Data Mining and Knowledge Discovery (2023) 37:710–735
https://doi.org/10.1007/s10618-022-00914-4

# Residual projection for quantile regression in vertically partitioned big data

**Ye Fan**[1] · **Jr-Shin Li**[2] · **Nan Lin**[3]

Received: 17 December 2021 / Accepted: 16 December 2022 / Published online: 17 January 2023


## Abstract

Standard regression techniques model only the mean of the response variable. Quantile regression (QR) is more powerful in that it depicts a comprehensive relationship between the response variable and independent covariates at different quantiles. It is particularly useful for non-normally distributed data with skewness or heterogeneity, which appear routinely in many scientific fields, such as economics, finance, public health and biology. Although its theory has been well developed in the literature, its computation in big data still faces multiple challenges, especially for vertically stored big data in modern distributed environments, where communication efficiency and security are usually the primary considerations. While the popular alternating direction method of multipliers (ADMM) provides a general computational solution, its slow convergence becomes a bottleneck when communication cost dominates local computational consumption, such as Internet of Things (IoT) networks. Motivated by the residual projection technique, in this paper we propose an innovative iterative parallel framework, PIQR, that converges faster and has a more secure data transmission plan, and establish its convergence property. This framework is further extended to composite quantile regression (CQR), a modified QR technique that improves estimation efficiency at extreme quantiles. Simulation studies show that both the ADMM-based method and the PIQR enjoy favorable estimation accuracy in distributed environments. While PIQR is inferior to the ADMM-based method at local computation, it requires much fewer iterations to achieve convergence, and hence significantly improves the overall computational efficiency when communication cost is the dominating factor.

Responsible editor: Aristides Gionis

✉ Nan Lin
nlin@wustl.edu

[1] School of Statistics, Capital University of Economics and Business, Beijing 100070, China

[2] Department of Electrical and Systems Engineering, Washington University in St. Louis, St. Louis, MO 63130, USA

[3] Department of Mathematics and Statistics, Washington University in St. Louis, St. Louis, MO 63130, USA

Moreover, PIQR transmits only data involving the residual information between different machines, and can better prevent the leakage of important data information compared with the ADMM-based method.

**Keywords** ADMM · Parallel framework · Privacy preservation · Quantile regression · Residual projection · Vertically distributed big data

## 1 Introduction

Due to the rapid development of data collection technology, modern big data are massive in volume and also high-dimensional. Horizontally distributed storage and vertically distributed storage are two popular techniques for tackling the storage problem of big data that aim at different application scenarios (Gamal and Lai 2015; Trofimov and Genkin 2015, 2017; Huang and Huo 2019). In horizontally distributed settings, big data are split by samples and distributively stored across multiple local machines in a horizontal way. On the other hand, vertically distributed storage requires that the samples are partitioned by features. For example, in a prediction study for product purchase, important features may include the user's income and expenditure behavior, credit rating, and browsing and purchasing history. These features are often obtained from different venues and hence distributively and vertically stored at different departments, such as a bank or an e-commerce company. For distributively stored big data, parallel analysis technique processes the data blocks stored on local machines in a parallel way and implements the aggregation of low-dimensional local summary statistics on a central machine (see Fig. 1). While extensive research has been done on the parallel analysis of horizontally stored distributed big data, e.g., Xi et al. (2008), Lin and Xi (2011), Zhang et al. (2013a, b), Chen and Xie (2014), Lee et al. (2017) and Jordan et al. (2019), far less attention was given to vertically stored distributed data, i.e., data split by features. It is worth noting that communication efficiency and privacy preservation are more prominent factors for analyzing vertically stored distributed data than horizontally stored distributed data due to the constraint on data transmission rate in modern networks and potential competition between different data owners. In this work, we consider the popular quantile regression (QR) and composite quantile regression (CQR) estimation problems, and the objective is to develop secure parallel analysis methods with low communication cost to support their computation in different real application scenarios.

QR technique, first proposed in Koenker and Bassett (1978), provides a powerful tool for analyzing non-normally distributed data, especially for those with skewness or heterogeneity frequently appearing in economics, finance, public health and biology (Yu et al. 2003; Peng and Huang 2008; Allen et al. 2009; Fitzenberger et al. 2013; Sherwood et al. 2013; Briollais and Durrieu 2014). Compared to the conventional mean regression, QR only imposes some mild distributional assumptions and gives a more comprehensive portrayal for the relationship between the response and the covariates through modeling the conditional quantiles of the response at different levels. While the mean regression problem is usually solved by gradient-based methods to optimize a smooth objective function, e.g., ordinary least square, the QR estimation can not

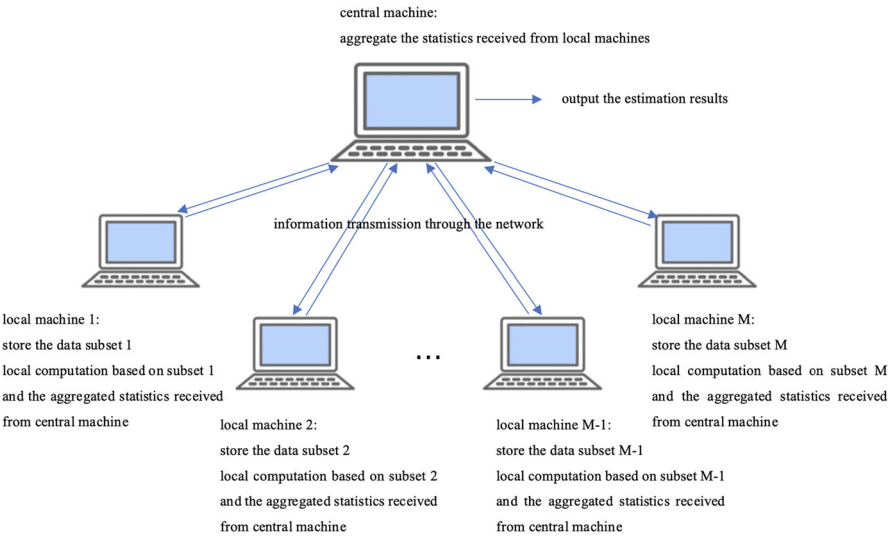


**Fig. 1** The parallel analysis for distributed big data

due to minimizing a non-smooth check loss function. In small or moderate-size data, the QR estimation problem is traditionally solved by linear programming, such as the interior point (IP) method (Portnoy and Koenker 1997). For a data set of size $n$ and dimension $p$, IP has complexity $\mathcal{O}\left(n^{1+\alpha}p^3\log n\right)$ $(\alpha \in (0, 1/2))$ (He et al. 2021) due to full factorization of a weighted design matrix in every iteration of the algorithm, and is thus computationally too expensive in big data (Chen and Wei 2005), especially under high-dimensional settings. Besides, IP is a centralized algorithm and may be infeasible for distributed big data due to the expensive communication cost or the privacy constraint in transmitting local raw data to a central machine.

As a variant of the standard QR, CQR (Zou and Yuan 2008; Gu and Zou 2020) considers simultaneous modeling at a sequence of quantile levels by assuming identical slope parameters but not the intercept, and can remedy the low estimation efficiency of standard QR at extreme quantiles or when the error distribution is extremely heavy-tailed, e.g., the Cauchy distribution. Similar to QR, the traditional solver for CQR estimation is also based on the IP method (Koenker 2017). However, its computation is even more time-consuming because CQR involves simultaneous estimation at multiple quantile levels.

### 1.1 Related works

QR estimation in big data has recently drawn significant research interest. Major strategies to tackle its expensive computation and communication problems include smoothing, subsampling, divide-and-conquer and the alternating direction method of multipliers (ADMM) (Boyd et al. 2011) based parallel/distributed analysis. The smoothing technique reduces the computational complexity of QR estimation through

converting the original check loss into a smoothed function by utilizing kernel functions. For example, see He et al. (2021), Pan et al. (2022), Jiang and Yu (2022) and Tan et al. (2022). The subsampling procedure lessens the computational and communication burdens of QR by projecting the entire big data to a lower dimensional space or selecting a relatively informative subset to analyze. See Yang et al. (2014), Ai et al. (2021) and Wang and Ma (2021). The divide-and-conquer technique processes the big data block-by-block in a distributed way and then conducts an aggregation to obtain an approximation to the global estimator. Methods following this paradigm include Wang and Li (2017), Chen et al. (2019), Volgushev et al. (2019), Chen and Zhou (2020) and Chen et al. (2020). The ADMM-based parallel/distributed analysis approach implements the parallelization of the computation through decomposing the original estimation problem into a series of iterative local subproblems and can obtain an exact solution to the original problem. See Yu et al. (2017) and Hu et al. (2021) for its application to QR estimation. However, all these researches focus on non-distributed environments or horizontally distributed scenarios.

For vertically stored distributed data, Yu and Lin (2017) pointed out that ADMM can be used to implement the parallel computation of QR estimation, but they did not study this issue in-depth, especially the convergence speed of this implementation. In this paper, we derive the algorithmic details of the ADMM-based parallel method for QR estimation in vertically stored distributed big data. However, our numerical experiment results show that this method often requires hundreds or thousands of iterations to converge even when the number of local machines is small or moderate. Another concern is that the ADMM-based parallel solution needs to transmit both the intermediate gradients and covariate information between the central and local machines, which is more prone to leakage of important data information (Yang et al. 2019). This then leaves it an important and urgent task to develop communication-efficient and secure parallel algorithms for QR estimation in distributed environments with expensive communication cost and/or privacy constraint. For example, Internet of Things (IoT) networks usually have low data throughput and transmission rate, and aim for stable and long-term private data transmission between wearable electronics, wireless sensors or handhold devices (Zou et al. 2019; Shi et al. 2020; Ye et al. 2021).

Compared to the standard QR, research on CQR estimation in big data is still relatively rare. Recently, Pietrosanu et al. (2021) systematically analyzed the computation problem of CQR, and developed the majorization-minimization (MM) (Hunter and Lange 2000), coordinate descent (CD) (Wu and Lange 2008) and ADMM-based algorithms. However, none of them apply to parallel computation for vertically stored distributed big data.

### 1.2 Contribution

- To tackle the problems mentioned in Sect. 1.1 for QR estimation in vertically distributed big data, in this paper we propose an innovative iterative parallel framework, PIQR, based on the idea of residual projection (Chen et al. 1989; Miao et al. 2020), and systematically compare it with the ADMM-based parallel method in

estimation accuracy, convergence speed and computational efficiency to provide practical guidance.

  - Simulation studies show that both methods enjoy favorable estimation accuracy in vertically distributed environments.
  - Although PIQR is inferior to the ADMM-based parallel method at local computation, it often requires much fewer iterations to converge, particularly when the number of local machines is not too large. This makes PIQR especially attractive for the distributed environments with expensive communication cost.
  - During the whole estimation process, PIQR only transmits data involving the residual information between the central and local machines, which provides an effective protection for the privacy of the raw data.

- We also extend the PIQR framework to CQR. To our knowledge, computation of the CQR estimator in vertically partitioned distributed big data remains an open issue prior to our method proposed in this paper.

### 1.3 Organization

The rest of this paper is organized as follows. In Sect. 2, we develop the algorithmic details of the ADMM-based parallel method for QR estimation in vertically stored distributed big data. In Sect. 3, we propose the iterative parallel framework, PIQR. In Sect. 4, we further extend PIQR to the CQR model. In Sect. 5, we analyze the convergence property of PIQR. Section 6 contains simulation studies that evaluate the performance of the ADMM-based parallel method and the PIQR for QR and CQR. Section 7 gives the concluding remarks.

## 2 The ADMM-based parallel method for distributed QR

In this section, we give the algorithmic details of the ADMM-based parallel method for QR estimation in vertically stored distributed big data mentioned in Yu and Lin (2017).

QR assumes that the $\tau$th $(0 < \tau < 1)$ conditional quantile of the response $y \in \mathbb{R}$ is a function of the covariate vector $\boldsymbol{x} = \left(x_1, x_2, \cdots, x_p\right)^{\mathrm{T}} \in \mathbb{R}^p$. For example, a linear QR model is given as

$$Q_\tau\left(y \mid \boldsymbol{x}\right) = \beta_{0\tau} + \boldsymbol{x}^{\mathrm{T}}\boldsymbol{\beta}_\tau, \tag{1}$$

where $Q_\tau\left(y \mid \boldsymbol{x}\right)$ represents the $\tau$th conditional quantile of $y$ given $\boldsymbol{x}$, $\beta_{0\tau} \in \mathbb{R}$ is the unknown intercept, and $\boldsymbol{\beta}_\tau = \left(\beta_{1\tau}, \beta_{2\tau}, \cdots, \beta_{p\tau}\right)^{\mathrm{T}} \in \mathbb{R}^p$ is the vector of unknown slope parameters. In this work, we focus on analyzing linear QR (1), which will provide a general guidance for solving other variants in the QR family. For example, nonparametric QR (Koenker 2005; Takeuchi et al. 2006) assumes $Q_\tau\left(y \mid \boldsymbol{x}\right)$ as a nonparametric function of $\boldsymbol{x}$, which may be represented through splines. As the spline method is essentially a linear approach, the method developed in this section as well as those in the following Sects. 3 and 4 are expected to extend naturally, though the

theory needs to be further developed due to errors introduced in the spline approximation. This will be our future research topic. Given a sample with response values $\boldsymbol{y} \in (y_1, y_2, \cdots, y_n)^{\mathrm{T}} \in \mathbb{R}^n$ and covariates $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)^{\mathrm{T}} \in \mathbb{R}^{n\times p}$, the regression parameters $\left(\beta_{0\tau}, \boldsymbol{\beta}_\tau\right)$ in (1) at a quantile level $\tau$ is estimated by

$$\left(\hat{\beta}_{0\tau}, \hat{\boldsymbol{\beta}}_\tau\right) = \underset{\beta_0\in\mathbb{R}, \boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \sum_{i=1}^{n} \rho_\tau\left(y_i - \beta_0 - \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}\right), \tag{2}$$

where $\rho_\tau(u) = [\tau - \mathbb{I}(u < 0)]\, u$ $(u \in \mathbb{R})$ is the check loss function and $\mathbb{I}(\cdot)$ is the indicator function. For simplicity, we denote the terms $\left(\beta_0, \boldsymbol{\beta}^{\mathrm{T}}\right)^{\mathrm{T}}$, $\left(\hat{\beta}_{0\tau}, \hat{\boldsymbol{\beta}}_\tau^{\mathrm{T}}\right)^{\mathrm{T}}$ and $\sum_{i=1}^{n} \rho_\tau\left(y_i - \beta_0 - \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}\right)$ in (2) as $\boldsymbol{\beta}_{\mathrm{QR}} \in \mathbb{R}^{p+1}$, $\hat{\boldsymbol{\beta}}_{\mathrm{QR}} \in \mathbb{R}^{p+1}$ and $\rho_\tau\left(\boldsymbol{y} - X^*\boldsymbol{\beta}_{\mathrm{QR}}\right)$, respectively. Here $X^* = [\mathbf{1}_n, X] \in \mathbb{R}^{n\times(p+1)}$ and $\mathbf{1}_n$ represents the $n$-dimensional vector whose elements are all 1.

We assume that the design matrix $X$ is vertically and distributively stored at $M$ local machines with $X_m \in \mathbb{R}^{n\times p_m}$ being the sub-matrix stored at the $m$th $(m = 1, 2, \cdots, M)$ local machine. Obviously, we have $\sum_{m=1}^{M} p_m = p$. Under this setup, the QR estimation problem (2) can be written as

$$\hat{\boldsymbol{\beta}}_{\mathrm{QR}} = \underset{\{\boldsymbol{\beta}_{\mathrm{QR}m}\}_M}{\arg\min}\ \rho_\tau\left(\boldsymbol{y} - \sum_{m=1}^{M} X_m^*\boldsymbol{\beta}_{\mathrm{QR}m}\right), \tag{3}$$

where $X_1^* = [\mathbf{1}_n, X_1] \in \mathbb{R}^{n\times(p_1+1)}$, $X_m^* = X_m \in \mathbb{R}^{n\times p_m}$ $(m = 2, 3, \cdots, M)$, $\left(\boldsymbol{\beta}_{\mathrm{QR}1}^{\mathrm{T}}, \boldsymbol{\beta}_{\mathrm{QR}2}^{\mathrm{T}}, \cdots, \boldsymbol{\beta}_{\mathrm{QR}M}^{\mathrm{T}}\right)^{\mathrm{T}}$ is a partition of $\boldsymbol{\beta}_{\mathrm{QR}}$ with $\boldsymbol{\beta}_{\mathrm{QR}1} \in \mathbb{R}^{p_1+1}$ and $\boldsymbol{\beta}_{\mathrm{QR}m} \in \mathbb{R}^{p_m}$ $(m = 2, 3, \cdots, M)$, and $\left\{\boldsymbol{\beta}_{\mathrm{QR}m}\right\}_M$ represents the set $\left\{\boldsymbol{\beta}_{\mathrm{QR}m} \mid m = 1, 2, \cdots, M\right\}$. Without loss of generality, we will hereinafter use $\boldsymbol{a}_m$ to represent the $m$th $(m = 1, 2, \ldots, M)$ component in the partition of a vector $\boldsymbol{a}$ unless special remark and no longer elaborate its meaning in the following but only specify its dimension to avoid ambiguity. To formulate (3) into the general ADMM form, Yu and Lin (2017) proposed to introduce auxiliary variables $\boldsymbol{z}_{\mathrm{QR}m} = X_m^*\boldsymbol{\beta}_{\mathrm{QR}m} \in \mathbb{R}^n$ $(m = 1, 2, \cdots, M)$ and rewrite this problem as

$$\begin{aligned} &\min_{\{\boldsymbol{\beta}_{\mathrm{QR}m}\}_M, \{\boldsymbol{z}_{\mathrm{QR}m}\}_M} \rho_\tau\left(\boldsymbol{y} - \sum_{m=1}^{M} \boldsymbol{z}_{\mathrm{QR}m}\right), \\ &\text{s.t.} \qquad X_m^*\boldsymbol{\beta}_{\mathrm{QR}m} = \boldsymbol{z}_{\mathrm{QR}m},\ m = 1, 2, \cdots, M. \end{aligned} \tag{4}$$

The augmented Lagrangian function corresponding to (4) is

$$\mathcal{L}_\eta\left(\boldsymbol{\beta}_{\mathrm{QR}m}, z_{\mathrm{QR}m}, \boldsymbol{u}_{\mathrm{QR}m}\right) = \rho_\tau\left(\boldsymbol{y} - \sum_{m=1}^{M} z_{\mathrm{QR}m}\right) + \sum_{m=1}^{M} \boldsymbol{u}_{\mathrm{QR}m}^{\mathrm{T}}\left(X_m^* \boldsymbol{\beta}_{\mathrm{QR}m} - z_{\mathrm{QR}m}\right) + \frac{\eta}{2}\sum_{m=1}^{M}\left\|X_m^* \boldsymbol{\beta}_{\mathrm{QR}m} - z_{\mathrm{QR}m}\right\|_2^2,$$

where $\boldsymbol{u}_{\mathrm{QR}m} \in \mathbb{R}^n$ $(m = 1, 2, \cdots, M)$ are the dual variables and $\eta > 0$ is the augmentation parameter. Then the problem (4) can be solved by the ADMM with the $(k+1)$th iteration being

$$\boldsymbol{\beta}_{\mathrm{QR}m}^{(k+1)} = \arg\min_{\boldsymbol{\beta}_{\mathrm{QR}m}} \frac{\eta}{2}\left\|X_m^* \boldsymbol{\beta}_{\mathrm{QR}m} - z_{\mathrm{QR}m}^{(k)} + \boldsymbol{u}_{\mathrm{QR}m}^{(k)}/\eta\right\|_2^2, \tag{5}$$

$$z_{\mathrm{QR}}^{(k+1)} = \arg\min_{\{z_{\mathrm{QR}m}\}_M} \rho_\tau\left(\boldsymbol{y} - \sum_{m=1}^{M} z_{\mathrm{QR}m}\right) + \frac{\eta}{2}\sum_{m=1}^{M}\left\|X_m^* \boldsymbol{\beta}_{\mathrm{QR}m}^{(k+1)} - z_{\mathrm{QR}m} + \boldsymbol{u}_{\mathrm{QR}m}^{(k)}/\eta\right\|_2^2, \tag{6}$$

$$\boldsymbol{u}_{\mathrm{QR}m}^{(k+1)} = \boldsymbol{u}_{\mathrm{QR}m}^{(k)} + \eta\left(X_m^* \boldsymbol{\beta}_{\mathrm{QR}m}^{(k+1)} - z_{\mathrm{QR}m}^{(k+1)}\right), \tag{7}$$

where $z_{\mathrm{QR}} = \left(z_{\mathrm{QR}1}^{\mathrm{T}}, z_{\mathrm{QR}2}^{\mathrm{T}}, \cdots, z_{\mathrm{QR}M}^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{(Mn)}$. Note that the $z_{\mathrm{QR}}$-update (6) is an $(Mn)$-dimensional optimization problem and requires iterative numerical algorithms for the computation. Thus, it is usually inflexible in big data. To simplify this computation, Yu and Lin (2017) further introduced an auxiliary variable $\overline{z}_{\mathrm{QR}} = M^{-1}\sum_{m=1}^{M} z_{\mathrm{QR}m}$ following the proposal in Boyd et al. (2011) and reformulated the subproblem (6) as

$$\min_{\{z_{\mathrm{QR}m}\}_M, \overline{z}_{\mathrm{QR}}} \rho_\tau\left(\boldsymbol{y} - M\overline{z}_{\mathrm{QR}}\right) + \frac{\eta}{2}\sum_{m=1}^{M}\left\|X_m^* \boldsymbol{\beta}_{\mathrm{QR}m}^{(k+1)} - z_{\mathrm{QR}m} + \boldsymbol{u}_{\mathrm{QR}m}^{(k)}/\eta\right\|_2^2,$$
$$\text{s.t.} \quad \overline{z}_{\mathrm{QR}} = M^{-1}\sum_{m=1}^{M} z_{\mathrm{QR}m}.$$

Optimizing this problem over $z_{\mathrm{QR}m}$'s with $\overline{z}_{\mathrm{QR}}$ fixed has the solution,

$$z_{\mathrm{QR}m} = X_m^* \boldsymbol{\beta}_{\mathrm{QR}m}^{(k+1)} + \boldsymbol{u}_{\mathrm{QR}m}^{(k)}/\eta + \overline{z}_{\mathrm{QR}} - \overline{X^*\boldsymbol{\beta}_{\mathrm{QR}}}^{(k+1)} - \overline{\boldsymbol{u}}_{\mathrm{QR}}^{(k)}/\eta, \quad m = 1, 2, \ldots, M, \tag{8}$$

where $\overline{X^*\boldsymbol{\beta}_{\mathrm{QR}}}^{(k+1)} = M^{-1}\sum_{m=1}^{M} X_m^* \boldsymbol{\beta}_{\mathrm{QR}m}^{(k+1)}$ and $\overline{\boldsymbol{u}}_{\mathrm{QR}}^{(k)} = M^{-1}\sum_{m=1}^{M} \boldsymbol{u}_{\mathrm{QR}m}^{(k)}$. Therefore, the $z_{\mathrm{QR}}$-update (6) can be converted into the following unconstrained

optimization problem,

$$\overline{z}_{\mathrm{QR}}^{(k+1)} = \underset{\overline{z}_{\mathrm{QR}}}{\arg\min}\ \rho_\tau\left(\boldsymbol{y} - M\overline{z}_{\mathrm{QR}}\right) + \frac{\eta}{2}\sum_{m=1}^{M}\left\|\overline{z}_{\mathrm{QR}} - \overline{X^*\boldsymbol{\beta}_{\mathrm{QR}}}^{(k+1)} - \overline{\boldsymbol{u}}_{\mathrm{QR}}^{(k)}/\eta\right\|_2^2.$$

Substitute the Eq. (8) into the $\boldsymbol{\beta}_{\mathrm{QR}m}$-update (5) and the $\boldsymbol{u}_{\mathrm{QR}m}$-update (7), and we obtain that the final ADMM updates for QR estimation in vertically stored distributed data are as follows.

$$\boldsymbol{\beta}_{\mathrm{QR}m}^{(k+1)} = \underset{\boldsymbol{\beta}_{\mathrm{QR}m}}{\arg\min}\ \frac{\eta}{2}\left\|X_m^*\boldsymbol{\beta}_{\mathrm{QR}m} - X_m^*\boldsymbol{\beta}_{\mathrm{QR}m}^{(k)} - \overline{z}_{\mathrm{QR}}^{(k)} + \overline{X^*\boldsymbol{\beta}_{\mathrm{QR}}}^{(k)} + \overline{\boldsymbol{u}}_{\mathrm{QR}}^{(k)}/\eta\right\|_2^2, \quad (9)$$

$$\overline{z}_{\mathrm{QR}}^{(k+1)} = \underset{\overline{z}_{\mathrm{QR}}}{\arg\min}\ \rho_\tau\left(\boldsymbol{y} - M\overline{z}_{\mathrm{QR}}\right) + \frac{\eta}{2}\sum_{m=1}^{M}\left\|\overline{z}_{\mathrm{QR}} - \overline{X^*\boldsymbol{\beta}_{\mathrm{QR}}}^{(k+1)} - \overline{\boldsymbol{u}}_{\mathrm{QR}}^{(k)}/\eta\right\|_2^2, \quad (10)$$

$$\overline{\boldsymbol{u}}_{\mathrm{QR}}^{(k+1)} = \overline{\boldsymbol{u}}_{\mathrm{QR}}^{(k)} + \eta\left(\overline{X^*\boldsymbol{\beta}_{\mathrm{QR}}}^{(k+1)} - \overline{z}_{\mathrm{QR}}^{(k+1)}\right). \quad (11)$$

Both the reformulated subproblems (9) and (10) have closed-form solutions. They respectively are

$$\boldsymbol{\beta}_{\mathrm{QR}m}^{(k+1)} = \boldsymbol{\beta}_{\mathrm{QR}m}^{(k)} + \left(X_m^{*\mathrm{T}}X_m^*\right)^{-1}\left[X_m^{*\mathrm{T}}\left(\overline{z}_{\mathrm{QR}}^{(k)} - \overline{X^*\boldsymbol{\beta}_{\mathrm{QR}}}^{(k)} - \overline{\boldsymbol{u}}_{\mathrm{QR}}^{(k)}/\eta\right)\right],$$

and

$$\begin{aligned}\overline{z}_{\mathrm{QR}}^{(k+1)} = \boldsymbol{y}/M &- \left[\boldsymbol{y}/M - \overline{X^*\boldsymbol{\beta}_{\mathrm{QR}}}^{(k+1)} - \overline{\boldsymbol{u}}_{\mathrm{QR}}^{(k)}/\eta - \tau\mathbf{1}_n/\eta\right]_+ \\ &+ \left[-\boldsymbol{y}/M + \overline{X^*\boldsymbol{\beta}_{\mathrm{QR}}}^{(k+1)} + \overline{\boldsymbol{u}}_{\mathrm{QR}}^{(k)}/\eta + (\tau-1)\,\mathbf{1}_n/\eta\right]_+,\end{aligned}$$

where $[\cdot]_+$ represents the positive part of a vector.

From (9)–(11), we see that all the $\boldsymbol{\beta}_{\mathrm{QR}m}$-update ($m = 1, 2, \cdots, M$) in the ADMM-based method are accomplished on local machines in parallel, and the $\overline{z}_{\mathrm{QR}}$-update and $\overline{\boldsymbol{u}}_{\mathrm{QR}}$-update are conducted on a central machine. During the computation, only the $n$-dimensional vectors, $\overline{z}_{\mathrm{QR}}^{(k+1)}$, $\overline{\boldsymbol{u}}_{\mathrm{QR}}^{(k+1)}$, $\overline{X^*\boldsymbol{\beta}_{\mathrm{QR}}}^{(k+1)}$ and $X_m^*\boldsymbol{\beta}_{\mathrm{QR}m}^{(k+1)}$'s, need to be transmitted between the central and local machines. However, as pointed out in Yang et al. (2019), data transmission involving both the intermediate gradients and covariate information is prone to leakage of important raw data information. More importantly, our numerical experiment results (presented in Sect. 6) show that even for small or moderate partition number, the ADMM-based parallel method needs hundreds or thousands of iterations to converge. This makes this method inflexible in distributed environments with expensive communication cost, especially when communication cost dominates local computational consumption, such as computing operations in IoT networks. To enable secure and communication-efficient QR estimation in these cases, we further propose an innovative iterative parallel framework, PIQR, and will give its details in the next section.

## 3 The PIQR framework for distributed QR

In this section, we elaborate the developed iterative parallel framework, PIQR, for QR estimation in vertically stored distributed big data.

The basic idea behind our PIQR is inspired by the parallel residual projection (PRP) method proposed in Miao et al. (2020) for solving large-scale linear inverse problems (LIPs). PRP treats a LIP as a projection operation of the response $\boldsymbol{y}$ onto the column space of the design matrix $X$ and implements it through iteratively projecting the LIP residuals onto this column space in parallel. Although the QR estimation problem (2) is not a LIP, motivated by the general idea of PRP, we may view (2) as an information extraction operation on $\boldsymbol{y}$ and solve it by iteratively performing a sequence of quantile regressions, each of which extracts the useful information remained in the residual from the previous iteration. This leads to the following iterative algorithm with the $(k+1)$th iteration given as

$$\widehat{\Delta\boldsymbol{\beta}}_{\mathrm{QR}}^{(k+1)} = \underset{\Delta\boldsymbol{\beta}_{\mathrm{QR}}\in\mathbb{R}^{p+1}}{\arg\min}\ \rho_\tau\left(\boldsymbol{e}^{(k)} - X^*\Delta\boldsymbol{\beta}_{\mathrm{QR}}\right), \tag{12}$$

and

$$\hat{\boldsymbol{\beta}}_{\mathrm{QR}}^{(k+1)} = \hat{\boldsymbol{\beta}}_{\mathrm{QR}}^{(k)} + \widehat{\Delta\boldsymbol{\beta}}_{\mathrm{QR}}^{(k+1)}, \tag{13}$$

where $\widehat{\Delta\boldsymbol{\beta}}_{\mathrm{QR}}^{(k+1)}$ is the increment for $\hat{\boldsymbol{\beta}}_{\mathrm{QR}}^{(k)}$ in the $(k+1)$th iteration, i.e., the remaining useful information for regression parameter extracted from the residual $\boldsymbol{e}^{(k)}$, and $\boldsymbol{e}^{(k)} = \boldsymbol{y} - X^*\hat{\boldsymbol{\beta}}_{\mathrm{QR}}^{(k)}$ is available from the previous iteration. Note that following this iterative optimization procedure, the check loss in QR will be reduced in every iteration, i.e.,

$$\rho_\tau\left(\boldsymbol{y} - X^*\hat{\boldsymbol{\beta}}_{\mathrm{QR}}^{(k+1)}\right) \le \rho_\tau\left(\boldsymbol{y} - X^*\hat{\boldsymbol{\beta}}_{\mathrm{QR}}^{(k)}\right),\ k = 0, 1, \cdots.$$

This naturally follows from the fact that $\widehat{\Delta\boldsymbol{\beta}}_{\mathrm{QR}}^{(k+1)}$ minimizes the objective function in (12).

While parallelization is rather straightforward in the PRP framework due to additivity in the linear objective function, the disassembly of (12) is more challenging as the additivity no longer holds under the check loss framework. That is, in general $\rho_\tau\left(\sum_{m=1}^M v_m\right) \neq \sum_{m=1}^M \rho_\tau\left(v_m\right)$ for a sequence of constants $v_m$'s. However, we note that $\sum_{m=1}^M \rho_\tau\left(v_m\right)$ is actually an upper bound of $\rho_\tau\left(\sum_{m=1}^M v_m\right)$ (see Lemma 1 in Section 1 in the Supplementary Information). Applying this to the iterative QR problem (12), we get

$$\begin{aligned}\rho_\tau\left(\boldsymbol{e}^{(k)} - X^*\Delta\boldsymbol{\beta}_{\mathrm{QR}}\right) &= \rho_\tau\left(\sum_{m=1}^M\left(w_m\boldsymbol{e}^{(k)} - X_m^*\Delta\boldsymbol{\beta}_{\mathrm{QR}m}\right)\right)\\ &\le \sum_{m=1}^M \rho_\tau\left(w_m\boldsymbol{e}^{(k)} - X_m^*\Delta\boldsymbol{\beta}_{\mathrm{QR}m}\right),\end{aligned} \tag{14}$$

where $\Delta\boldsymbol{\beta}_{\mathrm{QR}1} \in \mathbb{R}^{p_1+1}$ and $\Delta\boldsymbol{\beta}_{\mathrm{QR}m} \in \mathbb{R}^{p_m}$ $(m = 2, 3, \cdots, M)$. Besides, in (14), $w_m$ is the weight assigned to the local residuals for the $m$th machine and satisfy $\sum_{m=1}^{M} w_m = 1$. In vertically distributed computing, weights are usually introduced to indicate the relevance of attributes on different local machines to the response (Zheng et al. 2010). In our simulation studies in Sect. 6, as all predictors are basically randomly partitioned onto different local machines, it is natural to use the equal weights $w_1 = w_2 = \cdots = w_M = \frac{1}{M}$. And we will systematically and rigorously study the influence of the selection of $w_m$'s on the final estimation accuracy of the developed iterative parallel method in our future work. Intuitively, in the $(k+1)$th iteration, we may obtain a nearly optimal estimate for $\Delta\boldsymbol{\beta}_{\mathrm{QR}}$ by minimizing the upper bound in (14) instead of the original objective in (12). Therefore, we propose to iteratively optimize the following surrogate in computing the QR estimator $\hat{\boldsymbol{\beta}}_{\mathrm{QR}}$,

$$\widehat{\Delta\boldsymbol{\beta}}_{\mathrm{QR}}^{(k+1)} = \underset{\{\Delta\boldsymbol{\beta}_{\mathrm{QR}m}\}_M}{\arg\min} \sum_{m=1}^{M} \rho_\tau \left( w_m \boldsymbol{e}^{(k)} - X_m^* \Delta\boldsymbol{\beta}_{\mathrm{QR}m} \right). \tag{15}$$

The problem (15) can then be solved in parallel after being decomposed into $M$ low-dimensional subproblems, i.e.,

$$\widehat{\Delta\boldsymbol{\beta}}_{\mathrm{QR}m}^{(k+1)} = \underset{\Delta\boldsymbol{\beta}_{\mathrm{QR}m}}{\arg\min}\, \rho_\tau \left( w_m \boldsymbol{e}^{(k)} - X_m^* \Delta\boldsymbol{\beta}_{\mathrm{QR}m} \right), \ m = 1, 2, \cdots, M. \tag{16}$$

Our idea of searching a nearly optimal solution through optimizing an upper bound of the original objective is inspired by the MM technique (Lange et al. 2000; Hunter and Lange 2000), which uses surrogate majorizing functions to perform minimization. It is worth pointing out that although Hunter and Lange (2000) proposed an MM algorithm for QR, their goal was to expedite the computation rather than supporting parallel estimation. They constructed the majorizing function using a smooth quadratic function tangent to the original check loss, but the algorithm is still a centralized one and does not apply in our context.

As each problem (16) is iteratively solved by any QR solver, e.g., IP, ADMM, MM and CD, we name this general **i**terative **p**arallel estimation framework for **QR** as PIQR and give its details in Algorithm 1 and Fig. 2 In every iteration of PIQR, after receiving the local residuals, i.e., $\boldsymbol{e}_m^{(k+1)}$'s in Algorithm 1, the central machine calculates the global residuals $\boldsymbol{e}^{(k+1)}$ and then broadcasts this vector back to local machines. All the rest of the computations are done in parallel on local machines, involving estimating $\widehat{\Delta\boldsymbol{\beta}}_{\mathrm{QR}m}^{(k+1)}$'s and updating $\hat{\boldsymbol{\beta}}_{\mathrm{QR}m}^{(k+1)}$'s and $\boldsymbol{e}_m^{(k+1)}$'s. That is, only the $n$-dimensional global residuals $\boldsymbol{e}^{(k+1)}$ and local residuals $\boldsymbol{e}_m^{(k+1)}$'s need to be transmitted between the central and local machines. This effectively reduces the risk of information leakage of the raw data, and thus PIQR is especially suitable for analyzing highly confidential distributed data, in which security is usually much more important than the computational speed and estimation accuracy of the algorithm. In Sect. 5, we prove that the PIQR generates a sequence of estimates that decreases the check loss and converges to the optimal QR estimator. Simulation results in Sect. 6 verify our convergence theory and also show

that for a small or moderate partition number $M$, PIQR requires much fewer iterations to converge than the ADMM-based parallel method, which implies that PIQR may also be preferable for distributed environments with expensive communication cost.

**Algorithm 1** The PIQR framework for QR.

**Input:** the vertically and distributively stored design matrices $X_m$'s ($m = 1, 2, \cdots, M$), the response $\boldsymbol{y}$, the quantile level $\tau$, the initial estimate $\hat{\boldsymbol{\beta}}_{\mathrm{QR}}^{(0)}$ for $\boldsymbol{\beta}_{\mathrm{QR}}$, the preassigned weights $w_m$'s, the maximum number of iterations $\max_{iter}$ and the permission error $\varepsilon$.

1: Set $\boldsymbol{e}^{(0)} = \boldsymbol{y} - \sum_{m=1}^{M} X_m^* \hat{\boldsymbol{\beta}}_{\mathrm{QR}m}^{(0)}$, $d = +\infty$ and $k = 0$, where $X_1^* = [\mathbf{1}_n, X_1]$ and $X_m^* = X_m$ $(m = 2, 3, \cdots, M)$.
2: **while** $d > \varepsilon$ and $k < \max_{iter}$ **do**
3: **for** $m = 1, 2, \cdots, M$ **do**
4: Optimize $\widehat{\Delta\boldsymbol{\beta}}_{\mathrm{QR}m}^{(k+1)} = \underset{\Delta\boldsymbol{\beta}_{\mathrm{QR}m}}{\arg\min}\, \rho_\tau\left(w_m \boldsymbol{e}^{(k)} - X_m^* \Delta\boldsymbol{\beta}_{\mathrm{QR}m}\right)$.
5: Update $\hat{\boldsymbol{\beta}}_{\mathrm{QR}m}^{(k+1)} = \hat{\boldsymbol{\beta}}_{\mathrm{QR}m}^{(k)} + \widehat{\Delta\boldsymbol{\beta}}_{\mathrm{QR}m}^{(k+1)}$.
6: Calculate $\boldsymbol{e}_m^{(k+1)} = w_m \boldsymbol{e}^{(k)} - X_m^* \widehat{\Delta\boldsymbol{\beta}}_{\mathrm{QR}m}^{(k+1)}$ and $d_m = \|\widehat{\Delta\boldsymbol{\beta}}_{\mathrm{QR}m}^{(k+1)}\|_1$.
7: **end for**
8: Calculate $\boldsymbol{e}^{(k+1)} = \sum_{m=1}^{M} \boldsymbol{e}_m^{(k+1)}$ and $d = \sum_{m=1}^{M} d_m$.
9: $k = k + 1$.
10: **end while**
11: **return** the final estimate $\hat{\boldsymbol{\beta}}_{\mathrm{QR}} = \left(\left(\hat{\boldsymbol{\beta}}_{\mathrm{QR}1}^{(k)}\right)^{\mathrm{T}}, \left(\hat{\boldsymbol{\beta}}_{\mathrm{QR}2}^{(k)}\right)^{\mathrm{T}}, \cdots, \left(\hat{\boldsymbol{\beta}}_{\mathrm{QR}M}^{(k)}\right)^{\mathrm{T}}\right)^{\mathrm{T}}$.

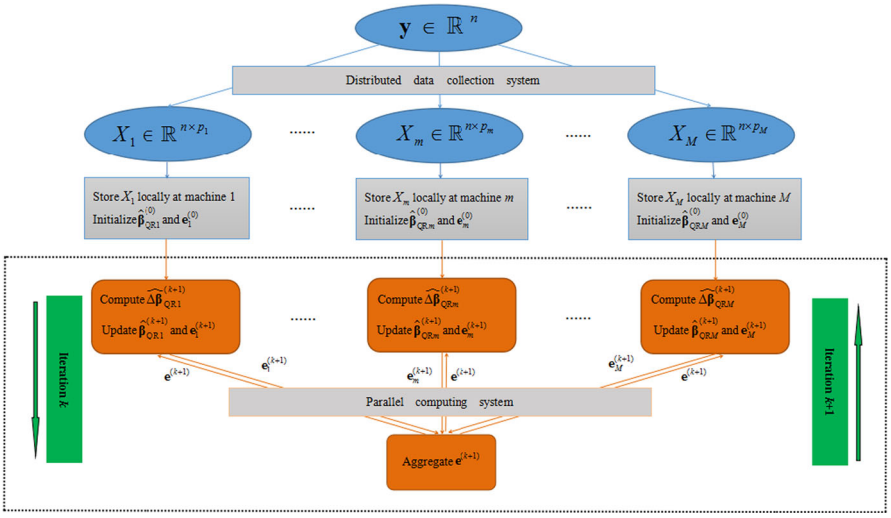


**Fig. 2** The iterative parallel PIQR framework

## 4 The extension of PIQR to distributed CQR

In this section, we extend the PIQR framework to CQR model, which considers simultaneous modeling at many quantile levels $0 < \tau_1 < \tau_2 < \cdots < \tau_S < 1$. To improve estimation efficiency, CQR assumes identical slope parameter $\boldsymbol{\beta} = \left(\beta_1, \beta_2, \cdots, \beta_p\right)^{\mathrm{T}} \in \mathbb{R}^p$ at different quantile levels $\tau_s$'s but allows the intercept $\beta_{0\tau_s}$ $(s = 1, 2, \cdots, S)$ to vary. Parameters in CQR are estimated by solving

$$\left(\hat{\beta}_{0\tau_1}, \hat{\beta}_{0\tau_2}, \cdots, \hat{\beta}_{0\tau_S}, \hat{\boldsymbol{\beta}}\right) = \underset{\{\beta_{0\tau_s}\}_S, \boldsymbol{\beta}}{\arg\min} \sum_{s=1}^{S} \sum_{i=1}^{n} \rho_{\tau_s}\left(y_i - \beta_{0\tau_s} - \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}\right). \tag{17}$$

We denote the terms $\sum_{s=1}^{S}\sum_{i=1}^{n}\rho_{\tau_s}\left(y_i - \beta_{0\tau_s} - \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}\right)$ and $\left(\beta_{0\tau_1}, \beta_{0\tau_2}, \cdots, \beta_{0\tau_S}\right)^{\mathrm{T}}$ as $\sum_{s=1}^{S}\rho_{\tau_s}\left(\boldsymbol{y} - \mathbf{1}_n\beta_{0\tau_s} - X\boldsymbol{\beta}\right)$ and $\boldsymbol{\beta}_{0S} \in \mathbb{R}^S$ and further let $\boldsymbol{\beta}_{\mathrm{CQR}} = \left(\boldsymbol{\beta}_{0S}^{\mathrm{T}}, \boldsymbol{\beta}^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{(S+p)}$ to simplify notations.

Note that the intercept in a linear model corresponds to vertically shifting the response values and has no impact on slope parameter estimation. Therefore, one may estimate the slope parameter without knowing the intercept (Pietrosanu et al. 2021). For the ease of parallelization, our extended PIQR framework for CQR will first estimate the slope parameter $\boldsymbol{\beta}$ in its iterative process and in the end obtain the intercept estimates on the central machine directly based on the sample quantile of the residuals. First, we can find the slope estimate using the PIQR idea but with a slightly different objective function, i.e.,

$$\left(\hat{\boldsymbol{\beta}}_{0S}^{(k+1)}, \widehat{\Delta\boldsymbol{\beta}}^{(k+1)}\right) = \underset{\boldsymbol{\beta}_{0S}, \Delta\boldsymbol{\beta}}{\arg\min} \sum_{s=1}^{S} \rho_{\tau_s}\left(\boldsymbol{e}^{(k)} - \mathbf{1}_n\beta_{0\tau_s} - X\Delta\boldsymbol{\beta}\right), \tag{18}$$

and

$$\hat{\boldsymbol{\beta}}^{(k+1)} = \hat{\boldsymbol{\beta}}^{(k)} + \widehat{\Delta\boldsymbol{\beta}}^{(k+1)}, \tag{19}$$

where $\widehat{\Delta\boldsymbol{\beta}}^{(k+1)} \in \mathbb{R}^p$ is the increment for $\hat{\boldsymbol{\beta}}^{(k)}$ in the $(k+1)$th iteration, or more concretely, the remaining useful information for the slope parameter $\boldsymbol{\beta}$ extracted from the residual $\boldsymbol{e}^{(k)} = \boldsymbol{y} - X\hat{\boldsymbol{\beta}}^{(k)}$. Note that the intercept estimate $\hat{\boldsymbol{\beta}}_{0S}^{(k+1)}$ in (18) will simply be discarded. Even though one may opt to updating the intercept estimates together with the slope parameter, our empirical results show that such intercept estimates are inaccurate, likely due to the sensitivity to extreme values at extreme quantile levels. After the final slope estimate $\hat{\boldsymbol{\beta}}$ is obtained through iteratively computing (18) and updating (19), in the end we obtain the intercept estimate as the sample quantile of the residuals, i.e., $\hat{\beta}_{0\tau_s} = Q_{\tau_s}^*\left(\boldsymbol{y} - X\hat{\boldsymbol{\beta}}\right)$ $(s = 1, 2, \cdots, S)$, where $Q_{\tau_s}^*(\cdot) \in \mathbb{R}$ represents the $\tau_s$th sample quantile.

To solve (18) in parallel, we introduce auxiliary variables $\beta_{0\tau_s m} = w_m\beta_{0\tau_s}$ $(s = 1, 2, \ldots, S, m = 1, 2, \cdots, M)$, in order to ensure that the decomposed low-dimensional subproblems still possess the standard form of CQR estimation. Then

similarly as (16), by employing the inequality

$$\sum_{s=1}^{S}\rho_{\tau_s}\left(\boldsymbol{e}^{(k)}-\mathbf{1}_n\beta_{0\tau_s}-X\Delta\boldsymbol{\beta}\right)\le\sum_{m=1}^{M}\sum_{s=1}^{S}\rho_{\tau_s}\left(w_m\boldsymbol{e}^{(k)}-\mathbf{1}_n\beta_{0\tau_s m}-X_m\Delta\boldsymbol{\beta}_m\right),$$

we propose to estimate the variable $\Delta\boldsymbol{\beta}$ in (18) through optimizing the following low-dimensional subproblems,

$$\left(\hat{\boldsymbol{\beta}}_{0Sm}^{(k+1)},\widehat{\Delta\boldsymbol{\beta}}_m^{(k+1)}\right)=\underset{\boldsymbol{\beta}_{0Sm},\Delta\boldsymbol{\beta}_m}{\arg\min}\sum_{s=1}^{S}\rho_{\tau_s}\left(w_m\boldsymbol{e}^{(k)}-\mathbf{1}_n\beta_{0\tau_s m}-X_m\Delta\boldsymbol{\beta}_m\right),\quad m=1,2,\cdots,M,\tag{20}$$

where $\boldsymbol{\beta}_{0Sm}=\left(\beta_{0\tau_1 m},\beta_{0\tau_2 m},\cdots,\beta_{0\tau_S m}\right)^{\mathrm{T}}\in\mathbb{R}^S$ and $\Delta\boldsymbol{\beta}_m\in\mathbb{R}^{p_m}$ $(m=1,2,\cdots,M)$. Correspondingly, we set

$$\hat{\boldsymbol{\beta}}_m^{(k+1)}=\hat{\boldsymbol{\beta}}_m^{(k)}+\widehat{\Delta\boldsymbol{\beta}}_m^{(k+1)},\ m=1,2,\cdots,M,\tag{21}$$

$$\boldsymbol{e}^{(k+1)}=\boldsymbol{y}-\sum_{m=1}^{M}X_m\hat{\boldsymbol{\beta}}_m^{(k+1)}=\sum_{m=1}^{M}\left(w_m\boldsymbol{e}^{(k)}-X_m\widehat{\Delta\boldsymbol{\beta}}_m^{(k+1)}\right),\tag{22}$$

and the intercept estimate is

$$\hat{\beta}_{0\tau_s}=Q_{\tau_s}^{*}\left(\boldsymbol{y}-\sum_{m=1}^{M}X_m\hat{\boldsymbol{\beta}}_m\right),\ s=1,2,\cdots,S,$$

where $\hat{\boldsymbol{\beta}}_m^{(k)},\hat{\boldsymbol{\beta}}_m\in\mathbb{R}^{p_m}$ $(m=1,2,\cdots,M)$ and $\left(\hat{\boldsymbol{\beta}}_1^{\mathrm{T}},\hat{\boldsymbol{\beta}}_2^{\mathrm{T}},\cdots,\hat{\boldsymbol{\beta}}_M^{\mathrm{T}}\right)^{\mathrm{T}}$ is the final estimate for the slope parameter $\boldsymbol{\beta}$ obtained by (20)–(22). Algorithm 2 summarizes this extended PIQR framework for CQR. Actually, we also extend the ADMM-based parallel method developed in Sect. 2 to CQR model for comparison. Please see Section 4 in the Supplementary Information for details.

## 5 The convergence theory of PIQR

In this section, we show that the PIQR enjoys the favorable descent property for both QR and CQR and establish its convergence property under these two models.

We first assume a mild boundedness condition on the design matrix $X$, which is commonly used for regression problems, for example, see (Zou and Yuan 2008; Wu and Liu 2009; Wang et al. 2012). For big data, this condition is satisfied if the values in $X$ are generated from distributions with finite variances.

**Algorithm 2** The PIQR framework for CQR.

**Input:** the vertically and distributively stored design matrices $X_m$'s ($m = 1, 2, \cdots, M$), the response $\boldsymbol{y}$, the quantile levels $\tau_s$'s ($s = 1, 2, \cdots, S$), the initial estimate $\hat{\boldsymbol{\beta}}^{(0)}$ for $\boldsymbol{\beta}$, the preassigned weights $w_m$'s, the maximum number of iterations $\max_{iter}$ and the permission error $\varepsilon$.

1: Set $\boldsymbol{e}^{(0)} = \boldsymbol{y} - \sum_{m=1}^{M} X_m \hat{\boldsymbol{\beta}}_m^{(0)}$, $d = +\infty$ and $k = 0$.
2: **while** $d > \varepsilon$ and $k < \max_{iter}$ **do**
3: **for** $m = 1, 2, \cdots, M$ **do**
4: Optimize

$$\left(\hat{\boldsymbol{\beta}}_{0Sm}^{(k+1)}, \widehat{\Delta\boldsymbol{\beta}}_m^{(k+1)}\right) = \underset{\boldsymbol{\beta}_{0Sm}, \Delta\boldsymbol{\beta}_m}{\arg\min} \sum_{s=1}^{S} \rho_{\tau_s}\left(w_m \boldsymbol{e}^{(k)} - \mathbf{1}_n \beta_{0\tau_s m} - X_m \Delta\boldsymbol{\beta}_m\right).$$

5: Update $\hat{\boldsymbol{\beta}}_m^{(k+1)} = \hat{\boldsymbol{\beta}}_m^{(k)} + \widehat{\Delta\boldsymbol{\beta}}_m^{(k+1)}$.
6: Calculate $\boldsymbol{e}_m^{(k+1)} = w_m \boldsymbol{e}^{(k)} - X_m \widehat{\Delta\boldsymbol{\beta}}_m^{(k+1)}$ and $d_m = \|\widehat{\Delta\boldsymbol{\beta}}_m^{(k+1)}\|_1$.
7: **end for**
8: Calculate $\boldsymbol{e}^{(k+1)} = \sum_{m=1}^{M} \boldsymbol{e}_m^{(k+1)}$ and $d = \sum_{m=1}^{M} d_m$.
9: $k = k + 1$.
10: **end while**
11: Set $\hat{\beta}_{0\tau_s}^{(k)} = Q_{\tau_s}^{*}\left(\boldsymbol{y} - \sum_{m=1}^{M} X_m \hat{\boldsymbol{\beta}}_m^{(k)}\right)$ $(s = 1, 2, \cdots, S)$.
12: **return** the final estimate

$$\hat{\boldsymbol{\beta}}_{\mathrm{CQR}} = \left(\left(\hat{\boldsymbol{\beta}}_{0S}^{(k)}\right)^{\mathrm{T}}, \left(\hat{\boldsymbol{\beta}}_1^{(k)}\right)^{\mathrm{T}}, \left(\hat{\boldsymbol{\beta}}_2^{(k)}\right)^{\mathrm{T}}, \cdots, \left(\hat{\boldsymbol{\beta}}_M^{(k)}\right)^{\mathrm{T}}\right)^{\mathrm{T}}.$$

**Assumption 1** Let $\lambda_{\min}(\cdot)$ and $\lambda_{\max}(\cdot)$ be the minimum and maximum eigenvalues of a positive definite matrix. Then we assume

$$c \le \lambda_{\min}\left(\frac{1}{n} X^{\mathrm{T}} X\right) \le \lambda_{\max}\left(\frac{1}{n} X^{\mathrm{T}} X\right) \le C,$$

where $c$ and $C$ are two positive constants.

Next, we give the convergence property of Algorithms 1 and 2 in two theorems.

**Theorem 1** *The PIQR algorithm developed for QR (i.e., Algorithm 1 in Sect. 3) enjoys the descent property. That is, the inequality*

$$\rho_\tau\left(\boldsymbol{y} - X^{*} \hat{\boldsymbol{\beta}}_{\mathrm{QR}}^{(k+1)}\right) \le \rho_\tau\left(\boldsymbol{y} - X^{*} \hat{\boldsymbol{\beta}}_{\mathrm{QR}}^{(k)}\right),$$

*holds for all $k = 0, 1, \cdots$. Further, if the design matrix $X$ satisfies Assumption 1, any cluster point of the estimate sequence $\left\{\hat{\boldsymbol{\beta}}_{\mathrm{QR}}^{(k)}\right\}$ generated by PIQR is a stationary point of $\rho_\tau\left(\boldsymbol{y} - X^{*} \boldsymbol{\beta}_{\mathrm{QR}}\right)$.*

***Proof*** The technical proof of this theorem is given in the Supplementary Information. □

Theorem 1 shows that through the iterative parallel PIQR algorithm, we can get a sequence of estimates for the QR parameter $\boldsymbol{\beta}_{\mathrm{QR}}$ that decreases the objective of the QR estimation problem. And more importantly, this sequence converges to a stationary point of the objective. That is, by using PIQR we can obtain an optimal estimate for $\boldsymbol{\beta}_{\mathrm{QR}}$.

Under the QR model, a very mild condition on the design matrix $X$ ensures that the estimate sequence $\left\{\hat{\boldsymbol{\beta}}_{\mathrm{QR}}^{(k)}\right\}$ converges to a stationary point of the QR objective function. For CQR, the result is slightly weaker. That is, if we further assume that the sample size $n$ is large enough, the cluster point $\left(Q_S^*\left(\boldsymbol{\beta}^*\right), \boldsymbol{\beta}^*\right)$ of the generated estimate sequence $\left\{\left(Q_S^*\left(\hat{\boldsymbol{\beta}}^{(k)}\right), \hat{\boldsymbol{\beta}}^{(k)}\right)\right\}$ by PIQR is also a stationary point of the CQR objective function, where the function $Q_S^*(\boldsymbol{\beta}) = \left(Q_{\tau_1}^*\left(\boldsymbol{y}-X\boldsymbol{\beta}\right), Q_{\tau_2}^*\left(\boldsymbol{y}-X\boldsymbol{\beta}\right), \cdots, Q_{\tau_S}^*\left(\boldsymbol{y}-X\boldsymbol{\beta}\right)\right)^{\mathrm{T}}$. We give this property in the following theorem.

**Theorem 2** *Let $\hat{\boldsymbol{\beta}}_{0S}^{*(k)} = \sum_{m=1}^{M} \hat{\boldsymbol{\beta}}_{0Sm}^{(k)}$, where $\hat{\boldsymbol{\beta}}_{0Sm}^{(k)}$ is the estimate for the temporary parameter $\boldsymbol{\beta}_{0Sm}$ obtained by (20). Then for the CQR model, the generated estimate sequence $\left\{\left(\hat{\boldsymbol{\beta}}_{0S}^{*(k)}, \hat{\boldsymbol{\beta}}^{(k)}\right)\right\}$ by PIQR satisfies the following inequality,*

$$\sum_{s=1}^{S} \rho_{\tau_s}\left(\boldsymbol{y}-\mathbf{1}_n\hat{\beta}_{0\tau_s}^{*(k+1)}-X\hat{\boldsymbol{\beta}}^{(k+1)}\right) \leq \sum_{s=1}^{S} \rho_{\tau_s}\left(\boldsymbol{y}-\mathbf{1}_n\hat{\beta}_{0\tau_s}^{*(k)}-X\hat{\boldsymbol{\beta}}^{(k)}\right), k=0,1,\cdots,$$

*where $\hat{\beta}_{0\tau_s}^{*(k)}$ is the $s$th ($s=1,2,\cdots,S$) entry of $\hat{\boldsymbol{\beta}}_{0S}^{*(k)}$. That is, under CQR the PIQR algorithm (i.e., Algorithm 2 in Sect. 4) also enjoys the descent property. Besides, if the design matrix $X$ satisfies Assumption 1, and further the sample size $n$ is large enough, the cluster point $\left(Q_S^*\left(\boldsymbol{\beta}^*\right), \boldsymbol{\beta}^*\right)$ of the sequence $\left\{\left(Q_S^*\left(\hat{\boldsymbol{\beta}}^{(k)}\right), \hat{\boldsymbol{\beta}}^{(k)}\right)\right\}$ is a stationary point of $\sum_{s=1}^{S} \rho_{\tau_s}\left(\boldsymbol{y}-\mathbf{1}_n\beta_{0\tau_s}-X\boldsymbol{\beta}\right)$.*

***Proof*** The technical proof of this theorem is also shown in the Supplementary Information. □

Theorem 2 shows that the estimate sequence for the CQR parameter $\boldsymbol{\beta}_{\mathrm{CQR}}$ we get by using the PIQR algorithm can also decrease the original objective of the estimation problem. And the final estimate we employed for estimating $\boldsymbol{\beta}_{\mathrm{CQR}}$ is actually an optimal one.

## 6 Simulation studies

In this section, we evaluate the performance of the ADMM-based parallel method developed in Sect. 2 and the PIQR algorithm proposed in Sects. 3–4 for QR and CQR through simulations. All the simulation experiments are conducted in R (R Development Core Team 2013) on a Mac with the Apple M1 chip (8-core CPU, 8-core GPU and 16-core Neural Engine) and a 16GB RAM. The source code of PIQR is available at https://github.com/nanlin999/PIQR.

The synthetic data sets in the experiment are generated from the following regression model,

$$y = \beta_0 + \sum_{j=1}^{p} x_j \beta_j + \epsilon, \tag{23}$$

with size $(n, p) = (10000, 500)$, $(10000, 1000)$ and $(50000, 500)$. The covariates $\left(x_1, x_2, \cdots, x_p\right)$ are generated from $N\left(\mathbf{0}_p, \mathbf{\Sigma}\right)$, $T_3(\mathbf{0}_p, \mathbf{\Sigma})$ or $\chi_1^2(\mathbf{\Sigma})$, where $T_3(\mathbf{0}_p, \mathbf{\Sigma})$ and $\chi_1^2(\mathbf{\Sigma})$ respectively represent the $p$-dimensional Student's $t$-distribution with three degrees of freedom (Kibria and Joarder 2006) and chi-square distribution with one degree of freedom (Royen 1995), and $\mathbf{\Sigma}$ is the correlation matrix with $\Sigma_{gh} = 0.5^{|g-h|}$ $(1 \le g, h \le p)$. The regression coefficients $\left(\beta_0, \beta_1, \cdots, \beta_p\right)$ are independently generated from the standard normal $N(0, 1)$, the Student's $t$-distribution $t(3)$ or the chi-square distribution $\chi^2(1)$. We also consider several choices on the distribution of the random error $\epsilon$, including $N(0, 1)$, $N(0, 4)$, $N(0, 9)$, $t(3)$, the normal mixture $\left(\epsilon = \sqrt{6}\epsilon^* \text{ and } \epsilon^* \sim 0.5N(0, 1) + 0.5N\left(0, 0.5^6\right)\right)$ and the standard Cauchy distribution C(0, 1). To mimic the vertically distributed environment, we split the generated sample covariate matrix into $M = 5, 10, 20, 50$ and 100 sub-matrices of equal size (i.e., size $n \times p_m$ with $p_m = p/M$ and $m = 1, 2, \cdots, M$) by features.

### 6.1 Results for QR

We first consider QR estimation problem at a specific quantile level $\tau = 0.3, 0.5$ or 0.7. Note that for the data generation model (23), we have $Q_\tau\left(y \mid x_1, x_2, \cdots, x_p\right) = \beta_0 + \sum_{j=1}^{p} x_j \beta_j + Q_\tau(\epsilon)$. Thus, according to the standard form of linear QR (1), we know that the true intercept of (23) under quantile level $\tau$ is actually $\{\beta_0 + Q_\tau(\epsilon)\}$. We employ the ADMM-based parallel method and the PIQR algorithm to estimate the regression coefficient in (23). In PIQR we use the equal weights $w_1 = w_2 = \cdots = w_M = \frac{1}{M}$ considering calculation simplicity and operation practicability, and adopt the classic IP (implemented in the R package `quantreg`) for the computation of the local subproblems (16). We initialize both algorithms at $\left(\hat{\beta}_0^{(0)}, \hat{\beta}_1^{(0)}, \cdots, \hat{\beta}_p^{(0)}\right)^{\mathrm{T}} = \mathbf{0}_{p+1}$, which contains no information about the regression coefficients, and stop the algorithm if the change of the coefficient estimate at two successive iterations $\sum_{j=0}^{p} \mid \hat{\beta}_j^{(k+1)} - \hat{\beta}_j^{(k)} \mid < \varepsilon = 10^{-2}$ or the number of iterations exceeds a large enough value $\max_{iter} = 5000$. The estimation accuracy and convergence speed of the algorithm are respectively evaluated by the absolute estimation error $\mathrm{AE} = \sum_{j=0}^{p} \mid \hat{\beta}_j - \beta_j \mid$ and the number of required iterations (Iteration). The communication cost is measured by the data volume (MB) transmitted between the central and local machines ($\mathrm{Commu}_{\mathrm{cost}}$) as used in Li et al. (2020). The computational efficiency is gauged by the average computing time ($\mathrm{T}_{\mathrm{computing}}$), communication time ($\mathrm{T}_{\mathrm{communication}}$) and total time cost ($\mathrm{T}_{\mathrm{total}}$). Tables 1, 2 and 3 and Figures S.1–S.6 (given in the Supplementary Information) report the simulation results of 50 runs for data generated with standard normal regression coefficients, covariates and error. As a benchmark, we also present the results obtained directly by the centralized IP, which requires accessing the raw data on one central

**Table 1** Comparison of PIQR, ADMM and IP for QR problem in vertically stored distributed big data under standard normal regression coefficients, covariates and error with $(n, p) = (10000, 500)$

| Method | $M$ | $\tau = 0.3$ | | | $\tau = 0.5$ | | | $\tau = 0.7$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | AE | Iteration | $\text{Commu}_{\text{cost}}$ | AE | Iteration | $\text{Commu}_{\text{cost}}$ | AE | Iteration | $\text{Commu}_{\text{cost}}$ |
| PIQR | 5 | 6.96 (0.28) | 91.14 (7.05) | 13.91 (1.08) | 6.55 (0.29) | 84.32 (6.68) | 12.87 (1.02) | 6.92 (0.32) | 88.88 (7.17) | 13.56 (1.09) |
| | 10 | 7.06 (0.25) | 155.28 (7.55) | 23.70 (1.15) | 6.63 (0.30) | 145.36 (6.65) | 22.18 (1.01) | 7.02 (0.32) | 154.14 (7.93) | 23.52 (1.21) |
| | 20 | 7.35 (0.27) | 280.72 (9.60) | 42.84 (1.46) | 6.86 (0.30) | 263.12 (8.56) | 40.15 (1.31) | 7.32 (0.29) | 280.04 (11.97) | 42.73 (1.83) |
| | 50 | 8.17 (0.29) | 623.74 (20.22) | 95.18 (3.09) | 7.58 (0.34) | 584.10 (20.43) | 89.13 (3.12) | 8.11 (0.37) | 621.92 (20.93) | 94.90 (3.19) |
| | 100 | 9.88 (0.48) | 1187.28 (36.88) | 181.18 (5.63) | 9.00 (0.37) | 1106.88 (30.32) | 168.91 (4.63) | 9.85 (0.50) | 1185.36 (41.58) | 180.89 (6.35) |
| ADMM | 5 | 7.02 (0.26) | 655.10 (10.12) | 199.94 (3.09) | 6.62 (0.25) | 571.22 (9.94) | 174.34 (3.03) | 6.95 (0.32) | 649.46 (9.53) | 198.22 (2.91) |
| | 10 | 7.03 (0.26) | 754.22 (13.31) | 230.19 (4.06) | 6.62 (0.25) | 669.76 (11.98) | 204.41 (3.66) | 6.96 (0.32) | 751.22 (13.53) | 229.27 (4.13) |
| | 20 | 7.03 (0.26) | 915.92 (16.28) | 279.54 (4.97) | 6.63 (0.25) | 829.26 (16.78) | 253.09 (5.12) | 6.97 (0.32) | 915.30 (18.33) | 279.35 (5.59) |
| | 50 | 7.04 (0.26) | 1249.48 (27.20) | 381.34 (8.30) | 6.64 (0.25) | 1153.96 (29.28) | 352.19 (8.94) | 6.97 (0.32) | 1240.78 (34.88) | 378.69 (10.65) |
| | 100 | 7.05 (0.27) | 1593.70 (35.65) | 486.40 (10.88) | 6.64 (0.25) | 1483.94 (39.06) | 452.90 (11.92) | 6.98 (0.31) | 1581.04 (40.52) | 482.53 (12.37) |
| IP | | 7.03 (0.25) | | | 6.62 (0.25) | | | 6.96 (0.32) | | |

The meanings of the notations used in this table are as follows. $M$: partition number; *AE*: absolute estimation error; *Iteration*: number of iterations; $\text{Commu}_{\text{cost}}$ *(MB)*: communication cost.
Numbers in the parentheses are the corresponding standard deviations.

**Table 2** Comparison of PIQR, ADMM and IP for QR problem in vertically stored distributed big data under standard normal regression coefficients, covariates and error with $(n, p) = (10000, 1000)$

| Method | $M$ | $\tau = 0.3$ | | | $\tau = 0.5$ | | | $\tau = 0.7$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | AE | Iteration | $\text{Commu}_{\text{cost}}$ | AE | Iteration | $\text{Commu}_{\text{cost}}$ | AE | Iteration | $\text{Commu}_{\text{cost}}$ |
| PIQR | 5 | 14.47 (0.31) | 138.06 (9.15) | 21.07 (1.40) | 13.64 (0.29) | 127.06 (8.41) | 19.39 (1.28) | 14.51 (0.47) | 140.28 (10.67) | 21.41 (1.63) |
| | 10 | 14.97 (0.26) | 230.30 (14.03) | 35.14 (2.14) | 14.02 (0.32) | 210.76 (10.88) | 32.16 (1.66) | 15.11 (0.48) | 233.56 (9.04) | 35.64 (1.38) |
| | 20 | 16.12 (0.43) | 424.86 (15.99) | 64.83 (2.44) | 14.80 (0.40) | 375.46 (15.26) | 57.30 (2.33) | 16.13 (0.59) | 424.24 (18.06) | 64.74 (2.76) |
| | 50 | 18.59 (0.52) | 966.22 (30.44) | 147.45 (4.65) | 16.87 (0.49) | 825.12 (24.05) | 125.91 (3.67) | 18.76 (0.68) | 955.54 (36.37) | 145.82 (5.55) |
| | 100 | 24.08 (0.86) | 1956.82 (63.32) | 298.61 (9.66) | 20.56 (0.56) | 1665.94 (59.43) | 254.22 (9.07) | 23.77 (0.69) | 1991.80 (89.23) | 303.95 (13.62) |
| ADMM | 5 | 14.24 (0.31) | 963.74 (15.67) | 294.13 (4.78) | 13.69 (0.26) | 845.68 (11.37) | 258.10 (3.47) | 14.44 (0.32) | 985.74 (14.59) | 300.85 (4.45) |
| | 10 | 14.24 (0.31) | 1104.90 (16.37) | 337.22 (5.00) | 13.70 (0.26) | 986.92 (17.81) | 301.21 (5.44) | 14.44 (0.32) | 1124.46 (12.60) | 343.19 (3.85) |
| | 20 | 14.26 (0.31) | 1321.22 (20.27) | 403.24 (6.19) | 13.71 (0.25) | 1196.46 (20.30) | 365.16 (6.20) | 14.45 (0.32) | 1330.50 (18.41) | 406.07 (5.62) |
| | 50 | 14.26 (0.31) | 1777.38 (38.19) | 542.46 (11.66) | 13.71 (0.25) | 1646.50 (31.04) | 502.51 (9.47) | 14.45 (0.32) | 1795.26 (33.49) | 547.91 (10.22) |
| | 100 | 14.26 (0.31) | 2345.76 (39.41) | 715.93 (12.03) | 13.72 (0.25) | 2211.12 (39.78) | 674.86 (12.14) | 14.46 (0.32) | 2352.92 (54.47) | 718.11 (16.62) |
| IP | | 14.25 (0.31) | | | 13.71 (0.25) | | | 14.45 (0.32) | | |

The notations $M$, AE, Iteration and $\text{Commu}_{\text{cost}}$ (MB) used here have the same meanings as those in Table 1.
Numbers in the parentheses are the corresponding standard deviations

**Table 3** Comparison of PIQR, ADMM and IP for QR problem in vertically stored distributed big data under standard normal regression coefficients, covariates and error with $(n, p) = (50000, 500)$

| Method | $M$ | $\tau = 0.3$ | | | $\tau = 0.5$ | | | $\tau = 0.7$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | AE | Iteration | $\text{Commu}_{\text{cost}}$ | AE | Iteration | $\text{Commu}_{\text{cost}}$ | AE | Iteration | $\text{Commu}_{\text{cost}}$ |
| PIQR | 5 | 3.05 (0.11) | 65.66 (2.28) | 50.10 (1.74) | 2.89 (0.14) | 65.02 (2.06) | 49.61 (1.57) | 3.05 (0.10) | 65.98 (1.70) | 50.34 (1.30) |
| | 10 | 3.07 (0.11) | 118.40 (2.68) | 90.34 (2.04) | 2.91 (0.14) | 118.16 (2.94) | 90.16 (2.24) | 3.07 (0.09) | 118.58 (3.71) | 90.48 (2.83) |
| | 20 | 3.14 (0.11) | 222.62 (5.90) | 169.86 (4.50) | 2.98 (0.13) | 218.72 (4.39) | 166.88 (3.35) | 3.14 (0.11) | 222.74 (5.50) | 169.95 (4.20) |
| | 50 | 3.46 (0.12) | 499.04 (8.81) | 380.77 (6.72) | 3.30 (0.11) | 484.22 (8.04) | 369.46 (6.13) | 3.47 (0.10) | 496.22 (7.90) | 378.62 (6.03) |
| | 100 | 4.22 (0.14) | 929.36 (12.54) | 709.10 (9.57) | 4.03 (0.14) | 912.66 (7.81) | 696.36 (5.96) | 4.22 (0.14) | 931.46 (10.61) | 710.70 (8.10) |
| ADMM | 5 | 3.07 (0.10) | 539.68 (4.24) | 823.55 (6.47) | 2.90 (0.16) | 465.76 (7.28) | 710.75 (11.11) | 3.07 (0.11) | 531.86 (6.64) | 811.62 (10.13) |
| | 10 | 3.07 (0.10) | 607.76 (7.66) | 927.44 (11.69) | 2.91 (0.16) | 535.12 (7.95) | 816.59 (12.13) | 3.07 (0.11) | 603.60 (5.48) | 921.09 (8.36) |
| | 20 | 3.09 (0.11) | 721.60 (12.09) | 1101.16 (18.45) | 2.92 (0.16) | 655.54 (8.17) | 1000.35 (12.47) | 3.09 (0.12) | 718.22 (8.81) | 1096.00 (13.44) |
| | 50 | 3.11 (0.11) | 944.88 (24.32) | 1441.89 (37.11) | 2.94 (0.16) | 870.54 (14.78) | 1328.44 (22.55) | 3.10 (0.11) | 930.44 (21.37) | 1419.85 (32.61) |
| | 100 | 3.13 (0.10) | 1165.46 (21.25) | 1778.49 (32.43) | 2.99 (0.15) | 1099.96 (25.36) | 1678.54 (38.70) | 3.16 (0.10) | 1160.84 (23.41) | 1771.44 (35.72) |
| IP | | 3.06 (0.11) | | | 2.90 (0.15) | | | 3.07 (0.11) | | |

The notations $M$, AE, Iteration and $\text{Commu}_{\text{cost}}$ (MB) used here have the same meanings as those in Table 1.
Numbers in the parentheses are the corresponding standard deviations

machine. Results under other data generation distributions are shown in Tables S.1–S.9 in the Supplementary Information.

Tables 1, 2 and 3 show that for small or moderate partition number, e.g., $M = 5$, 10 and 20, PIQR performs similarly to the ADMM-based parallel method and even the centralized IP in estimation accuracy. As $M$ continues to increase, the estimation error of PIQR slowly inflates, likely caused by the approximation in the surrogate objective function (15) in the parallelization of PIQR. Recall that using an upper bound of the objective in (12) as the surrogate makes it decomposable into $M$ low-dimensional subproblems depending only on local data. While $M$ increases, the local data dimensionality $p_m$ reduces, and then the intermediate PIQR estimate $\left(\left(\widehat{\Delta\boldsymbol{\beta}}_{\mathrm{QR}1}^{(k+1)}\right)^{\mathrm{T}}, \left(\widehat{\Delta\boldsymbol{\beta}}_{\mathrm{QR}2}^{(k+1)}\right)^{\mathrm{T}}, \cdots, \left(\widehat{\Delta\boldsymbol{\beta}}_{\mathrm{QR}M}^{(k+1)}\right)^{\mathrm{T}}\right)^{\mathrm{T}}$ obtained by (16) may deviate from the exact solution directly from (12), i.e., $\widehat{\Delta\boldsymbol{\beta}}_{\mathrm{QR}}^{(k+1)}$. On the other hand, estimates from the ADMM-based parallel method do not show this error inflation with increasing $M$ because this method does not introduce any approximation and directly converts the original QR estimation problem into an equivalent linearly constrained optimization problem. So the estimate obtained by it is expected to behave closely to the exact solution to QR problem computed using the centralized IP with full data.

However, the ADMM-based parallel method often requires hundreds or thousands of iterations to converge even when $M$ is small or moderate, and this implies that the communication cost of this method is usually expensive, as its data transmission volume is $\mathcal{O}(In)$ with $I$ denoting the required number of iterations (see the presented results for "$\mathrm{Commu}_{\mathrm{cost}}$" in Tables 1, 2 and 3). In distributed environments where the computing and communication costs have the same order, for example, the LTE Cat 4 module typically used in the M2M domain (peak data throughput: 150 Mbps in downlink and 50 Mbps in uplink), the communication efficiency of the ADMM-based parallel method is acceptable. See the middle panel of Figures S.1–S.3 in the Supplementary Information. But when the communication cost dominates the computing consumption, this method is rather inflexible. See our reported results of communication time for the IoT network (peak data throughput: 26 Kbps in downlink and 66 Kbps in uplink) in the middle panel of Figures S.4–S.6 in the Supplementary Information. On the other hand, PIQR needs much fewer iterations to converge (see our reported results for "Iteration" in Tables 1, 2 and 3). To more clearly show the advantage of PIQR in convergence speed, we also plot the estimation error of PIQR and the ADMM-based parallel method versus the number of iterations till PIQR converges. See Figure S.7 in the Supplementary Information. The data transmission volume of PIQR is also $\mathcal{O}(In)$, and thus its communication cost is usually much lower than that for the ADMM-based parallel method. Therefore, this method is especially appealing for the distributed IoT environment with low data transmission rate. See our comparison for the time performance of PIQR and the ADMM-based parallel method in the IoT network in Figures S.4–S.6 in the Supplementary Information. Besides, during the whole computation, PIQR only needs to transmit the full residuals $\boldsymbol{e}^{(k+1)}$ and the local residuals $\boldsymbol{e}_m^{(k+1)}$'s between the central and local machines, which can effectively preserve the data privacy, while the ADMM-based parallel method requires transmitting

quantities involving both the intermediate gradient (i.e., $\overline{\boldsymbol{u}}_{\mathrm{QR}}^{(k)}$) and covariates information (i.e., $\overline{X^*\boldsymbol{\beta}_{\mathrm{QR}}}^{(k)}$ and $X_m^*\boldsymbol{\beta}_{\mathrm{QR}m}^{(k)}$'s), and this operation can easily leak important data information as pointed out in Yang et al. (2019).

As for the computing time, from the reported results for $\mathrm{T}_{\mathrm{computing}}$ in the left panel of Figures S.1–S.6 in the Supplementary Information, we see that the ADMM-based parallel method shows an obvious advantage over PIQR. This is because, in essence the ADMM-based parallel method only needs to solve one $p$-dimensional check loss minimization problem during the QR estimation, while PIQR requires conducting $I$ rounds of check loss minimization operation and in each round, it solves $M$ low-dimensional check loss optimization problems in parallel (see (16)). However, as analyzed before, in the distributed environments with low data transmission rate, such as the IoT network, the communication cost between different machines can be much higher than that of local computation, and thus the PIQR is actually the winner when we consider total computational efficiency. See the right panel of Figures S.4–S.6 in the Supplementary Information.

Based on the simulation results, we find that for vertically stored distributed big data, when the data transmission cost of the distributed environment is expensive or the sample data are highly confidential, e.g., the IoT network considered in this experiment and the common medical data analysis in reality, especially when $M$ is small or moderate, the PIQR is a preferred method. Otherwise, the ADMM-based parallel method is more in favor.

### 6.2 Results for CQR

Next, we investigate the performance of CQR at quantile levels $(\tau_1, \tau_2, \tau_3, \tau_4, \tau_5) = (0.1, 0.3, 0.5, 0.7, 0.9)$ using both the PIQR and the ADMM-based parallel method. Please refer to Section 4 in the Supplementary Information for details about the ADMM-based parallel method for CQR. As a benchmark centralized algorithm for comparison, we also include the MM algorithm in Pietrosanu et al. (2021) (implemented in the R package `cqrReg`). This MM solution is also used in PIQR for solving the local CQR subproblems (20). Table 4 and Figures S.8–S.9 (given in the Supplementary Information) report results for 50 Monte Carlo simulations under standard normal regression coefficients, covariates and error. Note that in the CQR model, we calculate the absolute estimation error by $\mathrm{AE} = \sum_{s=1}^{5} \mid \hat{\beta}_{0\tau_s} - \beta_{0\tau_s} \mid + \sum_{j=1}^{p} \mid \hat{\beta}_j - \beta_j \mid$. Results in this experiment give findings similar to those found for QR. That is, PIQR is the winner for distributed environments with expensive communication cost or strict privacy-preservation constraint under small or moderate $M$ and the ADMM-based parallel method is for large $M$.

Besides, supplementary simulation results presented in Table S.10 in the Supplementary Information show that the CQR technique which considers simultaneous modeling at multiple quantiles is more efficient than the standard QR in estimation accuracy, especially for cases with extreme quantiles, and thus the necessity of developing distributed estimation methods for CQR in big data. In addition, it is worth noting that the high estimation efficiency of CQR also results in the relatively insensitivity

**Table 4** Comparison of PIQR, ADMM and MM for CQR problem in vertically stored distributed big data with quantile levels $(\tau_1, \tau_2, \tau_3, \tau_4, \tau_5) = (0.1, 0.3, 0.5, 0.7, 0.9)$

| Method | $M$ | $(n, p) = (10000, 500)$ | | | $(n, p) = (10000, 1000)$ | | | $(n, p) = (50000, 500)$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | AE | Iteration | $\text{Commu}_{\text{cost}}$ | AE | Iteration | $\text{Commu}_{\text{cost}}$ | AE | Iteration | $\text{Commu}_{\text{cost}}$ |
| PIQR | 5 | 5.38 (0.21) | 57.34 (0.85) | 8.75 (0.13) | 11.08 (0.26) | 70.74 (1.14) | 10.79 (0.17) | 2.35 (0.09) | 49.20 (0.49) | 37.54 (0.37) |
| | 10 | 5.38 (0.21) | 110.54 (1.70) | 16.87 (0.26) | 11.08 (0.27) | 140.04 (1.77) | 21.37 (0.27) | 2.37 (0.09) | 94.18 (0.83) | 71.86 (0.63) |
| | 20 | 5.40 (0.22) | 212.16 (3.13) | 32.38 (0.48) | 11.10 (0.26) | 262.66 (3.20) | 40.08 (0.49) | 2.44 (0.09) | 183.40 (1.32) | 139.93 (1.01) |
| | 50 | 5.55 (0.23) | 491.26 (5.49) | 74.97 (0.84) | 11.15 (0.27) | 609.84 (7.63) | 93.06 (1.16) | 2.76 (0.10) | 442.68 (3.00) | 337.76 (2.29) |
| | 100 | 5.99 (0.24) | 958.44 (8.29) | 146.26 (1.27) | 11.44 (0.28) | 1138.48 (12.42) | 173.73 (1.90) | 3.42 (0.11) | 852.20 (5.72) | 650.23 (4.36) |
| ADMM | 5 | 5.59 (0.23) | 793.36 (8.01) | 1210.59 (12.22) | 11.54 (0.29) | 1189.90 (8.65) | 1815.67 (13.20) | 2.47 (0.08) | 738.78 (3.21) | 5636.52 (24.49) |
| | 10 | 5.59 (0.22) | 850.66 (10.03) | 1298.02 (15.30) | 11.54 (0.29) | 1299.46 (12.04) | 1982.85 (18.37) | 2.47 (0.08) | 768.88 (4.30) | 5866.17 (32.81) |
| | 20 | 5.59 (0.23) | 945.84 (13.35) | 1443.26 (20.37) | 11.55 (0.28) | 1460.84 (18.58) | 2229.10 (28.35) | 2.50 (0.08) | 843.92 (10.04) | 6438.69 (76.60) |
| | 50 | 5.61 (0.22) | 1141.46 (18.66) | 1741.75 (28.49) | 11.55 (0.29) | 1786.38 (25.46) | 2725.84 (38.85) | 2.52 (0.09) | 937.76 (4.97) | 7154.64 (37.92) |
| | 100 | 5.62 (0.23) | 1367.28 (25.03) | 2086.33 (38.19) | 11.56 (0.28) | 2126.18 (32.59) | 3244.34 (49.73) | 2.63 (0.09) | 1156.00 (8.55) | 8819.70 (65.23) |
| MM | | 5.43 (0.22) | | | 11.20 (0.27) | | | 2.36 (0.09) | | |

The notations $M$, AE, Iteration and $\text{Commu}_{\text{cost}}$ (MB) used here have the same meanings as those in Table 1.
Numbers in the parentheses are the corresponding standard deviations

of the intermediate PIQR estimate $\left(\left(\widehat{\Delta\boldsymbol{\beta}}_1^{(k+1)}\right)^{\mathrm{T}}, \left(\widehat{\Delta\boldsymbol{\beta}}_2^{(k+1)}\right)^{\mathrm{T}}, \cdots, \left(\widehat{\Delta\boldsymbol{\beta}}_M^{(k+1)}\right)^{\mathrm{T}}\right)^{\mathrm{T}}$ with respect to the partition number $M$, and the PIQR may then outperform the ADMM-based parallel method in estimation accuracy under CQR (e.g., see the reported results of "AE" for $(n, p) = (10000, 1000)$ in Table 4).

## 7 Conclusion and discussion

In this work, we gave the algorithmic details of the ADMM-based parallel method for QR estimation in vertically stored distributed big data and further proposed a novel iterative parallel framework, PIQR. Through decomposing a surrogate objective function, PIQR solves the QR estimation problem by iteratively performing local quantile regressions based on the residual projection idea. We systematically compared the ADMM-based parallel method and the PIQR framework through simulation studies to provide practical guidance. Simulation results show that PIQR enjoys favorable estimation accuracy in distributed environments and usually requires much fewer iterations to converge than the ADMM-based parallel method, particularly when the partition number $M$ is small or moderate. This faster convergence makes PIQR potentially very appealing in context like IoT or federated learning where communication time is a crucial factor (Li et al. 2020; Konečnỳ et al. 2016; Ivkin et al. 2019). In addition, the transmission mechanism of PIQR is less likely to leak important data information, and thus for QR problem in highly confidential data it is also more in favor. However, we do notice that for cases with relatively large $M$, the performance of PIQR deteriorates compared to the ADMM-based parallel method. Possibly due to the estimation error inflation in the local solutions in PIQR, it even entails a longer overall computing time. But it is worth noting that the estimation accuracy of PIQR may be further improved by using more appropriate weights when distributing the local residuals. In this paper, we have always used the naive equal weights. This will be an important issue to study in the future to expand its application to contexts like federated learning.

Besides, we should note that the PIQR is a general parallel framework, and any of the QR estimation algorithms can be applied to the computation of the decomposed low-dimensional check loss minimization subproblems. Results in this paper, e.g., Tables 1, 2 and 3 and Tables S.1–S.6 in the Supplementary Information, are based on using the IP method to solve these subproblems. We also tried the computationally and communication more efficient MM algorithm but found it is not so robust, particularly for data generated with non-normal errors, and hence the estimation error is usually larger than that of IP (see Tables S.7–S.9 and Figures S.11–S.16 in the Supplementary Information). Therefore, another important future topic is to investigate the impact of the local solver on the global solution.

Overall, we believe that the PIQR framework has great potential for vertically stored distributed big data for its fast convergence. Although its performance under large partitions still needs improvement, further refinement of this framework will bring it as a competitive alternative to the popular ADMM framework.

## Supplementary Information

In Sections 1–3 of the Supplementary Information, we respectively give Lemma 1 and the technical proofs for Theorems 1 and 2. In Section 4, we give the algorithmic details of the ADMM-based parallel method for CQR estimation. Section 5 presents some additional simulation results.

**Supplementary Information** The online version contains supplementary material available at https://doi.org/10.1007/s10618-022-00914-4.

**Funding** Nan Lin's work is supported by NVDIA GPU grant program. Ye Fan's work is supported by Initial Scientific Research Fund of Young Teachers in Capital University of Economics and Business [Grant No. XRZ2022062], and partly supported by Special Fund for Basic Scientific Research of Beijing Municipal Colleges in Capital University of Economics and Business [Grant No. QNTD202207]. Jr-Shin Li's work is supported by the Air Force Office of Scientific Research under the award FA9550-21-1-0335.

## Declarations

**Conflict of interest** The authors have no competing interests to declare that are relevant to the content of this work.

## References

Ai M, Wang F, Yu J, Zhang H (2021) Optimal subsampling for large-scale quantile regression. J Complex 62:101512

Allen DE, Gerrans P, Powell R, Singh AK (2009) Quantile regression: its application in investment analysis. Finsia J Appl Finance 1(4):7–12

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J (2011) Distributed optimization and statistical learning via the alternating direction method of multipliers. Found Trends Mach Learn 3(1):1–122

Briollais L, Durrieu G (2014) Application of quantile regression to recent genetic and -omic studies. Hum Genet 133(8):951–966

Chen C, Wei Y (2005) Computational issues for quantile regression. Sankhyā Indian J Stat 67(2):399–417

Chen X, Xie M-G (2014) A split-and-conquer approach for analysis of extraordinarily large data. Stat Sin 24(4):1655–1684

Chen L, Zhou Y (2020) Quantile regression in big data: a divide and conquer based strategy. Comput Stat Data Anal 144:106892

Chen S, Billings SA, Luo W (1989) Orthogonal least squares methods and their application to non-linear system identification. Int J Control 50(5):1873–1896

Chen X, Liu W, Zhang Y (2019) Quantile regression under memory constraint. Ann Stat 47(6):3244–3273

Chen X, Liu W, Mao X, Yang Z (2020) Distributed high-dimensional regression under a quantile loss function. J Mach Learn Res 21(182):1–43

Fitzenberger B, Koenker R, Machado JAF (2013) Economic applications of quantile regression. Physica-Verlag Heidelberg, New York

Gamal ME, Lai L (2015) Are Slepian–Wolf Rates necessary for distributed parameter estimation? In: 2015 53rd annual Allerton conference on communication, control, and computing (Allerton), IEEE. pp 1249–1255

Gu Y, Zou H (2020) Sparse composite quantile regression in ultrahigh dimensions with tuning parameter calibration. IEEE Trans Inf Theory 66(11):7132–7154

He X, Pan X, Tan KM, Zhou WX (2021) Smoothed quantile regression for large-scale inference. J Econom. https://doi.org/10.1016/j.jeconom.2021.07.010

Hu A, Jiao Y, Liu Y, Shi Y, Wu Y (2021) Distributed quantile regression for massive heterogeneous data. Neurocomputing 448:249–262

Huang C, Huo X (2019) A distributed one-step estimator. Math Program 174(1):41–76

Hunter DR, Lange K (2000) Quantile regression via an MM algorithm. J Comput Gr Stat 9(1):60–77
Hunter DR, Lange K (2000) Optimization transfer using surrogate objective functions: rejoinder. J Comput Gr Stat 9(1):52–59
Ivkin N, Rothchild D, Ullah E, Braverman V, Stoica I, Arora R (2019) Communication-efficient distributed SGD with sketching. In: Proceedings of the 33rd conference on neural information processing systems (NeurIPS), pp 1–11
Jiang R, Yu K (2022) Renewable quantile regression for streaming data sets. Neurocomputing 508:208–224
Jordan MI, Lee JD, Yang Y (2019) Communication-efficient distributed statistical inference. J Am Stat Assoc 114(526):668–681
Kibria BG, Joarder AH (2006) A short review of multivariate T-distribution. J Stat Res 40(1):59–72
Koenker R (2017) Quantreg: quantile regression. https://CRAN.R-project.org/package=quantreg
Koenker R (2005) Quantile regression. Cambridge University Press, New York
Koenker R, Bassett JG (1978) Regression quantiles. Econometrica 46(1):33–50
Konečnỳ J, McMahan HB, Yu FX, Richtárik P, Suresh AT, Bacon D (2016) Federated learning: strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492
Lange K, Hunter DR, Yang I (2000) Optimization transfer using surrogate objective functions. J Comput Gr Stat 9(1):1–20
Lee JD, Liu Q, Sun Y, Taylor JE (2017) Communication-efficient sparse regression. J Mach Learn Res 18(1):115–144
Lin N, Xi R (2011) Aggregated estimating equation estimation. Stat Interface 4(1):73–83
Li A, Sun J, Wang B, Duan L, Li S, Chen Y, Li H (2020) LotteryFL: personalized and communication-efficient federated learning with lottery ticket hypothesis on non-IID datasets. arXiv preprint arXiv:2008.03371
Miao W, Narayanan V, Li J-S (2020) Parallel residual projection: a new paradigm for solving linear inverse problems. Sci Rep 10(1):12846
Pan R, Ren T, Guo B, Li F, Li G, Wang H (2022) A note on distributed quantile regression by pilot sampling and one-step updating. J Bus Econ Stat 40(4):1691–1700
Peng L, Huang Y (2008) Survival analysis with quantile regression models. J Am Stat Assoc 103(482):637–649
Pietrosanu M, Gao J, Kong L, Jiang B, Niu D (2021) Advanced algorithms for penalized quantile and composite quantile regression. Comput Stat 36(1):333–346
Portnoy S, Koenker R (1997) The Gaussian hare and the Laplacian tortoise: computability of squared-error versus absolute-error estimators. Stat Sci 12(4):279–300
R Development Core Team (2013) R: a language and environment for statistical computing. http://www.R-project.org
Royen T (1995) On some central and non-central multivariate chi-square distributions. Stat Sin 5:373–397
Sherwood B, Wang L, Zhou X-H (2013) Weighted quantile regression for analyzing health care cost data with missing covariates. Stat Med 32(28):4967–4979
Shi L, Ye Y, Chu X, Lu G (2020) Computation bits maximization in a backscatter assisted wirelessly powered MEC network. IEEE Commun Lett 25(2):528–532
Takeuchi I, Le QV, Sears TD, Smola AJ (2006) Nonparametric quantile estimation. J Mach Learn Res 7(45):1231–1264
Tan KM, Battey H, Zhou WX (2022) Communication-constrained distributed quantile regression with optimal statistical guarantees. J Mach Learn Res 23:1–61
Trofimov I, Genkin A (2017) Distributed coordinate descent for generalized linear models with regularization. Pattern Recognit Image Anal 27(2):349–364
Trofimov I, Genkin A (2015) Distributed coordinate descent for L1-regularized logistic regression. In: International conference on analysis of images, social networks and texts, Springer. pp 243–254
Volgushev S, Chao S-K, Cheng G (2019) Distributed inference for quantile regression processes. Ann Stat 47(3):1634–1662
Wang H, Li C (2017) Distributed quantile regression over sensor networks. IEEE Trans Signal Inf Process Netw 4(2):338–348
Wang H, Ma Y (2021) Optimal subsampling for quantile regression in big data. Biometrika 108(1):99–112
Wang L, Wu Y, Li R (2012) Quantile regression for analyzing heterogeneity in ultra-high dimension. J Am Stat Assoc 107(497):214–222
Wu TT, Lange K (2008) Coordinate descent algorithms for lasso penalized regression. Ann Appl Stat 2(1):224–244

Wu Y, Liu Y (2009) Variable selection in quantile regression. Stat Sin 19(2):801–817
Xi R, Lin N, Chen Y (2008) Compression and aggregation for logistic regression analysis in data cubes. IEEE Trans Knowl Data Eng 21(4):479–492
Yang J, Meng X, Mahoney MW (2014) Quantile regression for large-scale applications. SIAM J Sci Comput 36(5):78–110
Yang Q, Liu Y, Chen T, Tong Y (2019) Federated machine learning: concept and applications. ACM Trans Intell Syst Technol 10(2):1–19
Ye Y, Shi L, Chu X, Li D, Lu G (2021) Delay minimization in wireless powered mobile edge computing with hybrid Backcom and AT. IEEE Wirel Commun Lett 10(7):1532–1536
Yu L, Lin N (2017) ADMM for penalized quantile regression in big data. Int Stat Rev 85(3):494–518
Yu K, Lu Z, Stander J (2003) Quantile regression: applications and current research areas. J R Stat Soc Ser D 52(3):331–350
Yu L, Lin N, Wang L (2017) A parallel algorithm for large-scale nonconvex penalized quantile regression. J Comput Gr Stat 26(4):935–939
Zheng H, Kulkarni SR, Poor HV (2010) Attribute-distributed learning: models, limits, and algorithms. IEEE Trans Signal Process 59(1):386–398
Zhang Y, Duchi JC, Wainwright MJ (2013a) Communication-efficient algorithms for statistical optimization. J Mach Learn Res 14(1):3321–3363
Zhang Y, Duchi JC, Jordan MI, Wainwright MJ (2013b) Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In: Proceedings of the 26th international conference on neural information processing systems (NIPS), pp 2328–2336
Zou H, Yuan M (2008) Composite quantile regression and the oracle model selection theory. Ann Stat 36(3):1108–1126
Zou Y, Xu J, Gong S, Guo Y, Niyato D, Cheng W (2019) Backscatter-aided hybrid data offloading for wireless powered edge sensor networks. In: 2019 IEEE global communications conference (GLOBECOM). IEEE, pp 1–6