Available online at www.sciencedirect.com
ScienceDirect
Computational Statistics & Data Analysis 52 (2008) 1292 – 1303
COMPUTATIONAL STATISTICS & DATA ANALYSIS
www.elsevier.com/locate/csda

Time-adaptive quantile regression

Jan Kloppenborg Møller*, Henrik Aalborg Nielsen, Henrik Madsen

Informatics and Mathematical Modelling, Richard Pedersens Plads, Technical University of Denmark Building 321, DK-2800 Lyngby, Denmark

Received 22 August 2006; received in revised form 20 June 2007; accepted 21 June 2007
Available online 19 July 2007

Abstract

An algorithm for time-adaptive quantile regression is presented. The algorithm is based on the simplex algorithm, and the linear optimization formulation of the quantile regression problem is given. The observations have been split to allow a direct use of the simplex algorithm. The simplex method and an updating procedure are combined into a new algorithm for time-adaptive quantile regression, which generates new solutions on the basis of the old solution, leading to savings in computation time. The suggested algorithm is tested against a static quantile regression model on a data set with wind power production, where the models combine splines and quantile regression. The comparison indicates superior performance for the time-adaptive quantile regression in all the performance parameters considered.[1]
© 2007 Elsevier B.V. All rights reserved.

Keywords: Quantile regression; Time-adaptive; Simplex; Wind power

1. Introduction

The most commonly used statistic is the mean, and least squares methods for estimating the mean are well known. However, the mean value does not give any information on the random behavior of the data, and more information is therefore often requested. Pinson et al. (2007c) show that using an advanced prediction tool for wind power prediction for the Dutch market led to 38% savings in trading cost. If a reliable estimate of the uncertainty of the future wind power was also available, a further saving of 39% of trading costs could be obtained. In this paper wind power predictions will be used to demonstrate this method.

One way of obtaining information on the random behavior of the prediction errors is to estimate higher order moments of the data. If some assumptions are made on the distribution of errors, then a finite number of moments will characterize the distribution. E.g. if the error is assumed to be Gaussian then the two first moments characterize the distribution completely.

However, it is often not reasonable to assume a specific, known distribution of the data. In this case quantile regression is a way of estimating quantiles directly using linear regression techniques. In addition to the distribution of data being unknown, it is often reasonable to assume that the distribution of data varies over time. For slowly varying non-stationary systems time-adaptive mean value regression as described by Ljung and Söderström (1983) has proven to be effective in many applications. This article offers an algorithm for time-adaptive quantile regression.

* Corresponding author. Tel.: +45 4525 3369.
E-mail addresses: jkm@imm.dtu.dk (J.K Møller), han@imm.dtu.dk (H.A. Nielsen), hm@imm.dtu.dk (H. Madsen).
[1] The Matlab algorithms can be obtained from http://www.imm.dtu.dk/~jkm.

0167-9473/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.csda.2007.06.027

The loss function for quantile regression is based on a weighting of the absolute values of the residuals. Such problems can be solved using linear optimization techniques. Portnoy and Koenker (1997) give a nice discussion of some of the algorithms available for quantile regression. The algorithms discussed in this paper are all available in the add-on package "`quantreg`" for "R" (available at http://www.r-project.org/). The paper shows that quantile regression methods can be very fast, also compared to mean value regression. However, these algorithms do not offer the possibility to use knowledge about the solution of the system. The aim of this article is to present an algorithm where knowledge about the solution at time t is used to solve the system at time $t+1$, thereby reducing the computational effort and increasing computation speed.

In this article we present the linear optimization formulations of the quantile regression problem and propose a solution using the simplex method directly. This enables a formulation suitable for computer implementation. The implementation of the simplex method is then combined with an updating procedure to update information related to the actual estimates.

An example of a system that has demonstrated a need for time-adaptive methods for mean value regression, is wind power predictions (see Madsen, 1996; Madsen et al., 2005). As was explained in the beginning of this section, there are huge potential economic benefits from good estimates of the uncertainty in wind power predictions. The last part of this article presents the results from applying the *time-adaptive quantile regression* methods, combined with splines basis functions to wind power predictions. In online settings for wind power prediction data often arrive every 15 min making it possible to update the solutions at each of these time points. The strength of the methods is illustrated in this example, where the time-adaptive method proves to be superior to corresponding static models. The section also provides some promising timing examples for the method.

Chambers et al. (2006) present an algorithm for updating univariate quantile estimates. The approach in this paper is to update empirical quantiles as new data become available. Even though the problem set up is somewhat different from the one presented here, the article motivates the need for time-adaptive quantile estimation. As a main difference time-adaptive quantile regression offers the inclusion of regression parameters.

2. Quantile regression

Quantile regression as presented by Koenker and Bassett (1978), is based on a linear model

$$y = \boldsymbol{x}^{\mathrm{T}}\hat{\boldsymbol{\beta}} + r = \hat{Q}(\tau;\boldsymbol{x}) + r, \tag{1}$$

which is fitted using an asymmetrical and piecewise linear loss function

$$\rho_\tau(r) = \begin{cases} \tau r, & r \geqslant 0, \\ (\tau-1)r, & r < 0. \end{cases} \tag{2}$$

The observation equations are written as

$$\boldsymbol{y} = X\hat{\boldsymbol{\beta}} + \boldsymbol{r}. \tag{3}$$

X is called the design matrix, $\hat{\boldsymbol{\beta}}$ is the coefficients to be estimated and $\boldsymbol{y}$ is the response from the system. The estimate of $\boldsymbol{\beta} \in \mathbb{R}^K$ given N observations is

$$\hat{\boldsymbol{\beta}}(\tau) = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{N} \rho_\tau(r_i) = \arg\min_{\boldsymbol{\beta}} S_\tau(\boldsymbol{\beta};\boldsymbol{r}). \tag{4}$$

This is a linear optimization problem. Koenker and Bassett (1978) show that a solution to Eq. (4) is

$$\hat{\boldsymbol{\beta}}(\tau) = X(h)^{-1}\boldsymbol{y}(h), \tag{5}$$

where h is a K element index set from the set $\Omega = \{1, 2, \ldots, N\}$. $X(h)$ refer to rows in X, as this notation is also used in Koenker (2005) and Koenker and Bassett (1978). Solutions like this are obtained using linear programming techniques. In general interior point methods are considered superior to the simplex algorithm for large problems, see Koenker (2005, p. 190). The simplex method is, however, expected to be a reasonable choice, if a good candidate for $\hat{\boldsymbol{\beta}}$ is available. This is referred to as a "*warm start*" problem by Nielsen (1999, p. 75).

This article presents an algorithm for time-adaptive quantile regression. The algorithm is based on the simplex method. We use the simplex algorithm to update the solution directly as new observations become available. The structure of the quantile regression problem is exploited to enable an efficient implementation of the simplex algorithm.

3. The linear optimization formulation

A linear optimization formulation of the quantile regression problem is

$$\min\{\tau\mathbf{1}^{\mathrm{T}}\boldsymbol{r}^{+}+(1-\tau)\mathbf{1}^{\mathrm{T}}\boldsymbol{r}^{-}:X\boldsymbol{\beta}+\boldsymbol{r}^{+}-\boldsymbol{r}^{-}=\boldsymbol{y},(\boldsymbol{r}^{+},\boldsymbol{r}^{-})\in\mathbb{R}_0^{2N},\boldsymbol{\beta}\in\mathbb{R}^K\},\tag{6}$$

with $r_i^+=I(r_i\geqslant 0)r_i$, $r_i^-=-I(r_i\leqslant 0)r_i$, and $\mathbf{1}$ a vector of ones. This formulation is also given by Koenker (2005, p. 181). Koenker (2005) goes on to formulate the dual problem, here we will examine the direct simplex formulation of Eq. (6). The formulation uses what Chvátal (1983) refer to as the revised simplex method. The close connection to the dual simplex formulation becomes clear if the results presented in this article are compared with the results presented by Koenker (2005, p. 182).

In a more compact notation Eq. (6) can be written as

$$\min\{\boldsymbol{c}^{\mathrm{T}}\boldsymbol{z}:A\boldsymbol{z}=\boldsymbol{y},(\boldsymbol{r}^{+},\boldsymbol{r}^{-})\in\mathbb{R}_0^{2N},\boldsymbol{\beta}\in\mathbb{R}^K\},\tag{7}$$

with

$$\boldsymbol{c}=\begin{bmatrix}\mathbf{0}_K\\ \tau\mathbf{1}\\ (1-\tau)\mathbf{1}\end{bmatrix},\quad \boldsymbol{z}=\begin{bmatrix}\boldsymbol{\beta}\\ \boldsymbol{r}^{+}\\ \boldsymbol{r}^{-}\end{bmatrix},\quad A=[X,I,-I].\tag{8}$$

The simplex algorithm moves through vertex solutions, i.e. solutions on the edge of the feasible region in the direction of better objective functions ($\boldsymbol{c}^{\mathrm{T}}\boldsymbol{z}$). At a vertex the equality constraints in Eq. (7) can be written as

$$A\boldsymbol{z}=B\boldsymbol{z}_{\mathscr{B}}+C\boldsymbol{z}_{\mathscr{C}}=\boldsymbol{y},\tag{9}$$

where $(\mathscr{B},\mathscr{C})$ is a partition of the index set $\{1,2,\ldots,2N+K\}$. This index set refers to columns in the matrix A; note here that $X\in\mathbb{R}^{N\times K}$ and $A\in\mathbb{R}^{N\times(2N+K)}$. This partition is such that $B=A_{:,\mathscr{B}}$, $A_{:,\mathscr{B}}$ refer to the columns of A defined by $\mathscr{B}$ is a square matrix with an inverse and $\boldsymbol{z}_{\mathscr{C}}=\mathbf{0}$.

To give an intuitive interpretation of the simplex algorithm for quantile regression, we write down the observation equations at a vertex

$$\boldsymbol{y}=X\boldsymbol{\beta}+\boldsymbol{r}=B\boldsymbol{z}_{\mathscr{B}}.\tag{10}$$

Now let an estimate of $\boldsymbol{\beta}$ be given by an index set h cf. Eq. (5), it is required that rank $X=K$. It is not demanded that $\boldsymbol{\beta}$ is the best estimate. With $\boldsymbol{\beta}$ written like this, we have

$$\boldsymbol{y}(h)=X(h)\boldsymbol{\beta}+\boldsymbol{r}(h)=X(h)X(h)^{-1}\boldsymbol{y}(h)+\boldsymbol{r}(h)=\boldsymbol{y}(h)+\boldsymbol{r}(h),\tag{11}$$

this immediately gives $\boldsymbol{r}(h)=0$. The matrices B and C can therefore (using Eqs. (10) and (11) and interchanging rows) be written as

$$B=\begin{bmatrix}X(h)&0\\ X(\bar{h})&P\end{bmatrix};\quad C=\begin{bmatrix}I_K&-I_K&0\\ 0&0&-P\end{bmatrix},\tag{12}$$

where $P=\mathrm{diag}(\boldsymbol{p})$ is a diagonal matrix with diagonal elements equal to the sign of $\boldsymbol{r}(\bar{h})$, i.e. $\boldsymbol{p}=\mathrm{sign}\{\boldsymbol{r}(\bar{h})\}$, $\bar{h}=\Omega\backslash h$ and I_K is the $K\times K$ identity matrix. To emphasize how this is connected to the linear model rewrite Eq. (10) as

$$\begin{bmatrix}\boldsymbol{y}(h)\\ \boldsymbol{y}(\bar{h})\end{bmatrix}=B\begin{bmatrix}\boldsymbol{z}_{\mathscr{B}(1:K)}\\ \boldsymbol{z}_{\mathscr{B}(K+1:N)}\end{bmatrix}=\begin{bmatrix}X(h)\\ X(\bar{h})\end{bmatrix}\boldsymbol{\beta}+\begin{bmatrix}0\\ P|\boldsymbol{r}(\bar{h})|\end{bmatrix}=\begin{bmatrix}X(h)\\ X(\bar{h})\end{bmatrix}\boldsymbol{\beta}+\begin{bmatrix}0\\ \boldsymbol{r}(\bar{h})\end{bmatrix}.\tag{13}$$

The simplex algorithm moves from vertex to vertex by moving along the edge of the feasible region. Moving along this edge is the same as moving one of the elements from $\boldsymbol{r}(\mathscr{C})$ away from zero, while keeping the rest of these at zero, until a new vertex is reached, i.e. until one element from $\boldsymbol{r}(\mathscr{B})$ reaches zero.

Following Chvátal (1983, p. 103) and Nielsen (1999, p. 68) the inverse of B is needed to perform a simplex step. In the case of quantile regression this is found by examining Eq. (12), and noting that $P = P^{-1}$ (the diagonal elements are either 1 or -1), the result is

$$B^{-1} = \begin{bmatrix} X(h)^{-1} & 0 \\ -PX(\bar{h})X(h)^{-1} & P \end{bmatrix}. \tag{14}$$

From an implementation point of view this is very important, since $X(h)$ will be very small compared to B (at least for the problems considered here). So B is inverted by inverting a small matrix and using simple matrix multiplications. The first part of the simplex algorithm is to identify if there are directions with negative slopes. The slopes along the edges of the feasible region are given by

$$\boldsymbol{\Delta} = \boldsymbol{c}_{\mathscr{C}} - C^{\mathrm{T}}\boldsymbol{g}; \quad \boldsymbol{g} = B^{-\mathrm{T}}\boldsymbol{c}_{\mathscr{B}}. \tag{15}$$

$\boldsymbol{\Delta}$ is the marginal cost of moving a zero element from $\boldsymbol{z}_{\mathscr{C}}$ away from zero. Since the system is at a vertex these are the active constraints which should be examined. By straightforward matrix multiplication $\boldsymbol{\Delta}$ can be written as

$$\boldsymbol{\Delta} = \begin{bmatrix} \rho(\mathbf{1}_K) \\ \rho(-\mathbf{1}_K) \\ \rho(-\boldsymbol{p}) \end{bmatrix} - \begin{bmatrix} -X^{-\mathrm{T}}(h)X^{\mathrm{T}}P\rho(\boldsymbol{p}) \\ X^{-\mathrm{T}}(h)X^{\mathrm{T}}P\rho(\boldsymbol{p}) \\ -\rho(\boldsymbol{p}) \end{bmatrix} = \begin{bmatrix} \tau\mathbf{1}_K - \boldsymbol{g}(h) \\ (1-\tau)\mathbf{1}_K + \boldsymbol{g}(h) \\ \rho(-\boldsymbol{p}) + \rho(\boldsymbol{p}) \end{bmatrix}. \tag{16}$$

Since $\rho(\cdot)$ is greater than or equal to zero, this shows that the only possible descent directions are the first $2K$ directions. These correspond to moving an element of $\boldsymbol{r}(h)$ in either a positive or a negative direction. It is therefore sufficient to look at the first $2K$ elements of $\boldsymbol{\Delta}$, we denote these by $\boldsymbol{\Delta}_r$.

The negative elements in $\boldsymbol{\Delta}_r$ determine the directions in which the objective function is improved. This residual is now increased until one element in $\boldsymbol{z}_{\mathscr{B}}$ becomes zero. When this happens, an element from $\mathscr{B}$ swaps place with an element from $\mathscr{C}$, and the algorithm starts over. When a negative element, s, from $\boldsymbol{\Delta}_r$ has been chosen, the corresponding direction can be determined. This direction is denoted $\boldsymbol{d}$ and is given by

$$\boldsymbol{d} = B^{-1}C_{:,s}. \tag{17}$$

The columns of C are vectors with all elements except one are equal to zero, this element is equal to either plus or minus one. The first $2K$ columns in C, correspond to moving an element of $\boldsymbol{r}(h)$ in either a positive or negative direction. These are (as shown in Eq. (16)) the relevant ones in this context. These are equal to the sign of the new residual when we move in the direction determined by s. Thereby the direction $\boldsymbol{d}$ becomes one of the first K columns from B^{-1} times the sign of the new residual. The last step is to find the largest α such that

$$|r(\bar{h})| - \alpha\boldsymbol{d}(K+1:N) \geqslant 0. \tag{18}$$

The element q of $\boldsymbol{d}$ which limits α determines which element $\mathscr{B}(q)$ of $\mathscr{B}$ has to be swapped with $\mathscr{C}(s)$. In our implementation, $\boldsymbol{d}$ is calculated in each descent direction giving a series of directions $\boldsymbol{d}_{s_1}, \ldots, \boldsymbol{d}_{s_J}$, $(J \leqslant K)$. α is calculated for each of these descent directions. The direction that gives the largest improvement in the objective function is chosen, i.e

$$s = \arg\min_j(\alpha_{s_j}\boldsymbol{\Delta}_{s_j}). \tag{19}$$

This concludes one step of the simplex algorithm and the method is ready to start over. The solution is optimal when $\boldsymbol{\Delta}_r \geqslant \mathbf{0}$ at which point the algorithm terminates.

The standard simplex strategy for choosing the descent direction is to choose the steepest descent direction. The strategy presented here gives the largest improvement in the objective function. This is possible because the number of descent directions is small. The strategy for choosing α is the standard simplex strategy, i.e. moving to the next vertex. Barrodale and Roberts (1973) present a strategy where more vertices are crossed in one step. This is possible in the quantile regression formulation because the constraints are somewhat artificial.

It can be illustrative to consider an analytical development of the quantile regression problem. This is done in Appendix A, where the conclusions above are reached using directional derivatives.

3.1. Remarks on the implementation

The implementation of the simplex algorithm for quantile regression follows very closely the general algorithms given in Chvátal (1983, p. 103) and Nielsen (1999, p. 68). However, the special structure of the quantile regression problem as discussed above is exploited to enable efficient implementation. Koenker (2005, pp. 182–184) gives the dual formulation of the quantile regression problem. A comparison between the formulation given there, and the formulation suggested here show a close connection between the revised simplex formulation and the dual simplex formulation. A detailed description of the steps is given in Møller (2006) and Møller et al. (2006). The first reference concentrates on the mathematical development, while the second focuses on the implementation aspects. These references also comment on numerical issues that need to be addressed in the development of the procedure.

4. The time-adaptive method

In this section the simplex method outlined above is extended to enable iteration to the new solution when one observation is changed as described in the Introduction. It will be assumed that we have the optimal solution at each time point, and hence the simplex method iterates from the optimal solution at time t to the optimal solution at time $t+1$.

To emphasize the role of time we rewrite Eq. (5) as

$$\hat{\boldsymbol{\beta}}_t(\tau) = X_t(h_t)^{-1}\boldsymbol{y}_t(h_t). \tag{20}$$

In order to estimate h_{t+1}, h_t is used as a start guess and the simplex algorithm then iterates through vertices to the optimal solution.

The design matrix is updated by letting one observation leave the design matrix every time a new observation becomes available. A simple way of doing this is to let the oldest observation leave the design matrix. This and other algorithms for forgetting old observations have been tested. In general let l_{t+1} denote the index of the row in X_t, which is marked for deletion at time $t+1$. If the row that is marked for deletion is in h_t, then it cannot be deleted, since h_t is needed to perform the first simplex step. In this case one simplex step away from this solution is taken. This is essentially done by setting the cost of the corresponding residual equal to zero, i.e. setting $\rho(r_{l_{t+1}}) = 0$. This will ensure that the objective function is improved by taking this step.

The simple strategy for updating the design matrix X_t is

$$X_t = \begin{bmatrix} \boldsymbol{x}_t^{\mathrm{T}} \\ \vdots \\ \boldsymbol{x}_{t-n}^{\mathrm{T}} \end{bmatrix}; \quad \boldsymbol{y}_t = \begin{bmatrix} y_t \\ \vdots \\ y_{t-n} \end{bmatrix}, \tag{21}$$

where n is a predefined number. This strategy just leaves out the oldest observation as a new observation becomes available. We will describe a more general strategy in this paper.

The steps of the adaptive procedure are to decide which observation should leave the design matrix. This is then marked for deletion and the variables needed for the simplex algorithm are updated. Finally the simplex steps needed to go to the next optimal solution are taken. In summary the general procedure is

(1) Decide which row, $\boldsymbol{x}_{l_{t+1}}$, has to leave the design matrix.
(2) If $l_{t+1} \in h_t$ takes one simplex step s.t. $l_{t+1} \notin h_t$.
(3) Update $X, \boldsymbol{y}, P, h$ and $\boldsymbol{c}_{\mathscr{B}}$.
(4) Perform the simplex steps needed to get to the optimal solution at time $t+1$.

The individual steps are described in some detail below.

Step 1: Different procedures can be chosen in this step. The procedure presented here uses an explanatory variable (not necessarily in the model), associated with the observations to decide which observation has to leave the system.

Denote the explanatory variable for the forgetting structure by u_t and let $\{\mathscr{I}_1, \dots, \mathscr{I}_p\}$ be a partition of the sample space of u. Let the maximum number of allowed observations from each bin, $\mathscr{I}_j$, be $n_{\max}$. Let further $\mathscr{J}_{t_0} = \{1, 2, \dots, t_0\}$, where t_0 is such that

$$\max_j \left\{ \sum_{i=1}^{t_0} I(u_i \in \mathscr{I}_j) \right\} \leqslant n_{\max}. \tag{22}$$

All observations until time t are collected in the matrix $\tilde{X}_t$ and the vector $\tilde{\boldsymbol{y}}_t$, these are given by

$$\tilde{X}_t = \begin{bmatrix} \boldsymbol{x}_t^{\mathrm{T}} \\ \vdots \\ \boldsymbol{x}_1^{\mathrm{T}} \end{bmatrix}; \quad \tilde{\boldsymbol{y}}_t = \begin{bmatrix} y_t \\ \vdots \\ y_1 \end{bmatrix}. \tag{23}$$

The design matrices X_t, as time goes by, are drawn from this collection of observations ($X_t = \tilde{X}(\mathscr{J}_t)$). The variable marked for deletion in each bin is

$$l_{t+1,j} = \begin{cases} \min\{i \mid u_i \in \mathscr{I}_j, i \in \mathscr{J}_t\} & \text{if } \sum_{i \in \mathscr{J}_t} I(u_i \in \mathscr{I}_j) = n_{\max}, \\ \emptyset & \text{otherwise} \end{cases} \tag{24}$$

and the index of the leaving variable at time $t+1$ is

$$l_{t+1} = \{l_{t+1,j} | u_{t+1} \in \mathscr{I}_j\}. \tag{25}$$

This construction checks if the bin of the new observation is full. If this is the case, then the oldest observation from this bin is marked for deletion. Otherwise no observation is marked for deletion.

There are now two possibilities either $l_{t+1} \notin h_t$ or $l_{t+1} \in h_t$. If $l_{t+1} \notin h_t$ then the algorithm goes directly to Step 3. Otherwise the algorithm goes to Step 2 and changes h_t such that $l_{t+1} \notin h_t$.

Step 2: If the leaving variable is in h_t, then it cannot be removed, since the solution depends on the inverse of $\tilde{X}_t(h_t)$. Therefore if $l_{t+1} \in h_t$ one simplex step is performed with a new objective function, where the loss on $\boldsymbol{r}(l_{t+1})$ is set equal to zero. This corresponds to setting $\boldsymbol{c}_{\mathscr{C}}(\{l_t, l_t + K\}) = \boldsymbol{0}$. The result is that two elements of the simplex vector $\boldsymbol{\Delta}_r$ are changed. The elements we have to change are the elements corresponding to l_{t+1}. Denote these by $\boldsymbol{\Delta}_r^l$, we then have

$$\boldsymbol{\Delta}_r^l = \begin{bmatrix} -\boldsymbol{g}(l_{t+1}) \\ \boldsymbol{g}(l_{t+1}) \end{bmatrix}. \tag{26}$$

It is therefore only necessary to calculate the two elements of $\boldsymbol{\Delta}_r^l$ corresponding to l_{t+1}. This determines the descent direction and gives us s. If $\boldsymbol{g}(l_{t+1}) = 0$ then this will not be a descent direction. However, we can take one simplex step away from this; the direction is then not important. With this we can find $\boldsymbol{d}$ and thereby α and q, which was the variable that had to enter h_t. Having swapped l_{t+1} and q we are ready to update the design matrix and the vectors needed for the simplex algorithm.

Step 3: The index set that defines the design matrix is now updated by

$$\mathscr{J}_{t+1} = \{\mathscr{J}_t \backslash l_{t+1}, t+1\}. \tag{27}$$

The design matrix and responses are changed to

$$X_{t+1} = \tilde{X}_{t+1}(\mathscr{J}_{t+1}); \quad \boldsymbol{y}_{t+1} = \tilde{\boldsymbol{y}}_{t+1}(\mathscr{J}_{t+1}). \tag{28}$$

The residuals at time $t+1$ are

$$\boldsymbol{r}_{t+1} = \boldsymbol{y}_{t+1} - X_{t+1}\hat{\boldsymbol{\beta}}_t(\tau). \tag{29}$$

With this the variables needed for the simplex algorithm can be written as

$$\boldsymbol{p}_{t+1} = \operatorname{sign}(\boldsymbol{r}_{t+1}) \tag{30}$$

and

$$\boldsymbol{c}_{\mathscr{B}}^{(t+1)} = \rho_\tau(\boldsymbol{p}_{t+1}). \tag{31}$$

This concludes Step 3.

Step 4: With the updated variables the simplex steps needed to get to the optimal solution are now taken according to the algorithms given in Section 3.

Remarks. As is seen from the algorithm, a large part of effort in both the updating procedure and the simplex algorithm is to keep track of indices. In this presentation the design matrix is drawn from a matrix with all previous observations. This was done for notational convenience, but it does not mean that we have to store all previous observations. When an index has left $\mathscr{I}_t$, it cannot return. It is therefore only necessary to have access to the observations in $X_t = \tilde{X}_t(\mathscr{I}_t) \in \mathbb{R}^{(n_{\max}\cdot p)\times K}$, i.e. the storage requirement is $n_{\max}$ times the number of bins times the number of observations.

The full Matlab implementation is given in Møller et al. (2006), where the implementation is presented line by line. There is currently ongoing work to implement the algorithms in $C++$, this work seems very promising.

5. A case study

The example presented here uses data from a wind power plant located at Tunø Knob near Samsø in Denmark. The data is presented in detail in Nielsen et al. (2006) and Møller (2006).

The data set consists of forecasted power production (`pow.fc`) from WPPT (Wind Power Prediction Tool, see Madsen et al., 2005) and prediction error (`pow.pe`) from WPPT. In addition, meteorological forecasts at the site are available, and the forecasted power is based on some of these meteorological data. The power forecasts are based on meteorological forecasts given every morning at 06:00 UTC with a horizon of 18–36 h and a resolution of 15 min. The data set consists of data from the period January 1–October 31, 2003. The total number of data points is 21,753. The nominal power for the wind power plant is 5 MW.

In general, the operational setting for such wind power predictions is that data arrive every hour or every quarter-of-an-hour. The meteorological forecasts are given every morning or possibly every 6 h. WPPT then forecasts future wind power production based on these forecasts and with a horizon of 1–48 h. There is therefore an opportunity to update the quantile estimates every hour or every quarter-of-an-hour. In addition, the system will work on several wind power plants with the data arriving in the same way. The requirement for the algorithm would therefore be to update the result every hour or quarter-of-an-hour simultaneously for all the wind power plants being monitored and all required quantiles. We only give the 25% and 75% quantiles here, this, however, will not be sufficient to enable an optimal trading strategy.

The aim is now to use time-adaptive quantile regression to model quantiles of prediction errors from WPPT. Such quantiles are expected to be unknown non-linear functions of the forecasted power. Spline basis functions are therefore used to model the relationship.

The quantile models studied here are given by

$$Q(\tau;\texttt{pow.pe}) = \beta_{0,t}(\tau) + f_t(\texttt{pow.fc},\tau) = \beta_{0,t}(\tau) + \sum_{j=1}^{K-1} b_j(\texttt{pow.fc})\beta_{j,t}(\tau), \tag{32}$$

where `pow.pe` is the prediction error from WPPT, `pow.fc` is the forecasted power, τ is the required quantile. The b_j's are natural cubic B-spline basis function (see de Boor, 1978), with fixed level such that $f(0)=0$. This construction ensures uniqueness if more input variables are added in an additive model (see Nielsen et al., 2006; Hastie and Tibshirani, 1990). K is the number of knots (including boundary knots) used for the spline basis functions.

The model examined here has the knots for the spline basis functions placed in steps of 20% quantiles of forecasted power on the training set, which is the first 10,000 data points. The considered quantiles are the 25% and the 75% quantiles. This model is now examined with different updating strategies, i.e. the total number of elements in the design matrix is changed and the placement of the bins for the updating procedure is changed.

Four different adaptive models and one static model are used to illustrate the strength of the adaptive model. The models are

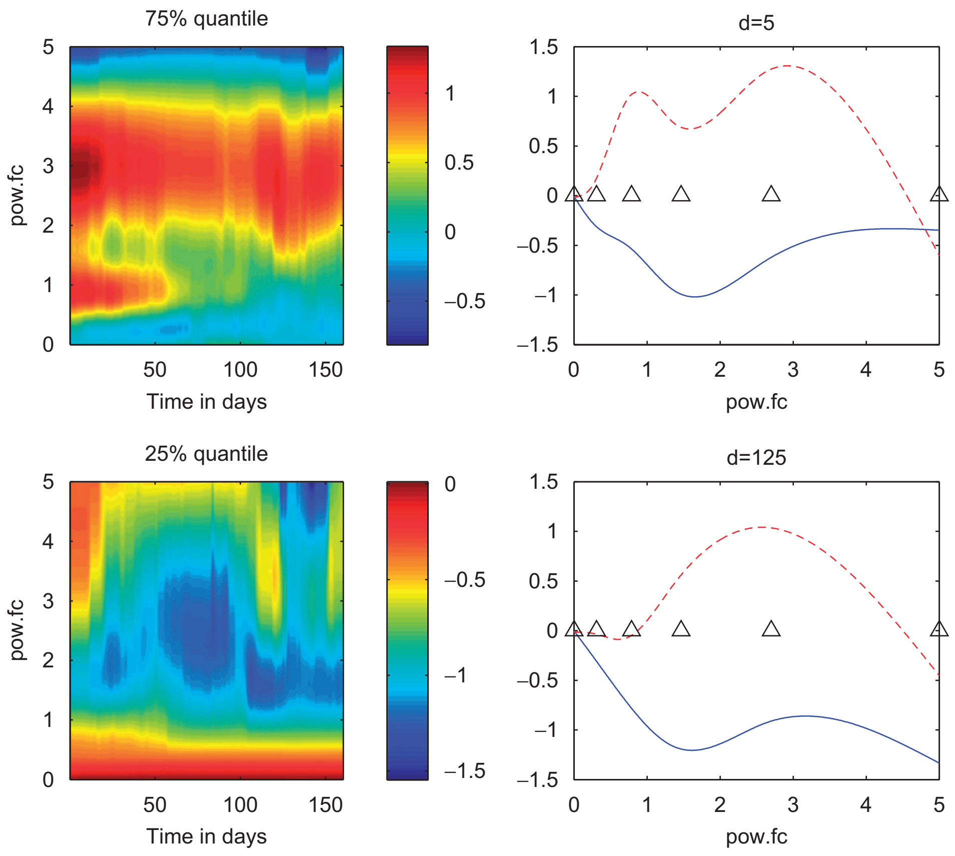

Fig. 1. The left panel of the figure shows the time evolution of model Adap 4 for the 1st and the 3rd quantiles, the right panel shows two cuts through the planes to help interpret the figure, the red line is the 75% quantile and the blue line is the 25% quantile. The triangles mark the placement of the knots for the splines. Units are in MW.

Reference: A static model with the first 10,000 data points used as training set, and the remaining part of the data set used as test set.

Adap 1: An adaptive model with a gliding window with 10,000 data points.

Adap 2: An adaptive model with a gliding window with 5000 data points.

Adap 3: An adaptive model with bins placed at the knots, i.e. the sequence of borders is $\{-\infty$, 308 kW, 789 kW, 1465.5 kW, 2701 kW, $\infty\}$. The number of elements allowed in each bin is 1200.

Adap 4: An adaptive model with the borders of the bins placed at $\{-\infty$, 800 kW, 1600 kW, 2400 kW, 3600 kW, $\infty\}$. The number of elements allowed in each bin is 1000.

Fig. 1 shows plots of model Adap 4 (on the test set). The left panel shows the time evolution for two quantiles, while the right panels show two cuts through these planes. The right panel also helps to interpret the two figures in the left panels. The figure shows that the model changes over time. This indicates that time-adaptive models should be used, at least with the explanatory variable chosen here.

The model is updated every 15 min, we will however only use the newest estimate available at 06:00 UTC to evaluate the quantiles. This corresponds to an operational setting for the presented data.

Quantiles and prediction intervals can be evaluated in many different ways and there is no single generally accepted protocol for such evaluations. Pinson et al. (2006, 2007a, b) go through some methods for evaluating quantiles and intervals. We will consider two measures of performance, namely reliability and skill score. Reliability measures the model's ability to split the observations as required by the nominal value of the quantile.

Obviously a good model should be able to split an observation set in the required proportions from a global perspective. This will be denoted as overall reliability and is addressed in Table 1. The model should also be able to split the

Table 1
The overall reliability for the time-adaptive models and the reference model

Model	Reference (%)	Adap 1 (%)	Adap 2 (%)	Adap 3 (%)	Adap 4(%)
Below **75**% (test)	83.9	78.6	**77.2**	77.6	**77.2**
Below **25**% (test)	22.6	22.9	25.7	25.1	**25.0**

The best performer in each row is marked in bold face.

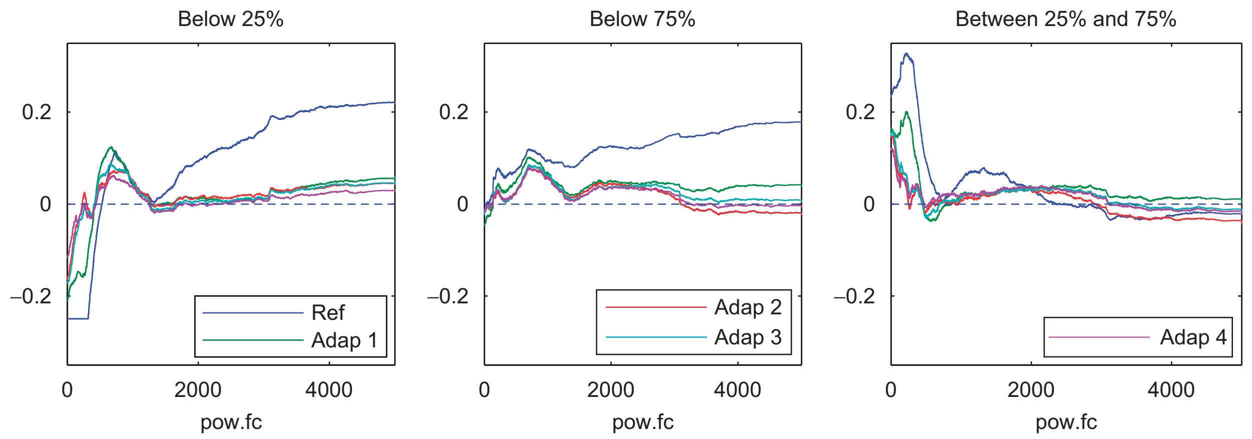

Fig. 2. The figure shows derivation of the local reliability from the required reliability for the models presented above. Units are in kW.

Table 2
The skill score for the time-adaptive models and the reference model

Model	Reference	Adap 1	Adap 2	Adap 3	Adap 4
$\overline{\rho}_{0.75}(\boldsymbol{r})$	260.3	**251.3**	251.8	251.8	251.4
$\overline{\rho}_{0.25}(\boldsymbol{r})$	209.6	201.3	199.5	199.6	**198.0**
$\overline{\rho}_{0.75}(\boldsymbol{r}) + \overline{\rho}_{0.25}(\boldsymbol{r})$	469.9	452.6	451.3	451.4	**449.4**

The best performer in each row is marked in bold face.

observations in the required proportions from a local perspective, i.e. the reliability of observation around some specific value of an explanatory variable. This issue is addressed in Fig. 2. In this paper we use a proportion of the data around each observation to calculate the local reliability, the precise definitions are given in Møller (2006, p. 68).

Sharpness and resolution of intervals are often used to evaluate intervals. These measures are intuitively appealing since they award narrow intervals and the ability to distinguish between situations with different uncertainties. However, there is no guarantee that they will award the best model and therefore they are not considered here.

Gneiting and Raftery (2005) present a skill score which should award reliability, sharpness and resolution and collect this in one number. This is defined for individual quantiles and for symmetric (around 50%) intervals. This skill score can be shown (see Pinson et al., 2006) to be equivalent to looking at the loss function for the quantile regression. The skill score for one quantile is the loss function (defined in Eq. (2)) for this quantile, and the skill score for an interval is the sum of the loss functions forming the interval. The loss functions are of course also relevant in their own right when we compare different quantile regression models. The average loss function on the test set is given in Table 2.

The overall reliability for the models presented above is given in Table 1. The table shows improvements in the overall reliability when adaptive models are used. There are clear improvements in the adaptive models compared to the static model. It can also be seen that the updating strategy is important for the result. The local reliability is presented in Fig. 2, the improvements in this picture are very clear. The static model is completely off at the 25% quantile for low and high forecasted power. For example, for low forecasted power there are no observations below the model.

Table 3
Mean cpu time (s) used per update of the solution as new observations become available

Model	Adap 1	Adap 2	Adap 3	Adap 4
`br`	0.71/0.33	0.24/0.13	0.30/0.18	0.19/0.16
`fn`	0.60/0.60	0.29/0.28	0.35/0.34	0.28/0.26
adap	0.15/0.08	0.09/0.06	0.06/0.06	0.07/0.07

The timings are given for the 25% quantile and the 75% quantile separated by "/".

The time-adaptive models perform much better, even though there is room for improvement in the low regime. For the 75% quantile the main improvements are for the high forecasted power, where the static model produce a empirical quantile of about 95%, while the adaptive models are only a few percent off.

Table 2 gives the values of the skill score for the models described above. It can be seen that the time-adaptive models are quite close and that all of these are better than the reference model. This confirms the results from Fig. 2 and Table 1. Comparing these loss functions is equivalent to comparing the mean square error for different mean value regression functions.

For time-adaptive models the cpu time used by the algorithm is also a performance parameter. The implementation of the time-adaptive procedure is done in Matlab. Of course the time-adaptive procedure should use less time than a procedure where everything is re-estimated at time $t+1$ without using knowledge about the solution at time t. The models mentioned above are timed and the results are shown in Table 3. For reference, the timing from the `rq`-methods `br` and `fn` in "R" (version 2.4.0) are given with the same updating strategies, but without using start guesses. The time-adaptive procedure is between two and five times faster than implementation with a naive approach, where information about the solution at time t is not used to estimate the solution at time $t+1$.

The results on timing are not directly comparable. It should, however, be stressed that the time-adaptive quantile regression is not optimized with respect to time and it is implemented directly in Matlab, while the quantile regression methods in "R" have been optimized and the main part of the algorithm is implemented in Fortran. This means that in order to do a complete comparison between the algorithms we would need to implement the optimized algorithms in Fortran and create an interface from "R" to these algorithms. This is out of the scope for this article. We would, however, expect the potential improvements to be larger than the factor of 2–5 presented here.

6. Discussion and conclusions

This paper describes a new method for time-adaptive quantile regression. The quantile regression problem is a linear optimization problem, and a formulation which is well suited for a direct solution using the simplex method is presented. Based on this formulation a time-adaptive method for quantile regression is suggested. The strength of the suggested method is illustrated by considering the problem of online prediction of wind power production. The linear optimization formulation presented is well known, but related to the formulation it is shown how the structure of the quantile regression problem can be exploited in the simplex algorithm. This observation leads to an efficient algorithm. Koenker (2005) presents a dual simplex formulation of the quantile regression problem, while we use a direct simplex formulation.

Using the proposed linear optimization formulation, an algorithm for time-adaptive quantile regression is suggested. The time-adaptive algorithm is presented for a broad class of forgetting structures. This structure depends on both the time and an explanatory variable, and in this way updating will depend on the rate of data inflow in different regions of the sample space of the explanatory variable. The best updating procedure depends heavily on the structure of the sample space, how to choose such a strategy is discussed in greater detail in Møller (2006).

To illustrate the strength of the methods developed, a real-life example is considered in the later part of the article. This study shows large improvements in the performance parameters, by using the time-adaptive quantile regression methods instead of static quantile models. Finally, the analysis also shows that the Matlab implementation of the method is significantly faster than naive re-estimation using a well-proven optimized code.

The method "`br`" implemented in "R" is a modified version (see Koenker and D'Orey, 1987) of the Barrodale and Roberts algorithm for l_1 regression (see Barrodale and Roberts, 1973). This is a simplex based algorithm, and therefore it could be expected that adopting the ideas presented in by Barrodale and Roberts (1973), could lead to improvements in the time-adaptive algorithm presented in this article.

There are two interior point methods available in "R", namely "`fn`" the Frisch–Newton interior point method and "`pfn`" the Frisch–Newton interior point method with preprocessing. These methods should be used for large problems, and as was seen in Table 3, the modified Barrodale and Roberts algorithm performs better than the Frisch–Newton algorithm for (most of) the problems considered. This is not expected to be the case for problems larger than those presented in this article. It is however not possible to use "warm start" algorithms for interior point methods. We would therefore expect our method to perform very well, compared to the Frisch–Newton interior point algorithm, for larger problems (window sizes) than those examined in this paper. The preprocessed Frisch–Newton algorithm is well suited for very large problems. In Portnoy and Koenker (1997) the algorithms are only presented with sample sizes larger than 20,000, while we used only 5000–10,000 in our case study. Preprocessing also has the drawback that it needs independent observations. This cannot be expected in the data we have presented here, and preprocessing is therefore not considered in this paper.

It is our expectation that the presented algorithm has the potential for very efficient implementation, both by optimizing the presented algorithm and by exploring the possibility of the "warm start" idea in the Barrodale and Roberts algorithm. This will facilitate online estimation of multiple quantiles on multiple sites simultaneously, even with fast real time flow of data through the system. Our case study is wind power prediction, but we expect that the method can be used in other areas of applications.

Appendix A. An analytic development

The development in Section 3 follows the linear programming formulation and the language developed for this. The close connection to slopes of the loss function was indicated, but a direct derivation of the slopes would be illustrative. To this end define the directional derivative of a vector function $f:\mathbb{R}^K\to\mathbb{R}$ in direction $\boldsymbol{w}$ as

$$\nabla f(\boldsymbol{x},\boldsymbol{w})=\frac{\mathrm{d}}{\mathrm{d}t}f(\boldsymbol{x}+t\boldsymbol{w})|_{t\to 0_+}. \tag{A.1}$$

Note that the sum of loss functions $S_\tau:\mathbb{R}^K\to\mathbb{R}$ is not differentiable at any point where a residual is zero and these are the points evaluated during the simplex algorithm. Therefore, we write down the directional derivative of S_τ:

$$\nabla S_\tau(\boldsymbol{\beta},\boldsymbol{w})=\frac{\mathrm{d}}{\mathrm{d}t}\sum_i\rho_\tau\{y_i-\boldsymbol{x}_i^{\mathrm{T}}(\beta-t\boldsymbol{w})\}|_{t\to 0_+} \tag{A.2}$$

$$=\frac{\mathrm{d}}{\mathrm{d}t}\sum_i\rho_\tau(r_i+\boldsymbol{x}_i^{\mathrm{T}}t\boldsymbol{w})|_{t\to 0_+} \tag{A.3}$$

$$=\left.\left(\frac{\mathrm{d}}{\mathrm{d}t}\sum_{r_i=0}\{\tau-I(\boldsymbol{x}_i^{\mathrm{T}}t\boldsymbol{w}<0)\}\boldsymbol{x}_i^{\mathrm{T}}t\boldsymbol{w}\right)\right|_{t\to 0_+}+\sum_{r_i<0}(\tau-1)\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w}+\sum_{r_i>0}\tau\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w} \tag{A.4}$$

$$=\sum_{r_i=0}\{\tau-I(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w}<0)\}\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w}+\sum_{r_i\neq 0}\operatorname{sign}(r_i)\rho_\tau\{\operatorname{sign}(r_i)\}\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w}. \tag{A.5}$$

If the observations are in what Koenker (2005, p. 35) calls a general position, then splitting the index set into h and $\bar{h}$, and the splitting in Eq. (A.5) are the same. If we are dealing with continuous random variables then the observations are in a general position with probability one. If the system is in a general position, then Eq. (A.5) can be written as

$$\nabla S_\tau(\boldsymbol{\beta},\boldsymbol{w})=\sum_{i\in h}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i\{\tau-I(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{w}<0)\}+\sum_{i\in\bar{h}}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i r_i\rho\{\operatorname{sign}(r_i)\} \tag{A.6}$$

$$=\boldsymbol{w}^{\mathrm{T}}X(h)^{\mathrm{T}}\{\tau\mathbf{1}-I(X(h)\boldsymbol{w}<0)\}+\boldsymbol{w}^{\mathrm{T}}X(\bar{h})^{\mathrm{T}}P\rho(\boldsymbol{p}). \tag{A.7}$$

We now identify directions where the directional derivative is less than zero. Because of the linearity of the problem the only directions to check are directions such that only one zero residual is changed away from zero; there are two directions for each element in h. These are $\boldsymbol{w} = \delta X(h)^{-1}\boldsymbol{e}_j$, $j = 1, \ldots, K$ and $\delta = \pm 1$. $\boldsymbol{e}_j$ is a vector with the jth element equal to one and all other elements equal to zero, and δ will be the sign of the new residual. Examining these $2K$ directions gives each of the elements of the vector $\boldsymbol{\Delta}_r$ presented in Eq. (16).

When a direction $\boldsymbol{w} = \delta X(h)^{-1}\boldsymbol{e}_j$ in terms of $\boldsymbol{\beta}$ has been chosen, we can find the direction in terms of $\boldsymbol{z}_{\mathscr{B}}$. Recall that $|\boldsymbol{r}(\bar{h})| = P\boldsymbol{r}(\bar{h})$, what we are now looking for is the vector $\Delta(\boldsymbol{w})$, such that when α is small enough the result of a change in $\boldsymbol{\beta}$ by $\alpha\boldsymbol{w}$, is that $\boldsymbol{z}_{\mathscr{B}}$ is changed to $\boldsymbol{z}_{\mathscr{B}} + \alpha\Delta(\boldsymbol{w})$. The directions examined are again $\boldsymbol{w} = \delta X(h)^{-1}\boldsymbol{e}_j$, and the result is

$$\Delta(\boldsymbol{w}) = \begin{bmatrix} \boldsymbol{w} \\ PX(\bar{h})\boldsymbol{w} \end{bmatrix} = \begin{bmatrix} \delta X(h)^{-1}_{:,j} \\ -PX(\bar{h})\delta X(h)^{-1}_{:,j} \end{bmatrix} = \delta B^{-1}_{:,j} \tag{A.8}$$

which is equal to $\boldsymbol{d}$ when a direction has been chosen. α and q are now updated as in Eq. (18).

Appendix B. Supplementary data

Supplementary data associated with this article can be found in the online version at 10.1016/j.csda.2007.06.027.

References

Barrodale, I., Roberts, F.D.K., 1973. An improved algorithm for discrete linear approximation. SIAM J. Numer. Anal. 10 (5), 839–848.

Chambers, J.M., James, D.A., Lambert, D., Wiel, S.V., 2006. Monotoring networked applications with incremental quantile estimation. Statist. Sci. 21, 463–475.

Chvátal, V., 1983. Linear Programming. W.H. Freeman and Company, New York.

de Boor, C., 1978. A practical Guide to Splines. Applied Mathematical Sciences, vol. 27. Springer, Berlin.

Gneiting, T., Raftery, A.E., 2005. Strictly proper scoring rules, prediction, and estimation. Technical Report No. 463, Department of Statistics, University of Washington, September 2005. URL:⟨http://www.stat.washington.edu/www/research/reports/2004/tr463R.pdf⟩.

Hastie, T., Tibshirani, R., 1990. Generalized Additive Models. Chapman & Hall, London.

Koenker, R., 2005. Quantile Regression. Cambridge University Press, Cambridge.

Koenker, R., Bassett Jr., G., 1978. Regression quantile. Econometrica 46 (1), 33–50.

Koenker, R., D'Orey, V., 1987. Computing regression quantile. Appl. Statist. 36 (3), 383–393.

Ljung, L., Söderström, T., 1983. Theory and Practice of Recursive Identification. Series in Signal Processing, Optimization, and Control, vol. 4. MIT Press, Cambridge, MA.

Madsen, H. (Ed.), 1996. Models and Methods for Predicting Wind Power. IMM/ELSAM, ISBN 87-87090-29-5, 88p.

Madsen, H., Nielsen, Aa.H., Nielsen, S.T., 2005. A tool for predicting the wind power production of off-shore wind plants. In: Proceedings of the Copenhagen Offshore Wind Conference & Exhibition, Copenhagen, October 2005. Danish Wind Industry Association. URL:⟨http://www.windpower.org/en/core.htm⟩.

Møller, J.K., 2006. Modeling of uncertainty in wind energy forecast. Master Thesis Informatics and Mathematical Modelling, Technical University of Denmark. URL:⟨http://www.imm.dtu.dk/pubdb/p.php?4428⟩.

Møller, J.K., Nielsen, H.Aa., Madsen, H., 2006. Algorithms for an adaptive quantile regression method. Technical Report 2006-08, Informatics and Mathematical Modelling, Technical University of Denmark. URL:⟨http://www2.imm.dtu.dk/pubdb/p.php?4727⟩.

Nielsen, H.B., 1999. Algorithms for linear optimization, an introduction. Course note for the DTU course. Optimization and Data Fitting, vol. 2. URL:⟨http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id = 654⟩.

Nielsen, H.Aa., Madsen, H., Nielsen, T.S., 2006. Using quantile regression to an existing wind power forecasting system with probabilistic forecast. Wind Energy 9, 95–108.

Pinson, P., Kariniotakis, G., Nielsen, H.Aa., Nielsen, T.S., Madsen, H., 2006. Properties of interval and quantile forecasts of wind generation and their evaluation. In: Proceedings of the European Wind Energy Conference & Exhibition, Athens, March 2006. EWEA. ⟨http://www.ewea.org⟩.

Pinson, P., Moller, J.K., Nielsen, H.Aa., Madsen, H., Kariniotakis, G.N., 2007a. Evaluation of nonparametric probalistic forecast of wind power. Technical Report 2007-02, Informatics and Mathematical Modelling, Technical University of Denmark. URL:⟨http://www2.imm.dtu.dk/pubdb/p.php?5024⟩.

Pinson, P., Nielsen, H.Aa., Møller, J.K., Madsen, H., Kariniotakis, G.N., 2007b. Nonparametric probabilistic forecasts of wind power: required properties and evaluation. Wind Energy, (in press).

Pinson, P., Chevallier, C., Kariniotakis, G.N., 2007c. Trading wind generation from short-term probabilistic forecast of wind power. IEEE Trans. on Power Systems, 22 (3).

Portnoy, S., Koenker, R., 1997. The Gaussian hare and the Laplacian tortoise: computability of squared-error versus absolute-error estimators. Statist. Sci. 12 (4), 279–296.