \underline{x}_1 \times \underline{x}_2^2)$$

$$f_4(\underline{x}) = cos^2(7 \times \underline{x}^2)$$

$$f_5(\underline{x}) = \sqrt{\underline{x}(1-\underline{x})} \sin\left(\frac{2\pi(1+2^{-\frac{7}{5}})}{\underline{x}+2^{-\frac{7}{5}}}\right)$$

Function f_1 can be found in Wood [2006]. f_5 can be found in Ruppert et al. [2003] p.298 and is used in Koenker [2011] and Muggeo et al. [2020]. And the following errors:

$\epsilon_{N,i} \sim Normal(\mu = 0, \sigma = 1)$.
$\epsilon_{A,i} \sim AlphaStable(\alpha_{\alpha Stable} = 1.5, \beta_{\alpha Stable} = 0, \gamma_{\alpha Stable} = 1, \delta_{\alpha Stable} =$

0).
$\epsilon_{B,i} \sim Binomial(p = 0.5)$.
$\epsilon_{G,i} \sim Gamma(shape = 1, rate = 1, scale = 1)$.
$\epsilon_{G_2,i} \sim Gamma(shape = 10, rate = 1, scale = 1)$.
$\epsilon_{N_{PS},x} \sim Normal(mean = 5 \times \sin(2\pi \underline{x}), sd = 0.5 + \exp(1.5 \times \sin(4\pi \underline{x})))$.

Function $\epsilon_{N_{PS},\underline{x}_i}$ is a Polson-Scott function as in Polson and Scott [2016] also described in Koenker [2020c]. The 13 One-dimensional examples used are:

- **Model 1**: $y_{1,i} = f_1(\underline{x}_{1,i}) + \epsilon_{N,i}$
- **Model 2:** $y_{2,i} = f_1(\underline{x}_{1,i}) + \epsilon_{A,i}$
- **Model 3**: $y_{3,i} = f_1(\underline{x}_{1,i}) + (\underline{x}_{1,i} + 1)^3 \epsilon_{N,i}$
- **Model 4**: $y_{4,i} = f_1(\underline{x}_{1,i}) + (\underline{x}_{1,i} + 1)^3 \epsilon_{A,i}$
- **Model 5**: $y_{5,i} = \underline{x}_{1,i} + \epsilon_{N,i}$
- **Model 6**: $y_{6,i} = f_2(\underline{x}_{1,i}) + 5 \times \underline{x}_{1,i} \times \epsilon_{N,i}$
- **Model 7**: $y_{7,i} = \underline{x}_{1,i} + \underline{x}_{1,i} \times \epsilon_{N,i}$
- **Model 8**: $y_{8,i} = 10^4 \times \underline{x}_{1,i}^2 \times (\underline{x}_{1,i} - 1/3)^2 \times (\underline{x}_{1,i} - 2/3)^2 \times (\underline{x}_{1,i} - 1)^2 \times \epsilon_{N,i}$
- **Model 9**: $y_{9,i} = \underline{x}_{1,i} + 10 \times (\epsilon_{B,i} - 0.5) + \epsilon_{N,i}$
- **Model 10**: $y_{10,i} = 3 \times sin(10 \times \underline{x}_{1,i}) + \epsilon_{N,i} + 10 \times \epsilon_{B,i}$
- **Model 11**: $y_{11,i} = 3 \times exp(3 \times \underline{x}_{1,i}) + \epsilon_{A,i}$
- **Model 12**: $y_{12,i} = \log(\underline{x}_{1,i}) + \epsilon_{G,i}$
- **Model 13**: $y_{13,i} = 100,000 \times (f_4(\underline{x}_i) + 2 \times \underline{x}_i^4) \times \epsilon_{G_2,i}$
- **Model 14**: $y_{14,i} = \epsilon_{N_{PS},x_{1,i}}$

Model 5 is also tested in Muggeo et al. [2020]. Model 12 is tested in Muggeo et al. [2020] with a different error. The 2 two-dimensional examples used:

- **Model 15**: $y_{15,i} = f_2(\underline{x}_{1,i}) + 2 \times \underline{x}_{2,i} + 2 \times \epsilon_{N,i}$
- **Model 16**: $y_{16,i} = f_3(\underline{x}_{1,i}, \underline{x}_{2,i}) + 0.5 \times \epsilon_{N,i}$

The three-dimensional example:

- **Model 17**: $y_{17,i} = f_3(\underline{x}_{1,i}, \underline{x}_{2,i}) + f_5(\underline{x}_{3,i}) + 0.5 \times \epsilon_{N,i}$

The five-dimensional example used:

- **Model 18**: $y_{18,i} = exp(-\underline{x}_{1,i} \times \underline{x}_{2,i}) + cos(\underline{x}_{3,i}) - \underline{x}_{4,i}^2 + (\underline{x}_{5,i}^2) + 0.5 \times \epsilon_{N,i}$

Figure 9.1 shows one instance of data generated using Model 1 with a QAM fitted at $\tau = 0.2$. Figure 9.2 shows a 3 covariate fit for $\tau = 0.5$ with a sample generated using Model 17. The simulations in this chapter were run on a cluster of computers. The simulation is split in 2,700 jobs ($50 \times 3 \times 18$)(JobNumber=1,...,2700) that are run by the cluster. The seed used is $(JobNumber + ModelNumber \times (10^5))$.

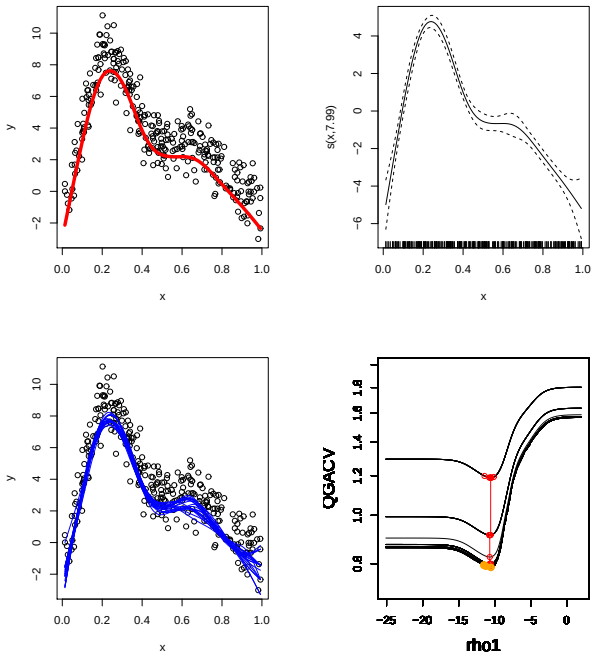


Fig. 9.1: This Figure shows a generated example using Model 1. The top left chart shows the generated data along with a fitted QAM at $\tau = 0.2$ using our methodology. The top right chart is a plot of the smooth with its confidence interval obtained using the method described in chapter 5. The bottom left Figure shows the 50 non-parametric quantile regression curves fitted after smoothing parameter relaxation on the non-parametric bootstrap samples used to obtain the confidence interval. The bottom right Figure shows the optimization path. The top curve is the QGACV curve as a function of the single smoothing parameter ρ_1 with rounding level $\alpha_{Start} = \hat{\alpha}$. The starting point of our algorithm happens to be close to the minimum of the criterion. Each successive curve represents each QGACV curve as a function of ρ_1 at successive lower levels of rounding α. The red curve is the optimization path, using Generalized Fellner-Schall. The orange path shows the path from the minimum of the QGACV to the relaxed smoothing parameter

.

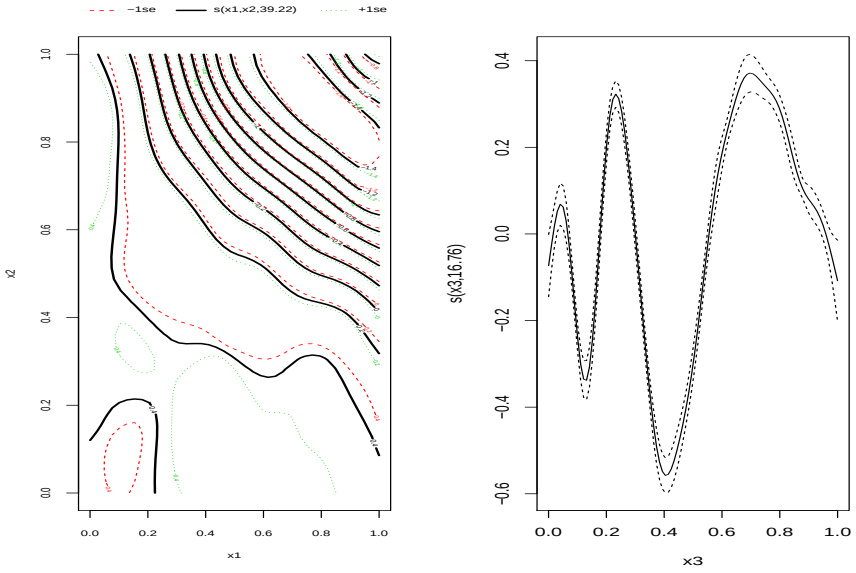


Fig. 9.2: An example of generated data using Model 17, n=20,000, 60 basis function for the surface, 20 basis functions for the univariate function. We fit a quantile additive model with $\tau = 0.5$.

9.2 Test 1: Comparing model fits to True Quantiles

The first test for our methodology consists in simulating examples as described in section 9.1. For each of the 18 models, and for a given number of datapoints ($n = 250,\ 500,\ 5,000$), we generate 50 samples from the model. Then for each quantile parameter value (for example ($\tau = 0.05, 0.5, 0.95$)) we fit a QAM. As the number of observations increases, the quantile parameters τ are changed to more extreme quantiles. For example, for Model 1, $n = 250$, we use $\tau = 0.25, 0.5, 0.75$. For $n = 500$, we fit more extreme quantiles: $\tau = 0.2, 0.5, 0.8$ and for $n = 2,500$, even more extreme quantiles: $\tau = 0.05, 0.5, 0.95$. We rank the fits based on the Mean Squared Error (MSE) compared to the true quantile. We then plot the true quantile in green, the best fit out of 50 in blue, the worst fit out of 50 in red. For the 1 dimensional models (Models 1 to 14). we also plot the median fit (25th out of 50 fit). Model 1 to Model 14 (1 dimensional models) are shown on Figures 9.3 to 9.16. The two 2d examples are shown on Figures 9.17 and 9.18. A 3d (2d+1d) example is shown on Figures 9.19, 9.20 and 9.21. A 5d example (2d+1d+1d+1d) is show on Figure 9.22, 9.23 and 9.24.

We note that as we increase the number of observations, even as we fit more extreme quantiles, the best and worst fits out of 50 are closer to the true quantiles. The example we tested in 1d include homoscedastic non-linear data with

Gaussian noise (Model 1), homoscedastic linear data with Gaussian noise (Model 5), homoscedastic non-linear data with alpha-stable noise (Model 2, Model 11), non linear data with heteroscedastic Gaussian noise (Model 3, Model 6, Model 7, Model 8), Bimodal data (Model 9, Model 10), non-linear model with data with homoscedastic Gamma noise (Model 12), heteroscedastic model with gamma noise with very large values (Model 13), heteroscedastic Gaussian noise (Model 14).

Multivariate models tested include the sum of two univariate functions with homoscedastic Gaussian noise (Model 15), a bivariate smooth with Gaussian noise (Model 16), the sum of a bivariate smooth and univariate smooth with homoscedastic Gaussian noise (Model 17) and the sum of a bivariate smooth and 3 univariate smooth with homoscedastic Gaussian noise (Model 18). Similarly to univariate models, as the number of datapoints increase, the best and worst fits our of 50 are closer to the true quantiles.

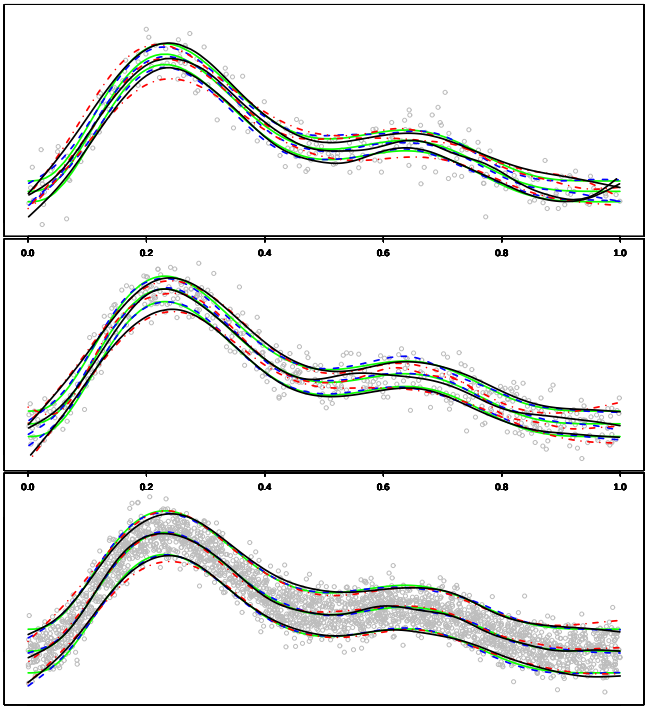


Fig. 9.3: Model 1: These charts show an example of fit for $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.25, 0.5, 0.75}$, $\mathbf{n = 250}$ (top chart), $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.2, 0.5, 0.8}$, $\mathbf{n = 500}$ (middle chart) $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.05, 0.5, 0.95}$ for $\mathbf{n = 2, 500}$ (bottom chart). Fellner Schall is used to optimize the QGACV and Generalized IRLS/Newton Raphson is used to fit the loss function. As the data is simulated, we know the true quantiles. We can then rank the fits in terms of mean squared error compared to the true quantiles. The green curves are the true quantiles, the blue curves the best fits, the black curves the 25th and the red curves the worst fit out of 50 different fits (we randomly generate data based on the same model).

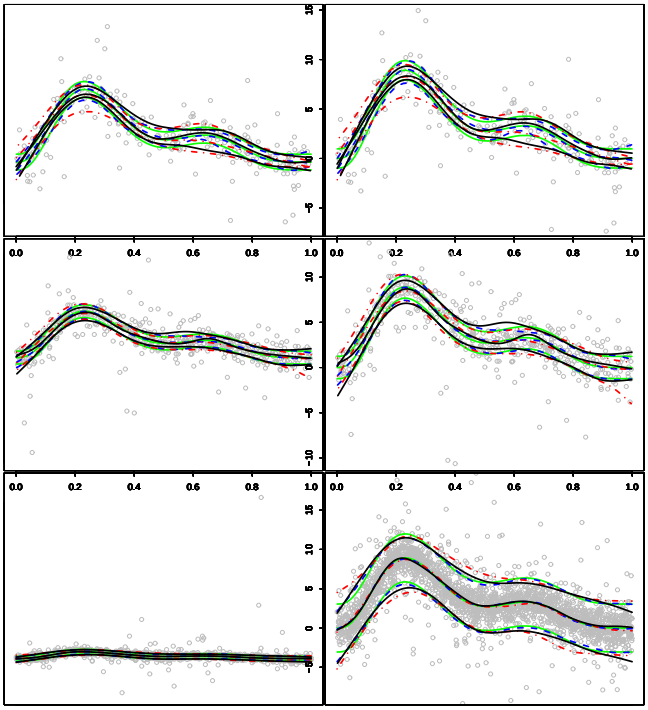


Fig. 9.4: Model 2: These charts show an example of fit for $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.25, 0.5, 0.75}$, $\mathbf{n = 250}$ (top chart), $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.2, 0.5, 0.8}$, $\mathbf{n = 500}$ (middle chart) $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.05, 0.5, 0.95}$ for $\mathbf{n = 2, 500}$ (bottom chart). Fellner Schall is used to optimize the QGACV and Generalized IRLS/Newton Raphson is used to fit the loss function. As the data is simulated, we know the true quantiles. We can then rank the fits in terms of mean squared error compared to the true quantiles. The green curves are the true quantiles, the blue curves the best fits, the black curves the 25th and the red curves the worst fit out of 50 different fits (we randomly generate data based on the same model). The right hand side charts are a zoom on the charts of the left hand side.

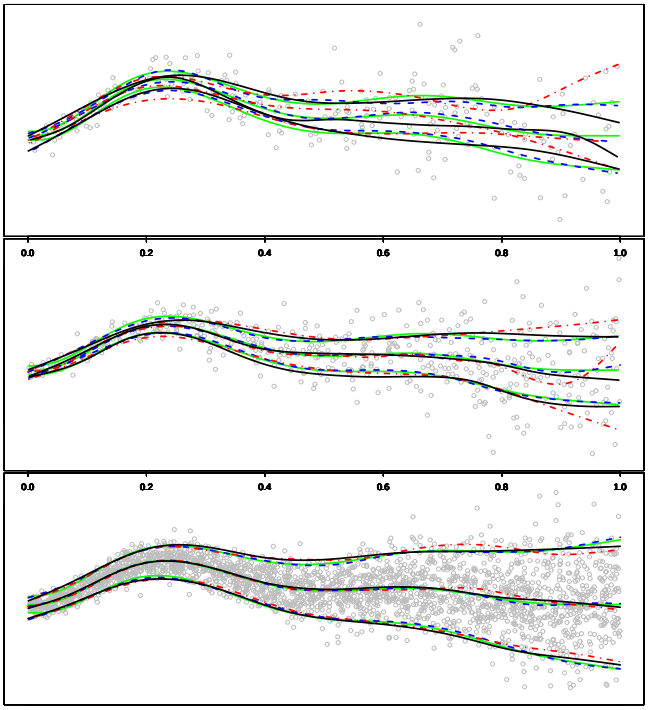


Fig. 9.5: Model 3: These charts show an example of fit for $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.25, 0.5, 0.75}$, $\mathbf{n = 250}$ (top chart), $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.2, 0.5, 0.8}$, $\mathbf{n = 500}$ (middle chart) $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.05, 0.5, 0.95}$ for $\mathbf{n = 2, 500}$ (bottom chart). Fellner Schall is used to optimize the QGACV and Generalized IRLS/Newton Raphson is used to fit the loss function. As the data is simulated, we know the true quantiles. We can then rank the fits in terms of mean squared error compared to the true quantiles. The green curves are the true quantiles, the blue curves the best fits, the black curves the 25th and the red curves the worst fit out of 50 different fits (we randomly generate data based on the same model).

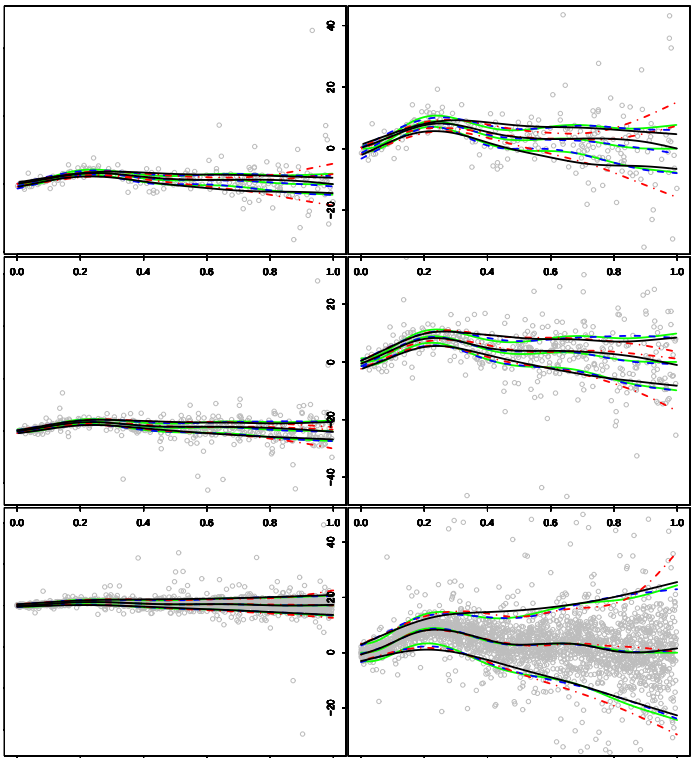


Fig. 9.6: Model 4: These charts show an example of fit for $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.25, 0.5, 0.75}$, $\mathbf{n = 250}$ (top chart), $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.2, 0.5, 0.8}$, $\mathbf{n = 500}$ (middle chart) $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.05, 0.5, 0.95}$ for $\mathbf{n = 2,500}$ (bottom chart). Fellner Schall is used to optimize the QGACV and Generalized IRLS/Newton Raphson is used to fit the loss function. As the data is simulated, we know the true quantiles. We can then rank the fits in terms of mean squared error compared to the true quantiles. The green curves are the true quantiles, the blue curves the best fits, the black curves the 25th and the red curves the worst fit out of 50 different fits (we randomly generate data based on the same model). The right hand side charts are a zoom on the charts of the left hand side.

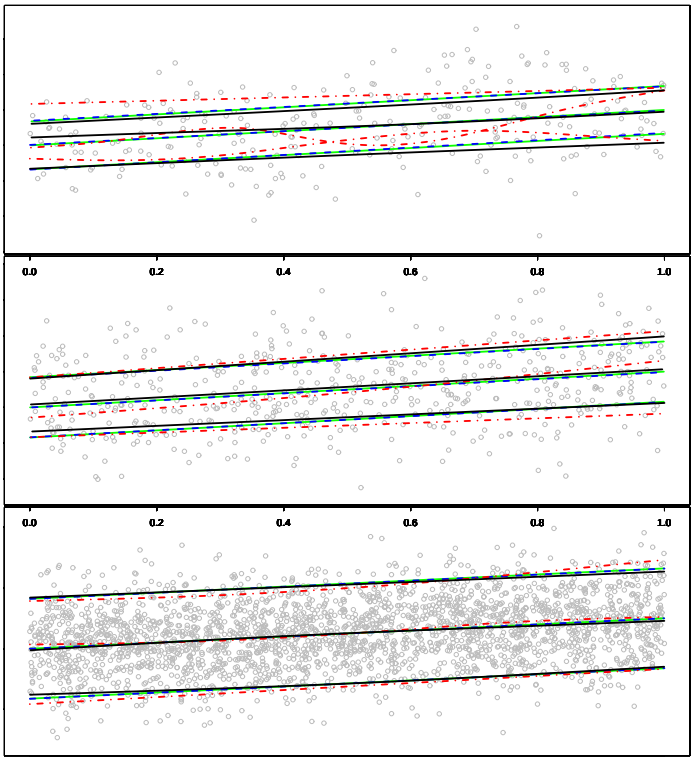


Fig. 9.7: Model 5: These charts show an example of fit for $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.25, 0.5, 0.75}$, $\mathbf{n = 250}$ (top chart), $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.2, 0.5, 0.8}$, $\mathbf{n = 500}$ (middle chart) $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.05, 0.5, 0.95}$ for $\mathbf{n = 2,500}$ (bottom chart). Fellner Schall is used to optimize the QGACV and Generalized IRLS/Newton Raphson is used to fit the loss function. As the data is simulated, we know the true quantiles. We can then rank the fits in terms of mean squared error compared to the true quantiles. The green curves are the true quantiles, the blue curves the best fits, the black curves the 25th and the red curves the worst fit out of 50 different fits (we randomly generate data based on the same model).

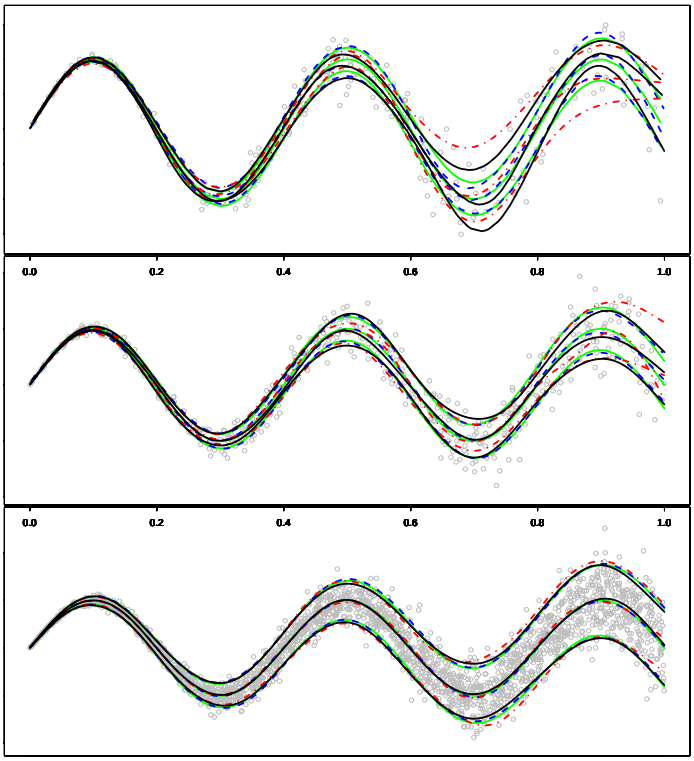


Fig. 9.8: Model 6: These charts show an example of fit for $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.25, 0.5, 0.75}$, $\mathbf{n = 250}$ (top chart), $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.2, 0.5, 0.8}$, $\mathbf{n = 500}$ (middle chart) $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.05, 0.5, 0.95}$ for $\mathbf{n = 2,500}$ (bottom chart). Fellner Schall is used to optimize the QGACV and Generalized IRLS/Newton Raphson is used to fit the loss function. As the data is simulated, we know the true quantiles. We can then rank the fits in terms of mean squared error compared to the true quantiles. The green curves are the true quantiles, the blue curves the best fits, the black curves the 25th and the red curves the worst fit out of 50 different fits (we randomly generate data based on the same model).

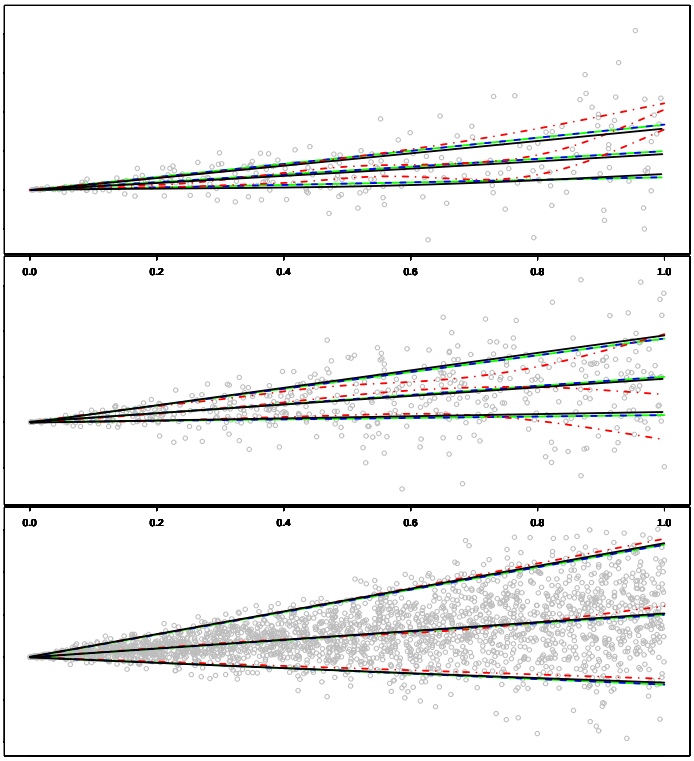


Fig. 9.9: Model 7: These charts show an example of fit for $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.25, 0.5, 0.75}$, $\mathbf{n = 250}$ (top chart), $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.2, 0.5, 0.8}$, $\mathbf{n = 500}$ (middle chart) $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.05, 0.5, 0.95}$ for $\mathbf{n = 2,500}$ (bottom chart). Fellner Schall is used to optimize the QGACV and Generalized IRLS/Newton Raphson is used to fit the loss function. As the data is simulated, we know the true quantiles. We can then rank the fits in terms of mean squared error compared to the true quantiles. The green curves are the true quantiles, the blue curves the best fits, the black curves the 25th and the red curves the worst fit out of 50 different fits (we randomly generate data based on the same model).

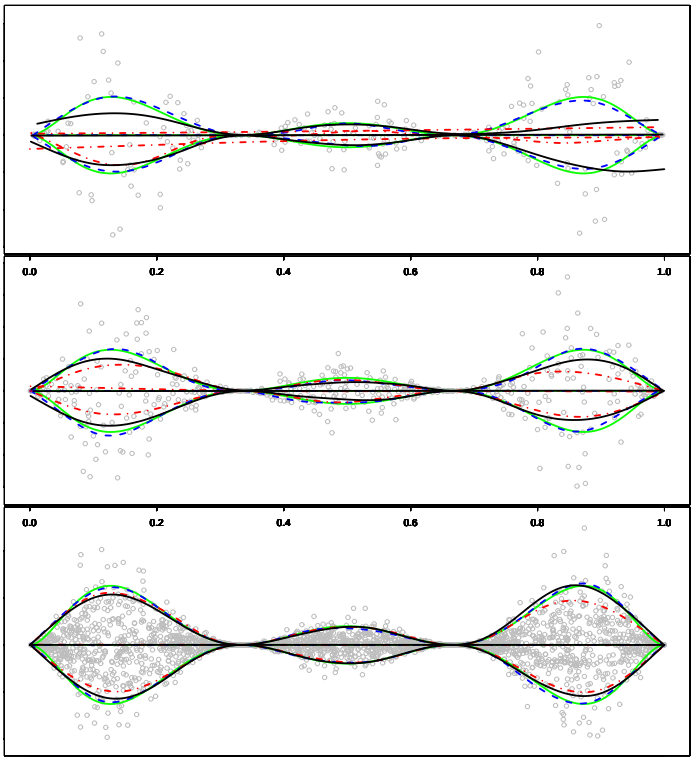


Fig. 9.10: Model 8: These charts show an example of fit for $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.25, 0.5, 0.75}$, $\mathbf{n = 250}$ (top chart), $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.2, 0.5, 0.8}$, $\mathbf{n = 500}$ (middle chart) $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.05, 0.5, 0.95}$ for $\mathbf{n = 2,500}$ (bottom chart). Fellner Schall is used to optimize the QGACV and Generalized IRLS/Newton Raphson is used to fit the loss function. As the data is simulated, we know the true quantiles. We can then rank the fits in terms of mean squared error compared to the true quantiles. The green curves are the true quantiles, the blue curves the best fits, the black curves the 25th and the red curves the worst fit out of 50 different fits (we randomly generate data based on the same model).

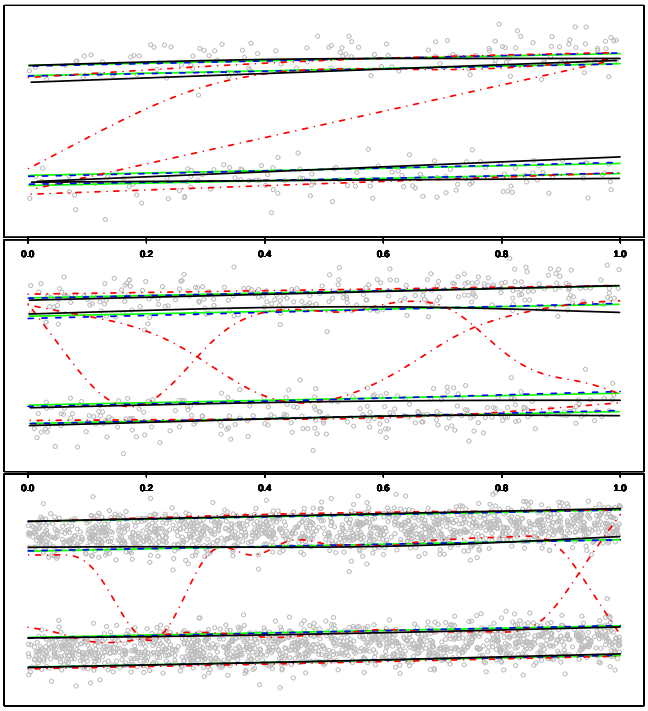


Fig. 9.11: Model 9: These charts show an example of fit for $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.25, 0.4, 0.6, 0.75}$, $\mathbf{n = 250}$ (top chart), $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.2, 0.45, 0.55, 0.8}$, $\mathbf{n = 500}$ (middle chart), $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.05, 0.45, 0.55, 0.95}$ for $\mathbf{n = 2,500}$ (bottom chart). Fellner Schall is used to optimize the QGACV and Generalized IRLS/Newton Raphson is used to fit the loss function. As the data is simulated, we know the true quantiles. We can then rank the fits in terms of mean squared error compared to the true quantiles. The green curves are the true quantiles, the blue curves the best fits, the black curves the 25th and the red curves the worst fit out of 50 different fits (we randomly generate data based on the same model).

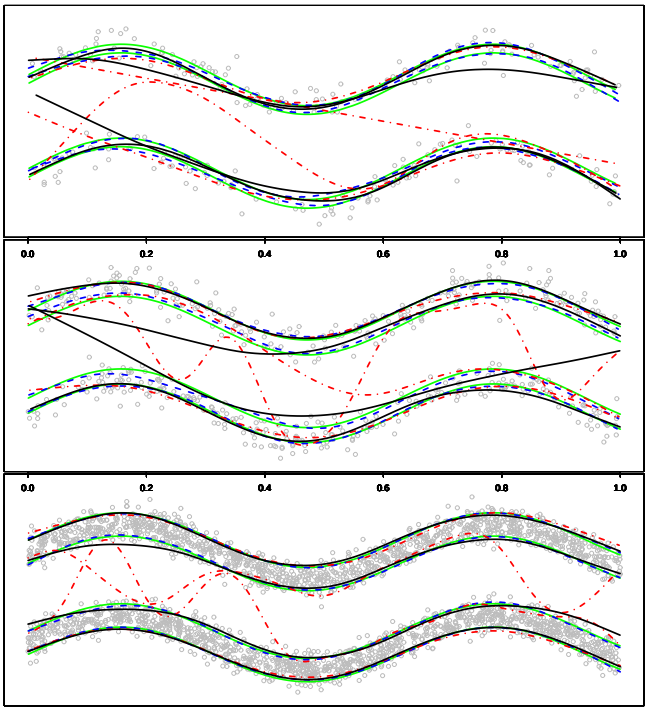


Fig. 9.12: Model 10: These charts show an example of fit for $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.25, 0.4, 0.6, 0.75}$, $\mathbf{n = 250}$ (top chart), $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.2, 0.45, 0.55, 0.8}$, $\mathbf{n = 500}$ (middle chart), $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.05, 0.45, 0.55, 0.95}$ for $\mathbf{n = 2,500}$ (bottom chart). Fellner Schall is used to optimize the QGACV and Generalized IRLS/Newton Raphson is used to fit the loss function. As the data is simulated, we know the true quantiles. We can then rank the fits in terms of mean squared error compared to the true quantiles. The green curves are the true quantiles, the blue curves the best fits, the black curves the 25th and the red curves the worst fit out of 50 different fits (we randomly generate data based on the same model).

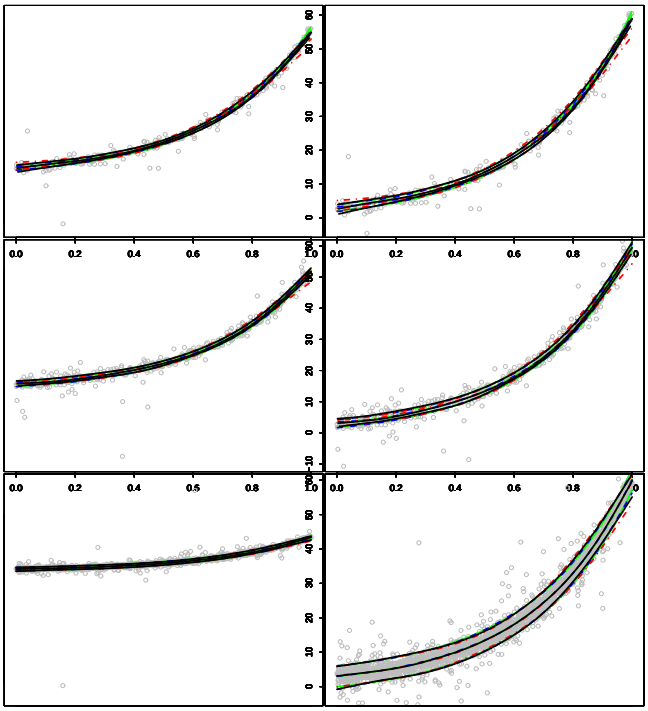


Fig. 9.13: Model 11: These charts show an example of fit for $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.25, 0.5, 0.75}$, $\mathbf{n = 250}$ (top chart), $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.2, 0.5, 0.8}$, $\mathbf{n = 500}$ (middle chart) $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.05, 0.5, 0.95}$ for $\mathbf{n = 2,500}$ (bottom chart). Fellner Schall is used to optimize the QGACV and Generalized IRLS/Newton Raphson is used to fit the loss function. As the data is simulated, we know the true quantiles. We can then rank the fits in terms of mean squared error compared to the true quantiles. The green curves are the true quantiles, the blue curves the best fits, the black curves the 25th and the red curves the worst fit out of 50 different fits (we randomly generate data based on the same model).

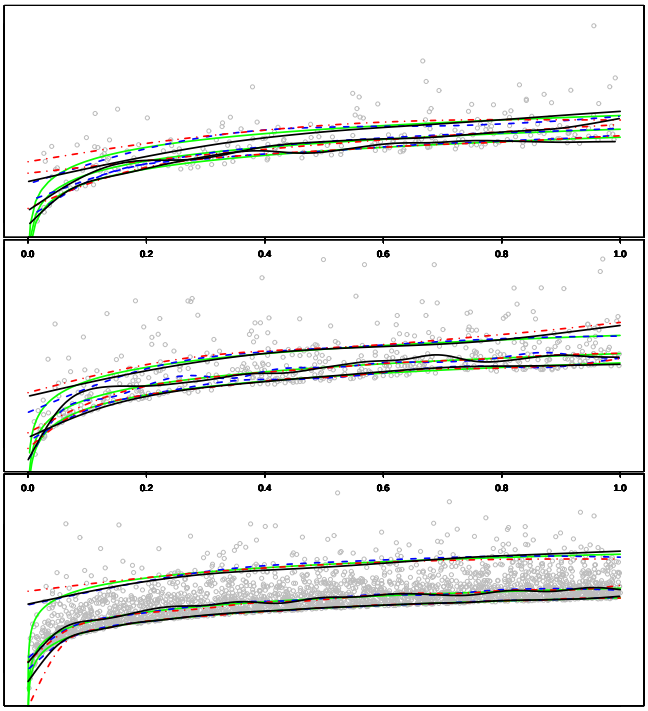


Fig. 9.14: Model 12: These charts show an example of fit for $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.25, 0.5, 0.75}$, $\mathbf{n = 250}$ (top chart), $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.2, 0.5, 0.8}$, $\mathbf{n = 500}$ (middle chart) $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.05, 0.5, 0.95}$ for $\mathbf{n = 2,500}$ (bottom chart). Fellner Schall is used to optimize the QGACV and Generalized IRLS/Newton Raphson is used to fit the loss function. As the data is simulated, we know the true quantiles. We can then rank the fits in terms of mean squared error compared to the true quantiles. The green curves are the true quantiles, the blue curves the best fits, the black curves the 25th and the red curves the worst fit out of 50 different fits (we randomly generate data based on the same model).

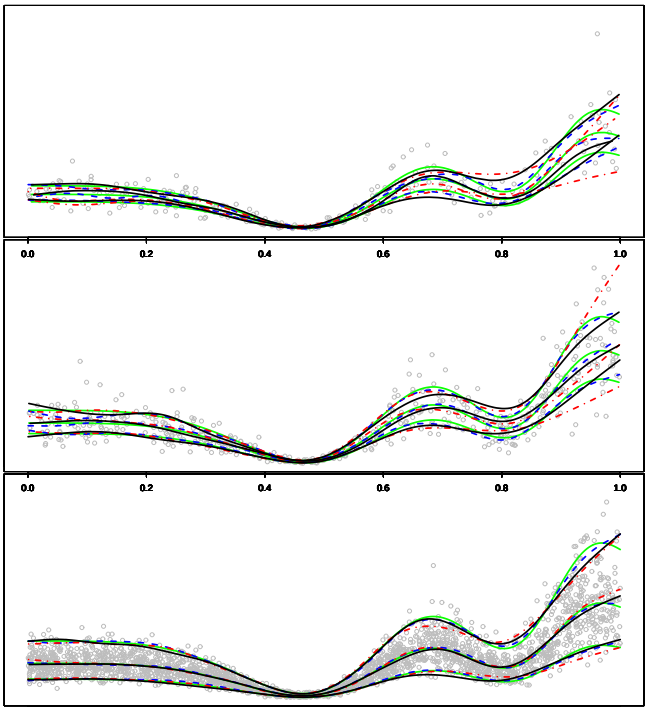


Fig. 9.15: Model 13: These charts show an example of fit for $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.25, 0.5, 0.75}$, $\mathbf{n = 250}$ (top chart), $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.2, 0.5, 0.8}$, $\mathbf{n = 500}$ (middle chart) $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.05, 0.5, 0.95}$ for $\mathbf{n = 2,500}$ (bottom chart). Fellner Schall is used to optimize the QGACV and Generalized IRLS/Newton Raphson is used to fit the loss function. As the data is simulated, we know the true quantiles. We can then rank the fits in terms of mean squared error compared to the true quantiles. The green curves are the true quantiles, the blue curves the best fits, the black curves the 25th and the red curves the worst fit out of 50 different fits (we randomly generate data based on the same model).

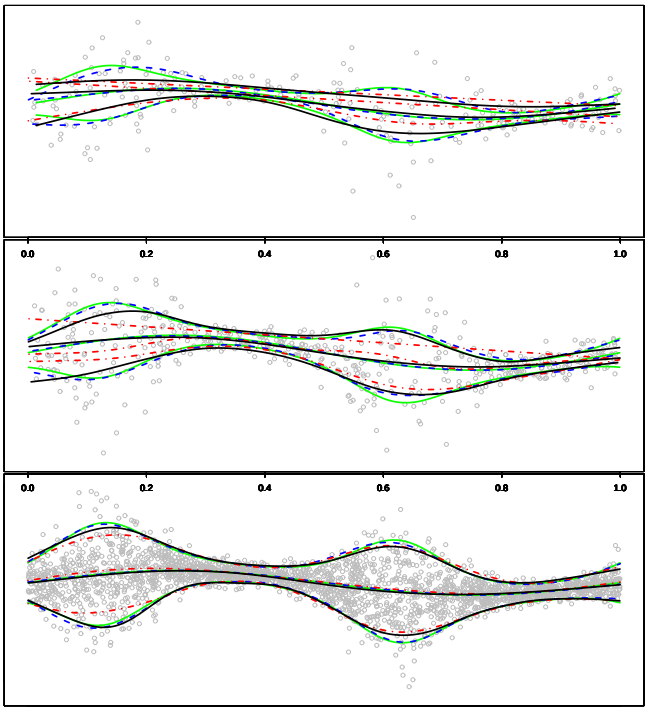
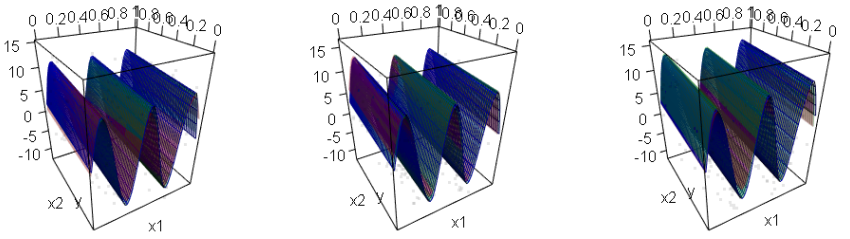
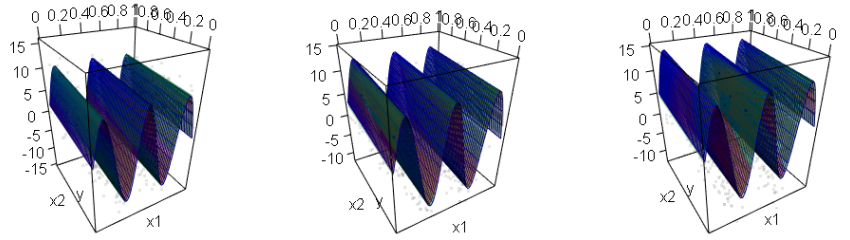


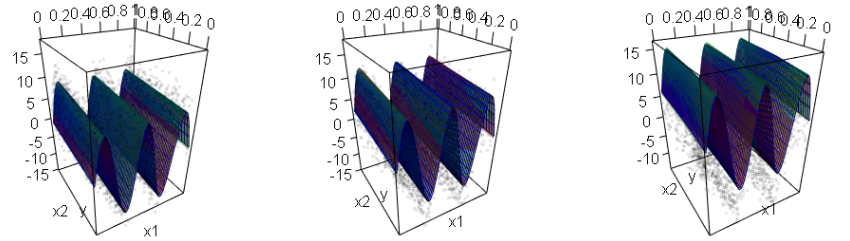
Fig. 9.16: Model 14: These charts show an example of fit for $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.25, 0.5, 0.75}$, $\mathbf{n = 250}$ (top chart), $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.2, 0.5, 0.8}$, $\mathbf{n = 500}$ (middle chart) $\boldsymbol{\tau_L, \tau_M, \tau_U} = \mathbf{0.05, 0.5, 0.95}$ for $\mathbf{n = 2,500}$ (bottom chart). Fellner Schall is used to optimize the QGACV and Generalized IRLS/Newton Raphson is used to fit the loss function. As the data is simulated, we know the true quantiles. We can then rank the fits in terms of mean squared error compared to the true quantiles. The green curves are the true quantiles, the blue curves the best fits, the black curves the 25th and the red curves the worst fit out of 50 different fits (we randomly generate data based on the same model).



(a) n=250, tau=0.25 *(b)* n=250, tau=0.5 *(c)* n=250, tau=0.75

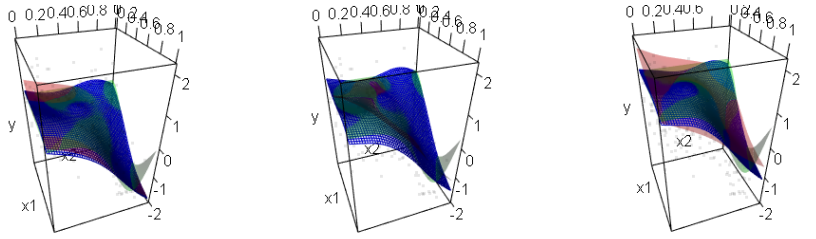


(d) n=500, tau=0.2 *(e)* n=500, tau=0.5 *(f)* n=500, tau=0.8

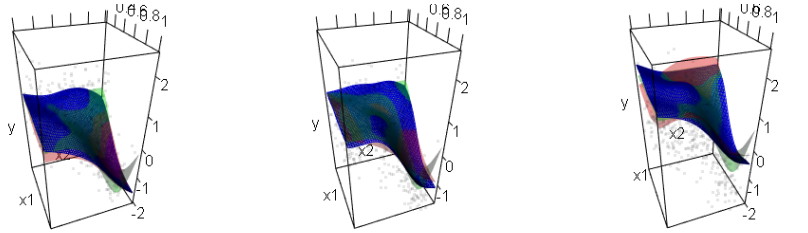


(g) n=2500, tau=0.05 *(h)* n=2500, tau=0.5 *(i)* n=2500, tau=0.95

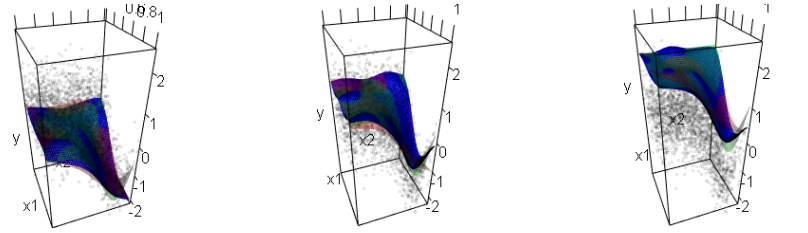
Fig. 9.17: Model 15: These charts show fit examples for $\boldsymbol{\tau_L, \tau_M, \tau_U = 0.25, 0.5, 0.75}$, $\mathbf{n = 250}$ (top chart), $\boldsymbol{\tau_L, \tau_M, \tau_U = 0.2, 0.5, 0.8}$, $\mathbf{n = 500}$ (middle chart) $\boldsymbol{\tau_L, \tau_M, \tau_U = 0.05, 0.5, 0.95}$ for $\mathbf{n = 2,500}$ (bottom chart). Fellner Schall is used to optimize the QGACV and Generalized IRLS/Newton Raphson is used to fit the loss function. As the data is simulated, we know the true quantiles. We can then rank the fits in terms of mean squared error compared to the true quantiles. The green surfaces are the true quantiles, the blue surfaces the best fits, and the red surfaces the worst fit out of 50 different fits (we randomly generate data based on the same model). We note that the worst and best fit are very close to the true quantile surface.



(a) n=250, tau=0.25 *(b)* n=250, tau=0.5 *(c)* n=250, tau=0.75

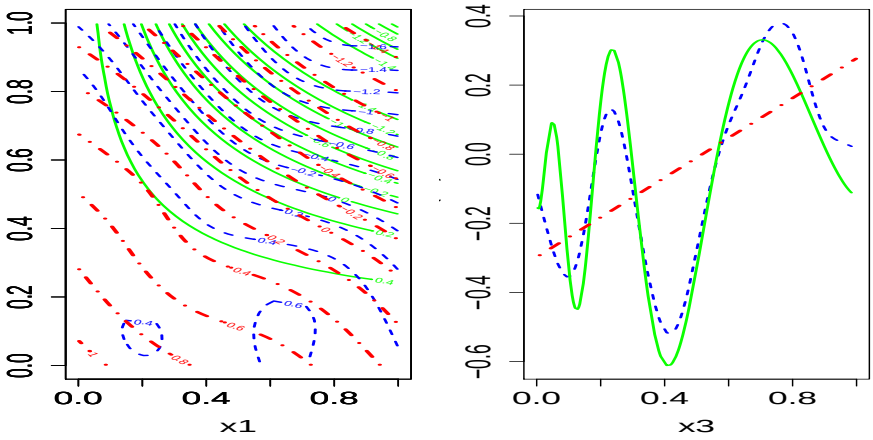


(d) n=500, tau=0.2 *(e)* n=500, tau=0.5 *(f)* n=500, tau=0.8

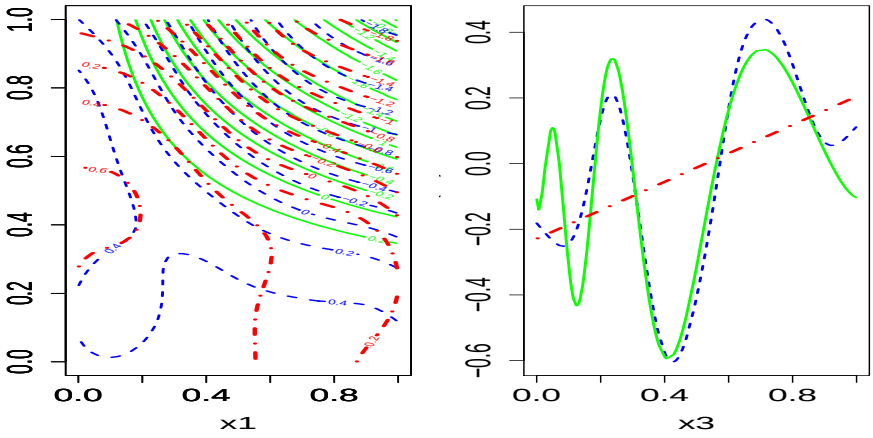


(g) n=2500, tau=0.05 *(h)* n=2500, tau=0.5 *(i)* n=2500, tau=0.95

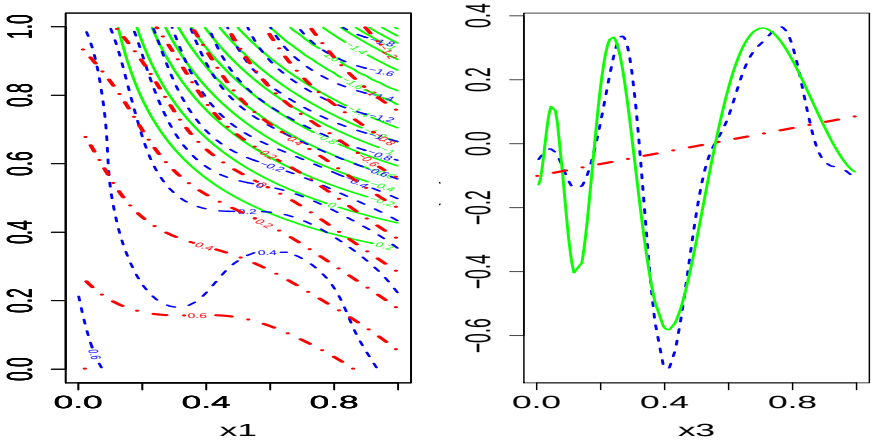
Fig. 9.18: Model 16: These charts show fit examples for $\boldsymbol{\tau_L, \tau_M, \tau_U = 0.25, 0.5, 0.75}$, $\mathbf{n = 250}$ (top chart), $\boldsymbol{\tau_L, \tau_M, \tau_U = 0.2, 0.5, 0.8}$, $\mathbf{n = 500}$ (middle chart) $\boldsymbol{\tau_L, \tau_M, \tau_U = 0.05, 0.5, 0.95}$ for $\mathbf{n = 2,500}$ (bottom chart). Fellner Schall is used to optimize the QGACV and Generalized IRLS/Newton Raphson is used to fit the loss function. As the data is simulated, we know the true quantiles. We can then rank the fits in terms of mean squared error compared to the true quantiles. The green surfaces are the true quantiles, the blue surfaces the best fits, and the red surfaces the worst fit out of 50 different fits (we randomly generate data based on the same model).We note that for n=250, $\tau = 0.75, 0.25$, and n=500, $\tau = 0.2, 0.8$ the worst fit out of 50 does not match the true quantile surface. For other fits, the worst surface out of 50 is quite close to the true quantile surface.



(a) n=250, tau=0.25

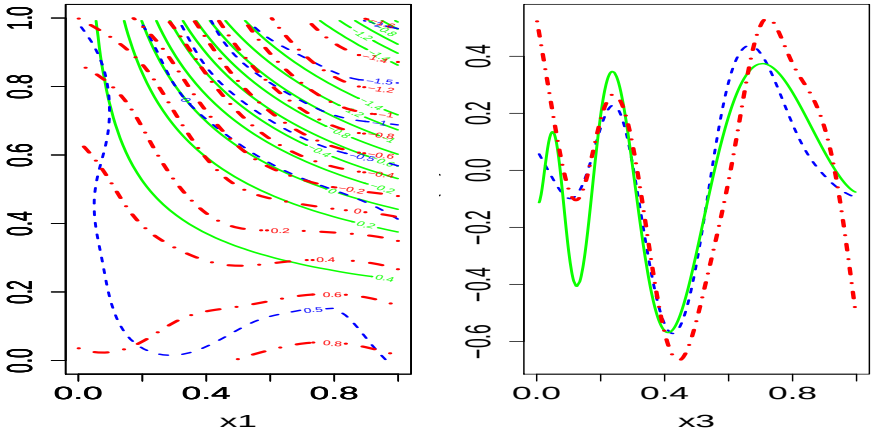


(b) n=250, tau=0.5

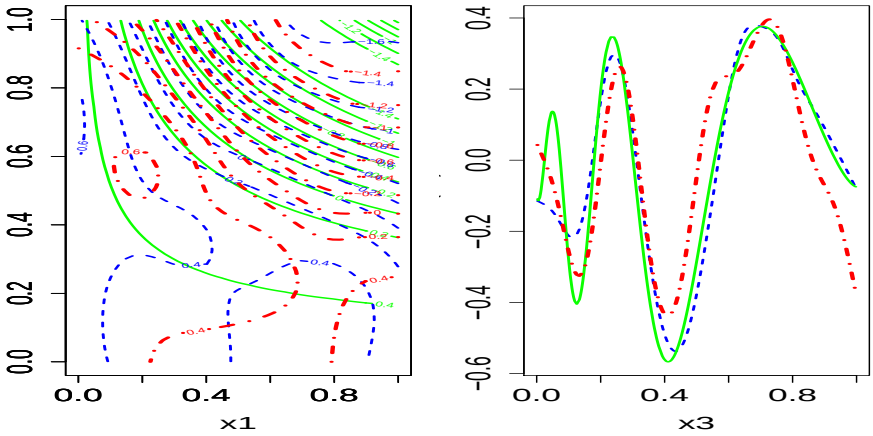


(c) n=250, tau=0.75

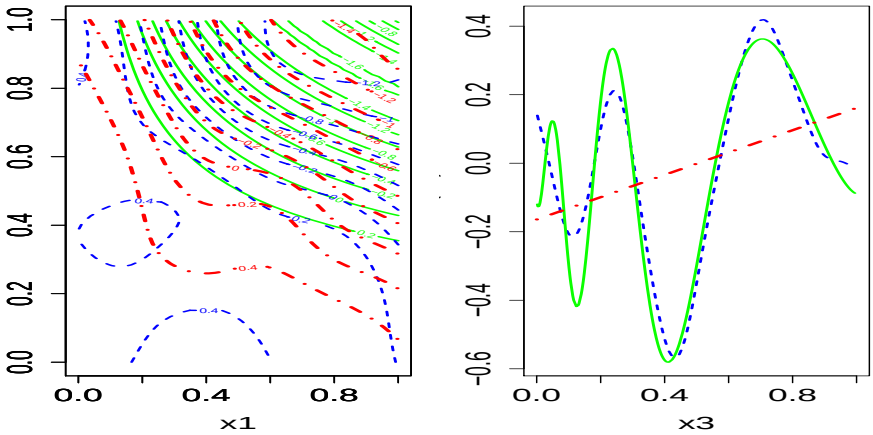
Fig. 9.19: Model 17: On this Figure, we present the results for n=250, tau=0.25, 0.5, 0.75. Each line shows the bivariate smooth of Model 17 represented as a contour map and the univariate smooth. The green lines show the true quantile for the smooth. The red line, the worst fit out of 50, the blue line, the best fit out of 50. results for n=500 and 2500 can be found on Figures 9.20 and 9.21, respectively.



(a) n=500, tau=0.20

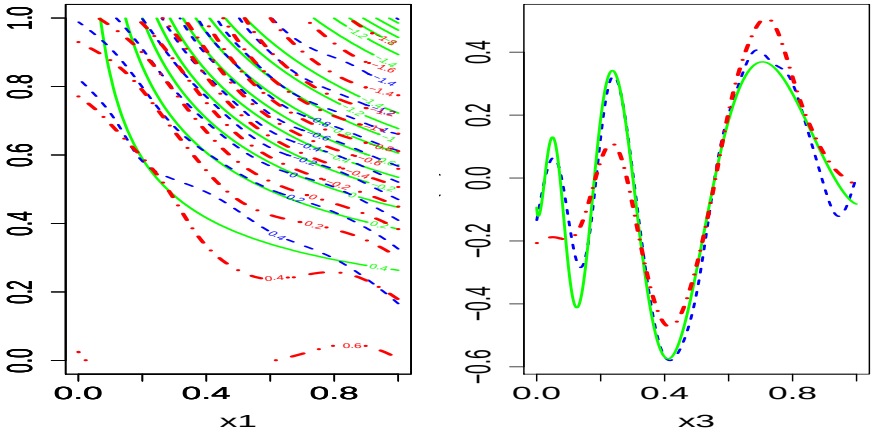


(b) n=500, tau=0.5

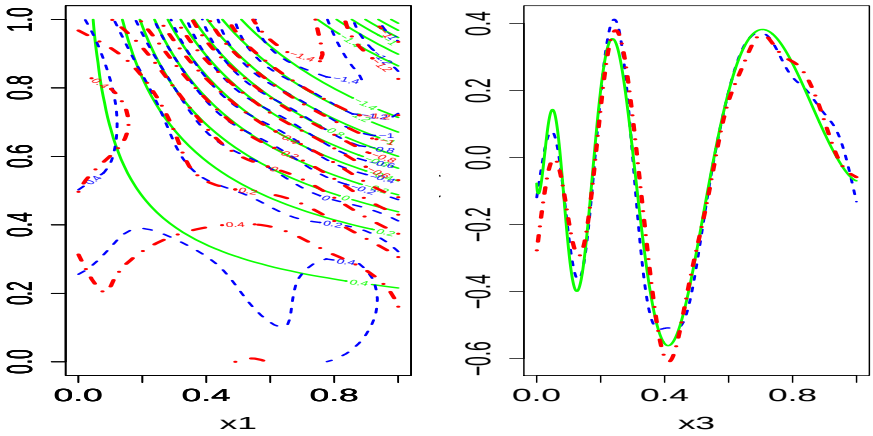


(c) n=500, tau=0.8

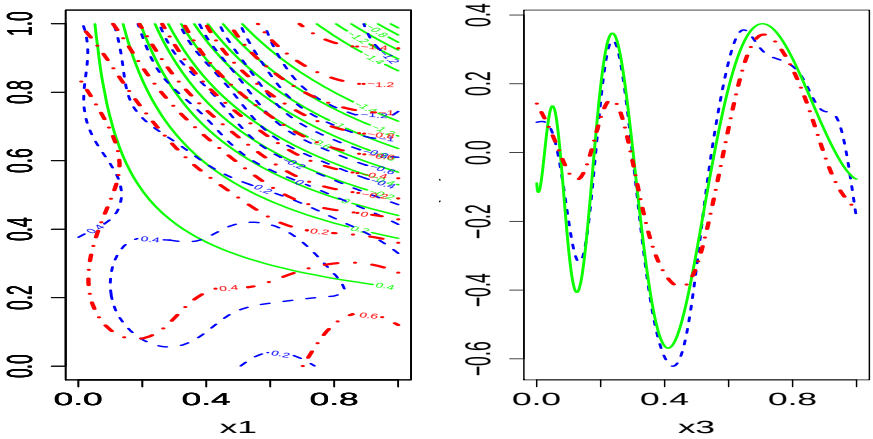
Fig. 9.20: Model 17: On this Figure, we present the results for n=500, tau=0.2, 0.5, 0.8. Each line shows the bivariate smooth of Model 17 represented as a contour map and the univariate smooth. The green lines show the true quantile for the smooth. The red line, the worst fit out of 50, the blue line, the best fit out of 50. results for n=250 and 2500 can be found on Figures 9.19 and 9.21, respectively.



(a) n=2500, tau=0.05

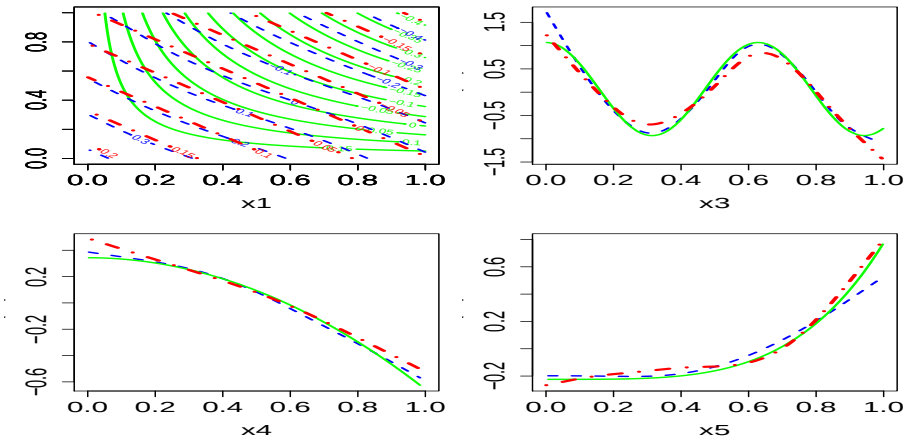


(b) n=2500, tau=0.5

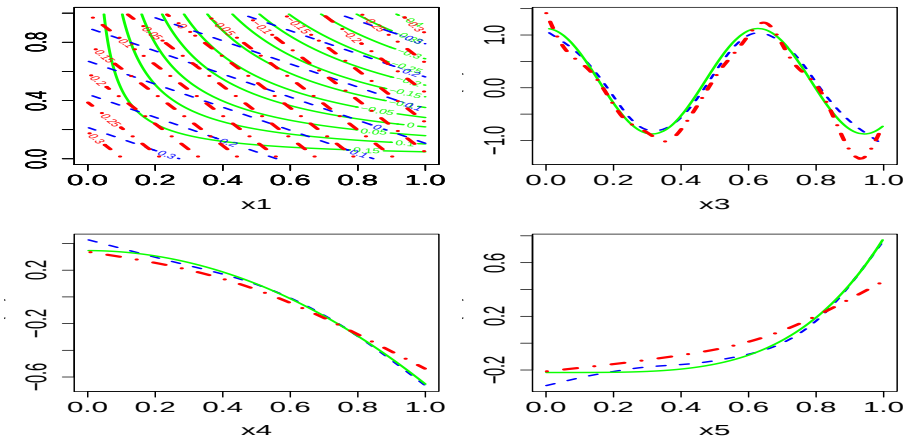


(c) n=2500, tau=0.95

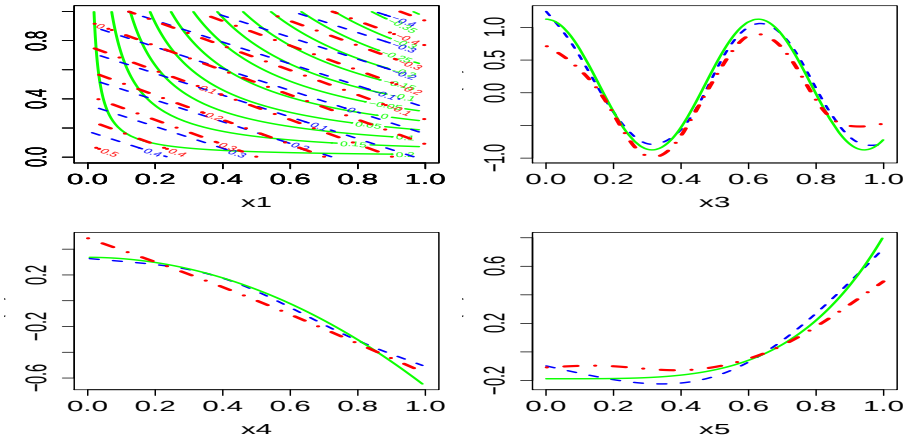
Fig. 9.21: Model 17: On this Figure, we present the results for n=2,500, tau=0.05, 0.5, 0.95. Each line shows the bivariate smooth of Model 17 represented as a contour map and the univariate smooth. The green lines show the true quantile for the smooth. The red line, the worst fit out of 50, the blue line, the best fit out of 50. results for n=250 and 500 can be found on Figures 9.19 and 9.20, respectively.



(a) n=250, tau=0.25

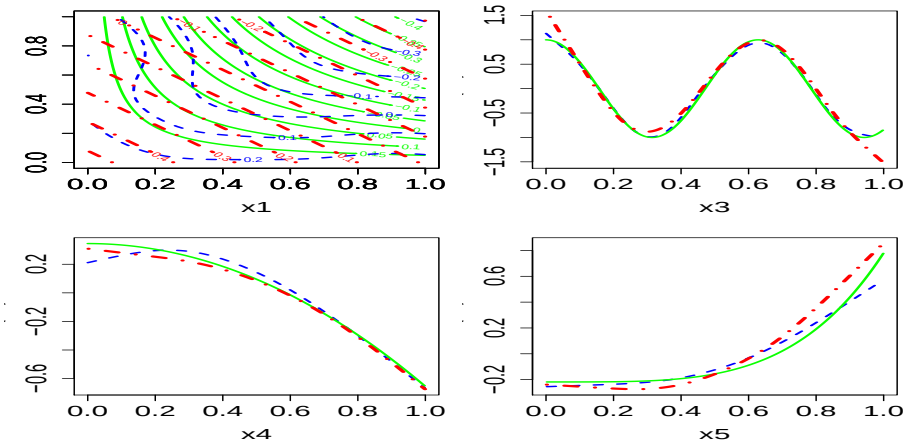


(b) n=250, tau=0.5

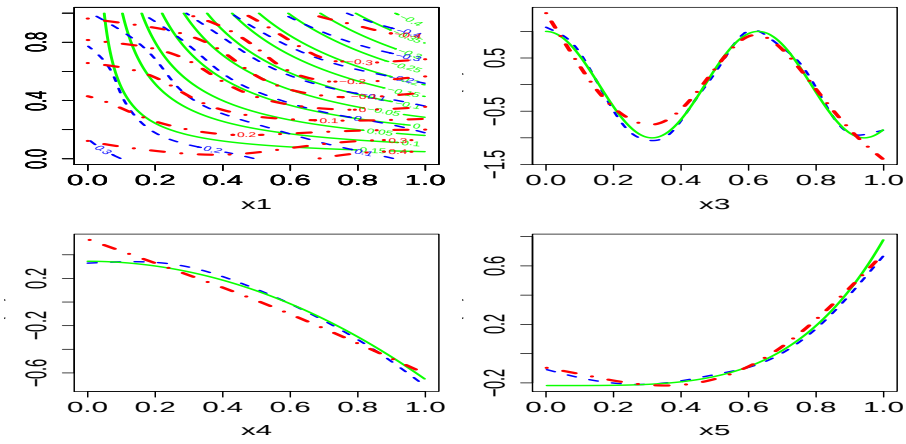


(c) n=250, tau=0.75

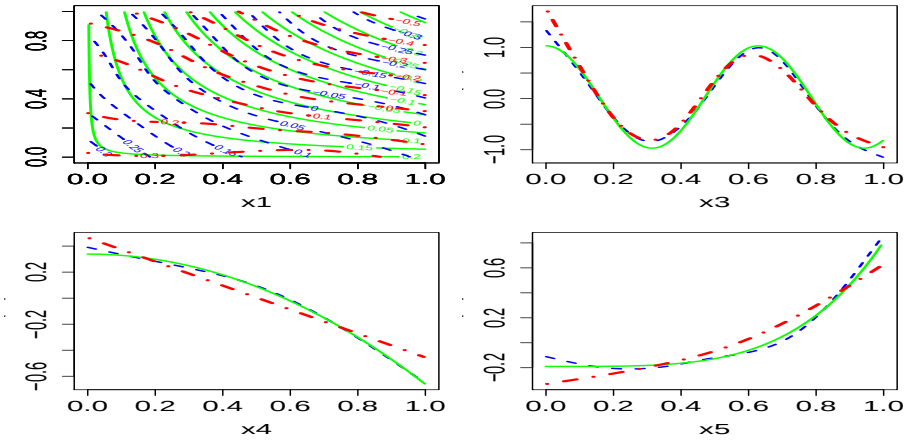
Fig. 9.22: Model 18: On this Figure, we present the results for n=250, tau=0.25, 0.5, 0.75. Each line shows the bivariate smooth of Model 18 represented as a contour map and 3 univariate smooths. The green lines show the true quantile for the smooth. The red line, the worst fit out of 50, the blue line, the best fit out of 50. results for n=500 and 2500 can be found on Figures 9.23 and 9.24, respectively.



(a) n=500, tau=0.20

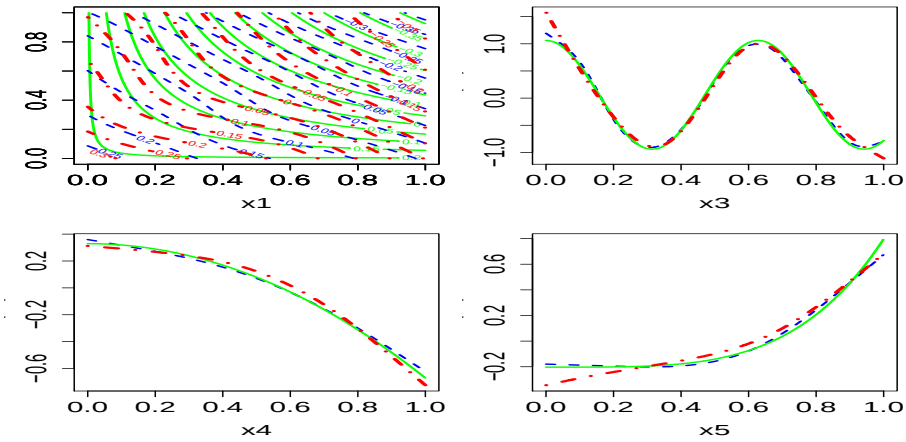


(b) n=500, tau=0.5

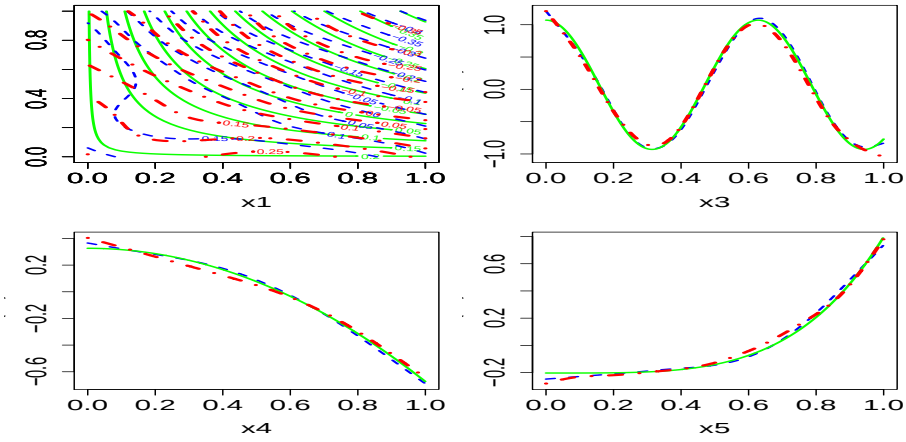


(c) n=500, tau=0.8

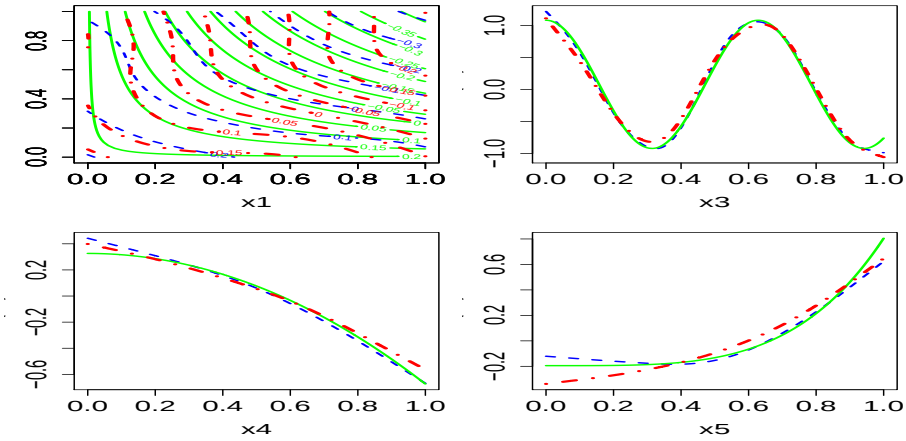
Fig. 9.23: Model 18: On this Figure, we present the results for n=500, tau=0.2, 0.5, 0.8. Each line shows the bivariate smooth of Model 18 represented as a contour map and 3 univariate smooths. The green lines show the true quantile for the smooth. The red line, the worst fit out of 50, the blue line, the best fit out of 50. results for n=250 and 2500 can be found on Figures 9.22 and 9.24, respectively.



(a) n=2500, tau=0.05



(b) n=2500, tau=0.5



(c) n=2500, tau=0.95

Fig. 9.24: Model 18: On this Figure, we present the results for n=2,500, tau=0.05, 0.5, 0.95. Each line shows the bivariate smooth of Model 17 represented as a contour map and 3 univariate smooths. The green lines show the true quantile for the smooth. The red line, the worst fit out of 50, the blue line, the best fit out of 50. results for n=250 and 500 can be found on Figures 9.22 and 9.23, respectively.

9.3 Test 2: Compare Packages with MSE

In this section, we compare the fits with aqmm, qgam and qgacv in terms of mean squared error compared to the true quantiles. Note that the MSE values in this section are absolute numbers, they can be used to compare qgacv with aqmm and qgam but cannot be relied on to compare the results of a model to another. Note that Model 13 results are expressed as a percentage of the mean of the qgacv fit. This is because the values for Model 13 are very large to test the robustness of functions aqmm:aqmm, qgacv:qam and qgam:qgam.

We note the following: for $n = 250, n = 500$, the MSE tends to be larger with qgacv than with qgam. However, for $n = 2500$, this is not the case any longer. Using a Bayesian update leads to fits that are closer to the true quantile than with the QGACV for a lower number of observations. For example, for Model 2, the MSE for qgacv, n=250, $\tau = 0.25, 0.5, 0.75$ are $0.24, 0.15, 0.21$ with qgacv compared to $0.19, 0.11, 0.16$ with qgam. However for $n = 2500$, for $\tau = 0.05, 0.5, 0.95$, the MSE for qgacv is $0.3, 0.02, 0.26$ to be compared to $0.32, 0.02, 0.33$ with qgam. A similar pattern appears with the other models. The QGACV tends to slightly oversmooth for a relatively low number of observations.

As will be discussed in the next section, qgam does not fit correctly central quantiles for bimodal data (Model 9 and Model 10). For example, for Model 9, we see that although the MSE is larger with qgacv than with qgam with $n = 250$, for $n = 2,500$, the MSE reduces with qgacv but increases with qgam.

With aqmm, we seem to obtain similar results to qgam or qgacv with Gaussian errors. However, aqmm seems to have high sensitivity to extreme datapoints (Model 2, Model 4, Model 11). We show an example Appendix D.4, Figure D.5. This example is a randomly drawn dataset that is based on Model 11. To use the logarithmic scale on the top chart of the Figure, we shift the data up by the minimum of the data and we replace the lowest datapoint by the mean. The fit is very sensitive to a single extreme datapoint and the fit changes significantly if this extreme datapoint is replaced by the average of the y values.

Another issue with the aqmm package can be observed when the degree of smoothness of central quantiles is different from the degree of smoothness of non-central quantiles. (Model 8, Model 13, Model 14). This issue was discussed in section 9 and Appendix D.1.

Last, we note that there are cases where aqmm outperforms both qgam and qgacv. This is the case for Model 9 (linear bimodal data), Model 10 (non-linear bimodal data), Model 12. For these 3 models, the degree of smoothness is the same for all quantiles. This gives an advantage to aqmm.

	Model 1								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qgacv	0.09	0.07	0.08	0.06	0.04	0.06	0.03	0.01	0.03
aqmm	0.08	0.07	0.08	0.05	0.04	0.06	0.03	0.01	0.03
qgam	0.07	0.05	0.07	0.05	0.03	0.05	0.02	0.01	0.02
	Model 2								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qgacv	0.24	0.15	0.21	0.15	0.09	0.16	0.3	0.02	0.26
aqmm	1	0.5	0.88	1.41	0.77	1.32	1.24	0.38	1.05
qgam	0.19	0.11	0.16	0.11	0.07	0.13	0.32	0.02	0.33
	Model 3								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qgacv	0.84	0.75	0.96	0.5	0.41	0.53	0.25	0.1	0.3
aqmm	0.84	0.83	0.9	0.45	0.43	0.52	0.29	0.11	0.31
qgam	0.77	0.65	0.87	0.46	0.37	0.52	0.37	0.1	0.41
	Model 4								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qgacv	2.2	1.47	2.18	1.3	0.82	1.36	2.21	0.19	2.45
aqmm	7.93	5.06	5.57	8.83	4.75	4.89	13.87	4.09	4.99
qgam	1.99	1.38	1.95	1.29	0.76	1.34	4	0.19	3.72
	Model 5								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qgacv	0.02	0.02	0.02	0.01	0.01	0.01	0	0	0
aqmm	0.02	0.01	0.01	0.01	0.01	0.01	0	0	0
qgam	0.02	0.01	0.02	0.01	0.01	0.01	0	0	0

Tab. 9.1: This tables shows the MSE simulations with the first 5 models. The three R packages tested are qgacv, aqmm and qgam. We calculate the mean squared error compared to the true quantile over 50 fits randomly sampled for each model. The numbers at the top (n=250,...) correspond to the number of datapoints. The numbers below are the values of the quantile parameters (0.25, 0.5,...).

	Model 6								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qgacv	1.02	0.73	0.91	0.44	0.34	0.46	0.24	0.07	0.23
aqmm	0.84	0.69	0.81	0.41	0.37	0.42	0.22	0.08	0.21
qgam	0.9	0.66	0.81	0.41	0.32	0.41	0.44	0.08	0.42
	Model 7								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qgacv	0.01	0.01	0.01	0	0	0	0	0	0
aqmm	0.01	0	0.01	0	0	0	0	0	0
qgam	0.01	0.01	0.01	0.01	0	0	0	0	0
	Model 8								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qgacv	0.06	0	0.08	0.02	0	0.02	0.02	0	0.02
aqmm	0.15	0	0.15	0.21	0	0.21	0.82	0	0.83
qgam	0.03	0.02	0.04	0.02	0.01	0.02	0.02	0	0.03

	Model 9											
	n=250				n=500				n=2500			
	0.25	0.4	0.6	0.75	0.2	0.45	0.55	0.8	0.05	0.45	0.55	0.95
qgacv	0.05	1.74	0.79	0.04	0.02	3.18	1.87	0.01	0	0.19	0.2	0.01
aqmm	0.05	0.71	0.13	0.04	0.02	2.36	0.56	0.01	0	0.03	0.03	0.01
qgam	0.22	0.79	0.44	0.15	0.06	3.8	2.84	0.07	0.01	2.99	3.02	0.02
	Model 10											
	n=250				n=500				n=2500			
	0.25	0.4	0.6	0.75	0.2	0.45	0.55	0.8	0.05	0.45	0.55	0.95
qgacv	0.23	3.32	5	0.18	0.11	8.72	5.66	0.11	0.03	1.76	1.66	0.04
aqmm	0.32	1.17	1.65	0.32	0.11	4.39	2.1	0.1	0.03	0.23	0.2	0.03
qgam	0.64	0.69	0.74	0.61	0.26	4.78	3.68	0.32	0.07	3.08	3.11	0.07

Tab. 9.2: This tables shows the MSE simulations for models 6 to 10. The three R packages tested are qgacv, aqmm and qgam. We calculate the mean squared error compared to the true quantile over 50 fits randomly sampled for each model. The numbers at the top (n=250,...) correspond to the number of datapoints. The numbers below are the values of the quantile parameters (0.25, 0.5,...). For models 9 and 10, because the data is bimodal, we do not fit the central quantile ($\tau = 0.5$) but quantiles just above and below the centre.

	Model 11								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qgacv	0.2	0.11	0.17	0.13	0.06	0.1	0.2	0.02	0.19
aqmm	4.48	2.52	3.39	5.17	3.27	5.33	5.02	2.24	11.41
qgam	0.14	0.09	0.13	0.08	0.05	0.08	0.18	0.01	0.21
	Model 12								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qgacv	0.05	0.07	0.15	0.04	0.06	0.13	0.02	0.03	0.12
aqmm	0.05	0.07	0.11	0.04	0.05	0.08	0.02	0.02	0.1
qgam	0.03	0.06	0.13	0.03	0.05	0.11	0.03	0.03	0.11
	Model 13								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qgacv	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0	0
aqmm	0.29	0.1	0.16	0.3	0.14	0.1	0.75	0.18	0.24
qgam	0.01	0.01	0.01	0.01	0	0.01	0.01	0	0
	Model 14								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qgacv	2.94	1.38	3.88	1.99	0.49	2.26	1.16	0.15	1.34
aqmm	10.15	2.42	4.82	7.91	1.07	5.64	9.16	0.23	9.7
qgam	2.27	1.16	2.44	1.49	0.5	1.6	1.11	0.16	1.19
	Model 15								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qgacv	0.43	0.37	0.47	0.24	0.2	0.27	0.13	0.05	0.14
aqmm	0.3	0.3	0.31	0.17	0.16	0.17	0.05	0.04	0.05
qgam	0.39	0.28	0.42	0.22	0.15	0.24	0.11	0.04	0.11

Tab. 9.3: This tables shows the MSE simulations for Models 11 to 15. The three R packages tested are qgacv, aqmm and qgam. We calculate the mean squared error compared to the true quantile over 50 fits randomly sampled for each model. The numbers at the top (n=250,...) correspond to the number of datapoints. The numbers below are the values of the quantile parameters (0.25, 0.5,...). Note that the MSE are absolute numbers. Model 13 has been designed to test the robustness of the packages to large values. Hence, the MSE numbers are very large. To display them in this table, the MSE numbers of Model 13 were divided by the mean of the fitted values using the qgacv to find the smoothing parameters.

	Model 16								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qgacv	0.03	0.03	0.03	0.02	0.02	0.02	0.02	0.01	0.01
aqmm	0.02	0.02	0.03	0.02	0.02	0.02	0.01	0.01	0.01
qgam	0.02	0.02	0.03	0.02	0.01	0.02	0.01	0	0.01
	Model 17								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qgacv	0.06	0.05	0.07	0.04	0.03	0.04	0.02	0.01	0.02
aqmm	0.05	0.04	0.05	0.03	0.03	0.03	0.02	0.01	0.02
qgam	0.04	0.03	0.04	0.03	0.02	0.03	0.02	0.01	0.01
	Model 18								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qgacv	0.03	0.03	0.03	0.02	0.02	0.02	0.01	0	0.01
aqmm	0.03	0.02	0.03	0.02	0.02	0.02	0.01	0	0.01
qgam	0.03	0.02	0.03	0.02	0.01	0.02	0.01	0	0.01

Tab. 9.4: This tables shows the MSE simulations for Models 16 to 18. The three R packages tested are qgacv, aqmm and qgam. We calculate the mean squared error compared to the true quantile over 50 fits randomly sampled for each model. The numbers at the top (n=250,...) correspond to the number of datapoints. The numbers below are the values of the quantile parameters (0.25, 0.5,...).

9.4 Test 3: Coverage compared to qgam

In this section, we run a simulation to compare the coverage of the confidence intervals with qgacv to the credible intervals obtained qgam. Credible intervals were not available for package 'aqmm' version 1.0. 'aqmm' version 1.1 includes a bootstrap procedure to obtain confidence intervals. Unfortunately, aqmm v1.1 has not been used to write this document. Hence the confidence intervals are not included in the analysis of coverage in this section.

The setup of the test is similar to the previous sections. We use the same examples, number of datapoints and values for quantile parameter τ. As mentioned in section 6.2, to calculate the confidence intervals, our methodology consists in first estimating the vector of smoothing parameters $\hat{\boldsymbol{\rho}}$ using the QGACV, then calculated a relaxed vector of smoothing parameters $\grave{\boldsymbol{\rho}}$ that is at most "1 standard error away" from $\hat{\boldsymbol{\rho}}$, then estimate the np bootstrap

samples $\grave{\boldsymbol{\beta}}^{*b}_{\hat{\boldsymbol{\rho}}}; b = 1, ..., B$ using $\grave{\boldsymbol{\rho}}$ as a vector of smoothing parameters. Then the covariance matrix to estimate the confidence interval is calculated using $\grave{\boldsymbol{\beta}}^{*b}_{\hat{\boldsymbol{\rho}}}; b = 1, ..., B$. The confidence interval is then calculated as $\hat{f}_i \pm 1.96 \times se_{\grave{f}_i}$, i.e. around the estimate of the the predicted value using the vector of smoothing parameter without relaxation $\hat{f}_i$, we calculate the confidence interval with the standard error of the predicted values calculated with relaxation $\grave{f}_i$.

We will now run a simulation to calculate the coverage of this confidence interval. However, we may want to compare it to other benchmarks. First, the method described above is designated in the list below as "qam Relax Covariance (RC) 100" and "qam Relax Covariance (RC) 20". The number indicates the number of bootstrap samples used to estimate the covariance matrix ($B = 100$ or $B = 20$).

Then, we compare this to the original method without relaxation, that is the base fit is $\hat{f}_i$, using $\hat{\boldsymbol{\rho}}$ and the standard error is also estimated using the covariance matrix calculated using the non-relaxed smoothing parameters $\hat{\boldsymbol{\rho}}$ ("qam No Relax Covariance (NRC) 100"). We can also compare our method to package qgam. Then we may want to compare the covariance approach to the percentile bootstrap. However, given that we are using a vector of smoothing parameters to estimate the $\hat{f}_i$ and another vector of smoothing parameters to estimate the covariance matrix, it is not clear how we could derive a percentile bootstrap. We then provide results with 3 different percentile bootstrap methods.

The first method is the "qam No Relax Percentile (NRPerc) 100" where we use the same vector of smoothing parameters $\hat{\boldsymbol{\rho}}$ for both the estimation of the predicted value and the estimation of the np bootstrap samples. The second method is the "qam Relax Percentile (RPerc) 100" where we are now using the relaxed vector of smoothing parameters $\grave{\boldsymbol{\rho}}$ for both the estimation of the predicted value and the estimation of the np bootstrap samples. The last method consists in calculating the percentile bootstrap bands as in the previous method using $\grave{\boldsymbol{\rho}}$. We then recentre the band around the estimate $\hat{\mathbf{f}}$, estimated using $\hat{\boldsymbol{\rho}}$ ("qam Relax Percentile (RPerc) recentre").

- **qam No Relax Covariance (NRC) 100:** CI calculated as

$$\hat{\boldsymbol{f}} \pm 1.96 se_{\hat{\boldsymbol{\rho}}}$$

  where $\hat{\boldsymbol{f}}$ is the predicted vector of values without parameter relaxation and $se_{\hat{\boldsymbol{\rho}}}$ is the standard error calculated based on the np bootstrap covariance matrix, where np bootstrap samples are calculated using the non-relaxed estimated smoothing parameters $\hat{\boldsymbol{\rho}}$. we use 100 bootstrap samples.

- **qgam:** The credible interval calculated using the qgam package version 1.3.1.

- **qam Relax Covariance (RC) 100:** This is the CI calculated as

$$\hat{\boldsymbol{f}} \pm 1.96 se_{\grave{\boldsymbol{\rho}}}$$

where $\hat{\boldsymbol{f}}$ is the predicted vector of values without parameter relaxation and $se_{\grave{\boldsymbol{\rho}}}$ is the standard error calculated based on the np bootstrap covariance matrix, where np bootstrap samples are calculated using the relaxed estimated sps $\grave{\boldsymbol{\rho}}$. we use 100 bootstrap samples.

- **qam Relax Covariance (RC) 20:** Same as previous but with 20 bootstrap samples.

- **qam No Relax Percentile (NRPerc) 100:** CI calculated as

$$quantiles_{(0.025,0.975)}(\hat{\boldsymbol{f}}^{*1}, ..., \hat{\boldsymbol{f}}^{*B})$$

where $\hat{\boldsymbol{f}}^{*b}$ are the bootstrap samples calculated using the non-relaxed vector of smoothing parameters $\hat{\boldsymbol{\rho}}$.

- **qam Relax Percentile (RPerc) 100:** CI calculated as

$$quantiles_{(0.025,0.975)}(\grave{\boldsymbol{f}}^{*1}, ..., \grave{\boldsymbol{f}}^{*B})$$

where the $\grave{\boldsymbol{f}}^{*b}$ are the bootstrap samples calculated using the relaxed vector of smoothing parameters. This is shown as a reference. Unfortunately, we cannot use this as the fit with relaxed smoothing parameter tends to overfit central quantiles.

- **qam Relax Percentile (RPerc) recentre 100:**
Same as qam Relax Percentile(RP) 100 but here, we take into account the difference between $\grave{f}$, the estimated model with the relaxed sps $\grave{\boldsymbol{\rho}}$ and the original estimate $\hat{f}$ without parameter relaxation. With this change, the bootstrap samples have the same distribution around $\hat{f}$ as they had around $\grave{f}$. The CI is then calculated as

$$quantiles_{(0.025,0.975)}(\grave{\boldsymbol{f}}^{*1} - (\grave{\boldsymbol{f}} - \hat{\boldsymbol{f}}), ..., \grave{\boldsymbol{f}}^{*B} - (\grave{\boldsymbol{f}} - \hat{\boldsymbol{f}}))$$

where $\grave{\boldsymbol{f}}^{b}$ are the bootstrap samples calculated using the relaxed vector of smoothing parameters and where $\grave{\boldsymbol{f}}$ is the fitted model with the relaxed smoothing parameters $\grave{\boldsymbol{\rho}}$. This is shows as a reference.

We note that with **qam NRC 100**, even with a model with an i.i.d Gaussian error (Model 1), the coverage is sometimes below 0.9, even with n=2,500. We note a poor coverage for Model 4 (heteroscedastic with alpha stable error), Model 13 (heteroscedastic errors), Model 16 (2d model) and Model 17 (2d+1d model). This is why we proposed to relax the smoothing parameter to obtain confidence intervals.

Overall, **qgam**, seems to outperform the CI calculated using qam for a low number of observations and Gaussian homoscedastic errors (for example, Model 1). However, for a higher number of observations (n=2,500) and with non-Gaussian errors, the coverage with qgam is worse than the coverage with **qam**

RC 100, our base method. (for example:
Model 3, n=2500, tau=0.05, qgam=0.78, qamRC100=0.87;
Model 4, n=2500, tau=0.05, qgam=0.64, qamRC100=0.80;
Model 6, n=2500, tau=0.05, qgam=0.68, qamRC100=0.92;
Model 7, n=2500, tau=0.05, qgam=0.66, qamRC100=0.93;
Model 13, n=2500, tau=0.05, qgam=0.56, qamRC100=0.85).
Furthermore, qgam does not adapt to multimodal data. We obtain very poor coverage for Models 9 and 10 (for example: Model 9, n=500, tau=0.45, qgam=0.016, qamRC100=0.93) (see Appendix D.2).

Other notable poor performance of qgam include Model 7 (linear with Gaussian heteroscedastic error). This poor performance is due to the bias in the estimation of the model with package 'qgam' with heteroscedastic data. This issue has been noted in Fasiolo et al. [2020c] who proposes to use a variable learning rate. Using a variable learning rate seems to reduce the bias issue (see an example on Figure 9.25). To compare qgam and qam, we kept the standard setting for both. Similarly to qgam, we may have improved the coverage of qam with non-standard setting for some examples, for example using adaptive splines.

qam RC 100, improves the coverage compared to **qam NRC 100**. It seems that we have slight overcoverage for quantiles that are straight lines with slope=0. For Model 8, we have a too large coverage for $\tau = 0.5$. This example has a central quantile exactly equal to 0. The coverage is also slightly too large for Model 9 for several quantiles/ number of datapoints.

To conclude, as discussed before, our aim was to compare method "qam Relax Covariance (RC) 100" and "qam Relax Covariance (RC) 20" to a series of benchmarks. Our method seems competitive overall and compared to these benchmarks. The benchmark that seems to give the best coverage is method "qam Relax Percentile (RPerc) 100". Unfortunately, using this method would require changing our original estimate of the vector of parameters $\hat{\boldsymbol{\beta}}$ for $\grave{\boldsymbol{\beta}}$. This would lead to overfitting, especially for central quantiles for the estimate $\grave{\boldsymbol{f}}$.

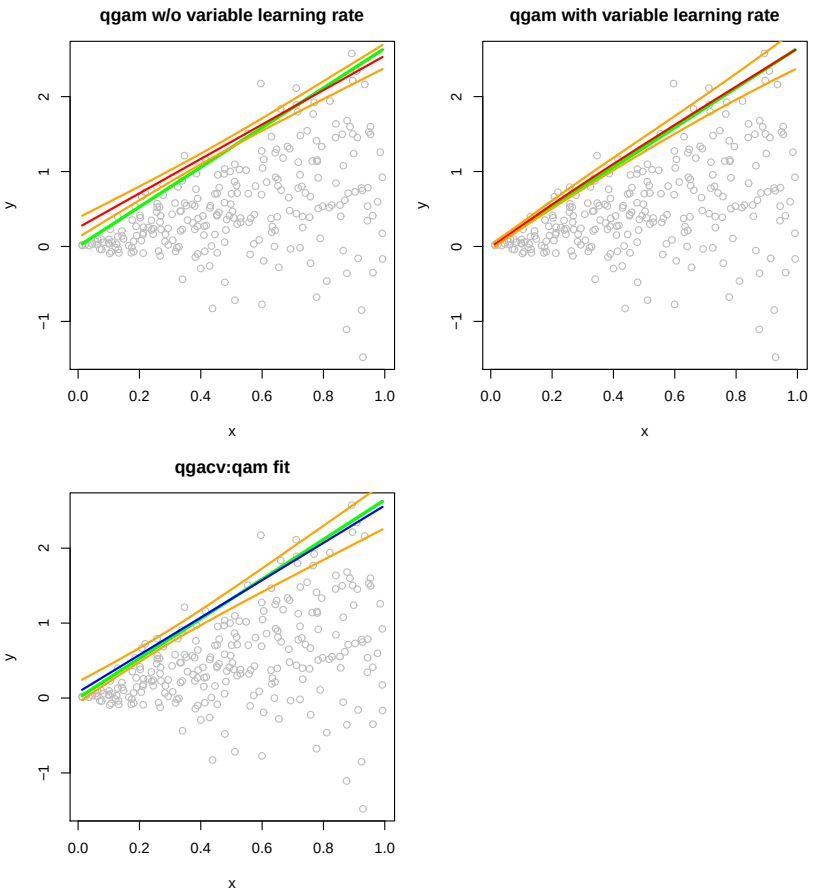


Fig. 9.25: On the top left chart, we can see a fit with qgam without variable learning rate. The fit (in red) is away from the true quantile (in green). The credible interval (orange) does not completely cover the true quantile. On the top right, we can see the same example but this time fitted using qgam with variable learning rate. We see that the fit is now very close to the true quantile and the credible interval contains the true quantile. On the bottom chart, we can see the fit with qgacv::qam. The fit is similar to qgam with variable learning rate.

	Model 1								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qam NRC 100	0.89	0.92	0.89	0.87	0.91	0.87	0.87	0.92	0.87
qgam	0.93	0.95	0.93	0.93	0.95	0.93	0.95	0.95	0.95
qam RC 100	0.93	0.95	0.93	0.92	0.95	0.93	0.91	0.94	0.91
qam RC 20	0.91	0.93	0.91	0.90	0.94	0.92	0.90	0.93	0.89
qam NRPerc 100	0.88	0.91	0.87	0.85	0.91	0.87	0.87	0.91	0.86
qam RPerc 100	0.93	0.94	0.92	0.93	0.93	0.93	0.92	0.93	0.92
qam RPerc recenter 100	0.91	0.94	0.91	0.90	0.93	0.91	0.90	0.93	0.89
	Model 2								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qam NRC 100	0.84	0.88	0.85	0.85	0.86	0.86	0.82	0.88	0.82
qgam	0.93	0.94	0.93	0.93	0.94	0.93	0.88	0.96	0.88
qam RC 100	0.89	0.92	0.90	0.90	0.91	0.90	0.87	0.90	0.88
qam RC 20	0.88	0.91	0.89	0.88	0.89	0.88	0.85	0.89	0.87
qam NRPerc 100	0.79	0.85	0.83	0.84	0.85	0.83	0.79	0.88	0.81
qam RPerc 100	0.89	0.92	0.91	0.91	0.93	0.93	0.90	0.90	0.92
qam RPerc recenter 100	0.85	0.89	0.86	0.86	0.90	0.87	0.84	0.90	0.86
	Model 3								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qam NRC 100	0.84	0.83	0.81	0.85	0.88	0.84	0.81	0.89	0.79
qgam	0.92	0.93	0.91	0.94	0.95	0.92	0.78	0.95	0.76
qam RC 100	0.90	0.91	0.89	0.89	0.93	0.89	0.87	0.91	0.85
qam RC 20	0.88	0.89	0.86	0.87	0.91	0.88	0.86	0.90	0.83
qam NRPerc 100	0.81	0.83	0.79	0.84	0.87	0.82	0.80	0.88	0.79
qam RPerc 100	0.93	0.94	0.92	0.92	0.94	0.92	0.91	0.92	0.89
qam RPerc recenter 100	0.86	0.89	0.87	0.87	0.92	0.88	0.87	0.90	0.83
	Model 4								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qam NRC 100	0.81	0.81	0.78	0.78	0.80	0.79	0.76	0.85	0.73
qgam	0.90	0.91	0.88	0.90	0.92	0.89	0.64	0.95	0.67
qam RC 100	0.86	0.87	0.83	0.84	0.87	0.85	0.80	0.86	0.80
qam RC 20	0.84	0.87	0.81	0.84	0.85	0.83	0.79	0.85	0.78
qam NRPerc 100	0.79	0.81	0.75	0.77	0.78	0.76	0.74	0.84	0.71
qam RPerc 100	0.90	0.90	0.87	0.89	0.90	0.90	0.86	0.86	0.84
qam RPerc recenter 100	0.84	0.84	0.80	0.81	0.84	0.83	0.78	0.84	0.76

Tab. 9.5: This tables shows the simulations of the coverage for the 95% confidence intervals for Models 1 to 4. Each simulation relies on 50 random samples of the model. We simulate examples with n=250, 500, 2500 datapoints. The number below (0.25,0.5,..) is the value of the quantile parameter τ.

	Model 5								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qam NRC 100	0.91	0.94	0.94	0.92	0.91	0.93	0.96	0.94	0.92
qgam	0.91	0.93	0.88	0.86	0.94	0.90	0.91	0.93	0.89
qam RC 100	0.92	0.95	0.96	0.94	0.92	0.93	0.95	0.93	0.91
qam RC 20	0.90	0.95	0.99	0.91	0.92	0.91	0.93	0.92	0.90
qam NRPerc 100	0.92	0.92	0.95	0.92	0.91	0.93	0.94	0.93	0.91
qam RPerc 100	0.93	0.92	0.94	0.91	0.92	0.93	0.94	0.91	0.89
qam RPerc recenter 100	0.94	0.94	0.96	0.93	0.93	0.94	0.94	0.91	0.91
	Model 6								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qam NRC 100	0.88	0.91	0.90	0.91	0.93	0.92	0.88	0.93	0.89
qgam	0.89	0.90	0.90	0.93	0.95	0.94	0.68	0.95	0.69
qam RC 100	0.93	0.95	0.94	0.94	0.95	0.94	0.92	0.95	0.91
qam RC 20	0.92	0.93	0.92	0.93	0.94	0.92	0.89	0.93	0.89
qam NRPerc 100	0.87	0.90	0.90	0.89	0.93	0.90	0.88	0.92	0.87
qam RPerc 100	0.93	0.94	0.95	0.94	0.95	0.94	0.92	0.93	0.92
qam RPerc recenter 100	0.90	0.92	0.91	0.92	0.94	0.91	0.90	0.93	0.90
	Model 7								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qam NRC 100	0.92	0.96	0.91	0.94	0.93	0.92	0.92	0.91	0.94
qgam	0.82	0.90	0.81	0.81	0.92	0.79	0.66	0.88	0.61
qam RC 100	0.95	0.97	0.92	0.97	0.94	0.93	0.93	0.91	0.96
qam RC 20	0.91	0.94	0.94	0.96	0.91	0.93	0.93	0.90	0.95
qam NRPerc 100	0.89	0.94	0.90	0.93	0.97	0.94	0.91	0.91	0.97
qam RPerc 100	0.89	0.94	0.89	0.93	0.94	0.92	0.92	0.88	0.93
qam RPerc recenter 100	0.90	0.95	0.90	0.95	0.96	0.93	0.94	0.88	0.95
	Model 8								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qam NRC 100	0.71	0.98	0.63	0.87	0.99	0.87	0.81	1.00	0.80
qgam	0.82	0.74	0.78	0.87	0.72	0.84	0.63	0.80	0.61
qam RC 100	0.75	0.97	0.67	0.93	0.99	0.91	0.86	1.00	0.85
qam RC 20	0.74	0.97	0.64	0.91	0.99	0.90	0.85	0.99	0.84
qam NRPerc 100	0.68	0.93	0.59	0.84	0.92	0.84	0.79	0.93	0.78
qam RPerc 100	0.79	0.93	0.68	0.92	0.92	0.92	0.86	0.93	0.85
qam RPerc recenter 100	0.72	0.93	0.62	0.88	0.92	0.89	0.82	0.93	0.83

Tab. 9.6: This tables shows the simulations of the coverage for the 95% confidence intervals for Models 5 to 8. Each simulation relies on 50 random samples of the model. We simulate examples with n=250, 500, 2500 datapoints. The number below (0.25,0.5,..) is the value of the quantile parameter τ.

	Model 9											
	n=250				n=500				n=2500			
	0.25	0.4	0.6	0.75	0.2	0.45	0.55	0.8	0.05	0.45	0.55	0.95
qam NRC 100	0.94	0.94	0.94	0.93	0.97	0.93	0.94	0.97	0.96	0.95	0.95	0.89
qgam	0.92	0.82	0.87	0.93	0.773	0.016	0.048	0.659	6.7e-01	0.0e+00	6.4e-05	6.1e-01
qam RC 100	0.95	0.96	0.94	0.95	0.96	0.93	0.95	0.97	0.96	0.97	0.94	0.90
qam RC 20	0.95	0.95	0.93	0.94	0.94	0.91	0.94	0.95	0.93	0.93	0.93	0.87
qam NRPerc 100	0.92	0.92	0.93	0.93	0.95	0.91	0.94	0.97	0.94	0.93	0.90	0.90
qam RPerc 100	0.92	0.94	0.92	0.93	0.95	0.92	0.95	0.95	0.95	0.93	0.89	0.91
qam RPerc recenter 100	0.92	0.92	0.93	0.94	0.96	0.90	0.94	0.96	0.95	0.92	0.89	0.91
	Model 10											
	n=250				n=500				n=2500			
	0.25	0.4	0.6	0.75	0.2	0.45	0.55	0.8	0.05	0.45	0.55	0.95
qam NRC 100	0.88	0.83	0.86	0.91	0.89	0.81	0.86	0.85	0.80	0.92	0.92	0.82
qgam	0.94	0.95	0.94	0.96	0.89	0.23	0.31	0.82	0.6833	0.0037	0.0079	0.6757
qam RC 100	0.94	0.85	0.89	0.94	0.93	0.85	0.90	0.89	0.84	0.95	0.94	0.85
qam RC 20	0.91	0.83	0.87	0.92	0.91	0.84	0.89	0.88	0.83	0.94	0.94	0.83
qam NRPerc 100	0.84	0.79	0.82	0.83	0.85	0.83	0.87	0.83	0.77	0.91	0.91	0.79
qam RPerc 100	0.92	0.80	0.85	0.92	0.92	0.86	0.92	0.92	0.88	0.93	0.94	0.89
qam RPerc recenter 100	0.88	0.78	0.83	0.87	0.89	0.81	0.86	0.86	0.82	0.92	0.93	0.84

Tab. 9.7: This tables shows the simulations of the coverage for the 95% confidence intervals for Models 9 to 10. Each simulation relies on 50 random samples of the model. We simulate examples with n=250, 500, 2500 datapoints. Because the data is bimodal, we do not estimate the central quantile $\tau = 0.5$ but quantile above and below the centre (4 quantile parameter values instead of 3 for other Models).

	Model 11								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qam NRC 100	0.91	0.92	0.89	0.89	0.92	0.92	0.90	0.91	0.85
qgam	0.95	0.96	0.95	0.96	0.97	0.96	0.93	0.96	0.91
qam RC 100	0.93	0.95	0.94	0.90	0.95	0.95	0.92	0.92	0.91
qam RC 20	0.91	0.93	0.91	0.91	0.93	0.93	0.91	0.90	0.90
qam NRPerc 100	0.89	0.91	0.88	0.88	0.91	0.90	0.90	0.91	0.86
qam RPerc 100	0.93	0.94	0.93	0.92	0.94	0.94	0.92	0.91	0.91
qam RPerc recenter 100	0.91	0.93	0.91	0.90	0.94	0.92	0.91	0.91	0.87
	Model 12								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qam NRC 100	0.87	0.90	0.84	0.90	0.89	0.84	0.89	0.90	0.86
qgam	0.94	0.94	0.86	0.93	0.92	0.88	0.81	0.93	0.87
qam RC 100	0.92	0.95	0.88	0.93	0.94	0.91	0.90	0.92	0.90
qam RC 20	0.91	0.94	0.86	0.92	0.92	0.90	0.89	0.90	0.88
qam NRPerc 100	0.87	0.90	0.85	0.89	0.90	0.84	0.86	0.91	0.86
qam RPerc 100	0.92	0.95	0.88	0.92	0.92	0.91	0.84	0.90	0.89
qam RPerc recenter 100	0.91	0.94	0.88	0.92	0.93	0.92	0.86	0.92	0.89
	Model 13								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qam NRC 100	0.85	0.87	0.84	0.82	0.88	0.85	0.79	0.92	0.85
qgam	0.90	0.90	0.89	0.90	0.92	0.90	0.56	0.94	0.72
qam RC 100	0.90	0.93	0.90	0.88	0.94	0.90	0.85	0.94	0.90
qam RC 20	0.89	0.92	0.88	0.87	0.92	0.88	0.83	0.93	0.88
qam NRPerc 100	0.83	0.88	0.84	0.81	0.88	0.83	0.79	0.90	0.84
qam RPerc 100	0.91	0.93	0.91	0.90	0.94	0.92	0.90	0.94	0.93
qam RPerc recenter 100	0.86	0.90	0.87	0.87	0.92	0.89	0.84	0.93	0.87

Tab. 9.8: This tables shows the simulations of the coverage for the 95% confidence intervals for Models 11 to 13. Each simulation relies on 50 random samples of the model. We simulate examples with n=250, 500, 2500 datapoints. The number below (0.25,0.5,..) is the value of the quantile parameter τ.

	Model 14								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qam NRC 100	0.83	0.88	0.69	0.85	0.93	0.83	0.86	0.90	0.84
qgam	0.88	0.91	0.85	0.89	0.94	0.88	0.83	0.94	0.83
qam RC 100	0.92	0.92	0.75	0.92	0.96	0.89	0.91	0.92	0.91
qam RC 20	0.89	0.90	0.74	0.89	0.95	0.87	0.90	0.92	0.89
qam NRPerc 100	0.83	0.86	0.69	0.84	0.91	0.82	0.85	0.89	0.85
qam RPerc 100	0.91	0.91	0.76	0.91	0.93	0.89	0.92	0.90	0.92
qam RPerc recenter 100	0.88	0.91	0.73	0.89	0.94	0.86	0.89	0.90	0.88
	Model 15								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qam NRC 100	0.92	0.93	0.91	0.92	0.93	0.91	0.89	0.93	0.89
qgam	0.93	0.95	0.91	0.93	0.95	0.93	0.94	0.95	0.93
qam RC 100	0.95	0.95	0.94	0.95	0.95	0.94	0.92	0.95	0.92
qam RC 20	0.94	0.94	0.93	0.93	0.94	0.93	0.91	0.93	0.90
qam NRPerc 100	0.92	0.93	0.92	0.92	0.93	0.91	0.89	0.92	0.89
qam RPerc 100	0.95	0.95	0.94	0.94	0.95	0.94	0.93	0.95	0.93
qam RPerc recenter 100	0.94	0.95	0.93	0.94	0.95	0.93	0.91	0.95	0.91
	Model 16								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qam NRC 100	0.81	0.85	0.78	0.81	0.85	0.82	0.80	0.91	0.82
qgam	0.93	0.95	0.92	0.93	0.95	0.93	0.93	0.95	0.94
qam RC 100	0.88	0.91	0.86	0.88	0.92	0.88	0.90	0.95	0.88
qam RC 20	0.87	0.90	0.84	0.87	0.91	0.87	0.89	0.94	0.88
qam NRPerc 100	0.82	0.84	0.77	0.79	0.86	0.80	0.80	0.91	0.81
qam RPerc 100	0.90	0.91	0.88	0.89	0.93	0.88	0.89	0.94	0.90
qam RPerc recenter 100	0.86	0.90	0.83	0.86	0.91	0.86	0.88	0.94	0.87

Tab. 9.9: This tables shows the simulations of the coverage for the 95% confidence intervals for Models 14 to 16. Each simulation relies on 50 random samples of the model. We simulate examples with n=250, 500, 2500 datapoints. The number below (0.25,0.5,..) is the value of the quantile parameter τ.

	Model 17								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qam NRC 100	0.78	0.85	0.73	0.80	0.86	0.79	0.80	0.91	0.79
qgam	0.89	0.92	0.89	0.90	0.93	0.91	0.92	0.94	0.92
qam RC 100	0.84	0.91	0.79	0.87	0.92	0.85	0.87	0.95	0.86
qam RC 20	0.82	0.89	0.79	0.85	0.91	0.84	0.86	0.93	0.85
qam NRPerc 100	0.77	0.84	0.72	0.80	0.86	0.78	0.80	0.91	0.79
qam RPerc 100	0.87	0.93	0.82	0.90	0.93	0.88	0.89	0.94	0.90
qam RPerc recenter 100	0.82	0.89	0.77	0.85	0.91	0.84	0.85	0.94	0.84
	Model 18								
	n=250			n=500			n=2500		
	0.25	0.5	0.75	0.2	0.5	0.8	0.05	0.5	0.95
qam NRC 100	0.89	0.91	0.88	0.88	0.89	0.87	0.85	0.91	0.87
qgam	0.90	0.94	0.89	0.91	0.95	0.91	0.94	0.97	0.93
qam RC 100	0.93	0.94	0.91	0.92	0.94	0.91	0.91	0.94	0.92
qam RC 20	0.92	0.93	0.90	0.91	0.93	0.90	0.90	0.93	0.90
qam NRPerc 100	0.89	0.91	0.87	0.88	0.90	0.87	0.85	0.91	0.86
qam RPerc 100	0.94	0.94	0.91	0.92	0.93	0.92	0.91	0.93	0.92
qam RPerc recenter 100	0.92	0.93	0.90	0.91	0.94	0.91	0.89	0.93	0.90

Tab. 9.10: This tables shows the simulations of the coverage for the 95% confidence intervals for Models 17 and 18. Each simulation relies on 50 random samples of the model. We simulate examples with n=250, 500, 2500 datapoints. The number below (0.25,0.5,..) is the value of the quantile parameter τ.

10. CONCLUSION AND FUTURE WORKS

We introduced a complete methodology to fit Quantile Additive Models together with confidence intervals. We introduce a new Quantile Generalized Approximation Cross Validation (QGACV) criterion.[1] We introduced a method to obtain a rounding level for the calculation of the effective degrees of freedom. We then proposed to used Generalized Iteratively Reweighted Least Squares and Graduated Optimization to obtain simultaneously the vector of smoothing parameters and vector of parameters of the model. The confidence intervals are determined using non-parametric bootstrap. It was proposed to improve the coverage of the confidence interval by relaxing the smoothing parameter using the "1 standard error rule" as described in Hastie et al. [2009], Wood [2017]. The methodology has been implemented in package 'qgacv'. We ran several tests to check our methodology: a first test was consisting in generating examples from 18 different univariate and multivariate generated models and observe the best, median and worst fit out of 50. The second test compares the fits to packages 'aqmm' and 'qgacv' in terms of MSE error. The third test compares the coverage of the CI compared to package 'qgam'. Our methodology is competitive and has advantages compared to other methodologies proposed previously.

From this thesis, several extensions could be envisaged. First, we showed that the surrogate loss we used fits in the framework of He et al. [2020] as it can be obtained by convolving the pinball loss with a shifted logistic kernel (see Appendix E). As an extension we could derive other surrogate losses using other infinitely differentiable kernels for example using the Gaussian kernel:

$$K_{Gauss}(u) = \frac{\exp\left(-\frac{u^2}{2}\right)}{\sqrt{2\pi}}$$

and the Sigmoid Kernel:

$$K_{Sig}(u) = \frac{1}{2\pi}\frac{1}{\exp(-u) + \exp(u)}$$

(see for example Sonabend et al. [retrieved 2021]). These surrogate losses could then be compared to the loss used in this document.

[1] Note that a poster describing a preliminary version of the this work (including a preliminary version of the QGACV criterion) was presented at the Data Science Summer School at Ecole Polytechnique and is available online at Nortier and Wood [2018]

Second, other smoothing techniques have been proposed in Horowitz [1998], Sahiner et al. [2018], Sahiner et al. [2019], Yilmaz and Sahiner [2019], Yilmaz and Sahiner [2020], Sahiner et al. [2020] and could be used as an alternative to surrogate loss 3.7. Applying some of these proposals to our case would result in surrogate losses that are C^2 (that could potentially be extended to a higher order of differentiability) or losses that are directly infinitely differentiable. One proposal in particular for piecewise differentiable functions based on an arctan function in Yilmaz and Sahiner [2019] is applied to an absolute value loss and can be extended to a pinball loss. Using the technique described in Yilmaz and Sahiner [2019], we can replace the pinball loss $\mathcal{L}_\tau(u) = u(\tau - I(u < 0))$ by (see Figure 10.1):

$$\begin{aligned}\mathcal{L}^{YS}_{\alpha,\tau}(u) =& u\tau \times \frac{\arctan(\frac{u}{\alpha}) + \frac{\pi}{2}}{\pi} + u(\tau - 1) \times \frac{\arctan(-\frac{u}{\alpha}) + \frac{\pi}{2}}{\pi} \\ =& u\tau - u \times \frac{\arctan(-\frac{u}{\alpha}) + \frac{\pi}{2}}{\pi}\end{aligned} \tag{10.1}$$

It is important to note that choosing a different smooth surrogate loss as discussed in the previous paragraphs and in equation 10.1 may have consequences for the different parts of the algorithm. What may differ between all these different losses is mainly their location compared to the exact pinball loss (above or below) and if they are always positive or not. For example, surrogate loss 3.7 is well adapted to replace the expected loss as described in section 3.4. This is because the loss has a minimum of $-\alpha\left(\tau \log(\tau) + (1-\tau)\log(1-\tau)\right)$ in 0 (see section 4.1). This minimum bounded away from zero seems to be sufficient to match the expected loss. The loss of Figure 10.1 has a minimum below zero that may not facilitate the determination of $\hat{\alpha}$ using equation 3.15. Similarly, we have seen in section 4.1 that shifting the loss to be always below the pinball loss (the loss we call $\mathcal{L}^C$ in section 4.1) can result in possible overfitting for central quantiles with the QGACV criterion.

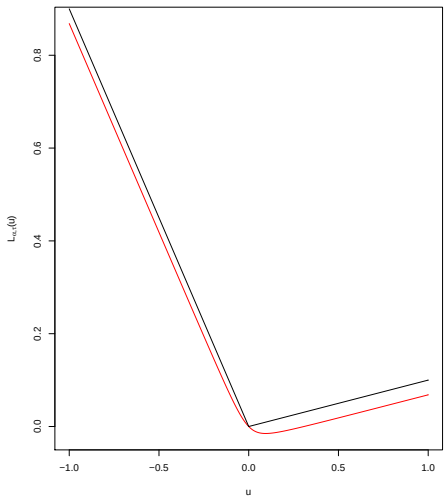


Fig. 10.1: Surrogate loss $\mathcal{L}^{YS}_{\alpha,\tau}(u) = u\tau - u \times \frac{\arctan(-\frac{u}{\alpha})+\frac{\pi}{2}}{\pi}$ derived using the technique from Yilmaz and Sahiner [2019] for $\alpha = 0.1$, $\tau = 0.1$ (red) vs pinball loss (black)

Third, the algorithms described in chapter 7 have a computational order of $\mathcal{O}\left(\left(\frac{m^2}{2} + (B + B_0 + m)\, n\right) p^2 + (B + B_0)\, p^3 + n^{\frac{4}{3}}\right)$, were m is the number of smooths, B is the number of bootstrap samples for the smoothing parameters relaxation, B_0 is the number of bootstrap samples for the confidence interval estimation, n is the number of observations and p is the length of vector $\boldsymbol{\beta}$. To achieve a speed-up of the algorithm, several avenues could be explored. First, the optimization algorithms were programmed in R. They could be reprogrammed in C, for example using Rcpp. Second, other methods such as the ones discussed in section 5.1 could be used to estimate the vector of parameters given fixed values for the vector of smoothing parameters (see also Koenker and Mizera [2014], Koenker [2020a]). This optimization method could then be used to estimate precisely the vector of parameters once the vector of smoothing parameters has been estimated (once we reach a rounding value of α_U in the graduated optimization). It could also be used to accelerate the calculation non-parametric bootstrap estimates. The new optimization method would replace the Generalized IRLS for each non-parametric bootstrap estimate as these calculations assume a fixed vector of smoothing parameters.

Fourth, the comparisons of Chapter 9 could be extended, first, by running the simulation of the coverage of the confidence interval of the aqmm method using package 'aqmm' v1.1 (Geraci [2019b], Geraci [2019a]). Then the simulations of Chapter 9 could be extended to the method of Muggeo et al. [2020], that is now implemented as part of package 'quantRegGrowth' (Muggeo [2021]).

Then as seen in section 4.6.3, the first order approximation used to derive

an ACV criterion may not always be appropriate for every datapoint. We could investigate higher order expansions as in Giordano et al. [2019]. If higher order expansions were to improve the approximation of the LOOCV,it is not clear if a criterion amenable to optimization could be derived. In particular, the QGACV criterion crucially depends on the effective degrees of freedom, itself estimated as the trace of the hat matrix. The trace of the hat matrix can be differentiated and gradient and Hessian for the QGACV can be derived. Ideally, a corrected criterion with higher order expansions would also be a function of the effective degrees of freedom.

Also, other inference methods could be designed to estimate confidence intervals. The non-parametric bootstrap used to estimate confidence intervals is computationally intensive. As in package 'quantreg' (Koenker et al. [2021]) other methods could be implemented in R package 'qgacv' and the user could select the confidence interval method. Potential reference articles include articles comparing different methods: Koenker [1994], Kocherginsky et al. [2005], Tarr [2012], Koenker [2020a], Zhang et al. [2020b], as well as articles focusing on a specific method: Zeng and Lin [2008] who develop resampling strategy for non-smooth estimating equations, Lim and Oh [2015] (based on Krivobokova et al. [2010]) and Koenker [2011] that extend Hotelling tubes to quantile regression, Guo et al. [2021] who apply a bias correction to obtain better confidence intervals and finally Medarametla and Candès [2021], who extend conformal prediction Vovk et al. [2005] to median confidence intervals (see also Romano et al. [2019], Koenker [2020b]).

Another possible extension consists in adapting the methodology to the total variation penalized empirical loss. In section 4.10, we saw that the QGACV could be applicable to the total variation penalized problem as investigated in Koenker [2011], Muggeo et al. [2020]. This seems to indicate that the methodology of this document could be applicable to this case. The difficulty here is that both the loss and the penalty are non-differentiable. We may need to smooth the penalty, using some of the proposals discussed previously (for example, Yilmaz and Sahiner [2019], Amemiya [1982], He et al. [2020] or alternatively Fountoulakis and Gondzio [2016]).

In this document, we noted that using the QGACV, we may obtain models that are too smooth for extreme quantiles. Several proposals have been made to handle extreme quantiles (see for example Chavez-Demoulin and Davison [2005], Yee and Stephenson [2007],Youngman [2020], Yoshida [2020a], Yoshida [2020b]) and could be investigated and integrated in our framework. The extreme quantiles would then be dealt with separately from non-extreme quantiles.

In section 3.4, we replaced the loss function with its expectation, itself replaced by a surrogate loss with a rounding parameter $\hat{\alpha}$ determined using equation 3.15. As we showed in Appendix E that surrogate loss 3.7 can be obtained by kernel smoothing, it could be interesting to contrast our method to obtain

the rounding of the surrogate loss function $\hat{\alpha}$ using equation 3.15 with other methods used to estimate the optimal kernel window as for example in Abo-El-Hadid [2018] or Fasiolo et al. [2020b].

In this document, we focused on estimating a model for a single quantile parameter τ at a time. However, if we estimate several quantiles models simultaneously, they could cross. Several articles have tackled this problem (see for example He [1997], Dette and Volgushev [2008], Shim et al. [2009], Takeuchi et al. [2006], Schnabel and Eilers [2013], Bondell et al. [2010], Cai and Jiang [2015], Chernozhukov et al. [2010]). One possibility consists in adding inequality constraints to the lower level problem to solve (of the type $\mathbf{X}\hat{\beta}_{\tau_1} \leq \mathbf{X}\hat{\beta}_{\tau_2}$).

Although improving speed of computation for our algorithm could be achieved with the methods discussed earlier, the method developed may not be directly applicable to large datasets. Subsampling/row sampling/randomized sketches (Wang and Ma [2020], Ai et al. [2021], Zhang et al. [2020a], Li et al. [2020], Zhang et al. [2021]), matrix-free calculations (Wagner et al. [2021]) or dimension reduction methods (Christou et al. [2021]) could then be studied for both point estimation and inference (see for example Bach et al. [2020]).

The Least Absolute Shrinkage and Selection Operator (LASSO) Tibshirani [1996] is an important method for variable selection. Marra and Wood [2011], Chouldechova and Hastie [2015] extend the LASSO to GAMs whilst Xu et al. [2018] use a LASSO penalty for data reduction for quantile regression. We could aim at extending the methodology described in this document to estimate QAMs to include a LASSO penalty.

Expectiles are an alternative to quantiles. The expectile loss function is a weighted square instead of a weighted absolute value: $\mathcal{L}_{\tau}^{expect}(u) = u^2(\tau - I(u < 0))$ (see Sobotka et al. [2013], Waltrup et al. [2015], Sobotka and Kneib [2012]). We could study the extension of the QGACV to expectile models.

Last, some applications of QAMs have already been investigated. They include childhood malnutrition (Koenker [2011]), electricity load forecasting (Fasiolo et al. [2020b]), children's physical activity (Geraci [2019b]), linguistics (Tomaschek et al. [2018], Miwa and Baayen [2020]), ecology (Waldock et al. [2019], Johnston et al. [2020]), sociology (Lundberg and Stewart [2020]), solar irradiance (Ranganai and Sigauke [2020]). These domains of application could be further investigated and some other applications could be explored. For example, Evangelou and Adams [2020] propose to use Quantile Regression Forests (QRFs) to detect unusual patterns in computer network connections. A preliminary analysis in Data Study Group Team [2019], section 4.5 discusses how QAMs could be used instead of QRFs to obtain flexible and interpretable quantile models of computer network activity. This could be extended by comparing QRFs to QAMs in this case and running the analysis on larger datasets.

APPENDIX

A. SMOOTHING THE LOSS FUNCTION

The table below contains a list of smooth surrogate losses that have been proposed to approximate the exact pinball loss:

Type	**Surrogate loss $\mathcal{L}_{\alpha,\tau}(u)$**	**Article**
piecewise linear/quadratic 1	$\begin{cases} (\tau-1)u & \left(u < -\frac{\alpha}{1-\tau}\right) \\ \frac{((1-\tau)u)^2}{2\alpha} + \frac{\alpha}{2} & \left(-\frac{\alpha}{1-\tau} \leq u < 0\right) \\ \frac{(\tau u)^2}{2\alpha} + \frac{\alpha}{2} & \left(0 \leq u \leq \frac{\alpha}{\tau}\right) \\ \tau u & \left(u > \frac{\alpha}{\tau}\right) \end{cases}$	Zhao et al. [2005]
piecewise linear/quadratic 2	$\begin{cases} (\tau-1)\left(u+\frac{\alpha}{2}\right) & (u < -\alpha) \\ \frac{(1-\tau)u^2}{2\alpha} & (-\alpha \leq u < 0) \\ \frac{\tau u^2}{2\alpha} & (0 \leq u < \alpha) \\ \tau\left(u-\frac{\alpha}{2}\right) & (u > \alpha) \end{cases}$	Oh et al. [2012], Mkhadri et al. [2017], Lee et al. [2012] ($c = \tau/\lambda$)
piecewise linear/quadratic 3	$\begin{cases} (\tau-1)u & (u < -\alpha\tau) \\ \frac{(1-\tau)u^2}{2\alpha\tau} + \frac{\alpha\tau(1-\tau)}{2} & (-\alpha\tau \leq u < 0) \\ \frac{\tau u^2}{2\alpha(1-\tau)} + \frac{\alpha\tau(1-\tau)}{2} & (0 \leq u < (1-\tau)\alpha) \\ \tau u & (u > \alpha(1-\tau)) \end{cases}$	Muggeo et al. [2012], Jennings et al. [1996]
Huberised Loss/Moreau Envelope	$\begin{cases} (1-\tau)\lvert u\rvert - \frac{\alpha(1-\tau)^2}{2} & u < (\tau-1)\alpha \\ \frac{u^2}{2\alpha} & u \in [(\tau-1)\alpha, \tau\alpha] \\ \tau u - \frac{\alpha\tau^2}{2} & u > \tau\alpha \end{cases}$	Aravkin et al. [2014b], Aravkin et al. [2014a], Aravkin et al. [2019], Ramamurthy et al. [2015], Yuan [2006], Reiss and Huang [2012], Chen [2007], Zhang et al. [2017]
Huber-type	$\begin{cases} (\tau-1)u & (x < -\alpha) \\ \frac{u^2}{4\alpha} + \left(\tau-\frac{1}{2}\right)u + \frac{\alpha}{4} & (-\alpha \leq x \leq \alpha) \\ \tau u & (x > \alpha) \end{cases}$	Zhang et al. [2017]

Type	**Surrogate loss $\mathcal{L}_{\alpha,\tau}(u)$**	**Article**
logarithm of one plus exponential	$\tau u + \alpha \log\left(1 + \exp\left(-\frac{u}{\alpha}\right)\right)$	Amemiya [1982], Zheng [2011], Wang and Ye [2016]
Approximation with logarithm of cosh	$\begin{cases} \alpha \log\left(\cosh\left(\frac{\tau u}{\alpha}\right)\right) & u \geq 0 \\ \alpha \log\left(\cosh\left(\frac{(\tau-1)u}{\alpha}\right)\right) & u < 0 \end{cases}$	Yue and Rue [2011]
Approximation used for MM algorithm	$\mathcal{L}_\tau(u) - \frac{\alpha}{2}\log(\alpha + \lvert u\rvert)$	Lv et al. [2016], Hunter and Lange [2000], Zhao and Lian [2016], Lian [2012], Jiang [2015], Xie et al. [2015], Lian et al. [2015], De Backer et al. [2019]
piecewise linear/quadratic 4	$\begin{cases} (\tau-1)u & (u < -\alpha) \\ (1-\tau)\frac{u^2}{\alpha} & -\alpha \leq u < 0 \\ \tau\frac{u^2}{\alpha} & 0 \leq u \leq \alpha \\ \tau\alpha & u > \alpha \end{cases}$	O'Connell and Nychka [1995], Nychka et al. [1995], Chen and Pinar [1998], Wu and Yu [2014]
Infinitely smooth function	$\begin{cases} u\left(\tau - \frac{1}{2} + \int_0^{\frac{u}{\alpha}} \frac{\sin(t)}{\pi t} dt\right) & (u > 0) \\ u\left(\tau - \frac{1}{2}\right) & (u = 0) \\ u\left(\tau - \frac{1}{2} - \int_{\frac{u}{\alpha}}^0 \frac{\sin(t)}{\pi t} dt\right) & (u < 0) \end{cases}$	Wang et al. [2012]
piecewise linear/quadratic 5	$\begin{cases} (\tau-1)u + \frac{\alpha\tau(\tau-1)}{2} & (u < -\tau\alpha) \\ \frac{(1-\tau)}{\tau}\frac{u^2}{2\alpha} & -\alpha\tau \leq u < 0 \\ \frac{\tau}{(1-\tau)}\frac{u^2}{2\alpha} & 0 \leq u < \alpha(1-\tau) \\ \tau u + \frac{\alpha\tau(\tau-1)}{2} & u \geq \alpha(1-\tau) \end{cases}$	Jung et al. [2020]. Note that this loss function corresponds to piecewise linear quadratic 3 Muggeo et al. [2012] minus $(\frac{\alpha\tau(1-\tau)}{2})$
Convolution based smoothing	$\frac{1}{\alpha}\int_{-\infty}^{+\infty} v\left(\tau - \mathcal{I}(v < 0)\right) K\left(\frac{v-u}{\alpha}\right) dv$, where K is a kernel, α is the window parameter (usually called h). Note that the logarithm of one plus exponential (Zheng [2011], Amemiya [1982]) is a special case of this framework.	Fernandes et al. [2021], He et al. [2020]

B. SQUARED DISTANCE BETWEEN THE SURROGATE LOSS WITH A FIXED ROUNDING AND THE SURROGATE LOSS WITH A VARYING ROUNDING

The Figure below shows that the sharp minimum observed on Figure 3.9 is an inherent characteristic of squared distance as long as there exists an α for which the surrogate loss can match the expected loss (itself replaced by an empirical loss). On the Figure, we show the squared distance between the surrogate loss with a fixed rounding α and the same surrogate loss with different values of α on a grid. We first take the surrogate loss with a parameter $\alpha = 0.5$ (displayed in dotted red) evaluated on a regularly spaced grid $u_{grid} = -2, -1.99, -1.98, ..., 2$ (we call the number of values G). We then evaluate a second surrogate loss on the same u_{grid} but with increasing values for α: $\alpha_{grid} = \{10^{-2}, 10^{-2.95}, ..., 10^{1}\}$. We then calculate:

$$SquaredDistance\left(\alpha\right) = \frac{1}{G}\sum_{i=1}^{G}\left(\mathcal{L}_{0.5,\tau}\left(u_i\right) - \mathcal{L}_{\alpha,\tau}\left(u_i\right)\right)^2$$

No value of α on α_{grid} is exactly equal to 0.5. However, for $\alpha = 0.5011$, we see that the Squared distance has a sharp minimum. This explains the sharp minimum that can be observed on Figure 3.9.

However, in some cases, it is impossible for the surrogate loss to match closely the expected loss replaced by an empirical loss. On Figure B.2, we show an example similar to Figure 3.9. However, instead of generating the data with model 1 (see section 9.1), we generate data using model 9 that is bimodal. We can see on the middle chart in red that the surrogate loss cannot match the empirical loss. On the bottom chart, we can then see that the squared distance has now a minimum that is less sharp.

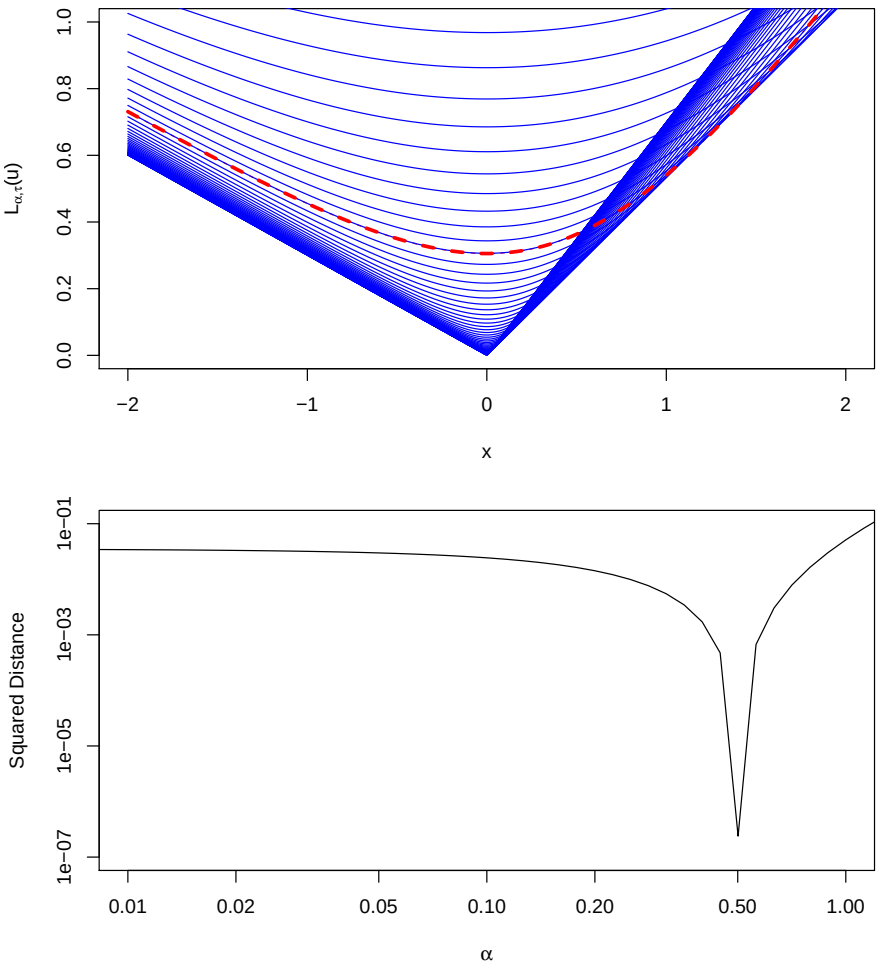


Fig. B.1: On these two charts, we calculate the squared distance between the surrogate loss with a parameter $\alpha = 0.5$ and the same surrogate loss but rounded with a parameter α that varies 10^{-3} and 10^{1}. We use $\tau = 0.7$. On the top chart, we can see the surrogate loss with rounding parameter $\alpha = 0.5$ in dotted red. The blue curves show the same surrogate loss with varying α parameters. On the bottom chart, we show the squared distance: $SquaredDistance\,(\alpha) = \frac{1}{G}\sum_{i=1}^{G}\left(\mathcal{L}_{0.5,\tau}\,(u_i) - \mathcal{L}_{\alpha,\tau}\,(u_i)\right)^2$. We can see that as in the example of Figure 3.9, the squared distance has a sharp minimum. This shows that if the surrogate loss can match well the expected loss itself replaced by an empirical loss, we can expect to obtain a sharp minimum to the squared distance.

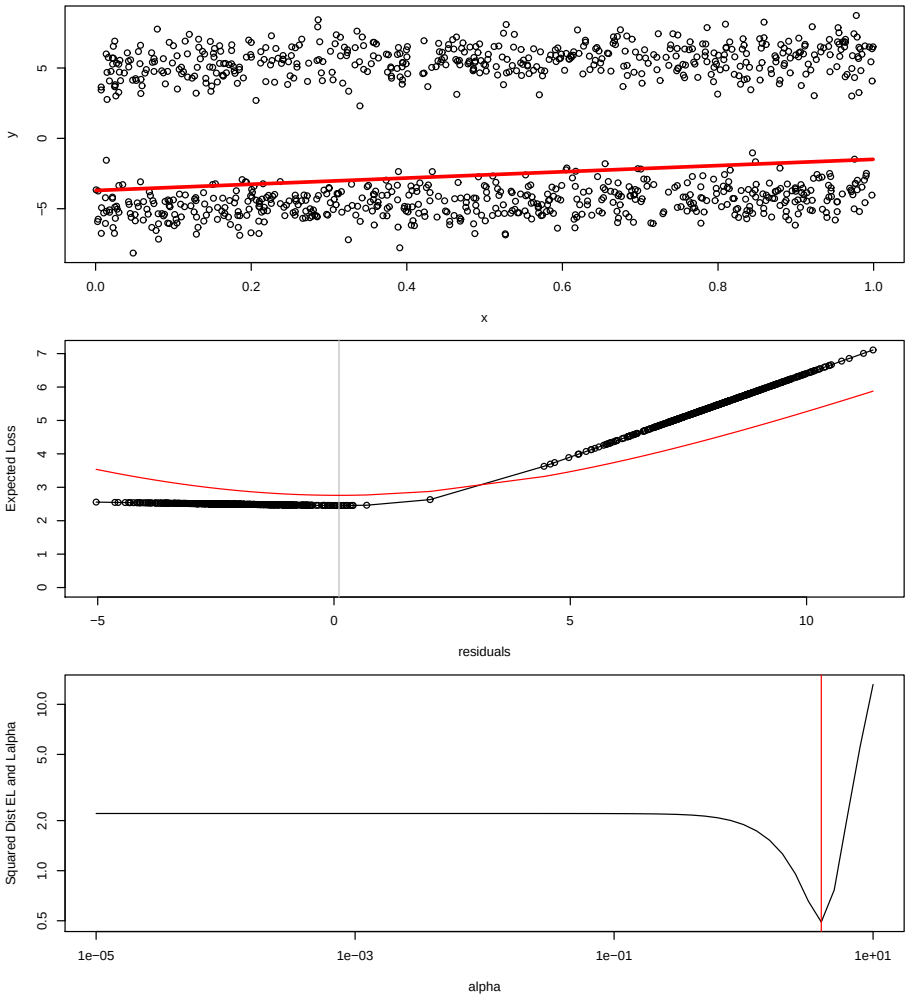


Fig. B.2: On this Figure, we show how in some cases, the surrogate loss is unable to match closely the expected loss evaluated as an empirical loss closely. This leads to a less sharp minimum for the squared distance. We generate data with example 9 (see section 9.1), $n = 1,000$, seed=2, $\tau = 0.5$ with qgacv function qgacvExamples. The data is bimodal. We use $n = 1,000$ datapoints generated with model 9. The top chart shows the fit. The middle chart shows the surrogate loss with the selected $\hat{\alpha}$ vs the expected loss replaced by an empirical loss (black). We can see that because the data is bimodal, the surrogate loss cannot match the empirical loss. On the bottom chart, we can see that the squared distance has a less sharp minimum because of the mismatch between the surrogate loss and the expected loss.

C. DERIVATIVES OF THE PENALIZED EMPIRICAL RISK AND THE QGACV

In this section, we derive the expressions for the derivatives necessary to implement the Generalized Fellner-Schall and Hyper-Newton algorithms. These derivatives are the gradient and Hessian of the penalized empirical risk (see section C.1 for the definition of $l^{Pen}_{\alpha_1,\alpha_2}$).

$$\mathcal{G}^{Pen}_{\alpha_1,\alpha_2} = \frac{\partial l^{Pen}_{\alpha_1,\alpha_2}}{\partial \boldsymbol{\beta}}$$

$$\mathcal{H}^{Pen}_{\alpha_1,\alpha_2} = \frac{\partial^2 l^{Pen}_{\alpha_1,\alpha_2}}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T}$$

and the total gradient and total Hessian of the QGACV. We refer to "total" as these derivatives are obtained using implicit differentiation. As in Wood [2011], for a give function h, we use notation $\frac{\partial h(x)}{\partial x}$ to indicate a partial derivative and $\frac{dh(x)}{dx}$ to indicate a total derivative. The techniques used to obtain the derivatives in this section are similar to the techniques used to obtain derivatives in Wood [2011].

$$\mathcal{G}^{QGACV}_{\alpha_1,\alpha_2} = \frac{dQGACV_{\alpha_1,\alpha_2}}{d\boldsymbol{\rho}}$$

$$\mathcal{H}^{QGACV}_{\alpha_1,\alpha_2} = \frac{d^2 QGACV_{\alpha_1,\alpha_2}}{d\boldsymbol{\rho} d\boldsymbol{\rho}^T}$$

Note that in the following section, we may also use the Hessian and gradient of the unpenalized empirical risk (see section C.1 for the definition of $l^{UnPen}_{\alpha_1,\alpha_2}$):

$$\mathcal{G}^{UnPen}_{\alpha_1,\alpha_2} = \frac{\partial l^{UnPen}_{\alpha_1,\alpha_2}}{\partial \boldsymbol{\beta}}$$

$$\mathcal{H}^{Pen}_{\alpha_1,\alpha_2} = \frac{\partial^2 l^{UnPen}_{\alpha_1,\alpha_2}}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T}$$

C.1 Formulae to differentiate

We start from the expression of the QGACV:

$$QGACV_{\hat{\alpha},\alpha} = \sum_{i=1}^{n} \frac{\mathcal{L}_{\alpha,\tau}\left(y_i - \mathbf{x}_i^T \tilde{\boldsymbol{\beta}}_\alpha\right)}{n.eff - ctr\left(\mathbf{A}_{\hat{\alpha},\alpha}\right)}$$

where:

$$\mathbf{A}_{\hat{\alpha},\alpha} = \mathbf{X}\left(\mathbf{X}^T\mathbf{W}_{\hat{\alpha},\alpha}\mathbf{X} + \mathbf{S}_{\boldsymbol{\lambda}}\right)^{-1}\mathbf{X}^T\mathbf{W}_{\hat{\alpha},\alpha}$$

Matrix $(\mathbf{A}_{\hat{\alpha},\alpha})$ is an $n \times n$ matrix. We only use the diagonal of this matrix in the formula. Instead of calculating the hat matrix directly, we can use a well known relationship. As in Wood [2017], section 6.1.2, pp.251-252, we define matrix $\mathbf{F}$ as:

$$\tilde{\mathbf{F}}_{\hat{\alpha},\alpha} = \left(\mathbf{X}^T\tilde{\mathbf{W}}_{\hat{\alpha},\alpha}\mathbf{X} + \mathbf{S}_{\boldsymbol{\lambda}}\right)^{-1}\left(\mathbf{X}^T\tilde{\mathbf{W}}_{\hat{\alpha},\alpha}\mathbf{X}\right) \tag{C.1}$$

We can then use a trace equality formula (see Petersen and Pedersen [2012], Wood [2017] p.419): if we have two matrices $\mathbf{M}_1$ and $\mathbf{M}_2$, then $tr(\mathbf{M}_1\mathbf{M}_2) = tr(\mathbf{M}_2\mathbf{M}_1)$. Then: $tr(\mathbf{A}_{\hat{\alpha},\alpha}) = tr(\mathbf{F}_{\hat{\alpha},\alpha})$. We will then use the following QGACV criterion:

$$QGACV_{\hat{\alpha},\alpha} = \sum_{i=1}^{n}\frac{\mathcal{L}_{\alpha,\tau}\left(y_i - \mathbf{x}_i^T\tilde{\boldsymbol{\beta}}_{\alpha}\right)}{n.eff - ctr\left(\mathbf{F}_{\hat{\alpha},\alpha}\right)}$$

where:

$$\mathbf{W}_{\hat{\alpha},\alpha} = diag\left(\mathcal{L}''_{\hat{\alpha},\tau}\left(y_1 - \mathbf{x}_1^T\tilde{\boldsymbol{\beta}}_{\alpha}\right), \ldots, \mathcal{L}''_{\hat{\alpha},\tau}\left(y_n - \mathbf{x}_n^T\tilde{\boldsymbol{\beta}}_{\alpha}\right)\right)$$

Calculating the diagonal is $\mathcal{O}(n)$. We also have:

$$\tilde{u}_{\alpha,i} = \left(y_i - \mathbf{x}_i^T\tilde{\boldsymbol{\beta}}_{\alpha}\right)$$

$$l^{UnPen}_{\alpha_1,\alpha_2} = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_{\alpha_1,\tau}\left(\tilde{u}_{\alpha_2,i}\right)$$

$$l^{Pen}_{\alpha_1,\alpha_2} = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_{\alpha_1,\tau}\left(\tilde{u}_{\alpha_2,i}\right) + \frac{1}{2}\boldsymbol{\beta}^T\left(\sum_{k=1}^{m}e^{\rho_k}\mathbf{S}_k\right)\boldsymbol{\beta}$$

$$\tilde{\boldsymbol{\beta}}_{\alpha} = \text{argmin}_{\beta}\,\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_{\alpha,\tau}\left(y_i - \mathbf{x}_i^T\boldsymbol{\beta}\right) + \frac{1}{2}\boldsymbol{\beta}^T\left(\sum_{k=1}^{m}e^{\rho_k}\mathbf{S}_k\right)\boldsymbol{\beta}$$

Computational Cost: the product of the diagonal matrix $\mathbf{W}_{\hat{\alpha},\alpha}$ by matrix $\mathbf{X}$ can be efficiently computed as a vector matrix multiplication by-rows. The cost is then $\mathcal{O}(np)$. The multiplication of $(p \times n)$ matrix $\mathbf{X}^T$ by $(n \times p)$ matrix $\mathbf{W}_{\hat{\alpha},\alpha}\mathbf{X}$ is then $\mathcal{O}(np^2)$ (see Wood [2015] p.220). The Cholesky decomposition to calculate the inverse can then be performed in $\mathcal{O}(p^2)$ (see Wood [2015] p.223). The product of the two $p \times p$ matrices can be made in $\mathcal{O}(p^3)$. In total, we have: $\mathcal{O}(np^2 + p^3)$.

C.2 First order derivative of the QGACV and derivatives of the penalized empirical risk

$$\frac{dQGACV_{\hat{\alpha},\alpha}}{d\rho_k} = \frac{\left(\sum_{i=1}^{n} \mathcal{L}'_{\alpha,\tau}\left(\tilde{u}_{\alpha,i}\right) \frac{d\tilde{u}_{\alpha,i}}{d\rho_k}\right)\left(n.eff - ctr\left(\mathbf{F}_{\hat{\alpha},\alpha}\right)\right)}{\left(n.eff - ctr\left(\mathbf{F}_{\hat{\alpha},\alpha}\right)\right)^2} + \frac{c\left(\sum_{i=1}^{n} \mathcal{L}_{\alpha,\tau}\left(\tilde{u}_{\alpha,i}\right)\right) \frac{dtr(\mathbf{F}_{\hat{\alpha},\alpha})}{d\rho_k}}{\left(n.eff - ctr\left(\mathbf{F}_{\hat{\alpha},\alpha}\right)\right)^2}$$

For each smoothing parameter, the computational cost of the formula is $\mathcal{O}(n)$ (only calculating the formula above, assuming all the components have already been calculated). The whole vector is then $\mathcal{O}(nm)$.

We are now going to detail each component of the first order derivative of the QGACV formula that has not yet been calculated. Using a similar reasoning to Wood [2011], Appendix C.2, the total derivative of the estimate of the vector of parameters with respect to smoothing parameter ρ_k can be obtained by implicit differentiation. (see also Gould et al. [2016]):

$$\left(\frac{d\tilde{u}_{\alpha,i}}{d\rho_k}\right) = \left(-\mathbf{x}_i^T \frac{d\tilde{\boldsymbol{\beta}}_\alpha}{d\rho_k}\right)$$

For each smoothing parameter and each observation, this calculation is an $\mathcal{O}(p)$ calculation. In total, we then have: $\mathcal{O}(mnp)$. As explained in Wood [2017] p.281, at $\tilde{\boldsymbol{\beta}}_\alpha$, we have:

$$\frac{dl^{Pen}_{\alpha,\alpha}}{d\boldsymbol{\beta}} = 0$$

We can differentiate this with respect to ρ_k:

$$\frac{d^2 l^{Pen}_{\alpha,\alpha}}{d\boldsymbol{\beta} d\rho_k} = \left(\frac{\partial^2 l^{Pen}_{\alpha,\alpha}}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T}\right)\frac{d\boldsymbol{\beta}}{d\rho_k} + \frac{\partial^2 l^{Pen}_{\alpha,\alpha}}{\partial\boldsymbol{\beta}\partial\rho_k} = 0$$

Hence:

$$\begin{aligned}
\frac{d\tilde{\boldsymbol{\beta}}_\alpha}{d\rho_k} &= -\left[\left(\frac{\partial^2 l^{Pen}_{\alpha,\alpha}}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T}\right)^{-1}\left(\frac{\partial^2 l^{Pen}_{\alpha,\alpha}}{\partial\boldsymbol{\beta}\partial\rho_k}\right)\right]_{\boldsymbol{\beta}=\tilde{\boldsymbol{\beta}}_\alpha} \\
&= -\left[\left(\frac{1}{n}\mathbf{X}^T\mathbf{W}_{\alpha,\alpha}\mathbf{X} + \mathbf{S}_{\boldsymbol{\rho}}\right)^{-1}\left(e^{\rho_k}\mathbf{S}_k\tilde{\boldsymbol{\beta}}_\alpha\right)\right] \\
&= -\left[\left(\mathcal{H}^{Pen}_{\alpha,\alpha}\right)^{-1}\left(e^{\rho_k}\mathbf{S}_k\tilde{\boldsymbol{\beta}}_\alpha\right)\right]
\end{aligned}$$

We assume that the inverse matrix has been calculated already. We then have a vector matrix product of order $\mathcal{O}(p^2)$ per smooth. The total cost is

then $\mathcal{O}(mp^2)$ to obtain the vector of total derivatives. To calculate the total derivative of the trace of $\mathbf{F}_{\hat{\alpha},\alpha}$ with respect to the smoothing parameters, we note that $\mathbf{F}_{\hat{\alpha},\alpha}$ can be written:

$$\mathbf{F}_{\hat{\alpha},\alpha} = \left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)^{-1} \left(\mathcal{H}^{UnPen}_{\hat{\alpha},\alpha}\right)$$

Using an equality that can be found in Gentle [2017] p.195, Petersen and Pedersen [2012]:

$$\frac{dtr\left(A^{-1}B\right)}{dx} = tr\left(-A^{-1}\frac{dA}{dx}A^{-1}B + A^{-1}\frac{dB}{dx}\right)$$

hence:

$$\begin{aligned}\frac{dtr\left(\mathbf{F}_{\hat{\alpha},\alpha}\right)}{d\rho_k} = tr\Bigg(&-\left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)^{-1}\frac{d\left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)}{d\rho_k}\left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)^{-1}\left(\mathcal{H}^{UnPen}_{\hat{\alpha},\alpha}\right)... \\ &+\left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)^{-1}\frac{d\left(\mathcal{H}^{UnPen}_{\hat{\alpha},\alpha}\right)}{d\rho_k}\Bigg)\end{aligned}$$

$$\begin{aligned}\frac{\partial l^{UnPen}_{\alpha_1,\alpha_2}}{\partial\boldsymbol{\beta}} &= \left(-\frac{1}{n}\sum_i \mathcal{L}'_{\alpha_1,\tau}\left(u_{\alpha_2,i}\right)\mathbf{x}_i\right) \\ \frac{\partial l^{Pen}_{\alpha_1,\alpha_2}}{\partial\boldsymbol{\beta}} &= \frac{\partial l^{UnPen}_{\alpha_1,\alpha_2}}{\partial\boldsymbol{\beta}} + \left(\sum_k e^{\rho_k}\mathbf{S}_k\right)\boldsymbol{\beta} \\ \mathcal{H}^{UnPen}_{\alpha_1,\alpha_2} &= \frac{\partial^2 l^{UnPen}_{\alpha_1,\alpha_2}}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T} = \frac{1}{n}\sum_i \mathcal{L}''_{\alpha_1,\tau}\left(\tilde{u}_{\alpha_2,i}\right)\mathbf{x}_i\mathbf{x}_i^T \\ \mathcal{H}^{Pen}_{\alpha_1,\alpha_2} &= \frac{\partial^2 l^{Pen}_{\alpha_1,\alpha_2}}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T} = \frac{\partial^2 l^{UnPen}_{\alpha_1,\alpha_2}}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T} + \left(\sum_k e^{\rho_k}\mathbf{S}_k\right)\end{aligned}$$

The surrogate loss and its derivatives up to 4th order are then:

$$\mathcal{L}_{\alpha,\tau}(u) = \tau(u+\gamma) + \alpha \log\left(1 + \exp\left(-\frac{(u+\gamma)}{\alpha}\right)\right)$$

$$\mathcal{L}'_{\alpha,\tau}(u) = \left(\tau - \frac{\exp\left(-\frac{(u+\gamma)}{\alpha}\right)}{1+\exp\left(-\frac{(u+\gamma)}{\alpha}\right)}\right) = \left(\tau - \frac{1}{\exp\left(+\frac{(u+\gamma)}{\alpha}\right)+1}\right)$$

$$\mathcal{L}''_{\alpha,\tau}(u) = \frac{1}{\alpha}\frac{\exp\left(-\frac{(u+\gamma)}{\alpha}\right)}{\left(1+\exp\left(-\frac{(u+\gamma)}{\alpha}\right)\right)^2} = \frac{1}{\alpha}\frac{\exp\left(+\frac{(u+\gamma)}{\alpha}\right)}{\left(\exp\left(+\frac{(u+\gamma)}{\alpha}\right)+1\right)^2}$$

$$\mathcal{L}'''_{\alpha,\tau}(u) = \frac{1}{\alpha^2}\left[\frac{\exp\left(-\frac{2(u+\gamma)}{\alpha}\right) - \exp\left(-\frac{(u+\gamma)}{\alpha}\right)}{\left(1+\exp\left(-\frac{(u+\gamma)}{\alpha}\right)\right)^3}\right] = \frac{1}{\alpha^2}\left[\frac{\exp\left(+\frac{(u+\gamma)}{\alpha}\right) - \exp\left(+\frac{2(u+\gamma)}{\alpha}\right)}{\left(\exp\left(+\frac{(u+\gamma)}{\alpha}\right)+1\right)^3}\right]$$

$$\begin{aligned}\mathcal{L}^{(4)}_{\alpha,\tau}(u) &= \frac{1}{\alpha^3}\left[\frac{\exp\left(-\frac{(u+\gamma)}{\alpha}\right) + \exp\left(-\frac{3(u+\gamma)}{\alpha}\right) - 4\exp\left(-\frac{2(u+\gamma)}{\alpha}\right)}{\left(1+\exp\left(-\frac{(u+\gamma)}{\alpha}\right)\right)^4}\right] \\ &= \frac{1}{\alpha^3}\left[\frac{\exp\left(+\frac{3(u+\gamma)}{\alpha}\right) + \exp\left(+\frac{((u+\gamma)+\gamma)}{\alpha}\right) - 4\exp\left(+\frac{2(u+\gamma)}{\alpha}\right)}{\left(\exp\left(+\frac{(u+\gamma)}{\alpha}\right)+1\right)^4}\right]\end{aligned}$$

Finally:

$$\frac{d\mathcal{H}^{UnPen}_{\alpha_1,\alpha_2}}{d\rho_k} = \frac{1}{n}\mathbf{X}^T\left(\frac{d\mathbf{W}_{\alpha_1,\alpha_2}}{d\rho_k}\right)\mathbf{X}$$

$$\frac{d\mathcal{H}^{UnPen}_{\hat{\alpha},\alpha}}{d\rho_k} = \frac{1}{n}\mathbf{X}^T\left(\frac{d\mathbf{W}_{\hat{\alpha},\alpha}}{d\rho_k}\right)\mathbf{X}$$

$$\frac{d\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}}{d\rho_k} = \frac{d\mathcal{H}^{UnPen}_{\hat{\alpha},\alpha}}{d\rho_k} + e^{\rho_k}\mathbf{S}_k$$

$$\frac{dw_{\hat{\alpha},\alpha,i}}{d\rho_k} = \frac{\partial^3\mathcal{L}_{\hat{\alpha},\tau}(\tilde{u}_{\alpha,i})}{\partial\tilde{u}^3_{\alpha,i}}\frac{\partial\tilde{u}_{\alpha,i}}{\partial\tilde{\boldsymbol{\beta}}^T_\alpha}\frac{d\tilde{\boldsymbol{\beta}}_\alpha}{d\rho_k} = -\mathcal{L}'''_{\hat{\alpha},\tau}(\tilde{u}_{\alpha,i})\,\mathbf{x}_i^T\frac{d\tilde{\boldsymbol{\beta}}_\alpha}{d\rho_k}$$

C.3 Second order derivatives of the QGACV

We can also calculate the second order derivatives:

$$\begin{aligned}\frac{d^2 QGACV_{\hat{\alpha},\alpha}}{d\rho_k d\rho_\ell} = & \frac{\sum_{i=1}^n \mathcal{L}''_{\alpha,\tau}\left(\tilde{u}_{\alpha,i}\right) \frac{d\tilde{u}_{\alpha,i}}{d\rho_k} \frac{d\tilde{u}_{\alpha,i}}{d\rho_\ell}}{\left(n.eff - ctr\left(\mathbf{F}_{\hat{\alpha},\alpha}\right)\right)} \\ & + \frac{\sum_{i=1}^n \mathcal{L}'_{\alpha,\tau}\left(\tilde{u}_{\alpha,i}\right) \frac{d^2\tilde{u}_{\alpha,i}}{d\rho_k d\rho_\ell}}{\left(n.eff - ctr\left(\mathbf{F}_{\hat{\alpha},\alpha}\right)\right)} \\ & + c\frac{\sum_{i=1}^n \mathcal{L}'_{\alpha,\tau}\left(\tilde{u}_{\alpha,i}\right) \frac{d\tilde{u}_{\alpha,i}}{d\rho_k} \frac{dtr(\mathbf{F}_{\hat{\alpha},\alpha})}{d\rho_\ell}}{\left(n.eff - ctr\left(\mathbf{F}_{\hat{\alpha},\alpha}\right)\right)^2} \\ & + c\frac{\sum_{i=1}^n \mathcal{L}'_{\alpha,\tau}\left(\tilde{u}_{\alpha,i}\right) \frac{d\tilde{u}_{\alpha,i}}{d\rho_\ell} \frac{dtr(\mathbf{F}_{\hat{\alpha},\alpha})}{d\rho_k}}{\left(n.eff - ctr\left(\mathbf{F}_{\hat{\alpha},\alpha}\right)\right)^2} \\ & + c\frac{\sum_{i=1}^n \mathcal{L}_{\alpha,\tau}\left(\tilde{u}_{\alpha,i}\right) \frac{d^2 tr(\mathbf{F}_{\hat{\alpha},\alpha})}{d\rho_k d\rho_\ell}}{\left(n.eff - ctr\left(\mathbf{F}_{\hat{\alpha},\alpha}\right)\right)^2} \\ & + \frac{2c^2 \sum_{i=1}^n \mathcal{L}_{\alpha,\tau}\left(\tilde{u}_{\alpha,i}\right) \frac{dtr(\mathbf{F}_{\hat{\alpha},\alpha})}{d\rho_k} \frac{dtr(\mathbf{F}_{\hat{\alpha},\alpha})}{d\rho_\ell}}{\left(n.eff - ctr\left(\mathbf{F}_{\hat{\alpha},\alpha}\right)\right)^3}\end{aligned}$$

$$\left(\frac{d^2\tilde{u}_{\alpha,i}}{d\rho_k d\rho_\ell}\right) = \left(-\mathbf{x}_i^T \frac{d^2\tilde{\boldsymbol{\beta}}_\alpha}{d\rho_k d\rho_\ell}\right)$$

$$\begin{aligned}\frac{d^2\tilde{\boldsymbol{\beta}}_\alpha}{d\rho_k d\rho_\ell} &= -\frac{d}{d\rho_\ell}\left[\left(\mathcal{H}_{\alpha,\alpha}^{Pen}\right)^{-1}\left(e^{\rho_k}\mathbf{S}_k\tilde{\boldsymbol{\beta}}_\alpha\right)\right] \\ &= \left[\left(\mathcal{H}_{\alpha,\alpha}^{Pen}\right)^{-1}\left[\frac{d\mathcal{H}_{\alpha,\alpha}^{Pen}}{d\rho_\ell}\right]\left(\mathcal{H}_{\alpha,\alpha}^{Pen}\right)^{-1}\left(e^{\rho_k}\mathbf{S}_k\tilde{\boldsymbol{\beta}}_\alpha\right) - \left(\mathcal{H}_{\alpha,\alpha}^{Pen}\right)^{-1}\left(e^{\rho_k}\mathbf{S}_k\frac{d\tilde{\boldsymbol{\beta}}_\alpha}{d\rho_\ell}\right)\right] \\ &= \left[\left(\mathcal{H}_{\alpha,\alpha}^{Pen}\right)^{-1}\left[\frac{d\mathcal{H}_{\alpha,\alpha}^{Pen}}{d\rho_\ell}\right]\left(\mathcal{H}_{\alpha,\alpha}^{Pen}\right)^{-1}\left(e^{\rho_k}\mathbf{S}_k\tilde{\boldsymbol{\beta}}_\alpha\right) + e^{\rho_k}e^{\rho_\ell}\left(\mathcal{H}_{\alpha,\alpha}^{Pen}\right)^{-1}\mathbf{S}_k\left(\mathcal{H}_{\alpha,\alpha}^{Pen}\right)^{-1}\mathbf{S}_\ell\tilde{\boldsymbol{\beta}}_\alpha\right]\end{aligned}$$

The current version of the algorithm calculates matrix-matrix products and calculates the matrix inverse using function 'solve'. This results in a computational cost of $\mathcal{O}(\frac{m(m+1)}{2}p^3)$. This could be optimized by forming only vector matrix products and using a Cholesky decomposition this could be reduced to $\mathcal{O}(\frac{m(m+1)}{2}p^2 + p^3)$.

$$\frac{d\mathcal{H}_{\alpha,\alpha}^{Pen}}{d\rho_k} = \frac{d\mathcal{H}_{\alpha,\alpha}^{UnPen}}{d\rho_k} + e^{\rho_k}\mathbf{S}_k$$

$$\begin{aligned}\frac{dw_{\alpha,\alpha,i}}{d\rho_k} &= \frac{\partial^3\mathcal{L}_{\alpha,\tau}\left(\tilde{u}_{\alpha,i}\right)}{\partial\tilde{u}_{\alpha,i}^3}\frac{\partial\tilde{u}_{\alpha,i}}{\partial\tilde{\boldsymbol{\beta}}_\alpha^T}\frac{d\tilde{\boldsymbol{\beta}}_\alpha}{d\rho_k} \\ &= -\mathcal{L}'''_{\alpha,\tau}\left(\tilde{u}_{\alpha,i}\right)\mathbf{x}_i^T\frac{d\tilde{\boldsymbol{\beta}}_\alpha}{d\rho_k}\end{aligned}$$

$$\begin{aligned}\frac{d^2 tr\left(\mathbf{F}_{\hat{\alpha},\alpha}\right)}{d\rho_k d\rho_\ell} = tr\Bigg(& \left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)^{-1} \frac{d\left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)}{d\rho_\ell} \left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)^{-1} \frac{d\left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)}{d\rho_k} \left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)^{-1} \left(\mathcal{H}^{UnPen}_{\hat{\alpha},\alpha}\right) \\ & - \left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)^{-1} \frac{d^2\left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)}{d\rho_k d\rho_\ell} \left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)^{-1} \left(\mathcal{H}^{UnPen}_{\hat{\alpha},\alpha}\right) \\ & + \left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)^{-1} \frac{d\left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)}{d\rho_k} \left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)^{-1} \frac{d\left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)}{d\rho_\ell} \left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)^{-1} \left(\mathcal{H}^{UnPen}_{\hat{\alpha},\alpha}\right) \\ & - \left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)^{-1} \frac{d\left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)}{d\rho_k} \left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)^{-1} \frac{d\left(\mathcal{H}^{UnPen}_{\hat{\alpha},\alpha}\right)}{d\rho_\ell} \\ & - \left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)^{-1} \frac{d\left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)}{d\rho_\ell} \left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)^{-1} \frac{d\left(\mathcal{H}^{UnPen}_{\hat{\alpha},\alpha}\right)}{d\rho_k} \\ & + \left(\mathcal{H}^{Pen}_{\hat{\alpha},\alpha}\right)^{-1} \frac{d^2\left(\mathcal{H}^{UnPen}_{\hat{\alpha},\alpha}\right)}{d\rho_k d\rho_\ell} \Bigg)\end{aligned}$$

$$\begin{aligned}\frac{d^2 w_{\hat{\alpha},\alpha,i}}{d\rho_k d\rho_\ell} &= \frac{\partial^4 \mathcal{L}_{\hat{\alpha},\tau}\left(\tilde{u}_{\alpha,i}\right)}{\partial \tilde{u}^4_{\alpha,i}} \frac{\partial \tilde{u}_{\alpha,i}}{\partial \tilde{\boldsymbol{\beta}}^T_\alpha} \frac{d\tilde{\boldsymbol{\beta}}_\alpha}{d\rho_\ell} \frac{\partial \tilde{u}_{\alpha,i}}{\partial \tilde{\boldsymbol{\beta}}^T_\alpha} \frac{d\tilde{\boldsymbol{\beta}}_\alpha}{d\rho_k} + \frac{\partial^3 \mathcal{L}_{\hat{\alpha},\tau}\left(\tilde{u}_{\alpha,i}\right)}{\partial \tilde{u}^3_{\alpha,i}} \frac{\partial \tilde{u}_{\alpha,i}}{\partial \tilde{\boldsymbol{\beta}}^T_\alpha} \frac{d^2\tilde{\boldsymbol{\beta}}_\alpha}{d\rho_k d\rho_\ell} \\ &= \mathcal{L}^{(4)}_{\hat{\alpha},\tau}\left(\tilde{u}_{\alpha,i}\right) \mathbf{x}_i^T \frac{d\tilde{\boldsymbol{\beta}}_\alpha}{d\rho_\ell} \mathbf{x}_i^T \frac{d\tilde{\boldsymbol{\beta}}_\alpha}{d\rho_k} - \mathcal{L}'''_{\hat{\alpha},\tau}\left(\tilde{u}_{\alpha,i}\right) \mathbf{x}_i^T \frac{d^2\tilde{\boldsymbol{\beta}}_\alpha}{d\rho_k d\rho_\ell}\end{aligned}$$

D. EXAMPLES FROM CHAPTER 9

This Appendix contains the examples discussed in Chapter 9.

D.1 An example where the smoothness of the central quantile is different from the smoothness of non-central quantiles

```
dataEx<-qgacvExamples(8,n=1000,seed=1)$data
dataEx<-data.frame(x=dataEx$x,y=dataEx$y,id=1)
aqmmFit<-aqmm(y~s(x,k=30),data=dataEx,random=~1,group=id,tau=0.95)
fit<-qam(y~s(x,k=30),data=dataEx,tau=0.95)
```

As can be seen from the chart below, the aqmm package fit relies on the central quantile. When the data has a very different degree of smoothness for the central quantile and non-central quantiles, the fits for non-central quantiles may not have the correct degree of smoothness.

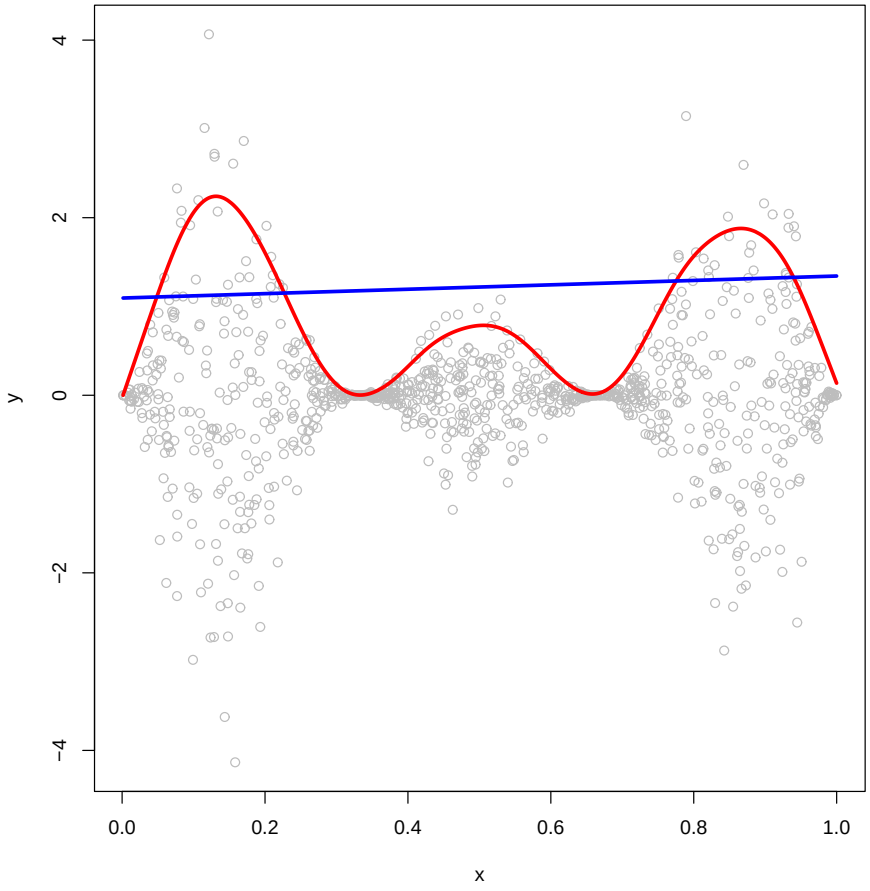


Fig. D.1: Example fit with aqmm, tau=0.95, n=1000. Red: qam fit. Blue: aqmm fit.

D.2 Multimodal data

```
dataEx<-qgacv::qgacvExamples(9,n=5000,seed=5)$data
#vgam with family=lms.bcn(zero=1) requires a positive y
dataEx$y<-dataEx$y-min(dataEx$y)+1
#vgam fits all quantiles simultaneously
fitvgam <- VGAM::vgam(y ~ s(x, df = 4), lms.bcn(zero=1), data = dataEx,maxit=100)
fitqam49<-qgacv::qam(y ~ s(x),data=dataEx,tau=0.49);
fitqam51<-qgacv::qam(y ~ s(x),data=dataEx,tau=0.51)
fitqgam49<-qgam::qgam(y ~ s(x),data=dataEx,qu=0.49);
fitqgam51<-qgam::qgam(y ~ s(x),data=dataEx,qu=0.51)
fitqr49<-quantreg::rq(y~x,data=dataEx,tau=0.49);
fitqr51<-quantreg::rq(y~x,data=dataEx,tau=0.51)
```

We fit quantiles $\tau = 0.49$ and $\tau = 0.51$ with qgacv, qgam, VGAM and quantreg. Because the underlying generating model is linear, we can also fit a linear quantile regression with package 'quantreg'.

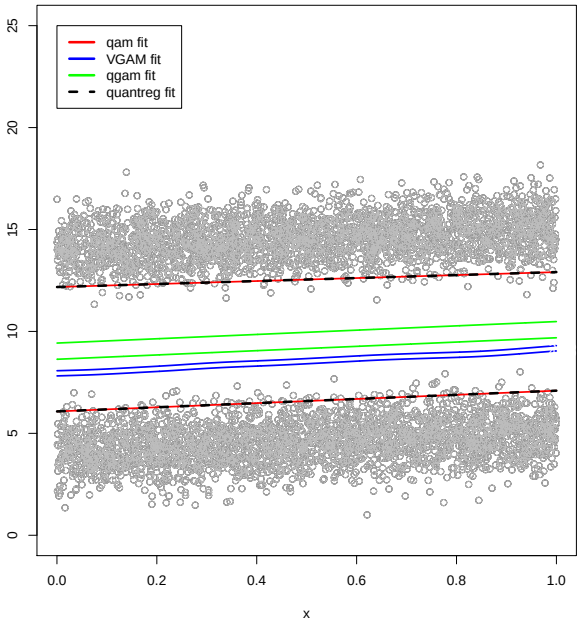


Fig. D.2: We fit two quantiles: $\tau = 0.49$ and $\tau = 0.51$ with 4 different packages. Because the underlying model is linear, we fit linear quantile regression (dotted black) with package 'quantreg'. We also with the data with package 'VGAM' (blue), 'qgam'(green) and 'qgacv'(red). Only the 'qgacv' package obtains fits at the correct quantile that match almost exactly the 'quantreg' package fit.

D.3 Biased estimation using qgam

```
n<-100;tau<-0.05;set.seed(33)
x<-sort(runif(n))
y<-x+((x+1)^2)*stabledist::rstable(n,alpha=1,beta=0)
dataEx<-data.frame(x=x,y=y)
qgamRes<-qgam(y~s(x,k=25),data=dataEx,qu=tau)
qamRes<-qam(y~s(x,k=25),data=dataEx,tau=tau)
rqfit<-rq(y~x,data=dataEx,tau=tau)
```

On the following chart, we fit the linear data with errors following a heteroscedastic alpha stable distribution with 'quantreg', 'qgacv' and 'qgam'. The fit with qgam is quite biased. We can see that 2 points out of 100 are below the model but the quantile is $\tau = 0.05$. We obtain approximately the same fit with 'qgacv' and 'quantreg'. With this fit, 2 points are interpolated, 4 points are below the fits and 94 points are above the fit.

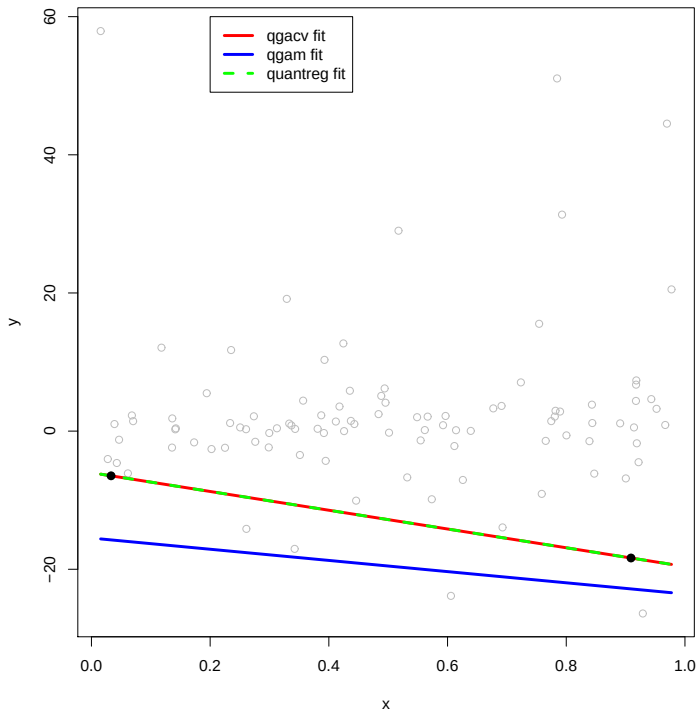


Fig. D.3: n=100, tau=0.05. The qgacv fits and quantreg match closely. The qgam fit is biased.

D.4 Robustness to outliers

In this section, we show examples of robustness issues for package 'qgam' and package 'aqmm'.

qgam

```
n<-100;tau<-0.05;set.seed(16)
x<-sort(runif(n))
y<-x+((x+1)^2)*stabledist::rstable(n,alpha=1,beta=0)
dataEx<-data.frame(x=x,y=y)
qgamRes<-qgam(y~s(x,k=25),data=dataEx,qu=tau)
qamRes<-qam(y~s(x,k=25),data=dataEx,tau=tau)
rqfit<-rq(y~x,data=dataEx,tau=tau)
```

In this example with $n = 100$, we fit a model at $\tau = 0.05$ with qgacv, qgam and quantreg. qgacv is robust to the outliers and matches the linear quantile regression fit from quantreg.

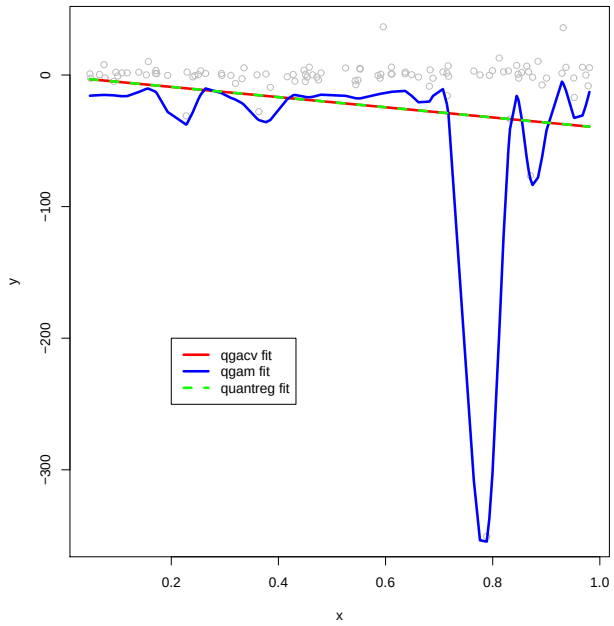


Fig. D.4: n=100, tau=0.05. We fit with qgacv::qam (red), quantreg::rq (dotted green) and qgam (blue). We see that qgacv::qam and quantreg::rq fits match.

aqmm

In the example below, an extreme datapoint can lead to different fits with package 'aqmm'.

```
n<-500;tau<-0.5
dataEx<-qgacvExamples(11,n=500,seed=237,tauArr=tau)
TQ<-dataEx$TrueQuantile
x<-dataEx$data$x
y<-dataEx$data$y
# We shift the values of y above 0 to use logarithmic
# scale for plot.
TQ<-TQ-min(y)+1;y<-y-min(y)+1
aqmmData<-data.frame(y=y,x=x,id=1)
aq<-aqmm(y~s(x,k=20),data=aqmmData,random=~1,group=id,
tau=tau)
aqmmData2<-aqmmData
# we replace the extreme datapoint by the mean of y
aqmmData2$y[which.max(y)]<-mean(y)
aq2<-aqmm(y~s(x,k=20),data=aqmmData2,random=~1,group=id,
tau=tau)
```

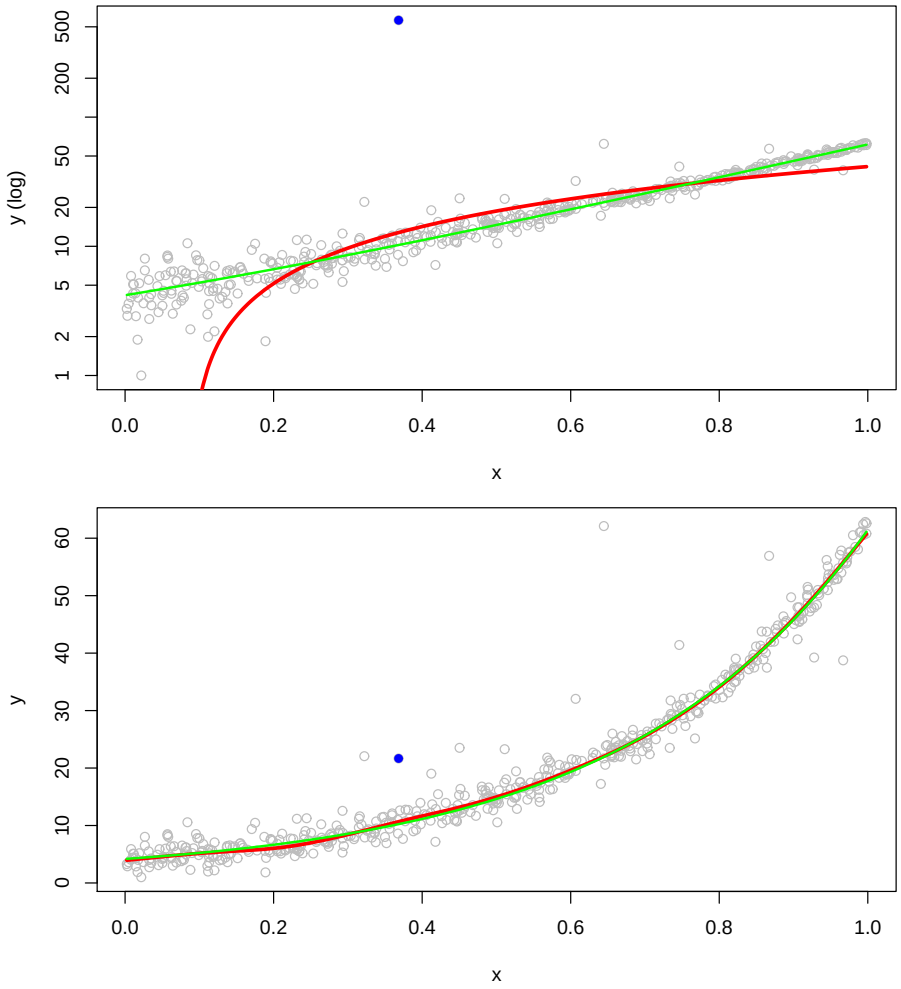


Fig. D.5: This Figure shows an example of sensitivity of a fit of package aqmm to a single data point. The data is generated using Model 11. To use the log scale for the top chart, we shift the data to positive values on the y axis and replace the minimum by its average. On the top chart, we can see that there is one extreme datapoint in the dataset (in blue). The fit using aqmm is the red curve. The green curve is the true quantile. With the dataset that includes this extreme datapoint, aqmm selects a straight line as the best fit. The MSE between the fitted curve and the true quantile is 35.70. On the bottom chart, we replace the value of the extreme datapoint on the y axis by the average of the y values in the dataset (in blue). After this change, we see that the aqmm fit is much closer to the true quantile and we now obtain an MSE of 0.085.

D.5 Increasing the number of basis functions

This example is taken from Koenker [2020c] and is also available in package 'quantreg' under 'demo/Polson.r' Koenker et al. [2021]:

```
set.seed(1848);n<-2000;x <- 0:n/n;
y <- rnorm(n+1, 5 * sin(2 * pi * x), 0.5 + exp(1.5 * sin(4 * pi * x)));
dataEx<-data.frame(x=x,y=y);taus = c(.02, .15, .5, .85, .98);mmArr<-c(10,20,50,200)
for(ii in 1:4){ mm<-mmArr[ii]
f98<-qam(y~s(x,k=mm),data=dataEx,tau=0.98);f2<-qam(y~s(x,k=mm),data=dataEx,tau=0.02);
f15<-qam(y~s(x,k=mm),data=dataEx,tau=0.15);f50<-qam(y~s(x,k=mm),data=dataEx,tau=0.5);
f85<-qam(y~s(x,k=mm),data=dataEx,tau=0.85)}
```

We use the same generating model as Koenker [2020c] that was originally proposed in Polson and Scott [2016], and increase the number of samples from 500 to 2,000 to make sure there is sufficient data to fit extreme quantiles correctly. The number of basis functions is then increased from 10 to 200 and the fits remain stable.

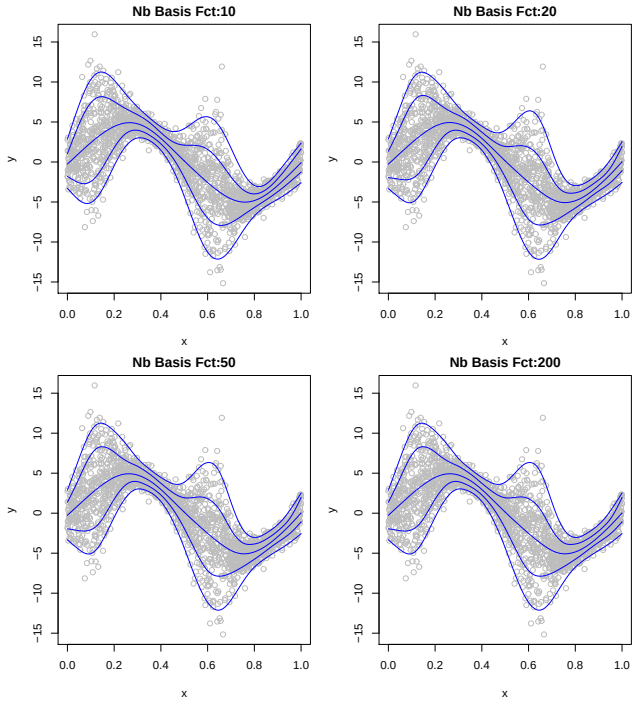


Fig. D.6: On this chart, we take an examples from Koenker [2020c]. n=2,000, tau=0.01, 0.15, 0.5, 0.85, 0.98. As the number of basis function is increased, the fits using 'qgacv::qam' remain relatively invariant.

D.6 Two dimensional fit with quantile RKHS approaches

Note that part of the code below is directly taken from the cosso package Zhang and Lin [2013]. Package 'cosso' implements a methodology developed in Lin and Zhang [2006] and Lin et al. [2013]. For an Reproducing Kernel Hilbert Space (RKHS) quantile regression, the expression implemented is:

$$\hat{f} \in \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\tau \left(y_i - f\left(\underline{\boldsymbol{x}}_i\right)\right) + \lambda_0 \sum_{k=1}^{p} \omega_j^2 \theta_j^{-1} \left\| P^j f \right\|^2$$

$$s.t. \sum_{k=1}^{p} \theta_j \leq M, \theta_j \geq 0$$

$$\omega_j = \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(P^j f\left(\underline{x}_{i,j}\right)\right)^2 \right\}^{-\frac{1}{2}}$$

We can see that the framework uses two tuning parameters: λ_0 and M. λ_0 is tuned when function 'cosso' is run whilst the second tuning parameter is calibrated with function 'tune.cosso'. We can see that the relative weight in each direction is calculated adaptively. There is not a single smoothing parameter in each direction.

```
PredictCosso<-function(x1,x2,fit,M){xnew<-cbind(x1,x2,rep(0,length(x1)))
#### this part of the code is taken from package 'cosso',
#### function 'predict.cosso'
fitObj = cosso::twostep.qr(fit$tau, fit$y, fit$Kmat, fit$tune$OptLam,
M, fit$wt)
predictor = as.numeric(fitObj$intercept +
 wsGram(bigGram(xnew,fit$x[fit$basis.id, ]),
fitObj$theta/(fit$wt^2)) %*% fitObj$coefs);return(predictor)}
####
PredictQAM<-function(x1,x2,fit){
xnew<-data.frame(x1=x1,x2=x2,x3=rep(0,length(x1)))
pred<-predict(fitQam,newdata=xnew);return(pred)}
n<-250;set.seed(1)
x<-as.matrix(cbind(sort(runif(n)),runif(n),0.05*runif(n),0.05*runif(n)))
y<-as.matrix(sin(x[,1]*45)+sin(x[,2]*4)+0.5*rnorm(n))
dataQam<-data.frame(x1=x[,1],x2=x[,2],x3=x[,3],y=y)
#### this part of the code is adapted from an example
#### in the documentation of package 'cosso'
fitCosso<-cosso::cosso(x[,1:3],y,tau=0.5,family="Quantile")
tuneObj <- cosso::tune.cosso(fitCosso, folds = 5, plot.it = FALSE)
M <- tuneObj$OptM
####
fitQam<-qam(y~s(x1,k=30)+s(x2,k=30)+s(x3,k=30),
data=dataQam,tau=0.5,BootSamplesCalc = T,CI_Nbboot = 20)
```

```
X1<-sort(x[,1]);X2<-sort(x[,2]);z <- outer(X1,X2,PredictCosso,
fit=fitCosso,M=M)
z2 <- outer(X1,X2 , PredictQAM,fit=fitQam)
```

To illustrate the issue with typical quantile RKHS approaches such as the one implemented in package 'cosso', we fit a 2 dimensional example. Note that function 'cosso' requires a minimum of 3 covariates. We then input 3 covariates (x_1, x_2, x_3). We generate the data as:

$$x_{ji} \sim \mathcal{U}[0,1] \text{ and } cov(x_{ji}, x_{kl}) = 0, \forall (j,i) \neq (k,l)$$
$$y_i = sin(45 \times x_{1i}) + sin(4 \times x_{2i}) + 0.5 \times \varepsilon_i, i = 1, \ldots, n.$$

We then fit the model using 'cosso:cosso' and 'qgacv:qam'. As can be seen on Figure D.8a, using function 'cosso', we overfit in one direction (see also Figure D.7 where we show the qam fit in all 3 directions). This is because we should have quite a different smoothing parameter in direction x_1 and direction x_2 and the adaptive method in 'cosso' does not seem to adapt correctly when the smoothness is very different in each direction whereas the qgacv fit has an independent smoothing parameter in each direction. Note that it would be straightforward to extend the typical quantile RKHS approach to have a smoothing parameter in each direction. In this case, the framework would be very close to the QAM framework. (see for example Hastie and Tibshirani [1990] or Berlinet and Thomas-Agnan [2011] for a discussion of the link between RKHS and additive models).

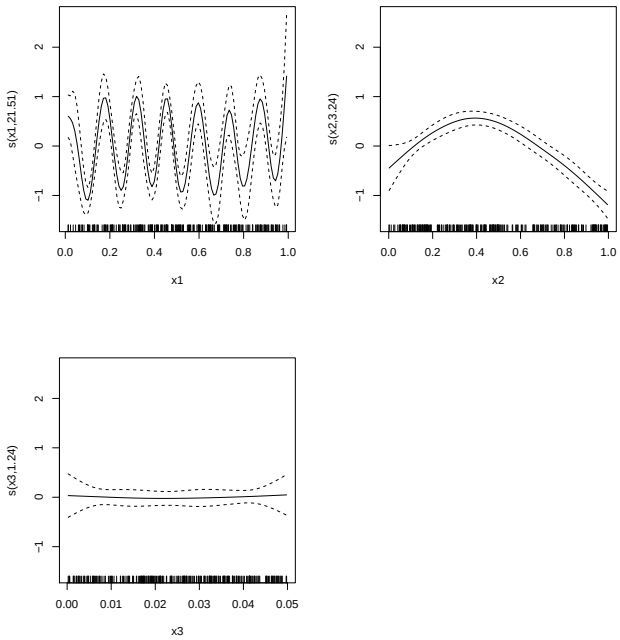
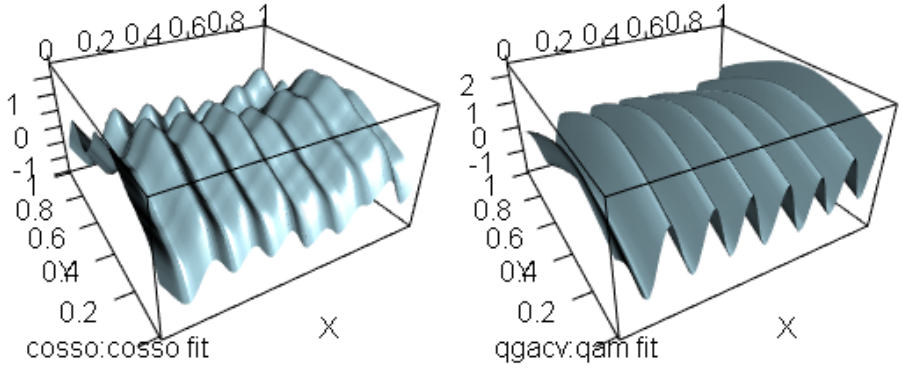


Fig. D.7: Fit with 'qgacv:qam' in all 3 directions.



(a) Fit with 'cosso:cosso' for the median *(b)* Fit with 'qgacv:qam' for the median

Fig. D.8: On the left hand side, as the degree of smoothness is quite different in the two directions, package cosso overfits in the y direction. On the right hand side, we obtain a correct with with package qgacv.

D.7 Central Quantile vs mean GAM

It has been reported that using the GCV formula for mean GAMs can sometimes lead to overfit with small sample sizes and correlated data (Lukas [2010]). We also observed overfitting for central quantiles using the GACV formula in our simulation 4.7. One of the proposals of Lukas et al. [2016] to reduce the issue is to multiply the edf by a factor $\times 2$ for $n \geq 100$. The QGACV formula derived has a $2\times$ multiplier for the edf when $\tau = 0.5$ which corresponds to the proposal of Lukas et al. [2016]. This leads to improved fits in some cases when compared to the mean GAM fit calibrated using 'GCV.Cp'. Below is an example where we generate 25 datasets that follow an autoregressive AR[2] process. We note that the median fitted using QGACV is smoother than the mean fitted using 'GCV.Cp'.

```
for(iii in 1:25)
{set.seed(iii);x<-sort(runif(n));y<-rep(0,n);err<-rnorm(n);
y[1]<-x[1];y[2]<-1.5*x[2]
for(i in 3:n){
y[abc]<-2*x[i]+(err[i]+err[i-1]+err[i-2])}
dataEx<-data.frame(x=x,y=y)
fitqam<-qam(y~s(x,k=15),data=dataEx,tau=0.5)
fitgam<-gam(y~s(x,k=15),data=dataEx)
plot(dataEx$x,dataEx$y,col="grey",xlab="x",ylab="y",ylim=c(-2,5))
lines(dataEx$x,predict(fitqam),col="red",lwd=3,lty=1)
lines(dataEx$x,predict(fitgam),col="blue",lwd=3,lty=1)}
```

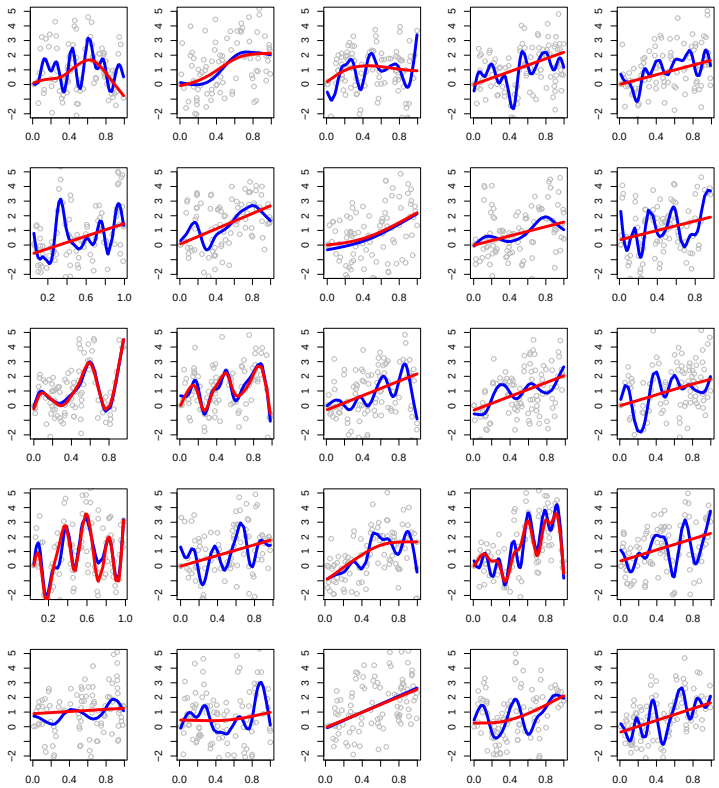


Fig. D.9: We generate 25 different examples with MA(q=2) time series with n=100 observations. The fit with mean GAM (mgcv::gam with method="GCV.Cp") is in blue. The fit with qgacv::qam is in red. We can see that using qgacv::qam with $\tau = 0.5$ leads to less overfitting with correlated error and small sample size than mean GAM fits with GCV.

E. DERIVATION OF THE SURROGATE LOSS FUNCTION $\mathcal{L}_{\alpha,\tau}$ VIA KERNEL CONVOLUTION

He et al. [2020] showed that the surrogate loss used in Amemiya [1982], Zheng [2011] results from the convolution of the exact pinball loss:

$$\mathcal{L}_{\tau}(u) = u(\tau - I(u < 0))$$

with a logistic kernel:

$$K_{Log}\left(u\right) = \frac{\exp\left(-u\right)}{\left(1+\exp\left(-u\right)\right)^{2}}$$

We here show that the shifted loss function used in this document results from the convolution of the exact pinball loss with a shifted logistic Kernel:

$$K_{SLog,\tau}\left(u\right) = \frac{\left(\frac{1-\tau}{\tau}\right)\exp\left(-u\right)}{\left(1+\left(\frac{1-\tau}{\tau}\right)\exp\left(-u\right)\right)^{2}} \tag{E.1}$$

In our case, the convolution formula of He et al. [2020] is:

$$\mathcal{L}_{\alpha,\tau}\left(u\right) = \int_{-\infty}^{+\infty} \mathcal{L}_{\tau}\left(v\right) \times \left(\frac{1}{\alpha} K_{SLog,\tau}\left(\frac{v-u}{\alpha}\right)\right) dv \tag{E.2}$$

replacing each term, we obtain:

$$\mathcal{L}_{\alpha,\tau}\left(u\right) = \int_{-\infty}^{+\infty} v\left(\tau - I\left(v<0\right)\right) \times \left(\frac{1}{\alpha}\frac{\left(\frac{1-\tau}{\tau}\right)\exp\left(-\frac{v-u}{\alpha}\right)}{\left(1+\left(\frac{1-\tau}{\tau}\right)\exp\left(-\frac{v-u}{\alpha}\right)\right)^{2}}\right) dv \tag{E.3}$$

First, we can transform the expression of the kernel to obtain a standard logistic kernel where u is shifted by $\gamma = \alpha log\left(\frac{1-\tau}{\tau}\right)$:

$$\frac{\left(\frac{1-\tau}{\tau}\right)\exp\left(-\frac{v-u}{\alpha}\right)}{\left(1+\left(\frac{1-\tau}{\tau}\right)\exp\left(-\frac{v-u}{\alpha}\right)\right)^{2}} = \frac{\exp\left(\frac{\alpha}{\alpha}log\left(\frac{1-\tau}{\tau}\right)\right)\exp\left(-\frac{v-u}{\alpha}\right)}{\left(1+\exp\left(\frac{\alpha}{\alpha}log\left(\frac{1-\tau}{\tau}\right)\right)\exp\left(-\frac{v-u}{\alpha}\right)\right)^{2}} = \frac{\exp\left(-\frac{v-(u+\gamma)}{\alpha}\right)}{\left(1+\exp\left(-\frac{v-(u+\gamma)}{\alpha}\right)\right)^{2}}$$

Then, we note that we can separate the integral in two parts:

$$\mathcal{L}_{\alpha,\tau}\left(u\right) = \frac{1}{\alpha}\int_{-\infty}^{+\infty} v\tau \frac{\exp\left(-\frac{v-(u+\gamma)}{\alpha}\right)}{\left(1+\exp\left(-\frac{v-(u+\gamma)}{\alpha}\right)\right)^{2}} dv - \frac{1}{\alpha}\int_{-\infty}^{0} v \frac{\exp\left(-\frac{v-(u+\gamma)}{\alpha}\right)}{\left(1+\exp\left(-\frac{v-(u+\gamma)}{\alpha}\right)\right)^{2}} dv$$

To calculate these two integrals, we can use a change of variable similar to the one that can be found in Decani and Stine [1986]:

$$F = \frac{1}{\left(1 + \exp\left(-\frac{v-(u+\gamma)}{\alpha}\right)\right)}$$

Johnson et al. [1995] note that this transformation can be found in Gumbel [1961] (see also Gumbel [1958] pp126-127). Gumbel [1958] notes that the logistic distribution can be expressed as $F(1-F)$. We then have the following:

$$dF = \frac{1}{\alpha} \frac{\exp\left(-\frac{v-(u+\gamma)}{\alpha}\right)}{\left(1 + \exp\left(-\frac{v-(u+\gamma)}{\alpha}\right)\right)^2} dv$$

$$v = (u+\gamma) - \alpha \log\left(\frac{1-F}{F}\right)$$

$$\begin{cases} v \to -\infty & F \to 0 \\ v = 0 & F = \frac{1}{\left(1+\exp\left(\frac{u+\gamma}{\alpha}\right)\right)} \\ v \to +\infty & F \to 1 \end{cases}$$

The integral can then be re-written:

$$\mathcal{L}_{\alpha,\tau}(u) = \tau \int_0^1 (u+\gamma) + \alpha\left(\log(F) - \log(1-F)\right) dF - \int_0^{\left(1+\exp\left(\frac{u+\gamma}{\alpha}\right)\right)^{-1}} (u+\gamma) + \alpha\left(\log(F) - \log(1-F)\right) dF$$

Decani and Stine [1986] then note the following:

$$\int \left(\log(F) - \log(1-F)\right) dF = F\log(F) + (1-F)\log(1-F)$$

They also note that:

$$\lim_{F\to 1} (1-F)\log(1-F) = 0$$

$$\lim_{F\to 0} F\log(F) = 0$$

The integral can then be re-written:

$$\mathcal{L}_{\alpha,\tau}(u) = (u+\gamma)\left(\tau \int_0^1 dF - \int_0^{\left(1+\exp\left(\frac{u+\gamma}{\alpha}\right)\right)^{-1}} dF\right) - \alpha\left[F\log(F) + (1-F)\log(1-F)\right]_0^{\left(1+\exp\left(\frac{u+\gamma}{\alpha}\right)\right)^{-1}}$$

This can then be simplified as:

$$\mathcal{L}_{\alpha,\tau}(u) = (u+\gamma)\left(\tau - \frac{1}{\left(1+\exp\left(\frac{u+\gamma}{\alpha}\right)\right)}\right) - \alpha\log\left(\left(1+\exp\left(\frac{u+\gamma}{\alpha}\right)\right)^{-1}\right) - \left(\frac{(u+\gamma)\exp\left(\frac{u+\gamma}{\alpha}\right)}{\left(1+\exp\left(\frac{u+\gamma}{\alpha}\right)\right)}\right)$$

We then obtain:

$$\mathcal{L}_{\alpha,\tau}(u) = (u+\gamma)(\tau-1) + \alpha\log\left(1+\exp\left(\frac{u+\gamma}{\alpha}\right)\right) \tag{E.4}$$

That can be re-arranged as:

$$\mathcal{L}_{\alpha,\tau}(u) = \tau(u+\gamma) + \alpha\log\left(1+\exp\left(-\frac{u+\gamma}{\alpha}\right)\right) \tag{E.5}$$

F. DERIVATION OF A SURROGATE LOSS USING THE ENTROPY PROX METHOD

In this Appendix, we show that the entropy prox smoothing method of Nesterov [2005] can be applied to obtain a similar loss to the smooth surrogate of Amemiya [1982], Zheng [2011]. This derivation is based on Hahn et al. [2020]. Nesterov [2005], Hahn et al. [2020] notice that piecewise linear functions can be rewritten as the max of linear functions. Hence in the case of the pinball loss, we have:

$$\mathcal{L}_\tau(u) = \max(\tau u, (\tau - 1)u) = \max_{i=1,2}(\mathbf{A}(u, 1)^T)_i$$

where

$$\mathbf{A} = \begin{bmatrix} \tau - 1 & 0 \\ \tau & 0 \end{bmatrix}$$

Then the entropy-prox surrogate loss

$$\begin{aligned} \mathcal{L}_{\alpha,\tau}^{Nesterov}(u) &= \alpha \log\left(\frac{1}{2}\sum_{i=1,2}\exp\left(\frac{\left(\mathbf{A}\left[u,1\right]^T\right)_i}{\alpha}\right)\right) \\ &= \alpha \log\left(\frac{1}{2}\left(\exp\left(\frac{\tau u}{\alpha}\right) + \exp\left(\frac{(\tau-1)u}{\alpha}\right)\right)\right) \\ &= \tau u + \alpha \log\left(1 + \exp\left(-\frac{u}{\alpha}\right)\right) - \alpha \log 2 \end{aligned}$$

We obtain the loss of Amemiya [1982], Zheng [2011] shifted down by $\alpha \log(2)$. Interestingly, this loss seems to minorize the pinball loss. In this document, two other infinitely smooth surrogate losses that have been derived minorize the pinball loss. The first one is Loss C described in Section 4.1:

$$\mathcal{L}_{\alpha,\tau}^{C}(u) = \tau u + \alpha \log\left((1-\tau) + \tau \exp\left(-\frac{u}{\alpha}\right)\right)$$

The second one is the derived the method of Yilmaz and Sahiner [2019] in Chapter 10:

$$\mathcal{L}_{\alpha,\tau}^{YS}(u) = u\tau - \frac{u}{\pi}\left(\arctan\left(-\frac{u}{\alpha}\right) + \frac{\pi}{2}\right)$$

For $\tau = 0.5$, $\mathcal{L}_{\alpha,\tau=0.5}^{Nesterov}(u) = \mathcal{L}_{\alpha,\tau=0.5}^{C}(u)$. For $\tau \neq 0.5$, the three losses do not approximate the loss function the same way. A chart showing the three surrogate losses for $\tau = 0.3$ is shows on Figure F.1:

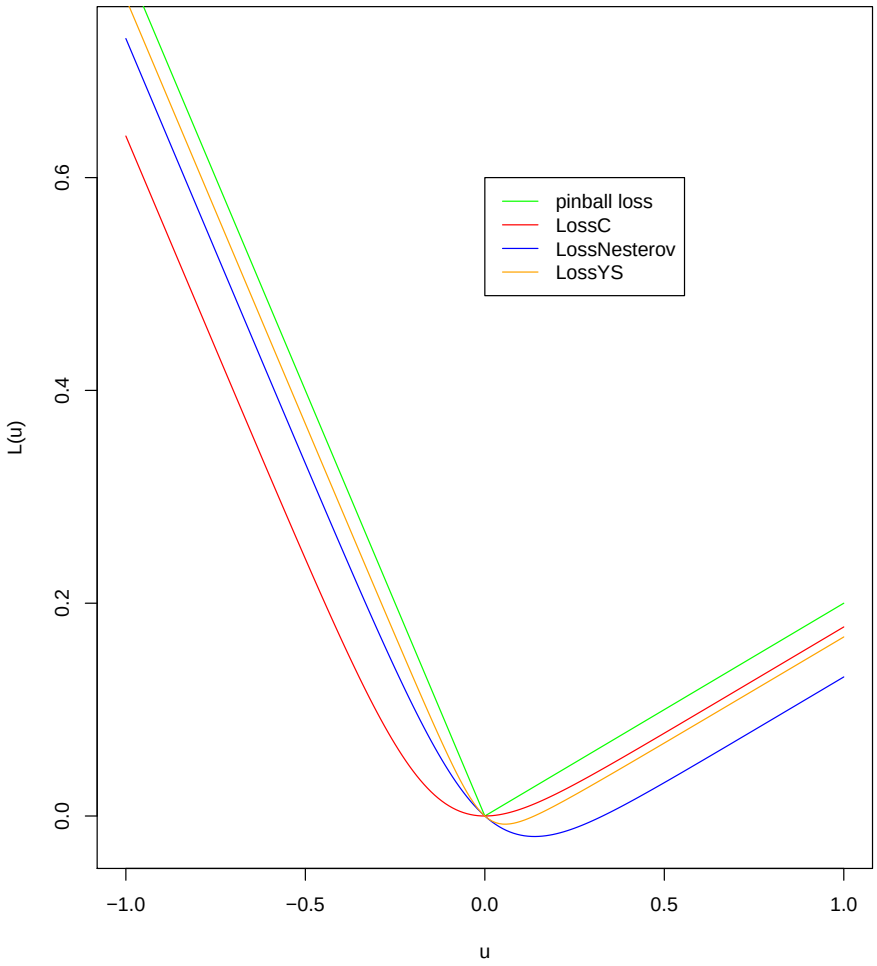


Fig. F.1: Comparison between Pinball loss (green), LossC (red), Loss derived using Yilmaz and Sahiner [2019] in orange and Loss derived using the entropy-prox methods of Nesterov [2005] in blue.

G. CONVERGENCE OF THE TR(A) TO THE NUMBER OF INTERPOLATED POINTS IN A SIMPLE CASE

In this section, we show how in the case of a linear quantile regression with a single covariate, where the quantile regression model interpolates two datapoints, the model with the sum of the diagonal of the hat matrix converges to 2 as $\alpha \to 0$. We first develop the expression of the hat matrix for the model with the surrogate loss rounded at α:

$$
\begin{aligned}
\mathbf{A} &= \mathbf{X}\left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{W} \\
&= \begin{bmatrix} 1 & \underline{x}_1 \\ \vdots & \vdots \\ 1 & \underline{x}_n \end{bmatrix} \left(\begin{bmatrix} \underline{x}_1 & \cdots & \underline{x}_n \end{bmatrix} diag(..., w_j, ...) \begin{bmatrix} 1 & \underline{x}_1 \\ \vdots & \vdots \\ 1 & \underline{x}_n \end{bmatrix} \right)^{-1} \left(\begin{bmatrix} 1 & \cdots & 1 \\ \underline{x}_1 & \cdots & \underline{x}_n \end{bmatrix} diag(..., w_j, ...) \right) \\
&= \begin{bmatrix} 1 & \underline{x}_1 \\ \vdots & \vdots \\ 1 & \underline{x}_n \end{bmatrix} \begin{pmatrix} \left(\sum_{i=1}^n w_i\right) & \left(\sum_{i=1}^n w_i\underline{x}_i\right) \\ \left(\sum_{i=1}^n w_i\underline{x}_i\right) & \left(\sum_{i=1}^n w_i\underline{x}_i^2\right) \end{pmatrix}^{-1} \begin{bmatrix} w_1 & \cdots & w_n \\ w_1\underline{x}_1 & \cdots & w_n\underline{x}_n \end{bmatrix} \\
&= \begin{bmatrix} 1 & \underline{x}_1 \\ \vdots & \vdots \\ 1 & \underline{x}_n \end{bmatrix} \begin{pmatrix} \left(\sum_{i=1}^n w_i\underline{x}_i^2\right) & \left(-\sum_{i=1}^n w_i\underline{x}_i\right) \\ \left(-\sum_{i=1}^n w_i\underline{x}_i\right) & \left(\sum_{i=1}^n w_i\right) \end{pmatrix} \begin{bmatrix} w_1 & \cdots & w_n \\ w_1\underline{x}_1 & \cdots & w_n\underline{x}_n \end{bmatrix} \\
&= \frac{\begin{bmatrix} 1 & \underline{x}_1 \\ \vdots & \vdots \\ 1 & \underline{x}_n \end{bmatrix} \begin{bmatrix} \cdots & w_j\left(\left(\sum_{i=1}^n w_i\underline{x}_i^2\right) - \underline{x}_j\left(\sum_{i=1}^n w_i\underline{x}_i\right)\right) & \cdots \\ \cdots & w_j\left(\underline{x}_j\left(\sum_{i=1}^n w_i\right) - \left(\sum_{i=1}^n w_i\underline{x}_i\right)\right) & \cdots \end{bmatrix}}{\left(\sum_{i=1}^n w_i\right)\left(\sum_{i=1}^n w_i\underline{x}_i^2\right) - \left(\sum_{i=1}^n w_i\underline{x}_i\right)^2} \\
&= \frac{\begin{bmatrix} 1 & \underline{x}_1 \\ \vdots & \vdots \\ 1 & \underline{x}_n \end{bmatrix} \begin{bmatrix} \cdots & w_j\left(\left(\sum_{i=1}^n w_i\underline{x}_i^2\right) - \underline{x}_j\left(\sum_{i=1}^n w_i\underline{x}_i\right)\right) & \cdots \\ \cdots & w_j\left(\underline{x}_j\left(\sum_{i=1}^n w_i\right) - \left(\sum_{i=1}^n w_i\underline{x}_i\right)\right) & \cdots \end{bmatrix}}{\left(\sum_{i=1}^n w_i\right)\left(\sum_{i=1}^n w_i\underline{x}_i^2\right) - \left(\sum_{i=1}^n w_i\underline{x}_i\right)^2} \\
&= \frac{\begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & w_j\left(\left(\sum_{i=1}^n w_i\underline{x}_i^2\right) - \underline{x}_j\left(\sum_{i=1}^n w_i\underline{x}_i\right)\right) + \underline{x}_j w_j\left(\underline{x}_j\left(\sum_{i=1}^n w_i\right) - \left(\sum_{i=1}^n w_i\underline{x}_i\right)\right) & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}}{\left(\sum_{i=1}^n w_i\right)\left(\sum_{i=1}^n w_i\underline{x}_i^2\right) - \left(\sum_{i=1}^n w_i\underline{x}_i\right)^2}
\end{aligned}
$$

Then, the diagonal elements of the Hat matrix $\mathbf{A}$ in the case of the linear quantile regression with a single covariate is:

$$\frac{w_i \left(\sum_{j=1}^{n} w_j \underline{x}_j^2\right) + w_i \underline{x}_i^2 \left(\sum_{j=1}^{n} w_j\right) - 2\underline{x}_i w_i \left(\sum_{j=1}^{n} w_j \underline{x}_j\right)}{\left(\sum_{j=1}^{n} w_j\right)\left(\sum_{i=1}^{n} w_j \underline{x}_j^2\right) - \left(\sum_{i=1}^{n} w_j \underline{x}_j\right)^2} \tag{G.1}$$

where the w_i are the diagonal elements of matrix $\mathbf{W}$ that is a matrix of second derivatives of the loss function, evaluated with a rounding α but where the vector of parameters is estimated with the exact pinball loss:

$$w_i = \frac{\partial^2 \mathcal{L}_{\alpha,\tau}(u_i)}{\partial u_i^2}\Big|_{u_i = y_i - \hat{f}_i} = \frac{1}{\alpha} \frac{\exp\left(-\frac{(y_i - \hat{f}_i + \gamma)}{\alpha}\right)}{\left(1 + \exp\left(-\frac{(y_i - \hat{f}_i + \gamma)}{\alpha}\right)\right)^2}$$

With a linear quantile regression in 1 dimension that are only 2 datapoints that will be interpolated. For $\alpha \to 0$, 2 of the weights will become very large compared to:

$$\begin{aligned}
&\frac{w_A \left(\sum_{j=1}^{n} w_j \underline{x}_j^2\right) + w_A \underline{x}_A^2 \left(\sum_{j=1}^{n} w_j\right) - 2\underline{x}_A w_A \left(\sum_{j=1}^{n} w_j \underline{x}_j\right)}{\left(\sum_{j=1}^{n} w_j\right)\left(\sum_{i=1}^{n} w_j \underline{x}_j^2\right) - \left(\sum_{i=1}^{n} w_j \underline{x}_j\right)^2} \\
&\approx_{\alpha \to 0} \frac{w_A \left(w_A \underline{x}_A^2 + w_B \underline{x}_B^2\right) + w_A \underline{x}_A^2 \left(w_A + w_B\right) - 2\underline{x}_A w_A \left(w_A \underline{x}_A + w_B \underline{x}_B\right)}{\left(w_A + w_B\right)\left(w_A \underline{x}_A^2 + w_B \underline{x}_B^2\right) - \left(w_A \underline{x}_A + w_B \underline{x}_B\right)^2} \\
&\approx_{\alpha \to 0} \frac{w_A w_B \left(\underline{x}_A - \underline{x}_B\right)^2}{w_A w_B \left(\underline{x}_A - \underline{x}_B\right)^2} = 1
\end{aligned}$$

We then see that in this case, the trace of the hat matrix converges to the number of interpolated points.

H. COMPARISON OF SORTING ALGORITHMS IN R

To compare the different sorting algorithms in function stats::sort, we generate random number that are $\mathcal{N}(0,1)$ with an increasing sequence length from $n = 10$ to $n = 10^8$. We then record the calculation time to sort the sequence with 4 different methods of function stats::sort. We then fit 5 different linear regressions of calculation time as a function of n, $n^{\frac{4}{3}}$, $n \times \log(n)$, $n^{1.5}$ and n^2. We then choose the model that has the highest R squared. We find that with this example, the closest match for the default algorithm of stat::sort is $n^{\frac{4}{3}}$.

	"default"	"quick"	"radix"	"shell"
n	0.975	0.992	0.983	0.982
$n^{\frac{4}{3}}$	**0.988**	0.985	**0.988**	**0.988**
$n \times \log(n)$	0.980	**0.994**	0.987	0.986
$n^{1.5}$	0.984	0.972	0.980	0.979
n^2	0.950	0.915	0.935	0.933

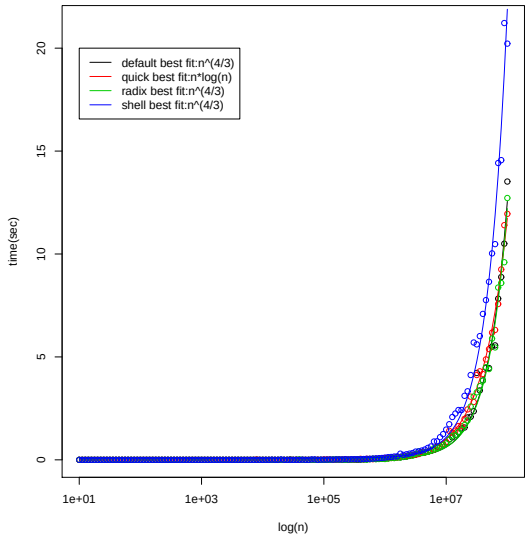


Fig. H.1: This function compares the different calculation times with function 'sort' for different algorithms.

Bibliography

Samah M Abo-El-Hadid. Logistic kernel estimator and bandwidth selection for density function. *International Journal of Contemporary Mathematical Sciences*, 13(6):279–286, 2018.

Mingyao Ai, Fei Wang, Jun Yu, and Huiming Zhang. Optimal subsampling for large-scale quantile regression. *Journal of Complexity*, 62:101512, 2021.

Alnur Ali, J Zico Kolter, and Ryan J Tibshirani. The multiple quantile graphical model. In *Advances in Neural Information Processing Systems*, pages 3747–3755, 2016.

David M Allen. The relationship between variable selection and data agumentation and a method for prediction. *technometrics*, 16(1):125–127, 1974.

Takeshi Amemiya. Two stage least absolute deviations estimators. *Econometrica: Journal of the Econometric Society*, pages 689–711, 1982.

Aleksandr Aravkin, Aurelie Lozano, Ronny Luss, and Prabhajan Kambadur. Orthogonal matching pursuit for sparse quantile regression. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 11–19. IEEE, 2014a.

Aleksandr Y Aravkin, Anju Kambadur, Aurelie C Lozano, and Ronny Luss. Sparse quantile huber regression for efficient and robust estimation. *arXiv preprint arXiv:1402.4624*, 2014b.

Aleksandr Y Aravkin, James V Burke, Dmitry Drusvyatskiy, Michael P Friedlander, and Scott Roy. Level-set methods for convex optimization. *Mathematical Programming*, 174(1):359–390, 2019.

Azam Asl and Michael L Overton. Behavior of limited memory bfgs when applied to nonsmooth functions and their nesterov smoothings. *arXiv preprint arXiv:2006.11336*, 2020.

Azam Asl and Michael L Overton. Analysis of limited-memory bfgs on a class of nonsmooth convex functions. *IMA Journal of Numerical Analysis*, 41(1): 1–27, 2021.

Philipp Bach, Sven Klaassen, Jannis Kueck, and Martin Spindler. Uniform inference in high-dimensional generalized additive models. *arXiv preprint arXiv:2004.01623*, 2020.

Yoshua Bengio and Yves Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. *Journal of machine learning research*, 5(Sep):1089–1105, 2004.

Alain Berlinet and Christine Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

Nicolai Bissantz, Lutz Dümbgen, Axel Munk, and Bernd Stratmann. Convergence analysis of generalized iteratively reweighted least squares algorithms on convex function spaces. *SIAM Journal on Optimization*, 19(4):1828–1845, 2009.

Pier Giovanni Bissiri, Chris C Holmes, and Stephen G Walker. A general framework for updating belief distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):1103–1130, 2016.

Andrew Blake and Andrew Zisserman. *Visual reconstruction*. MIT press, 1987.

Peter Bloomfield and William L Steiger. *Least absolute deviations: theory, applications, and algorithms*. Springer, 1983.

Marc-Olivier Boldi and Valérie Chavez-Demoulin. Improving the local scoring algorithm using gradient sampling. *arXiv preprint arXiv:1705.10082*, 2017.

Marc-Olivier Boldi, Valérie Chavez-Demoulin, and Olivier Gallay. Intraday retail sales forecast: An efficient algorithm for quantile additive modeling. *arXiv preprint arXiv:1912.07373*, 2019.

Howard D Bondell, Brian J Reich, and Huixia Wang. Noncrossing quantile regression curve estimation. *Biometrika*, 97(4):825–838, 2010.

Ronald J Bosch, Yinyu Ye, and George G Woodworth. A convergent algorithm for quantile regression with smoothing splines. *Computational statistics & data analysis*, 19(6):613–630, 1995.

Roger Joseph Boscovich. De litteraria expeditione per pontificiam ditionem, et synopsis amplioris operis, ac habentur plura ejus ex exemplaria etiam sensorum impessa. *Bononiensi Scientiarum et Artum Instuto Atque Academia Commentarii*, 4:353–396, 1757.

Léon Bottou and Chih-Jen Lin. Support vector machine solvers. *Large scale kernel machines*, 3(1):301–320, 2007.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1): 1–122, 2011.

Romain Brault, Alex Lambert, Zoltán Szabó, Maxime Sangnier, and Florence d'Alché Buc. Infinite task learning in rkhss. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1294–1302. PMLR, 2019.

James V Burke, Frank E Curtis, Adrian S Lewis, Michael L Overton, and Lucas EA Simões. Gradient sampling methods for nonsmooth optimization. *Numerical Nonsmooth Optimization*, pages 201–225, 2020.

Paul-Christian Bürkner. brms: An r package for bayesian multilevel models using stan. *Journal of Statistical Software*, 80(1):1–28, 2017.

Yuzhi Cai and Tao Jiang. Estimation of non-crossing quantile regression curves. *Australian & New Zealand Journal of Statistics*, 57(1):139–162, 2015.

Gavin C Cawley and Nicola LC Talbot. Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural networks*, 17(10): 1467–1475, 2004.

Volkan Cevher. Ee-731: Advanced topics in data sciences laboratory for information and inference systems spring 2016 lecture 7: Coordinate descent methods (cont.) and randomized linear algebra. `https://lions.epfl.ch/wp-content/uploads/2019/01/Scribe-Lecture-7.pdf`, 2016.

Olivier Chapelle. Training a support vector machine in the primal. *Neural computation*, 19(5):1155–1178, 2007.

Valérie Chavez-Demoulin and Anthony C Davison. Generalized additive modelling of sample extremes. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 54(1):207–222, 2005.

B Chen and M.Ç. Pinar. On newton's method for huber's robust m-estimation problems in linear regression. *BIT Numerical Mathematics*, 38(4):674–684, 1998.

Chunhui Chen and Olvi L Mangasarian. Smoothing methods for convex inequalities and linear complementarity problems. *Mathematical programming*, 71(1):51–69, 1995.

Colin Chen. A Finite Smoothing Algorithm for Quantile Regression. *Journal of Computational and Graphical Statistics*, 16(1):136–164, 2007.

Colin Chen and Ying Wei. Computational issues for quantile regression. *Sankhyā: The Indian Journal of Statistics*, pages 399–417, 2005.

Victor Chernozhukov, Iván Fernández-Val, and Alfred Galichon. Quantile and probability curves without crossing. *Econometrica*, 78(3):1093–1125, 2010.

Alexandra Chouldechova and Trevor Hastie. Generalized additive model selection. *arXiv preprint arXiv:1506.03850*, 2015.

Andreas Christmann and Ingo Steinwart. How svms can estimate quantiles and the median. In *Advances in neural information processing systems*, pages 305–312, 2008.

Eliana Christou, Annabel Settle, and Andreas Artemiou. Nonlinear dimension reduction for conditional quantiles. *Advances in Data Analysis and Classification*, pages 1–20, 2021.

DI Clark and MR Osborne. Finite algorithms for huber's m-estimator. *SIAM journal on scientific and statistical computing*, 7(1):72–85, 1986.

Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

Peter Craven and Grace Wahba. Smoothing noisy data with spline functions. *Numerische Mathematik*, 31(4):377–403, 1978.

Data Study Group Team. Bertrand Nortier, Camelia Simoiu, Edward Chuah, Francesco Sanna Passino, Ghita Berrada, Hanne Hoitzing, Henry Clausen, John Booth, Karl Halgren, Kaushik Jana, Keli Liu, Leigh Shlomovich, Qi (Katherine) He, Roberto Jordaney, Silvia Metelli, Victor-Alexandru Darvariu, Zhenzheng (Helen) Hu."Developing Data Science Tools for improving Enterprise Cyber-Security": Data Study Group Final Report: Imperial College London, Los Alamos National Laboratory, Heilbronn Institute. `http://doi.org/10.5281/zenodo.3558251`, 2019.

Cristina Davino, Marilena Furno, and Domenico Vistocco. *Quantile regression: theory and applications*, volume 988. John Wiley & Sons, 2013.

Mickaël De Backer, Anouar El Ghouch, and Ingrid Van Keilegom. An adapted loss function for censored quantile regression. *Journal of the American Statistical Association*, 114(527):1126–1137, 2019.

Carl De Boor. *A practical guide to splines*, volume 27. springer-verlag New York, 1978.

John S Decani and Robert A Stine. A note on deriving the information matrix for a logistic distribution. *The American Statistician*, 40(3):220–222, 1986.

Charles-Alban Deledalle, Samuel Vaiter, Jalal Fadili, and Gabriel Peyré. Stein unbiased gradient estimator of the risk (sugar) for multiple parameter selection. *SIAM Journal on Imaging Sciences*, 7(4):2448–2487, 2014.

Holger Dette and Stanislav Volgushev. Non-crossing non-parametric estimates of quantile curves. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(3):609–627, 2008.

Charles Dossal, Maher Kachour, MJ Fadili, Gabriel Peyré, and Christophe Chesneau. The degrees of freedom of the lasso for general design matrix. *Statistica Sinica*, pages 809–828, 2013.

Bradley Efron. How biased is the apparent error rate of a prediction rule? *Journal of the American statistical Association*, 81(394):461–470, 1986.

Bradley Efron. The estimation of prediction error: covariance penalties and cross-validation. *Journal of the American Statistical Association*, 99(467): 619–632, 2004.

Bradley Efron and Trevor Hastie. *Computer age statistical inference*, volume 5. Cambridge University Press, 2016.

Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

Paul HC Eilers. The truth about the effective dimension. *Statistica Neerlandica*, 72(3):201–209, 2018.

Paul HC Eilers and Brian D Marx. Flexible smoothing with b-splines and penalties. *Statistical science*, pages 89–102, 1996.

Paul HC Eilers, Iain D Currie, and Maria Durbán. Fast and compact smoothing on large multidimensional grids. *Computational Statistics & Data Analysis*, 50(1):61–76, 2006.

André Elisseeff and Massimiliano Pontil. Leave-one-out error and stability of learning algorithms with applications. *NATO science series sub series iii computer and systems sciences*, 190:111–130, 2003.

Marina Evangelou and Niall M Adams. An anomaly detection framework for cyber-security data. *Computers & Security*, 97:101941, 2020.

Matteo Fasiolo and Raphael Nedellec. An introduction to mgcviz: visual tools for gams. `https://mfasiolo.github.io/mgcViz/articles/mgcviz.html`, 2020.

Matteo Fasiolo, Raphaël Nedellec, Yannig Goude, and Simon N. Wood. Scalable visualization methods for modern generalized additive models. *Journal of computational and Graphical Statistics*, 29(1):78–86, 2020a.

Matteo Fasiolo, Simon N. Wood, Margaux Zaffran, Raphaël Nedellec, and Yannig Goude. Fast calibrated additive quantile regression. *Journal of the American Statistical Association*, pages 1–11, 2020b.

Matteo Fasiolo, Simon N. Wood, Margaux Zaffran, Raphaël Nedellec, and Yannig Goude. qgam: Bayesian non-parametric quantile regression modelling in r. *arXiv preprint arXiv:2007.03303*, 2020c.

Jean Feng and Noah Simon. Gradient-based regularization parameter selection for problems with nonsmooth penalty functions. *Journal of Computational and Graphical Statistics*, 27(2):426–435, 2018.

Marcelo Fernandes, Emmanuel Guerre, and Eduardo Horta. Smoothing quantile regressions. *Journal of Business & Economic Statistics*, 39(1):338–357, 2021.

Sabrina Fiege. *Minimization of Lipschitzian piecewise smooth objective functions*. PhD thesis, Universität Paderborn, 2017.

Kimon Fountoulakis and Jacek Gondzio. A second-order method for strongly convex l1-regularization problems. *Mathematical Programming*, 156(1-2):189–219, 2016.

Dominique Fourdrinier, William E Strawderman, and Martin T Wells. *Shrinkage estimation*. Springer, 2018.

Jueyu Gao. Computation in quantile and composite quantile regression models with or without regularization, master of science in statistical machine learning thesis, department of mathematical and statistical sciences, university of alberta. `https://era.library.ualberta.ca/items/24e3b3fc-ddd9-4320-9482-e60c2556f55b`, 2015.

Jueyu Gao and Linglong Kong. R Package 'cqrReg'. `https://cran.r-project.org/web/packages/cqrReg/`, 2015.

Xiaoli Gao and Yixin Fang. A note on the generalized degrees of freedom under the l1 loss function. *Journal of statistical planning and inference*, 141(2): 677–686, 2011.

Seymour Geisser. The predictive sample reuse method with applications. *Journal of the American statistical Association*, 70(350):320–328, 1975.

James E Gentle. Matrix transformations and factorizations. In *Matrix Algebra*, pages 227–263. Springer, 2017.

Marco Geraci. R package 'aqmm'. `https://github.com/marco-geraci/aqmm`, 2019a.

Marco Geraci. Additive quantile regression for clustered data with an application to children's physical activity. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 68(4):1071–1089, 2019b.

Ryan Giordano, Michael I Jordan, and Tamara Broderick. A higher-order swiss army infinitesimal jackknife. *arXiv preprint arXiv:1907.12116*, 2019.

Pontus Giselsson and Mattias Fält. Envelope functions: Unifications and further properties. *Journal of Optimization Theory and Applications*, 178(3):673–698, 2018.

Gene H Golub, Michael Heath, and Grace Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2): 215–223, 1979.

Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*, 2016.

Frithjof Gressmann, Franz J Király, Bilal Mateen, and Harald Oberhauser. Probabilistic supervised learning. *arXiv preprint arXiv:1801.00753*, 2018.

Yuwen Gu and Hui Zou. R Package 'fdhrq'. `http://users.stat.umn.edu/~zouxx019/ftpdir/code/fhdqr/`, 2017.

Yuwen Gu, Jun Fan, Lingchen Kong, Shiqian Ma, and Hui Zou. Admm for high-dimensional sparse penalized quantile regression. *Technometrics*, 60(3): 319–331, 2018.

Emil J Gumbel. *Statistics of extremes*. Columbia university press, 1958.

Emil J Gumbel. Bivariate logistic distributions. *Journal of the American Statistical Association*, 56(294):335–349, 1961.

Jiayi Guo. Smooth quasi-newton methods for nonsmooth optimization (phd thesis). *Cornell University*, 2018.

Jiayi Guo and A.S. Lewis. Bfgs convergence to nonsmooth minimizers of convex functions. *arXiv preprint arXiv:1703.06690*, 2017.

Jiayi Guo and A.S. Lewis. Nonsmooth variants of powell's bfgs convergence theorem. *SIAM Journal on Optimization*, 28(2):1301–1311, 2018a.

Jiayi Guo and A.S. Lewis. Rescaling nonsmooth optimization using bfgs and shor updates. *arXiv preprint arXiv:1802.06453*, 2018b.

Shaojun Guo, Yu Han, and Qingsong Wang. Better nonparametric confidence intervals via robust bias correction for quantile regression. *Stat*, 2021.

Marjo Haarala. Large-scale nonsmooth optimization: variable metric bundle method with limited memory, doctoral dissertation, university of jyväskylä, 2004.

Georg Hahn, Sharon M Lutz, Nilanjana Laha, and Christoph Lange. A framework to efficiently smooth l1 penalties for linear regression. *bioRxiv*, 2020.

Niels Richard Hansen. On stein's unbiased risk estimate for reduced rank estimators. *Statistics & Probability Letters*, 135:76–82, 2018.

Trevor Hastie and Robert Tibshirani. Generalized additive models: some applications. *Journal of the American Statistical Association*, 82(398):371–386, 1987.

Trevor Hastie and Robert Tibshirani. *Generalized additive models*. Chapman and Hall/CRC Monographs on Statistics and Applied Probability, 1990.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

Elad Hazan, Kfir Yehuda Levy, and Shai Shalev-Shwartz. On graduated optimization for stochastic non-convex problems. In *International conference on machine learning*, pages 1833–1841, 2016.

Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

Xuming He. Quantile curves without crossing. *The American Statistician*, 51 (2):186–192, 1997.

Xuming He, Xiaoou Pan, Kean Ming Tan, and Wen-Xin Zhou. Smoothed quantile regression with large-scale inference. *arXiv preprint arXiv:2012.05187*, 2020.

Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex analysis and minimization algorithms I: Fundamentals*, volume 305. Springer science & business media, 2013.

Thilo Hofmeister. R package 'qrsvm'. `https://cran.r-project.org/web/packages/qrsvm/`, 2017.

Joel L Horowitz. Bootstrap methods for median regression models. *Econometrica*, pages 1327–1351, 1998.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415. ACM, 2008.

Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425, 2002.

David R Hunter and Kenneth Lange. Quantile Regression via an MM Algorithm. *Journal of Computational and Graphical Statistics*, 9(1):60–77, 2000.

John K Hunter. Notes on partial differential equations, department of mathematics, university of california at davis. `https://www.math.ucdavis.edu/~hunter/pdes/pdes.html`, 2014.

Changha Hwang and Jooyong Shim. A simple quantile regression via support vector machine. *Advances in natural computation*, pages 418–418, 2005.

Lucas Janson, William Fithian, and Trevor J Hastie. Effective degrees of freedom: a flawed metaphor. *Biometrika*, 102(2):479–485, 2015.

LS Jennings, KH Wong, and KL Teo. Optimal control computation to account for eccentric movement. *The ANZIAM Journal*, 38(2):182–193, 1996.

Yunlu Jiang. Double-penalized quantile regression in partially linear models. *Open Journal of Statistics*, 5(02):158, 2015.

Norman L Johnson, Samuel Kotz, and Narayanaswamy Balakrishnan. *Continuous univariate distributions, volume 2*, volume 289. John wiley & sons, 1995.

A Johnston, T Auer, D Fink, M Strimas-Mackey, M Iliff, KV Rosenberg, S Brown, R Lanctot, AD Rodewald, and S Kelling. Comparing abundance distributions and range maps in spatial conservation planning for migratory species. *Ecological Applications*, 30(3):e02058, 2020.

Romain Juban, Henrik Ohlsson, Mehdi Maasoumy, Louis Poirier, and J Zico Kolter. A multiple quantile regression approach to the wind, solar, and price tracks of gefcom2014. *International Journal of Forecasting*, 32(3):1094–1102, 2016.

Yoonsuh Jung, Steven N MacEachern, and Hang Joon Kim. Modified check loss for efficient estimation via model selection in quantile regression. *Journal of Applied Statistics*, pages 1–21, 2020.

EE Kammann and MP Wand. Geoadditive models. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 52(1):1–18, 2003.

Kengo Kato. On the degrees of freedom in shrinkage estimation. *Journal of Multivariate Analysis*, 100(7):1338–1352, 2009.

Kengo Kato. Group lasso for high dimensional sparse quantile regression models. *arXiv preprint arXiv:1103.1458*, 2011.

S Kaufman and S Rosset. When does more regularization imply fewer degrees of freedom? sufficient conditions and counterexamples. *Biometrika*, 101(4): 771–784, 2014.

Tarak Kharrat, Georgi N Boshnakov, Ian McHale, and Rose Baker. Flexible regression models for count data based on renewal processes: The countr package. *Journal of Statistical Software*, 90(13), 2019.

Young-Ju Kim and Chong Gu. Smoothing spline gaussian regression: more scalable computation via efficient approximation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(2):337–356, 2004.

Marius Kloft, Ulf Brefeld, Sören Sonnenburg, and Alexander Zien. Lp-norm multiple kernel learning. *Journal of Machine Learning Research*, 12(Mar): 953–997, 2011.

Masha Kocherginsky, Xuming He, and Yunming Mu. Practical confidence intervals for regression quantiles. *Journal of Computational and Graphical Statistics*, 14(1):41–55, 2005.

Roger Koenker. Confidence intervals for regression quantiles. In *Asymptotic statistics*, pages 349–359. Springer, 1994.

Roger Koenker. Additive Models for Quantite Regression: Model Selection and Confidence Bandaids. *Brazilian Journal of Probability and Statistics*, pages 239–262, 2011.

Roger Koenker. Quantile regression in r: A vignette. `https://cran.r-project.org/web/packages/quantreg/vignettes/rq.pdf`, 2019.

Roger Koenker. Quantile regression methods: An r vinaigrette. `http://www.econ.uiuc.edu/~roger/research/vinaigrettes/QRMeth.pdf`, 2020a.

Roger Koenker. Conformal quantile regression: an r vinaigrette. `http://www.econ.uiuc.edu/~roger/research/vinaigrettes/conformal.pdf`, 2020b.

Roger Koenker. The group lasso for quantile regression: an r vinaigrette. `http://www.econ.uiuc.edu/~roger/research/bandaids/glasso.pdf`, 2020c.

Roger Koenker and Gilbert Bassett Jr. Regression quantiles. *Econometrica: journal of the Econometric Society*, pages 33–50, 1978.

Roger Koenker and Ivan Mizera. Convex optimization in r. *Journal of Statistical Software*, 60(5):1–23, 2014.

Roger Koenker and Pin Ng. A frisch-newton algorithm for sparse quantile regression. *Acta Mathematicae Applicatae Sinica*, 21(2):225–236, 2005.

Roger Koenker and Beum J Park. An interior point algorithm for nonlinear quantile regression. *Journal of Econometrics*, 71(1):265–283, 1996.

Roger Koenker, Pin Ng, and Stephen Portnoy. Quantile Smoothing Splines. *Biometrika*, 81(4):673–680, 1994.

Roger Koenker, Stephen Portnoy, Pin Tian Ng, Blaise Melly, Achim Zeileis, Philip Grosjean, Cleve Moler, Yousef Saad, Victor Chernozhukov, Fernández val Iván, and Brian D Ripley. Package 'quantreg'. `https://cran.r-project.org/web/packages/quantreg/quantreg.pdf`, 2021.

Linglong Kong, Haoxu Shu, Giseon Heo, and Qianchuan Chad He. Estimation for bivariate quantile varying coefficient model. *arXiv preprint arXiv:1511.02552*, 2015.

Ja-Yong Koo, Kwi Wook Park, Byung Won Kim, Kwang-Rae Kim, and Changyi Park. Structured kernel quantile regression. *Journal of Statistical Computation and Simulation*, 83(1):179–190, 2013.

Tatyana Krivobokova, Thomas Kneib, and Gerda Claeskens. Simultaneous confidence bands for penalized spline estimators. *Journal of the American Statistical Association*, 105(490):852–863, 2010.

Miroslav Kubat, Stan Matwin, et al. Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*, volume 97, pages 179–186. Citeseer, 1997.

Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-coordinate frank-wolfe optimization for structural svms. In *International Conference on Machine Learning*, pages 53–61. PMLR, 2013.

Ming-Jun Lai, Yangyang Xu, and Wotao Yin. Improved iteratively reweighted least squares for unconstrained smoothed ℓ^q minimization. *SIAM Journal on Numerical Analysis*, 51(2):927–957, 2013.

Kenneth Lange. *MM optimization algorithms*. SIAM, 2016.

Yoonkyung Lee, Steven N MacEachern, and Yoonsuh Jung. Regularization of case-specific parameters for robustness and efficiency. *Statistical Science*, pages 350–372, 2012.

Qi Lei, Ian En-Hsu Yen, Chao-yuan Wu, Inderjit S Dhillon, and Pradeep Ravikumar. Doubly greedy primal-dual coordinate descent for sparse empirical risk minimization. In *International Conference on Machine Learning*, pages 2034–2042, 2017.

Adrian S Lewis and Michael L Overton. Behavior of bfgs with an exact line search on nonsmooth examples. `http://www.cs.nyu.edu/overton/papers/pdffiles/bfgs_exactLS.pdf`, 2008.

Adrian S Lewis and Michael L Overton. Nonsmooth optimization via bfgs. `https://cs.nyu.edu/~overton/papers/pdffiles/bfgs_inexactLS.pdf`, 2009.

Adrian S Lewis and Michael L Overton. Nonsmooth optimization via quasi-newton methods. *Mathematical Programming*, 141(1):135–163, 2013.

Yi Li, Ruosong Wang, Lin Yang, and Hanrui Zhang. Nearly linear row sampling algorithm for quantile regression. In *International Conference on Machine Learning*, pages 5979–5989. PMLR, 2020.

Youjuan Li, Yufeng Liu, and Ji Zhu. Quantile regression in reproducing kernel hilbert spaces. *Journal of the American Statistical Association*, 102(477): 255–268, 2007.

Zhe Li, Tianbao Yang, Lijun Zhang, and Rong Jin. Fast and accurate refined nyström-based kernel svm. In *AAAI*, pages 1830–1836, 2016.

Heng Lian. Semiparametric estimation of additive quantile regression models by two-fold penalty. *Journal of Business & Economic Statistics*, 30(3):337–350, 2012.

Heng Lian, Jie Meng, and Zengyan Fan. Simultaneous estimation of linear conditional quantiles with penalized splines. *Journal of Multivariate Analysis*, 141:1–21, 2015.

Zhizheng Liang, Shixiong Xia, Jin Liu, Yong Zhou, and Lei Zhang. A majorization-minimization approach to lq norm multiple kernel learning. In *Pattern Recognition (ACPR), 2013 2nd IAPR Asian Conference on*, pages 366–370. IEEE, 2013.

Yaeji Lim and Hee-Seok Oh. Simultaneous confidence interval for quantile regression. *Computational Statistics*, 30(2):345–358, 2015.

Chen-Yen Lin, Howard Bondell, Hao Helen Zhang, and Hui Zou. Variable selection for non-parametric quantile regression via smoothing spline analysis of variance. *Stat*, 2(1):255–268, 2013.

Yi Lin and Hao Helen Zhang. Component selection and smoothing in multivariate nonparametric regression. *The Annals of Statistics*, 34(5):2272–2297, 2006.

Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2008.

Yongxin Liu, Peng Zeng, and Lu Lin. Degrees of freedom for regularized regression with huber loss and linear constraints. *Statistical Papers*, pages 1–23, 2020.

Zhe Liu and Ying Yang. Least absolute deviations estimation for uncertain regression with imprecise observations. *Fuzzy Optimization and Decision Making*, 19(1):33–52, 2020.

Mark A Lukas. Robust generalized cross-validation for choosing the regularization parameter. *Inverse Problems*, 22(5):1883, 2006.

Mark A Lukas. Strong robust generalized cross-validation for choosing the regularization parameter. *Inverse Problems*, 24(3):034006, 2008.

Mark A Lukas. Robust gcv choice of the regularization parameter for correlated data. *The Journal of integral equations and applications*, pages 519–547, 2010.

Mark A Lukas, Frank R de Hoog, and Robert S Anderssen. Efficient algorithms for robust generalized cross-validation spline smoothing. *Journal of computational and applied mathematics*, 235(1):102–107, 2010.

Mark A Lukas, Frank R de Hoog, and Robert S Anderssen. Practical use of robust gcv and modified gcv for spline smoothing. *Computational Statistics*, 31(1):269–289, 2016.

L Lukšan and J Vlček. Globally convergent variable metric method for convex nonsmooth unconstrained minimization. *Journal of Optimization Theory and Applications*, 102(3):593–613, 1999.

Ian Lundberg and Brandon M Stewart. Comment: Summarizing income mobility with multiple smooth quantiles instead of parameterized means. *Sociological Methodology*, 50(1):96–111, 2020.

A Luntz and V Brailovsky. On estimation of characters obtained in statistical procedure of recognition, technicheskaya kibernetica, vol. 3 (in russian), 1969.

Shaogao Lv, Xin He, and Junhui Wang. A unified penalized method for sparse additive quantile models: an rkhs approach. *Annals of the Institute of Statistical Mathematics*, pages 1–27, 2016.

Qin Lyu, Zhouchen Lin, Yiyuan She, and Chao Zhang. A comparison of typical ℓ_p minimization algorithms. *Neurocomputing*, 119:413–424, 2013.

Kaj Madsen and Hans Bruun Nielsen. A finite smoothing algorithm for linear l_1 estimation. *SIAM Journal on Optimization*, 3(2):223–235, 1993.

Subhransu Maji. Linearized smooth additive classifiers. In *Computer Vision–ECCV 2012. Workshops and Demonstrations*, pages 239–248. Springer, 2012.

Marko Mäkelä. Survey of bundle methods for nonsmooth optimization. *Optimization methods and software*, 17(1):1–29, 2002.

Jan Mankau and Friedemann Schuricht. A nonsmooth nonconvex descent algorithm. *arXiv preprint arXiv:1910.11199*, 2019.

Giampiero Marra and Simon N. Wood. Practical variable selection for generalized additive models. *Computational Statistics & Data Analysis*, 55(7): 2372–2387, 2011.

Dhruv Medarametla and Emmanuel J Candès. Distribution-free conditional median inference. *arXiv preprint arXiv:2102.07967*, 2021.

Aditya Krishna Menon. Large-scale support vector machines: algorithms and theory. *Technical report, University of California, San Diego*, 2009.

Mary Meyer and Michael Woodroofe. On the degrees of freedom in shape-restricted regression. *Annals of Statistics*, pages 1083–1104, 2000.

Frederik Riis Mikkelsen and Niels Richard Hansen. Degrees of freedom for piecewise lipschitz estimators. In *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, volume 54, pages 819–841. Institut Henri Poincaré, 2018.

Koji Miwa and Harald Baayen. Nonlinearities in bilingual visual word recognition: An introduction to generalized additive modeling. *Bilingualism: Language and Cognition*, pages 1–8, 2020.

Abdallah Mkhadri, Mohamed Ouhourane, and Karim Oualkacha. cdasqr r package. `https://karimoualkacha.wordpress.com/software/`, 2015.

Abdallah Mkhadri, Mohamed Ouhourane, and Karim Oualkacha. A coordinate descent algorithm for computing penalized smooth quantile regression. *Statistics and Computing*, 27(4):865–883, 2017.

Boris S Mordukhovich and Nguyen Mau Nam. *An easy path to convex analysis and applications*. Morgan & Claypool Publishers, 2013.

Boris S Mordukhovich and Nguyen Mau Nam. Geometric approach to convex subdifferential calculus. *Optimization*, 66(6):839–873, 2017.

Vito MR Muggeo. Package 'quantreggrowth'. `https://cran.r-project.org/web/packages/quantregGrowth/index.html`, 2021.

Vito MR Muggeo, Mariangela Sciandra, and Luigi Augugliaro. Quantile regression via iterative least squares computations. *Journal of Statistical Computation and Simulation*, 82(11):1557–1569, 2012.

Vito MR Muggeo, Federico Torretta, Paul HC Eilers, Mariangela Sciandra, and Massimo Attanasio. Multiple smoothing parameters selection in additive regression quantiles. *Statistical Modelling*, page 1471082X20929802, 2020.

Bhaskar Mukhoty, Govind Gopakumar, Prateek Jain, and Purushottam Kar. Globally-convergent iteratively reweighted least squares for robust regression problems. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 313–322. PMLR, 2019.

Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

Hien D Nguyen and Geoffrey J McLachlan. Iteratively-reweighted least-squares fitting of support vector machines: A majorization–minimization algorithm approach. *Future Technologies Conference (FTC), Vancouver, Canada*, 2017.

Bertand Nortier and Simon N. Wood. Automated smoothing parameter selection for quantile additive models. `http://2018.ds3-datascience-polytechnique.fr/wp-content/uploads/2018/06/DS3-512.pdf`, 2018.

Doug Nychka, Gerry Gray, Perry Haaland, David Martin, and Michael O'connell. A Nonparametric Regression Approach to Syringe Grading for Quality Improvement. *Journal of the American Statistical Association*, 90 (432):1171–1178, 1995.

Mícheál Ó Searcóid. *Metric spaces*. Springer Science & Business Media, 2006.

Peter Ochs, René Ranftl, Thomas Brox, and Thomas Pock. Bilevel Optimization with Nonsmooth Lower Level Problems. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 654–665. Springer, 2015.

Peter Ochs, René Ranftl, Thomas Brox, and Thomas Pock. Techniques for Gradient-Based Bilevel Optimization with Non-smooth Lower Level Problems. *Journal of Mathematical Imaging and Vision*, pages 1–20, 2016.

Michael O'Connell and Douglas Nychka. A generalized linear classification model with a smooth link function and predictors obtained from quantile spline fits to high-dimensional data. *Journal of statistical planning and inference*, 47(1-2):153–164, 1995.

Hee-Seok Oh, Thomas CM Lee, and Douglas W Nychka. Fast nonparametric quantile regression with arbitrary smoothing methods. *Journal of Computational and Graphical Statistics*, 2012.

Edgar Osuna, Robert Freund, and Federico Girosi. An improved training algorithm for support vector machines. In *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*, pages 276–285. IEEE, 1997.

Balamurugan Palaniappan and Francis Bach. Stochastic variance reduction methods for saddle-point problems. In *Advances in Neural Information Processing Systems*, pages 1416–1424, 2016.

Kaare Brandt Petersen and Michael Syskind Pedersen. The matrix cookbook (version: November 15, 2012), 2012.

Matthew Pietrosanu, Jueyu Gao, Linglong Kong, Bei Jiang, and Di Niu. Advanced algorithms for penalized quantile and composite quantile regression. *Computational Statistics*, 36(1):333–346, 2021.

Nicholas G Polson and James G Scott. Mixtures, envelopes and hierarchical duality. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(4):701–727, 2016.

Julien Prados. R package 'bmrm'. `https://cran.r-project.org/web/packages/bmrm/`, 2017.

Hu Qinghui, Wei Shiwei, Li Zhiyuan, and Liu Xiaogang. Quasi-newton method for ℓ^p multiple kernel learning. *Neurocomputing*, 194:218–226, 2016.

Karthikeyan Natesan Ramamurthy, Aleksandr Y Aravkin, and Jayaraman J Thiagarajan. Automatic inference of the quantile parameter. *arXiv preprint arXiv:1511.03990*, 2015.

Edmore Ranganai and Caston Sigauke. Capturing long-range dependence and harmonic phenomena in 24-hour solar irradiance forecasting: A quantile regression robustification via forecasts combination approach. *IEEE Access*, 8: 172204–172218, 2020.

Christian H Reinsch. Smoothing by spline functions. *Numerische mathematik*, 10(3):177–183, 1967.

Philip T Reiss and Lei Huang. Smoothness Selection for Penalized Quantile Regression Splines. *The international journal of biostatistics*, 8(1), 2012.

RA Rigby, DM Stasinopoulos, and V Voudouris. Discussion: A comparison of gamlss with quantile regression. *Statistical Modelling*, 13(4):335–348, 2013.

Adrian P. Robson. The flowchart package: Flowchart shapes for tikz. `https://www.ctan.org/pkg/flowchart`, 2015.

Yaniv Romano, Evan Patterson, and Emmanuel J Candès. Conformalized quantile regression. *Advances in Neural Information Processing Systems 32*, pages 3543–3553, 2019.

Saharon Rosset. Bi-level Path Following for Cross Validated Solution of Kernel Quantile Regression. *Journal of Machine Learning Research*, 10(Nov):2473–2505, 2009.

David Ruppert, Matt P Wand, and Raymond J Carroll. *Semiparametric regression*. Cambridge university press, 2003.

Ahmet Sahiner, Nurullah Yilmaz, and Shehab Ahmed IBRAHEM. Smoothing approximations to non-smooth functions. *Journal of Multidisciplinary Modeling and Optimization*, 1(2):69–74, 2018.

Ahmet Sahiner, Nurullah Yilmaz, and Gulden Kapusuz. A novel modeling and smoothing technique in global optimization. *Journal of Industrial & Management Optimization*, 15(1):113, 2019.

Ahmet Sahiner, Idris A Masoud Abdulhamid, and Nurullah Yilmaz. Escaping from current minimizer by using an auxiliary function smoothed by bezier curves. In *Numerical Solutions of Realistic Nonlinear Phenomena*, pages 87–105. Springer, 2020.

Maxime Sangnier, Olivier Fercoq, and Florence d'Alché Buc. Joint quantile regression in vector-valued rkhss. In *Advances in Neural Information Processing Systems*, pages 3693–3701, 2016.

Sabine K Schnabel and Paul HC Eilers. Simultaneous estimation of quantile curves using quantile sheets. *AStA Advances in Statistical Analysis*, 97(1):77–87, 2013.

Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013.

Saahil Shenoy and Dimitry Gorinevsky. Stochastic optimization of power market forecast using non-parametric regression models. In *Power & Energy Society General Meeting, 2015 IEEE*, pages 1–5. IEEE, 2015.

Saahil Shenoy, Dimitry Gorinevsky, and Stephen Boyd. Non-parametric regression modeling for stochastic optimization of power grid load forecast. In *American Control Conference (ACC), 2015*, pages 1010–1015. IEEE, 2015.

Hao-Jun Michael Shi, Shenyinying Tu, Yangyang Xu, and Wotao Yin. A primer on coordinate descent algorithms. *arXiv preprint arXiv:1610.00040*, 2016.

Jooyong Shim, Changha Hwang, and Kyung Ha Seok. Non-crossing quantile regression via doubly penalized kernel machine. *Computational Statistics*, 24 (1):83–94, 2009.

Martin Siebenborn and Julian Wagner. A multigrid preconditioner for tensor product spline smoothing. *Computational Statistics*, pages 1–33, 2021.

Anders Skajaa. Limited memory bfgs for nonsmooth optimization. *Master's thesis, Courant Institute of Mathematical Science, New York University*, 2010.

Fabian Sobotka and Thomas Kneib. Geoadditive expectile regression. *Computational Statistics & Data Analysis*, 56(4):755–767, 2012.

Fabian Sobotka, Göran Kauermann, Linda Schulze Waltrup, and Thomas Kneib. On confidence intervals for semiparametric expectile regression. *Statistics and Computing*, 23(2):135–148, 2013.

Raphael Sonabend, Franz Király, Peter Ruckdeschel, Matthias Kohl, Nurul Ain Toha, Shen Chen, Jordan Deenichin, Chengyang Gao, Chloe Zhaoyuan Gu, Yunjie He, Xiaowen Huang, Shuhan Liu, Runlong Yu, Chijing Zeng, and Qian Zhou. Sigmoid kernel: part of the distr6 r package. `https://alan-turing-institute.github.io/distr6/reference/Sigmoid.html`, retrieved 2021.

Karthik Sriram and RV Ramamoorthi. On square root n consistency for bayesian quantile regression based on the misspecified asymmetric laplace likelihood. *arXiv preprint arXiv:1812.03652*, 2018.

Karthik Sriram, RV Ramamoorthi, and Pulak Ghosh. Posterior consistency of bayesian quantile regression based on the misspecified asymmetric laplace density. *Bayesian Analysis*, 8(2):479–504, 2013.

Charles M Stein. Estimation of the mean of a multivariate normal distribution. *The annals of Statistics*, pages 1135–1151, 1981.

Ingo Steinwart and Andreas Christmann. Estimating conditional quantiles with the help of the pinball loss. *Bernoulli*, 17(1):211–225, 2011.

Lorenzo Stella and Panagiotis Patrinos. 'forbes' package. `http://kul-forbes.github.io/ForBES/`, 2016.

Lorenzo Stella, Andreas Themelis, and Panagiotis Patrinos. Forward–backward quasi-newton methods for nonsmooth optimization problems. *Computational Optimization and Applications*, 67(3):443–487, 2017.

Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133, 1974.

Ying Sun, Prabhu Babu, and Daniel P Palomar. Majorization-minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing*, 65(3):794–816, 2016.

Ichiro Takeuchi and Takeshi Furuhashi. Non-crossing quantile regressions by svm. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 1, pages 401–406. IEEE, 2004.

Ichiro Takeuchi, Quoc V Le, Timothy D Sears, and Alexander J Smola. Nonparametric Quantile Estimation. *Journal of Machine Learning Research*, 7 (Jul):1231–1264, 2006.

Garth Tarr. Small sample performance of quantile regression confidence intervals. *Journal of Statistical Computation and Simulation*, 82(1):81–94, 2012.

Choon Hui Teo, SVN Vishwanthan, Alex J Smola, and Quoc V Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 11(Jan):311–365, 2010.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Ryan J Tibshirani. Advanced methods for data analysis: Spring 2014 statistics 36-402/36-608. `https://www.stat.cmu.edu/~ryantibs/advmethods/notes/errval.pdf`, 2014.

Ryan J Tibshirani. Degrees of freedom and model search. *Statistica Sinica*, pages 1265–1296, 2015.

Ryan J Tibshirani and Saharon Rosset. Excess optimism: how biased is the apparent error of an estimator tuned by sure? *Journal of the American Statistical Association*, 114(526):697–712, 2019.

Fabian Tomaschek, Benjamin V Tucker, Matteo Fasiolo, and R Harald Baayen. Practice makes perfect: The consequences of lexical proficiency for articulation. *Linguistics Vanguard*, 4(s2), 2018.

Léonard Torossian, Victor Picheny, Robert Faivre, and Aurélien Garivier. A review on quantile regression for stochastic computer experiments. *Reliability Engineering & System Safety*, 201:106858, 2020.

Federico Torretta. P-spline quantile regression: a new algorithm for smoothing parameter selection. *PhD Thesis, Università degli Studi di Palermo*, 2016.

Federico Torretta, Vito M.R. Muggeo, and Paul H.C. Eilers. P-spline quantile regression with a mixed model algorithm. *Proceedings 30th International Workshop on Statistical Modelling, Linz*, 2015.

Shanshan Tu, Yunzhang Zhu, and Yoonkyung Lee. Cross validation for penalized quantile regression with a case-weight adjusted solution path. *arXiv preprint arXiv:1902.07770*, 2019.

Berwin A. Turlach and Andreas Weingessel. R package 'quadprog', version 1.5-5. `https://cran.r-project.org/web/packages/quadprog/`, 2015.

Nikolaus Umlauf, Thomas Kneib, Nadja Klein, Felix Heinzl, Andreas Brezger, and Daniel Sabanes Bove. Package bayesx. `https://cran.r-project.org/web/packages/BayesX/index.html`, 2018.

Nikolaus Umlauf, Nadja Klein, Thorsten Simon, and Achim Zeileis. bamlss: A lego toolbox for flexible bayesian regression (and beyond). *arXiv preprint arXiv:1909.11784*, 2019.

Samuel Vaiter, Charles-Alban Deledalle, Gabriel Peyré, Charles Dossal, and Jalal Fadili. Local behavior of sparse analysis regularization: Applications to risk estimation. *Applied and Computational Harmonic Analysis*, 35(3): 433–451, 2013.

Samuel Vaiter, Charles Deledalle, Jalal Fadili, Gabriel Peyré, and Charles Dossal. The degrees of freedom of partly smooth regularizers. *Annals of the Institute of Statistical Mathematics*, 69(4):791–832, 2017.

Vladimir Vapnik. Statistical learning theory wiley. *New York*, 1:624, 1998.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. Conformal prediction. *Algorithmic learning in a random world*, pages 17–51, 2005.

Julian Wagner, Göran Kauermann, and Ralf Münnich. Matrix-free penalized spline smoothing with multiple covariates. *arXiv preprint arXiv:2101.06034*, 2021.

Grace Wahba. Support vector machines, reproducing kernel hilbert spaces and the randomized gacv. *Advances in Kernel Methods-Support Vector Learning*, 6:69–87, 1999.

Elisabeth Waldmann. Quantile regression: a short story on how and why. *Statistical Modelling*, 18(3-4):203–218, 2018.

Elisabeth Waldmann, Thomas Kneib, Yu Ryan Yue, Stefan Lang, and Claudia Flexeder. Bayesian semiparametric additive quantile regression. *Statistical Modelling*, 13(3):223–252, 2013.

Conor Waldock, Rick D Stuart-Smith, Graham J Edgar, Tomas J Bird, and Amanda E Bates. The shape of abundance distributions across temperature gradients in reef fishes. *Ecology letters*, 22(4):685–696, 2019.

Linda Schulze Waltrup, Fabian Sobotka, Thomas Kneib, and Göran Kauermann. Expectile and quantile regression-david and goliath? *Statistical Modelling*, 15(5):433–456, 2015.

Haiying Wang and Yanyuan Ma. Optimal subsampling for quantile regression in big data. *Biometrika*, 2020.

Huixia Judy Wang, Leonard A Stefanski, and Zhongyi Zhu. Corrected-loss estimation for quantile regression with covariate measurement errors. *Biometrika*, 2012.

Zhefeng Wang and Wanzhou Ye. An efficient smooth quantile boost algorithm for binary classification. *Advances in Pure Mathematics*, 6(9):615–624, 2016.

Jack Warga. Minimizing certain convex functions. *Journal of the Society for Industrial and Applied Mathematics*, 11(3):588–593, 1963.

Simon N. Wood. *Generalized additive models: an introduction with R (first edition)*. Chapman & Hall, 2006.

Simon N. Wood. A toolbox of smooths. `https://www.maths.ed.ac.uk/~swood34/mgcv/tampere/smooth-toolbox.pdf`, 2010.

Simon N. Wood. Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(1):3–36, 2011.

Simon N. Wood. *Core statistics*, volume 6. Cambridge University Press, 2015.

Simon N. Wood. *Generalized Additive Models: an Introduction with R, Second Edition*. CRC press, 2017.

Simon N. Wood. Package 'gamair'. `https://cran.r-project.org/web/packages/gamair/index.html`, 2019.

Simon N. Wood. R package mgcv. `https://cran.r-project.org/web/packages/mgcv/index.html`, 2020.

Simon N. Wood. Low rank gaussian process smooths. `https://stat.ethz.ch/R-manual/R-patched/library/mgcv/html/smooth.construct.gp.smooth.spec.html`, retrieved 2020a.

Simon N. Wood. Smooth terms in gam. `https://stat.ethz.ch/R-manual/R-devel/RHOME/library/mgcv/html/smooth.terms.html`, retrieved 2020b.

Simon N. Wood and Matteo Fasiolo. A generalized fellner-schall method for smoothing parameter optimization with application to tweedie location, scale and shape models. *Biometrics*, 73(4):1071–1081, 2017.

Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

Chaojiang Wu and Yan Yu. Partially Linear Modeling of Conditional Quantiles using Penalized Splines. *Computational Statistics & Data Analysis*, 77:170–187, 2014.

Teng Wu and Naveen N Narisetty. Bayesian multiple quantile regression for linear models using a score likelihood. *Bayesian Analysis*, 2020.

Tong Tong Wu and Kenneth Lange. Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, 2(1):224–244, 2008.

Dong Xiang and Grace Wahba. A Generalized Approximate Cross Validation for Smoothing Splines with Non-Gaussian Data. *Statistica Sinica*, pages 675–692, 1996.

Shangyu Xie, Alan TK Wan, and Yong Zhou. Quantile regression methods with varying-coefficient models for censored data. *Computational Statistics & Data Analysis*, 88:154–172, 2015.

Yuchen Xie and Andreas Waechter. On the convergence of bfgs on a class of piecewise linear non-smooth functions. *arXiv preprint arXiv:1712.08571*, 2017.

Qifa Xu, Chao Cai, Cuixia Jiang, Fang Sun, and Xue Huang. Sampling lasso quantile regression for large-scale data. *Communications in Statistics-Simulation and Computation*, 47(1):92–114, 2018.

Zheng Xu, Mário AT Figueiredo, and Tom Goldstein. Adaptive admm with spectral penalty parameter selection. In *Artificial Intelligence and Statistics*, pages 718–727. PMLR, 2017a.

Zheng Xu, Mário AT Figueiredo, Xiaoming Yuan, Christoph Studer, and Tom Goldstein. Adaptive relaxed admm: Convergence theory and practical implementation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7389–7398, 2017b.

Tianbao Yang, Mehrdad Mahdavi, Rong Jin, and Shenghuo Zhu. An efficient primal dual prox method for non-smooth optimization. *Machine Learning*, 98(3):369–406, 2015.

Jianming Ye. On measuring and correcting the effects of data mining and model selection. *Journal of the American Statistical Association*, 93(441):120–131, 1998.

Thomas W Yee. Vgam family functions for quantile regression. `https://www.stat.auckland.ac.nz/~yee/VGAM/doc/qreg.pdf`, 2008.

Thomas W Yee and Alec G Stephenson. Vector generalized linear and additive extreme value models. *Extremes*, 10(1-2):1–19, 2007.

Show-Jane Yen and Yue-Shi Lee. Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset. In *Intelligent Control and Automation*, pages 731–740. Springer, 2006.

Congrui Yi and Jian Huang. Semismooth newton coordinate descent algorithm for elastic-net penalized huber loss regression and quantile regression. *Journal of Computational and Graphical Statistics*, 26(3):547–557, 2017.

Nurullah Yilmaz and Ahmet Sahiner. New smoothing approximations to piecewise smooth functions and applications. *Numerical Functional Analysis and Optimization*, 40(5):513–534, 2019.

Nurullah Yilmaz and Ahmet Sahiner. On a new smoothing technique for nonsmooth, non-convex optimization. *Numerical Algebra, Control & Optimization*, 10(3):317, 2020.

Takuma Yoshida. Additive models for extremal quantile regression with paretotype distributions. *AStA Advances in Statistical Analysis*, pages 1–32, 2020a.

Takuma Yoshida. Simultaneous confidence bands for extremal quantile regression with splines. *Extremes*, 23(1):117–149, 2020b.

Benjamin D Youngman. evgam: An r package for generalized additive extreme value models. *arXiv preprint arXiv:2003.04067*, 2020.

Dengdeng Yu, Li Zhang, Ivan Mizera, Bei Jiang, and Linglong Kong. Sparse wavelet estimation in quantile regression with multiple functional predictors. *Computational Statistics & Data Analysis*, 136:12–29, 2019.

Jin Yu. New quasi-newton optimization methods for machine learning, phd thesis, the australian national university, 2009.

Jin Yu, SVN Vishwanathan, Simon Günter, and Nicol N Schraudolph. A quasi-newton approach to nonsmooth convex optimization problems in machine learning. *Journal of Machine Learning Research*, 11(Mar):1145–1200, 2010.

Keming Yu and Rana A Moyeed. Bayesian quantile regression. *Statistics & Probability Letters*, 54(4):437–447, 2001.

Liqun Yu and Nan Lin. Admm for penalized quantile regression in big data. *International Statistical Review*, 2017.

Liqun Yu, Nan Lin, and Lan Wang. A parallel algorithm for large-scale nonconvex penalized quantile regression. *Journal of Computational and Graphical Statistics*, 26(4):935–939, 2017.

Ming Yuan. GACV for Quantile Smoothing Splines. *Computational statistics & data analysis*, 50(3):813–829, 2006.

Yu Ryan Yue and Håvard Rue. Bayesian inference for additive mixed quantile regression models. *Computational Statistics & Data Analysis*, 55(1):84–96, 2011.

Donglin Zeng and DY Lin. Efficient resampling methods for nonsmooth estimating functions. *Biostatistics*, 9(2):355–363, 2008.

Fode Zhang, Xuejun Wang, Rui Li, and Heng Lian. Randomized sketches for sparse additive models. *Neurocomputing*, 385:80–87, 2020a.

Fode Zhang, Rui Li, and Heng Lian. Approximate nonparametric quantile regression in reproducing kernel hilbert spaces via random projection. *Information Sciences*, 547:244–254, 2021.

Hao Helen Zhang and Chen-Yen Lin. Package cosso. `https://cran.r-project.org/web/packages/cosso/`, 2013.

Kai Zhang, Liang Lan, Zhuang Wang, and Fabian Moerchen. Scaling up kernel svm on limited resources: A low-rank linearization approach. In *Artificial Intelligence and Statistics*, pages 1425–1434, 2012.

Lianjun Zhang, Y Li, L Kong, and Shenglong Zhou. A smoothing iterative method for quantile regression with nonconvex ℓ^p penalty. *J. Ind. Manag. Optim*, 13(1):93–112, 2017.

Likun Zhang, Enrique del Castillo, Andrew J Berglund, Martin P Tingley, and Nirmal Govind. Computing confidence intervals from massive data via penalized quantile smoothing splines. *Computational Statistics & Data Analysis*, 144:106885, 2020b.

GH Zhao, Kok Lay Teo, and KS Chan. Estimation of conditional quantiles by a new smoothing approximation of asymmetric loss functions. *Statistics and Computing*, 15(1):5–11, 2005.

Kaifeng Zhao and Heng Lian. Variable selection in additive quantile regression using nonconcave penalty. *Statistics*, 50(6):1276–1289, 2016.

Songfeng Zheng. Gradient descent algorithms for quantile regression with smooth approximation. *International Journal of Machine Learning and Cybernetics*, 2(3):191, 2011.

Songfeng Zheng. A generalized newton algorithm for quantile regression models. *Computational Statistics*, 29(6):1403–1426, 2014.

Ying-hui Zhou, Zhong-xin Ni, and Yong Li. Quantile regression via the em algorithm. *Communications in Statistics-Simulation and Computation*, 43 (10):2162–2172, 2014.

Yunzhang Zhu. An augmented admm algorithm with application to the generalized lasso problem. *Journal of Computational and Graphical Statistics*, 26 (1):195–204, 2017.

Zeyuan Allen Zhu, Weizhu Chen, Gang Wang, Chenguang Zhu, and Zheng Chen. P-packsvm: Parallel primal gradient descent kernel svm. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 677–686. IEEE, 2009.