FAST QUANTILE REGRESSION FOR GENOMICS RESEARCH

BY ETHAN X. FANG[1,*,a] TIANCHEN XU[2,†,c], XIAOYU SONG[3,‡,d] AND YING WEI [2,†,b]
DUKE UNIVERSITY*, COLUMBIA UNIVERSITY† AND MOUNT SINAI‡

[1] *Department of Biostatistics & Bioinformatics, Duke University, Durham, NC 27705,* [a]*xingyuan.fang@duke.edu*

[2] *Department of Biostatistics, Columbia University, New York, NY 10032,* [b]*yw2148@cumc.columbia.edu*
[c]*tx2155@columbia.edu*

[3] *Department of Population Health Science & Policy, Icahn School of Medicine at Mount Sinai, New York, NY 10029,*
[d]*xiaoyu.song@mountsinai.org*

We propose a novel fast estimating algorithm for quantile regressions that share common covariates. We begin with a brief review of the standard linear programming (LP) algorithm for quantile regression. With this knowledge, we propose an adaptive LP algorithm for solving a set of quantile regression problems: we fit the common variables first and then add other predictors to fitted results to obtain the optimal solutions efficiently. We demonstrate the efficiency of the proposed algorithm through comprehensive simulation studies. An application to the Genomics of Drug Sensitivity in Cancer project provides valuable insights into the computation advantages of the proposed algorithms.

1. Introduction. In recent years, the field of genomics research has experienced explosive growth, driven by advancements in high-throughput sequencing technologies and the accessibility of extensive genomic repositories. Researchers are dedicated to decode the intricate genetic basis of complex traits, from disease susceptibility to drug response. As our exploration of the genome deepens, the scale and complexity of genomic data have grown exponentially, creating a pressing need for efficient computational solutions.

To address these challenges, scientists have been actively developing a range of efficient methods for diverse genomic analyses. For example, they have introduced accelerated techniques for genome-wide association studies (GWAS) (Lippert et al., 2011), expression quantitative trait loci (eQTL) studies (Shabalin, 2012), and single-cell studies (Korsunsky et al., 2019), and many other general analyses (Mbatchou et al., 2021). These computationally efficient methods have made it feasible to analyze vast datasets encompassing millions of samples and significantly accelerated the pace of the scientific discoveries.

Quantile regression, initially proposed by Koenker and Bassett (1978), has emerged as a crucial tool in genomics research (Song et al., 2017; Zhang et al., 2020; Ling et al., 2021; Wang, Ionita-Laza and Wei, 2022). It focuses on modeling the conditional quantiles of a response variable $\boldsymbol{Y}$ given its covariates $\boldsymbol{X}$ (Mosteller and Tukey, 1977). This method serves as a valuable complement to traditional mean-based analyses, enabling the discovery of

arXiv: arXiv:0000.0000
Keywords and phrases: Linear Programming, Quantile Regression, Shared design matrix, Simplex method.

quantile-specific, heterogeneous, and high-order associations (Cade and Noon, 2003). However, no tools are currently available to expedite the computation for quantile regression for genomic analyses.

This study is motivated by the quest to screen for genes associated with drug response, using the potent tool of quantile regression. In drug response investigations, scientists often study how cell lines react to various drugs, by quantifying these responses using metrics like the half-maximal inhibitory concentration. In this context, lower quantiles of these metrics hold exceptional significance, as they suggest the possibility of achieving therapeutic effects with lower drug doses. This prospect not only holds the potential to reduce side effects but also enhances patient tolerance - a highly coveted outcome, especially in clinical settings. To embark on this quest, we harness data from the Genomics of Drug Sensitivity in Cancer (GDSC) project (Iorio et al., 2016). This dataset encompasses expression levels of $17,736$ genes and roughly 266 different anticancer compounds. Analyzing this extensive dataset entails performing approximately five million regressions for each quantile level of interest. If we explore multiple quantile levels, the computational workload further multiplies, underscoring the formidable computational challenges in genomics study.

To solve the computational challenges, we propose to develop a computational expedition method for quantile regression-based genomic analyses. The core concept underlying this method is to capitalize the shared structure in numerous parallel regressions in genetics studies. Specifically, our approach takes root in genomics studies that typically an outcome Y is regressed with individual predictors (one at a time) or a small group of predictors, while accounting for potential confounding factors. For instance, in GWAS, the outcome variable Y represents a specific trait, the individual predictors are genetic variants, and the covariates include variables that capture the population structure. Similarly, in drug screening studies, the outcome variable Y pertains to drug response, the individual predictors are gene expression, and the covariates might include different cancer types. These studies have the same features: the number of regressions is large, while the model dimension of each regression is limited. In the meantime, the design matrices share a considerable similarity among these regressions, since they share the controlling variables. Such a fact motivates us to develop an adaptive algorithm that utilizes the information from analogous regressions, and boosts up the whole fitting procedure.

To describe the problem in mathematical equations, we let $(Y_i, \boldsymbol{X}_i, \boldsymbol{Z}_i)_{i=1,\ldots,n}$ be an observational data set of sample size n, where Y_i is an outcome of interest, $\boldsymbol{X}_i$ is a high-dimensional primary predictor, and $\boldsymbol{Z}_i$ is a q-dimensional vector of covariates including intercept for potential confounding effects. In exploratory analysis, $\boldsymbol{X}$ is often partitioned into

m small segments, i.e. $\boldsymbol{X}_i = (\boldsymbol{X}_i^{(1)}, ..., \boldsymbol{X}_i^{(m)})$ to conduct a sequence of m regressions by

$$g(Y_i) = \boldsymbol{X}_i^{(\ell)}\boldsymbol{\beta}_\ell + \boldsymbol{Z}_i\boldsymbol{\alpha}_\ell,\ \ell = 1, \cdots, m, \tag{1.1}$$

where $g(\cdot)$ is the link function. Different link functions can be used. When Y is a continuous modeled with linear regression, $g(Y) = \mathrm{E}(Y)$ links to the mean function of Y. Without loss of generality, we assume the dimension of $\boldsymbol{X}_i^{(\ell)}$ is p for all $\ell \in \{1, ..., m\}$. If $p = 1$, then Model (1.1) explores the pair-wise association between Y and each component of $\boldsymbol{X}$. For quantile regression, we rewrite Model (1.1) as

$$Q_Y(\tau|\boldsymbol{X}_i^{(\ell)}, \boldsymbol{Z}_i) = \boldsymbol{X}_i^{(\ell)}\boldsymbol{\beta}_{\ell,\tau} + \boldsymbol{Z}_i\boldsymbol{\alpha}_{\ell,\tau},\ \ell = 1, \cdots, m, \tag{1.2}$$

where $Q_Y(\tau|\boldsymbol{X}_i^{(\ell)}, \boldsymbol{Z}_i)$ is the τ-th quantile of Y given $\boldsymbol{X}_i^{(\ell)}$ and $\boldsymbol{Z}_i$; and $\boldsymbol{\beta}_{\ell,\tau}$ and $\boldsymbol{\alpha}_{\ell,\tau}$ are quantile specific coefficients.

The exploratory models in (1.1) and (1.2) have the following features: the number of regressions, m, is large, while the model dimension of each regression is limited. In the meantime, we notice that the design matrices share a considerable similarity among these regressions, since they share the controlling variables $\boldsymbol{Z}_i$. Such a fact motivates us to develop an adaptive algorithm that utilizes the information from analogous regressions, and boosts up the whole fitting procedure.

The estimation of quantile regression falls in the category of M-estimation, where its loss function at the τ-th quantile level can be written as $\rho_\tau(\xi) = \xi \cdot \{\tau - \mathbb{I}(\xi < 0)\}$, and is non-differentiable at 0. Consequently, the estimated quantile coefficients do not have a closed-form solution, and cannot be estimated by the classical Newton-Raphson algorithm. Instead, it is re-casted as a linear programming problem, which optimizes a linear function under linear constraints. Compared with mean regressions, the computation cost of linear programming increases quickly with sample size and the number of covariates. Hence there is a great need to reduce the computation time of quantile regression, so that it could be a feasible analysis tool for big data applications.

There are mainly two categories of linear programming algorithms: (i) Exterior-point algorithm based on simplex or modified simplex methods (Barrodale and Roberts, 1974; Koenker and d'Orey, 1987, 1994); (ii) Frisch-Newton interior-point algorithm (Frisch, 1956), including its modified versions such as preprocessing algorithm for the Frisch-Newton algorithm, sparse implementation of the Frisch-Newton algorithm (Portnoy et al., 1997) and so forth.

A simplex algorithm is often recommended for regressions with limited dimensions, since it is computationally stable. The modified simplex method is quite efficient for problems up to several thousand observations, and may be used to compute the full quantile regression process. In addition, unlike interior-point-type algorithms, simplex methods can better utilize an initial solution for more efficient warm starts. Therefore, we focus on the simplex

algorithm in this paper, and propose a new adaptive simplex algorithm that could greatly reduce the computation time by incorporating "shared" design matrices. The basic idea is to warm-start each problem through previously obtained solutions from a similar design matrix.

The rest of the paper is organized as follows. We will take a review of linear programming and the simplex method first, and then introduce our adaptive algorithm. Finally, we conduct a numerical study on the cancer drug sensitivity dataset to illustrate the improvement of the computation efficiency.

2. Review of Simplex Algorithm for Quantile Regression. A quantile regression model assumes that the conditional quantile of Y given $\boldsymbol{X}$ is a linear function of $\boldsymbol{X}$, i.e $Q_Y(\tau|\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{\beta}_\tau$. The quantile coefficient $\boldsymbol{\beta}_\tau$ can be defined as

$$\boldsymbol{\beta}_\tau = \arg\min_{\boldsymbol{\beta}} \mathrm{E}_Y \left\{ \rho_\tau(Y - \boldsymbol{X}\boldsymbol{\beta}) \right\},$$

where $\rho_\tau(\xi) = \xi \cdot \{\tau - \mathbb{I}(\xi < 0)\}$ is the quantile loss function. With a representative random sample $\{(Y_i, \boldsymbol{X}_i)\}_{i=1}^n$, one can estimate $\boldsymbol{\beta}_\tau$ consistently by minimizing the sample quantile loss

$$\widehat{\boldsymbol{\beta}}_\tau = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \rho_\tau(y_i - \boldsymbol{x}_i\boldsymbol{\beta}). \tag{2.1}$$

As we mentioned in the Introduction, the quantile loss function ρ_τ is non-smooth. The classical Newton-Raphson algorithm cannot be applied to solve (2.1). Instead, we use linear programming (LP) techniques to solve (2.1) to obtain an estimator of $\boldsymbol{\beta}_\tau$. In what follows, we briefly review the concept of LP, and illustrate how the quantile regression optimization can be translated into a linear programming problem, followed by a review of the general simplex method, which is the fundamental approach for solving LP problems.

2.1. *Quantile Regression and Linear Programming.* Linear Programming is a widely-used optimization method to optimize a linear function under linear constraints. A standard LP program takes the following form

$$\min_{\boldsymbol{\eta}} \ \boldsymbol{c}^\top \boldsymbol{\eta}, \quad \text{s.t. } \boldsymbol{A}\boldsymbol{\eta} = \boldsymbol{Y}, \ \boldsymbol{\eta} \geq 0, \tag{2.2}$$

where $\boldsymbol{c}$ and $\boldsymbol{Y}$ are known vectors, and $\boldsymbol{A}$ is a known matrix. The estimation of $\boldsymbol{\beta}_\tau$ in (2.1) can be equivalently formulated as

$$\begin{aligned} \hat{\boldsymbol{\beta}}_\tau = \arg\min_{\boldsymbol{\beta}_\tau} \ & \tau \mathbf{1}_n^\top \boldsymbol{u} + (1-\tau)\mathbf{1}_n^\top \boldsymbol{v}, \\ \text{subject to } & \boldsymbol{Y} = \boldsymbol{X}(\boldsymbol{\beta}_\tau^+ - \boldsymbol{\beta}_\tau^-) + \boldsymbol{u} - \boldsymbol{v}, \\ & \boldsymbol{\beta}_\tau = \boldsymbol{\beta}_\tau^+ - \boldsymbol{\beta}_\tau^-, \\ & \boldsymbol{\beta}_\tau^+ \geq 0, \boldsymbol{\beta}_\tau^- \geq 0, \boldsymbol{u} \geq 0, \boldsymbol{v} \geq 0, \end{aligned} \tag{2.3}$$

where $\mathbf{1}_n$ is a vector of all 1's with dimension n, $\boldsymbol{Y} = (y_1, y_2, ..., y_n)^\top \in \mathbb{R}^n$, $\boldsymbol{X} = \left(\boldsymbol{x}_1^\top, \boldsymbol{x}_2^\top, ..., \boldsymbol{x}_n^\top\right)^\top \in \mathbb{R}^{n\times p}$, and the first value of $\boldsymbol{x}_i$ is 1 to incorporate intercept. We rewrite problem (2.3) into the standard linear programming form (2.2), where

$$\begin{aligned}
\boldsymbol{c}^\top &= (\underbrace{0, 0, \ldots, 0}_{2p\ 0's}, \underbrace{\tau, \tau, \ldots, \tau}_{n\ \tau's}, \underbrace{1-\tau, 1-\tau, \ldots, 1-\tau}_{n\ (1-\tau)'s}) \in \mathbb{R}^{2p+2n},\\
\boldsymbol{A} &= (\boldsymbol{X}, -\boldsymbol{X}, \boldsymbol{I}_n, -\boldsymbol{I}_n) \in \mathbb{R}^{n\times(2p+2n)},\\
\boldsymbol{\eta} &= (\boldsymbol{\beta}_\tau^{+\top}, \boldsymbol{\beta}_\tau^{-\top}, \boldsymbol{u}^\top, \boldsymbol{v}^\top)^\top \in \mathbb{R}^{2p+2n}.
\end{aligned}$$

2.2. *Simplex algorithm.* Note that in the quantile regression optimization (2.2), the dimension of the parameter $\boldsymbol{\eta}$ is $(2p+2n)$, while the number of linear constraints in $\boldsymbol{A\eta} = \boldsymbol{Y}$ is n. If we let $(2p+n)$ elements in $\boldsymbol{\eta}$ equal to 0's and solve the resulting equations $\boldsymbol{A\eta} = \boldsymbol{Y}$, we have one feasible solution of $\boldsymbol{\eta}$ that satisfies the linear constraints. We call such a solution a *vertex* solution. It is well known that each LP problem has a finite number of vertex solutions, and if the LP problem admits an optimal solution, one of the vertex solutions is an optimal solution (Bertsimas and Tsitsiklis, 1997). The maximum number of vertex/feasible solutions is $\binom{2p+2n}{n}$. Naturally, one could compute all vertex/feasible solutions, evaluate their corresponding objective values, and find the optimal solution that produces the minimum objective value. However, this exhaustive-search approach is computationally daunting as p and n increase.

Alternatively, the simplex method provides an iterative optimization algorithm that starts at a vertex solution, and iteratively moves to another vertex solution that reduces the objective value until an optimal solution is found. The simplex algorithm significantly improves the computational efficiency from exhaustive-search approach, but its computational time is very sensitive to the choice of the starting feasible solution. Appropriately chosen starting solutions could greatly enhance the computation efficiency of the simplex algorithm, which is the foundation of the proposed methods.

Before going to the proposed algorithms, we first introduce related concepts and notations. We define $I = \{1, 2, 3, \ldots, (2p+2n)\}$ as the column index set of the matrix $\boldsymbol{A}$. We randomly partition I into two mutually-exclusive sub-sets B and D, where $I = B \cup D$, and the dimension of B is n, while the dimension of D is $2p+n$. We then define $\boldsymbol{A}[B]$ as the sub-matrix of $\boldsymbol{A}$, whose columns are in the index set B, and $\boldsymbol{A}[D]$ is the complementary sub-matrix whose column index is in D. Likewise, we define $\boldsymbol{\eta}_B$ and $\boldsymbol{\eta}_D$ as sub-vectors of $\boldsymbol{\eta}$ that are associated with $\boldsymbol{A}[B]$ and $\boldsymbol{A}[D]$, respectively. We also define $\boldsymbol{c}_B$ and $\boldsymbol{c}_D$ analogously. This way, we rewrite the constraint $\boldsymbol{A\eta} = \boldsymbol{Y}$ as:

$$\boldsymbol{A}[B]\boldsymbol{\eta}_B + \boldsymbol{A}[D]\boldsymbol{\eta}_D = \boldsymbol{Y}.$$

In what follows, we call the variables associated with the columns in $\boldsymbol{A}[B]$ as basic variables, and call those associated with $\boldsymbol{A}[D]$ non-basic variables. When we set $\boldsymbol{\eta}_D = 0$, and solve $\boldsymbol{A}[B]\boldsymbol{\eta}_B = \boldsymbol{Y}$ for $\boldsymbol{\eta}_B$, we obtain a vertex solution. If we move one basic variable in $\boldsymbol{A}[B]$ to $\boldsymbol{A}[D]$, and move one non-basic variable in $\boldsymbol{A}[D]$ to $\boldsymbol{A}[B]$, we obtain a neighboring vertex solution. The simplex algorithm searches for the optimal solution by moving from one basic vertex solution to another (Bertsimas and Tsitsiklis, 1997).

The rules for deciding the leaving and entering variables are called pivoting rules. Following the convention in LP literature, we use a tableau to present the simplex pivoting iteration. Table 1 illustrates the general tableau given basic variable index set B during pivoting iteration.

	η_{B_1}	η_{B_2}	$\cdots$	η_{B_n}	η_{D_1}	η_{D_2}	$\cdots$	$\eta_{D_{2p+n}}$	
η_{B_1}	$\boldsymbol{\Gamma}$ 1	0	$\cdots$	0	γ_{1,D_1}	γ_{1,D_2}	$\cdots$	$\gamma_{1,D_{2p+n}}$	$\boldsymbol{b}$ b_1
η_{B_2}	0	1	$\cdots$	0	γ_{2,D_1}	γ_{2,D_2}		$\gamma_{2,D_{2p+n}}$	b_2
$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$	$\vdots$
η_{B_n}	0	0	$\cdots$	1	γ_{n,D_1}	γ_{n,D_2}	$\cdots$	$\gamma_{n,D_{2p+n}}$	b_n
r	$\boldsymbol{r}$ 0	0	$\cdots$	0	r_{D_1}	r_{D_2}	$\cdots$	$r_{D_{2p+n}}$	$-\zeta$

Table 1: Simplex tableau with given basic variable index set B

As shown in Vanderbei et al. (2015), the tableau in Table 1 can be written compactly as

$$T(B) = \begin{bmatrix} \{\boldsymbol{A}[B]\}^{-1} & \boldsymbol{0} \\ -\boldsymbol{c}_B^\top \{\boldsymbol{A}[B]\}^{-1} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{A} & \boldsymbol{Y} \\ \boldsymbol{c}^\top & 0 \end{bmatrix}. \tag{2.4}$$

In the meantime, we let

$$\begin{aligned} \boldsymbol{\Gamma} = \{\boldsymbol{A}[B]\}^{-1}\boldsymbol{A}, \quad \boldsymbol{b} = \{\boldsymbol{A}[B]\}^{-1}\boldsymbol{Y}, \\ \boldsymbol{r} = \boldsymbol{c}^\top - \boldsymbol{c}_B^\top \{\boldsymbol{A}[B]\}^{-1}\boldsymbol{A}, \quad -\zeta = -\boldsymbol{c}_B^\top \{\boldsymbol{A}[B]\}^{-1}\boldsymbol{Y}. \end{aligned} \tag{2.5}$$

Here $\boldsymbol{\Gamma}$ is a row- and column-rotated transformation of $\boldsymbol{A}$, whose sub-matrix associated with B is an identity matrix. The linear spaces expanded from $\boldsymbol{A}$ and $\boldsymbol{\Gamma}$ are identical. The vector $\boldsymbol{b} = (b_1, ..., b_n)$ is the solution to $\boldsymbol{A}[B]\boldsymbol{\eta}_B = \boldsymbol{Y}$. The feasible solution given the basic variables B is $\widehat{\boldsymbol{\eta}} = (\widehat{\boldsymbol{\eta}}_B, \widehat{\boldsymbol{\eta}}_D) = (\boldsymbol{b}, 0)$, at which the objective value is $\boldsymbol{c}^\top\widehat{\boldsymbol{\eta}} = -\zeta$. Same as $\boldsymbol{\eta}$, we partition $\boldsymbol{r}$ by $\boldsymbol{r} = (\boldsymbol{r}_B, \boldsymbol{r}_D)$. By definition, $\boldsymbol{r}_B = 0$, while r_{D_j} measures the reduction of the objective function $\boldsymbol{c}^\top\boldsymbol{\eta}$ by changing η_{D_j} from 0 to 1. One popular pivoting rule is to choose D_j with the smallest r_{D_j} as the entering variable. Or, equivalently, we choose the t-th variable as the entering variable where

$$t = \arg\min_j \left\{ r_{D_j} \middle| 1 \le D_j \le 2p + n \right\}.$$

For the leaving basic variable, in order to maintain the non-negativity of the solution after pivoting, we choose the leaving variable as the k-th variable such that

$$k = \arg\min_i \left\{ \frac{b_i}{\gamma_{it}} \middle| \gamma_{it} > 0, 1 \le i \le n \right\}.$$

Note that after one pivot, variable t becomes a basic variable from a non-basic variable, and variable k becomes a non-basic variable from a basic variable. We then update the sets B and D accordingly. With the new set of B and D, we can update tabular $T(B)$ accordingly by (2.5). The simplex algorithm iteratively repeats this pivoting procedure until all elements of $\boldsymbol{r}$ are non-negative. That is, any pivot will not improve the objective value, when the optimal solution is reached.

2.3. *Simplex algorithm for quantile regression.* When it comes to quantile regression (2.2), we summarize the initial tableau in the following table:

	η_1	η_2	$\cdots$	η_p	η_{p+1}	η_{p+2}	$\cdots$	η_{2p}	η_{2p+1}		$\cdots$	η_{2p+n}			$\cdots$	η_{2p+2n}	
	β_1^+	β_2^+	$\cdots$	β_p^+	β_1^-	β_2^-	$\cdots$	β_p^-	u_1	u_2	$\cdots$	u_n	v_1	v_2	$\cdots$	v_n	
η_{2p+1}	1	x_{11}	$\cdots$	$x_{1(p-1)}$	-1	$-x_{11}$	$\cdots$	$-x_{1(p-1)}$	1				-1				y_1
η_{2p+2}	1	x_{21}	$\cdots$	$x_{2(p-1)}$	-1	$-x_{21}$	$\cdots$	$-x_{2(p-1)}$		1				-1			y_2
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$			$\ddots$				$\ddots$		$\vdots$
η_{2p+n}	1	x_{n1}	$\cdots$	$x_{n(p-1)}$	-1	$-x_{n1}$	$\cdots$	$-x_{n(p-1)}$				1				-1	y_n
r	0	0	$\cdots$	$\cdots$	$\cdots$	$\cdots$	0	0	τ	τ	$\cdots$	τ	$1-\tau$	$1-\tau$	$\cdots$	$1-\tau$	$-\zeta$

Table 2: Initial tableau of a quantile regression

The easiest way to start is to choose $B = \{2p+1, ..., 2p+n\}$ as the initial set of basic variables, since its associated $\boldsymbol{A}[B]$ is already an identity matrix. In Table 2, the first two rows of the table illustrate the relationship between $\boldsymbol{\eta}$ and $(\boldsymbol{\beta}, \boldsymbol{u}, \boldsymbol{v})$, which is defined in the last part of Section 2.1, and the red box highlights the initial basic variables, and the first column indicates the basic variables as well. The elements inside the dashed box represent the constraint $\boldsymbol{A\eta} = \boldsymbol{Y}$, and the last row includes the coefficients $\boldsymbol{c}^\top$ and the value of objective function $-\zeta = \boldsymbol{c}^\top \boldsymbol{\eta} = \sum_{i=1}^n \rho_\tau\{y_i - \boldsymbol{x}_i \boldsymbol{\beta}_\tau\}$. Starting from Table 2, we iteratively apply the pivot algorithm, until all $r_j \geq 0$. Once the algorithm terminates, we obtain an optimal solution $\hat{\boldsymbol{\eta}}$ and also the estimator $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta}^+ - \boldsymbol{\beta}^-$ for the quantile regression model.

REMARK 1. *Without loss of generality, we assume all y's are positive. If any $y_i < 0$, we can simply multiply the rows that the corresponding negative y_i's by -1, and replace the basic variable u_i by v_i.*

REMARK 2. *This simplex method provides a relatively efficient way to solve a quantile regression. We introduce extra variables $\boldsymbol{\beta}_\tau^+$ and $\boldsymbol{\beta}_\tau^-$ to replace $\boldsymbol{\beta}$ to satisfy the non-negativity constraints in the standard LP formulation. In Appendix, we present the method by using general $\boldsymbol{\beta}_\tau$ without the non-negative constraints.*

3. Adaptive Simplex Algorithms with Shared Design Matrix. In this section, we propose an adaptive simplex algorithm to simultaneously solve a large number of quantile re-

gressions with a shared design matrix. Recall the quantile models can be written as

$$Q_Y(\tau|\boldsymbol{X}_i^{(\ell)}, \boldsymbol{Z}_i) = \boldsymbol{X}_i^{(\ell)}\boldsymbol{\beta}_{\ell,\tau} + \boldsymbol{Z}_i\boldsymbol{\alpha}_{\ell,\tau},\ \ell = 1, \cdots, m, \tag{3.1}$$

where Y is the outcome of interest, $\boldsymbol{X}_i^{(\ell)}, \ell = 1, ..., m$ are m segments of the covariate of interest, and $\boldsymbol{Z}_i$ are the confounders. In the m regressions, Y and $\boldsymbol{Z}_i$ are the same, but $\boldsymbol{X}_i^{(\ell)}$ changes across regression models. In big data exploratory analyses, the dimensions of $\boldsymbol{X}_i^{(\ell)}$'s are often small but the number of regressions could be easily over millions as discussed in the two examples in the Introduction.

With the standard simplex algorithms, all the m regressions start their optimization searches from scratch (i.e., starting from Table 2). When m is large, it becomes time-consuming to fit all m models even with the help of parallel computing. Since those regressions share the response and majority of the design matrix, the solution of one regression could naturally be used to warm-start another similar regression. Comparing with starting all the regressions from scratch, a warm-start approach could potentially shorten the optimization path, which in turn leads to the reduction of computation time. We propose an adaptive simplex algorithm that borrows information from shared responses and a design matrix. In particular, we warm-start the m regressions in parallel from regressing Y using $\boldsymbol{Z}$ alone. we elaborate in detail below on how we construct the warm-start Tabular from a previously optimized solution.

Our main idea is to first conduct a regression with only shared data $\boldsymbol{Z}_i$:

$$Q_Y(\tau|Z_i) = \boldsymbol{Z}_i\boldsymbol{\alpha}_\tau.$$

The optimal solution of the problem serves as a warm-start of the problem of interest (3.1). The key motivation is that the objective function of the LP problem does not change by adding new predictors to the model, and we can get a feasible solution immediately after adding new predictors.

We start by solving the quantile regression for the sub-dataset $(Y_i, \boldsymbol{Z}_i)_{i=1,...,n}$, where all m datasets share these values. The corresponding LP problem is

$$\min_{\boldsymbol{\eta}}\ \boldsymbol{c}^\top\boldsymbol{\eta} \quad \text{s.t. } \boldsymbol{A}\boldsymbol{\eta} = \boldsymbol{Y},\ \boldsymbol{\eta} \geq 0, \tag{3.2}$$

where

$$\boldsymbol{A}_{n\times(2q+2n)} = \left(\boldsymbol{Z}_{n\times q}\ -\boldsymbol{Z}_{n\times q}\ \boldsymbol{I}_n\ -\boldsymbol{I}_n\right)$$

is the constraint matrix, $\boldsymbol{\eta} = (\boldsymbol{\alpha}_\tau^{+\top}, \boldsymbol{\alpha}_\tau^{-\top}, \boldsymbol{u}^\top, \boldsymbol{v}^\top)^\top$ is the vector of parameters, and

$$\boldsymbol{c}^\top = (\underbrace{0, 0, \ldots, 0}_{2q\ 0's}, \underbrace{\tau, \tau, \ldots, \tau}_{n\ \tau's}, \underbrace{1-\tau, 1-\tau, \ldots, 1-\tau}_{n\ (1-\tau)'s})$$

is the objective vector.

We construct the tabular at its optimal solution in Table 3 and let B denote the optimal set of basic variables.

	η_1	$\cdots$	η_q	η_{q+1}	$\cdots$	η_{2q}	η_{2q+1}	$\cdots$	η_{2q+2n}	
	α_1^+	$\cdots$	α_q^+	α_1^-	$\cdots$	α_q^-	u_1	$\cdots$	v_n	
η_{B_1}										b_1
$\vdots$										$\vdots$
η_{B_i}		$\boldsymbol{\Gamma}_{\alpha^+}$			$\boldsymbol{\Gamma}_{\alpha^-}$			$\boldsymbol{\Gamma}_{uv}$		b_i
$\vdots$										$\vdots$
η_{B_n}										b_n
r	r_1	$\cdots$	r_q	r_{q+1}	$\cdots$	r_{2q}	r_{2q+1}	$\cdots$	r_{2q+2n}	$-\zeta$

Table 3: Example, $T(B)$ at the optimal solution

To conduct quantile regression using the whole dataset $(Y_i, \boldsymbol{X}_i, \boldsymbol{Z}_i)_{i=1,\ldots,n}$, we have p new variables in the model. The new LP model of the ℓ-th regression is:

$$\min_{\boldsymbol{\eta}^{(\ell)}, \boldsymbol{\eta}} \ \boldsymbol{c}^\top \boldsymbol{\eta} \quad \text{s.t. } \boldsymbol{A}\boldsymbol{\eta} + \boldsymbol{A}^{(\ell)}\boldsymbol{\eta}^{(\ell)} = \boldsymbol{Y},\ \boldsymbol{\eta} \geq 0, \tag{3.3}$$

where

$$\boldsymbol{A}^{(\ell)} = \left(\boldsymbol{X}^{(\ell)}_{n\times p} - \boldsymbol{X}^{(\ell)}_{n\times p}\right),\ \boldsymbol{\eta}^{(\ell)} = (\boldsymbol{\beta}^{+\top}_{\ell,\tau}, \boldsymbol{\beta}^{-\top}_{\ell,\tau})^\top.$$

It is immediately seen that the objective in (3.3) is the same that in (3.2). Thus, we may just let $\boldsymbol{\eta}^{(\ell)} = 0$ be nonbasic variables and update the tableau. Meanwhile, note that the set B is an optimal set of basic variables for the problem (3.2), and it also generates a set of feasible basic variables for the new problem (3.3) when we set $\boldsymbol{\eta}^{(\ell)} = 0$. This provides a natural approach to update the tableau after obtaining an optimal solution to (3.2).

In specific, the algorithm runs in three steps. In Step 1, to ensure all $y_i's$ to be positive, we multiply the rows in $\boldsymbol{A}^{(\ell)}$ and $\boldsymbol{A}^{(\ell+1)}$ where $y_i < 0$ by -1, which results in new $\boldsymbol{A}^{(\ell)}$ and $\boldsymbol{A}^{(\ell+1)}$.

In Step 2, we update the tableau by adding $2p$ more predictors and their corresponding samples into the tableau by putting the matrix below to the right of $\boldsymbol{\Gamma}_{uv}$ part

$$\begin{pmatrix} \{\boldsymbol{A}[B]\}^{-1}\boldsymbol{A}^{(\ell)} \\ -\boldsymbol{c}_B^\top\{\boldsymbol{A}[B]\}^{-1}\boldsymbol{A}^{(\ell)} \end{pmatrix},$$

and we let all new $2p$ variables be nonbasic, where we have that the resulting tableau corresponds to a feasible solution. The new simplex tableau is illustrated in Table 4.

In Step 3, we continue to conduct pivoting steps as described in Section 2.3 until an optimal solution is achieved, and we get the predictors accordingly.

	η_1	$\cdots$	η_q	η_{q+1}	$\cdots$	η_{2q}	η_{2q+1}	$\cdots$	η_{2q+2n}	$\eta_{2q+2n+1}$	$\cdots$	$\eta_{2p+2q+2n}$	
	α_1^+	$\cdots$	α_q^+	α_1^-	$\cdots$	α_q^-	u_1	$\cdots$	v_n	β_1^+	$\cdots$	β_p^-	
η_{B_1}													b_1
$\vdots$													$\vdots$
η_{B_i}	$\boldsymbol{\Gamma}_{\alpha^+}$			$\boldsymbol{\Gamma}_{\alpha^-}$			$\boldsymbol{\Gamma}_{uv}$			$\{\boldsymbol{A}[B]\}^{-1}\boldsymbol{A}^{(\ell)}$			b_i
$\vdots$													$\vdots$
η_{B_n}													b_n
r	r_1	$\cdots$	r_q	r_{q+1}	$\cdots$	r_{2q}	r_{2q+1}	$\cdots$	r_{2q+2n}	$-\boldsymbol{c}_B^\top\{\boldsymbol{A}[B]\}^{-1}\boldsymbol{A}^{(\ell)}$			$-\zeta$

Table 4: Example, Step 2

We point out that in some applications, there is a natural order of the similarity of $\boldsymbol{X}_i^{(\ell)}$'s. For example, in genetic association tests, the neighboring SNPs are often highly correlated, so the chromosome locations naturally approximate the similarity of $\boldsymbol{X}_i^{(\ell)}$. In the Appendix, we provide a similar adaptive algorithm that "alternate" instead of "add" $\boldsymbol{X}_i^{(\ell)}$'s to fully utilize such ordering structures.

3.1. *Cross-quantile Adaptive Simplex Algorithm.* In many applications, it is often necessary to estimate quantile coefficients across different quantile levels. To do this, we needs to estimate the base model

$$Q_Y(\tau|\boldsymbol{Z}_i) = \boldsymbol{Z}_i\boldsymbol{\alpha}_\tau$$

for a set of quantile levels of interest, $0 < \tau_1 < \tau_2 < \cdots < \tau_k < 1$. In this section, we introduce a cross-quantile adaptive simplex algorithm, which aims to reduce the computing time when estimating regression quantiles on a set of pre-specified quantile levels.

We use the similar warm-start concept to speed up the estimation process. When two quantile levels are close to each other, we expect their solutions to be similar, so we can use the solution at one quantile level to warm-start the estimating in next quantile level.

We consider the corresponding LP problem at a quantile level τ_1, where

$$\min_{\boldsymbol{\eta}} \boldsymbol{c}_1^\top\boldsymbol{\eta} \quad \text{s.t. } \boldsymbol{A}\boldsymbol{\eta} = \boldsymbol{Y},\ \boldsymbol{\eta} \geq 0,$$

where

$$\boldsymbol{A}_{n\times(2q+2n)} = \begin{pmatrix}\boldsymbol{Z}_{n\times q} & -\boldsymbol{Z}_{n\times q} & \boldsymbol{I}_n & -\boldsymbol{I}_n\end{pmatrix}$$

is the constraint matrix, $\boldsymbol{\eta} = (\boldsymbol{\alpha}_\tau^{+\top}, \boldsymbol{\alpha}_\tau^{-\top}, \boldsymbol{u}^\top, \boldsymbol{v}^\top)^\top$ is the vector of parameters, and

$$\boldsymbol{c}_1^\top = (\underbrace{0,0,\ldots,0}_{2q\ 0's}, \underbrace{\tau_1,\tau_1,\ldots,\tau_1}_{n\ \tau_1's}, \underbrace{1-\tau_1,1-\tau_1,\ldots,1-\tau_1}_{n\ (1-\tau_1)'s})$$

is the objective vector. Suppose we successfully obtain the optimal set of basic variables for τ_1, denoted by $\mathcal{B}_1$. Thus, the optimal tableau can be expressed as:

$$T(\mathcal{B}_1) = \begin{bmatrix}\{\boldsymbol{A}[\mathcal{B}_1]\}^{-1} & \boldsymbol{0} \\ -\boldsymbol{c}_{1\mathcal{B}_1}^\top\{\boldsymbol{A}[\mathcal{B}_1]\}^{-1} & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{A} & \boldsymbol{Y} \\ \boldsymbol{c}_1^\top & 0\end{bmatrix}.$$

When estimating at a new quantile τ_2, the LP problem can be formulated as follows.

$$\min_{\boldsymbol{\eta}} \ \boldsymbol{c}_2^\top \boldsymbol{\eta} \quad \text{s.t. } \boldsymbol{A}\boldsymbol{\eta} = \boldsymbol{Y},\ \boldsymbol{\eta} \geq 0,$$

where

$$\boldsymbol{c}_2^\top = (\underbrace{0, 0, \ldots, 0}_{2q\ 0's}, \underbrace{\tau_2, \tau_2, \ldots, \tau_2}_{n\ \tau_2's}, \underbrace{1-\tau_2, 1-\tau_2, \ldots, 1-\tau_2}_{n\ (1-\tau_2)'s})$$

All other elements, $\boldsymbol{A}, \boldsymbol{Y}$, remain exactly the same as they were in the earlier LP problem. To fully adapt the optimal set of basic variables from τ_1-LP problem, we need to set $\mathcal{B}_1$ as the initial set of basic variables instead of $\{2p+1, ..., 2p+n\}$. With this change, the tableau can be expressed as:

$$T(\mathcal{B}_1) = \begin{bmatrix} \left\{\boldsymbol{A}[\mathcal{B}_1]\right\}^{-1} & \mathbf{0} \\ -\boldsymbol{c}_{2\mathcal{B}_1}^\top \left\{\boldsymbol{A}[\mathcal{B}_1]\right\}^{-1} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{A} & \boldsymbol{Y} \\ \boldsymbol{c}_2^\top & 0 \end{bmatrix}.$$

We then follow the simplex algorithm outlined in Section 2 to obtain the optimal solution for $\boldsymbol{\alpha}_{\tau_2}$. Starting the estimation at lower quantile levels is preferred, since computing time is usually less as it involves a smaller set of feasible solutions. We hence propose to start the estimate at the lowest quantile level τ_1, and then sequentially updates the quantile estimations, using the solution at τ_k to warm-start the the estimation at τ_{k+1}.

4. Simulation Study.

4.1. *Implementation and comparison algorithms.* In this section, we measure the performance of our algorithm and three other programs: (1) The 'Quantreg' package in R (Koenker, 2013). The package uses a modified simplex method of the Barrodale and Roberts algorithm for *l1*-regression by default. It has been developed for more than 20 years and is the most versatile and mature package in quantile regression analysis. The core part of the package is written in FORTRAN. (2) Standard linear programming introduced in implemented in Matlab by "linprog" function. The estimating algorithm is very similar to the Simplex algorithm used in 'Quantreg' package but is written in a high-level language. Therefore, this method helps to estimate the performance difference between FORTRAN and Matlab. (3) The 'rq_fnm' function in Matlab (Koenker, 2019). This function is initially written by Daniel Morillo & Roger Koenker and uses Frisch-Newton interior point method. The interior point method is expected to be faster than Simplex algorithm for large data. We include this algorithm to make sure that our proposed algorithm has better performance than all popular estimating methods on Matlab.

4.2. *Setting 1 with continuous X mimicing gene expressions.* The first simulated data contains three parts: (1) one million variables (gene expressions) following a multivariate

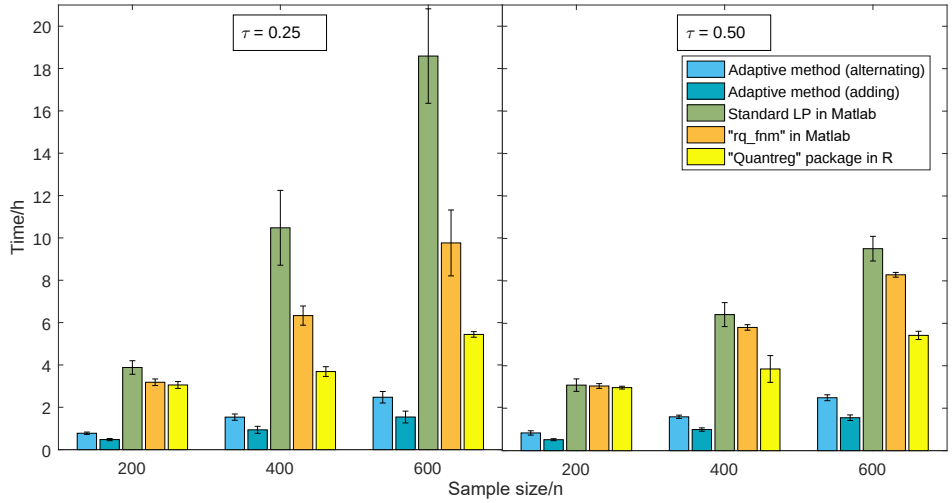


Fig 1: Estimation time with different sample sizes ($q = 60$, $\rho = 0$)
Note: Adaptive method (adding) is presented in the Section 3, while Adaptive method (alternating) is outlined in the Appendix

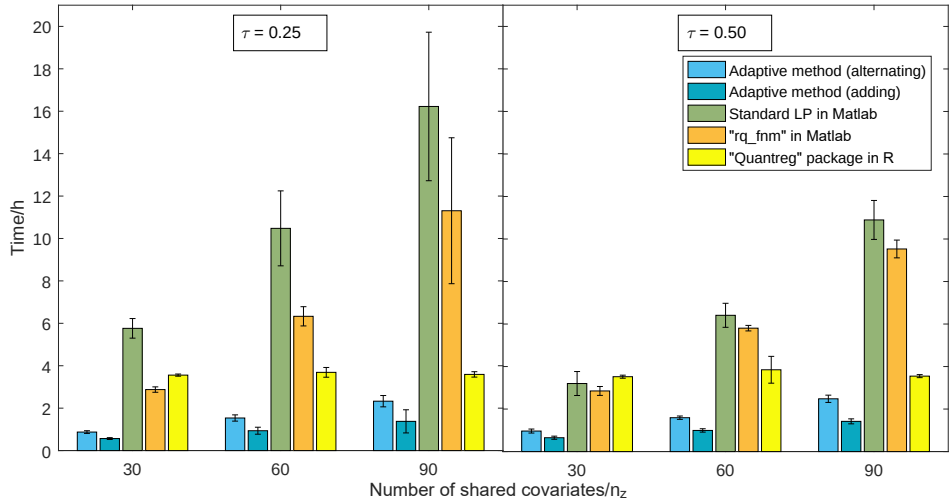


Fig 2: Estimation time with different numbers of covariates ($n = 400$, $\rho = 0$)
Note: Adaptive method (adding) is presented in the Section 3, while Adaptive method (alternating) is outlined in the Appendix

Poisson distribution with mean five and correlation coefficient ρ ($\rho = 0, 0.25, 0.5, 0.75$) between every two variables. This dataset is generated by the method proposed in (Macke et al., 2009); (2) q ($q = 30$, 60, 90) shared covariates following identical independent standard normal distributions; (3) Response variable following a standard normal distribution. The sample sizes are n ($n = 200$, 400, 600).

In Figure 1, we show the total fitting time by altering the sample sizes, with 60 fixed covariates and uncorrelated gene expressions ($\rho = 0$). The total fitting time is the hours used to fit a million regressions with all shared covariates and one gene expression variable each. It does

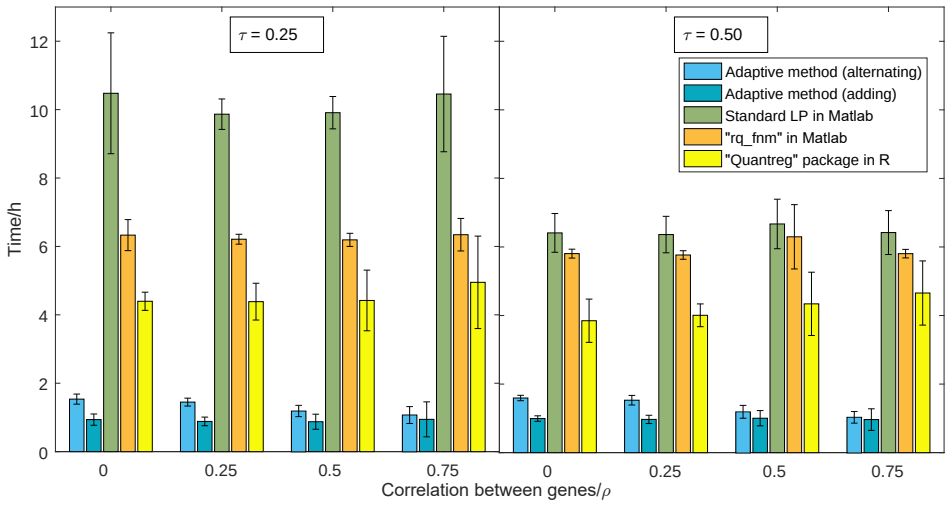


Fig 3: Estimation time with different correlations between genes ($q = 60$, $n = 400$)
Note: Adaptive method (adding) is presented in the Section 3, while Adaptive method (alternating) is outlined in the Appendix

not include the time for data loading. Parallel computing is not used. The total fitting time of our adaptive method and 'rq_fnm' function increases linearly with sample size. While the estimation time of standard linear programming increases faster, the time of 'Quantreg' package increases slower than the linear ra. This feature makes our method more advantageous when the sample size is relatively small to moderate. For example, the adaptive 'adding' method takes 84.4% less time than 'Quantreg' package for a 200-subject dataset, but is only 71.3% less time for a 600-subject dataset. The quantile level (τ) does not strongly affect our algorithm or the 'Quantreg' package.

In Figure 2, we compare the performance of different algorithms when increasing the number of covariates from 30 to 90, while keeping the sample size fixed at $n = 400$ and assuming uncorrelated gene expressions ($\rho = 0$). In most genetic analyses, the number of covariates is often limited. The fitting time of our algorithm increase slowly with respect to the number of covariates, while the other two LP algorithms implemented in Matlab (Standard Simplex and interior-point 'rq_fnm' algorithms) require much more computing time as the number of covariates increases. On the other hand, the 'Quantreg' package in R is not sensitive to the covariate dimension when $q \leq 90$. Therefore, we observed greater efficiency gains while comparing our proposed algorithm to 'Quantreg' in R when $q = 30$ or 60, which are common dimensions of covariates in genetic research. In Figure 3, we modified the gene-gene correlation while keeping the sample size at 400 and the number of covariates at 60. We observed that the three comparison methods and the adaptive Simplex Algorithm (adding) were not impacted by the correlation between genes. On the other hand, our adaptive alter-

nating method presented in the Appendix required less time when gene-gene correlations were present, which was expected.

4.3. *Setting 2 with discrete X mimicing genotypes.* In addition to considering continuous gene expressions as X variables, we also explored a simulation setting where the variables X were discrete genetic variants. For this, we simulated five thousand X variables following a binomial distribution with a minor allele frequency (MAF) that was randomly drawn from a uniform distribution over the range (0, 0.5). The distributions of Z and Y remained the same as before.

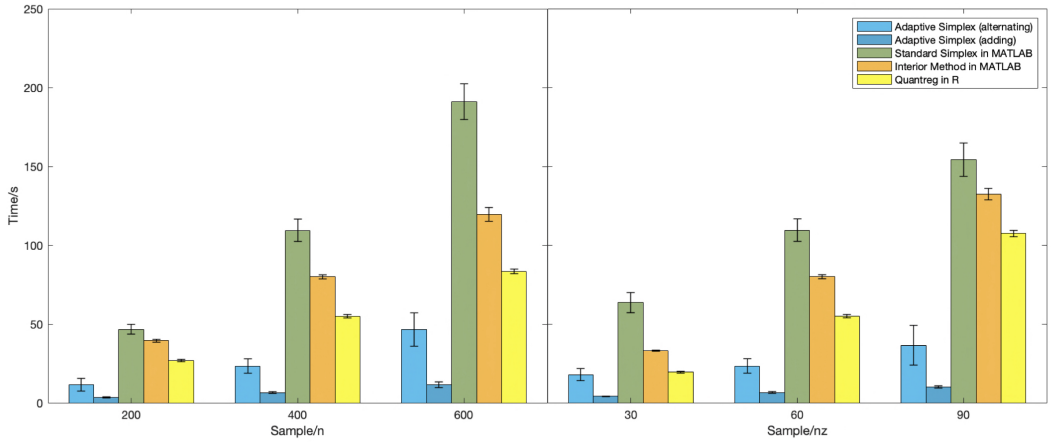


Fig 4: Estimation time for $\tau = 0.25$ between genes ($q = 60/n = 400$)
Note: Adaptive method (adding) is presented in the Section 3, while Adaptive method (alternating) is outlined in the Appendix

In Figure 4, we present the estimation times of the five algorithms at the 0.25 quantile level in the new setting. Same as before, the reported time excludes data loading time and does not utilize parallel computing. The left sub-figure demonstrates how computing time increases as the sample size n grows from 200 to 600, while the right sub-figure shows the computing time's response to an increasing dimension of covariates q. The observed results are consistent; the proposed adaptive algorithms significantly outperform the existing ones in speed. Notably, their computing times are less impacted by increases in both sample size and covariate dimensions.

4.4. *Assessment of Estimation Bias .* We also evaluated how closely the estimated coefficients from our new algorithms align with those from established existing algorithms. We calculated relative bias as the difference between the coefficients estimated by our proposed

Max Relative Bias over Covatiate $\boldsymbol{X}$	Standard Simplex	Interior Point
Adding Method	2.5567e-14	1.4631e-07
Alternating Method	3.9294e-14	1.4631e-07

TABLE 5
Max relative bias among $\tau = 0.25, 0.5$ *sample size*= $200, 400, 600$*, number of covariate* = $30, 60, 90$

adding (alternating) method and the standard simplex (interior point) method, divided by the estimates from the standard simplex (interior point) method. Table 5 shows the maximum average relative bias of β_x across all 18 simulation scenarios (3 sample sizes $\times$ 3 covariate dimensions $\times$ 2 X distributions). The maximum relative bias of β_x when compared to the Simplex algorithm is in the order of 10^{-14}, while it is around 10^{-7} when compared to the Interior Point algorithm. From this, we conclude that the coefficients estimated by the new algorithms are nearly identical to those produced by the existing algorithms.

4.5. *Performance of cross-quantile adaptive simplex.* In Section xx, we introduced the cross-quantile adaptive simplex algorithm. In this subsection we apply it to estimate the base model, $Q_Y(\tau) = Z_z^\beta$, across a quantile level grid ranging from 0.01 to 0.99, using a uniform step size of 0.02, covering 50 levels. We start with the lowest quantile level, and update the estimate sequentially moving up the quantile level. In Table **??**, we compare the average estimation time and the average number of iterations required by the cross-quantile adaptive Simplex algorithm against the standard simplex algorithm (average across 5k replicates and 50 quantile levels each replicate). Overall, the cross-quantile adaptive simplex method demonstrates a tenfold reduction in both computation time and the number of iterations compared to the standard Simplex algorithm.

Moreover, we explored the potential computational efficiency gain as the number of quantile levels rises. We measured the efficiency gain as the ratio of the computing time (and number of iterations) between the standard Simplex and the proposed adaptive algorithm. Higher ratios indicates greater relative efficiency gain. We applied both of the cross-quantile adaptive Simplex algorithm and the standard with various lengths of quantile level sequence ranging from 10 to 100. The cross-quantile adaptive simplex algorithm performs identical to the standard Simplex for the first quantile level but requires much fewer iterations for each subsequent quantile level. Therefore, as illustrated in the Figure 5, the algorithm achieves more efficient gain as the quantile level sequence lengthens.

In summary, the simulation studies demonstrate that our adaptive algorithms have better and unbiased computation efficiencies under various situations. The strongest competitor is the 'Quantreg' package in R. The core part of this package is written in FORTRAN and has been optimized by many contributors in the past two decades. It is widely believed that the performance of most programs implemented in a high-level language such as Matlab can

be greatly improved if they are rewritten using a low-level programming language such as C/FORTRAN. So there is still great potential for the implementation of our algorithms in the future.

	Average computing time per regression (Seconds)		Average number of iterations	
q, n	Standard	Proposed	Standard	Proposed
30, 200	0.0056	0.00071	146.6	15.1
60, 200	0.0086	0.00083	201.1	17.2
90, 200	0.0110	0.00081	233.4	15.7
30, 400	0.0113	0.00156	272.1	33.5
60, 400	0.0249	0.00273	392.9	42.5
90, 400	0.0287	0.00294	476.8	46.3
30, 600	0.0234	0.00285	394.2	50.8
60, 600	0.0387	0.00480	572.1	68.1
90, 600	0.0521	0.00596	705.2	76.3

TABLE 6

Average computing time and number of iterations with different sample size and number of covariates: proposed cross-quantile adaptive simplex vs standard simplex algorithm.

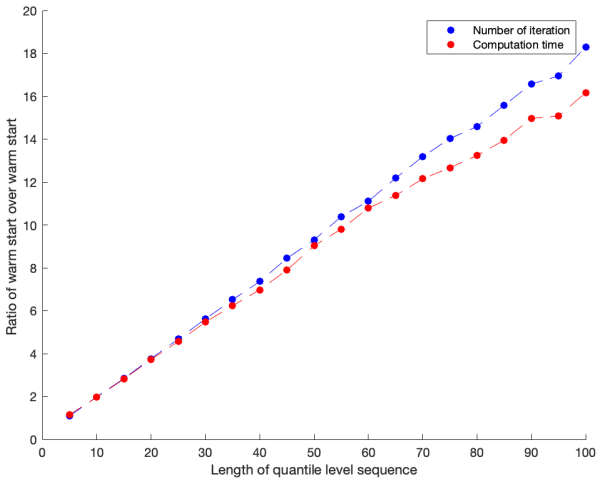


Fig 5: Ratio of average computation time and average number of iterations for no warm start over warm start respective to length of quantile level sequence

5. Application. We apply our adaptive algorithms to analyze data from the Genomics of Drug Sensitivity in Cancer (GDSC) project(Iorio et al., 2016). GDSC provides a large in vitro cancer pharmacogenomics database for investigating the influences of gene expressions on drug responses. It collects over 1000 human tumor cell lines, representing a wide range of common and rare types of adult and childhood cancers. Those cell lines have been extensively profiled with genome-wide gene expression levels across $G = 17,736$ genes. In the

meantime, the GDSC considered $\ell = 266$ anticancer compounds, covering a wide range of cancer drugs, either approved or under-development. Those compounds are applied to a subset of cell lines, respectively, to evaluate their responses. The number of cell lines per drug ranges from 353 to 937, depending on the drug of interest. The cell line responses to those drugs are recorded as the natural log half maximal inhibitory concentration (ln IC50). The goal of the study is to determine whether the expression level of a target gene is associated with drug responses. If so, such a gene is a potential therapeutic marker.

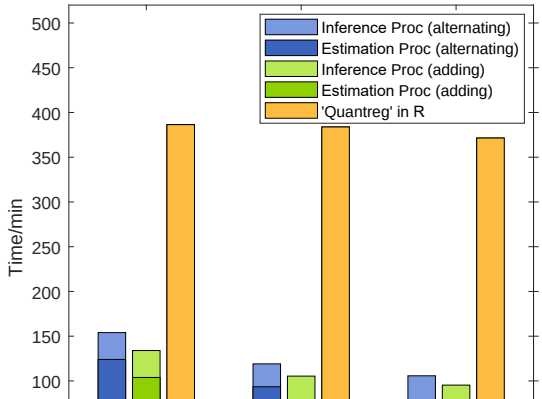
We make the following notions. Let Γ_ℓ be the collection of cell lines that are tested on the ℓth drug. We denote $Y_{i,l}$ as the $\ln IC50$ level of the ith cell line on the ℓth drug, where $i \in \Gamma_\ell$. In the meantime, we denote $x_{i,g}$ is the ith cell lines gene expression on the gth gene, where $g = 1, ...G$. The data also includes 19 covariates, z_i, characterizing the ith cell line. To systemically identify anti-cancer drug associated genes, we build the following linear quantile models for each (ℓ, g) gene-drug pairs

$$Y_{i,\ell} = X_{i,g}\beta_{\ell g,\tau} + Z_i^T \alpha_{\ell g,\tau} + e_{i,\ell g} \tag{5.1}$$

where the conditional τth quantile of $e_{i,\ell g}$ given X and Z is zero, and $\beta_{\ell g,\tau} \neq 0$ for any given $\tau \in (0, 1)$. The conditional quantile models complement the mean regressions, to examine how the expression level of a target gene impacts the distribution of the drug response $\ln IC50$. Researchers are particularly interested in the lower quantiles of $\ln IC50$, which represent high sensitivity at a low dose and are of great biological importance. We performed hypothesis testing $H_0 : \beta_{\ell g,\tau} = 0$ to determine whether gene g is significantly associated with the τ-th quantile of $\ln IC50$.

We considered three quantile levels $\tau = 0.10, 0.25, 0.50$ as in Williams (2012). At each quantile level, we performed a total number of $266 \times 17736 = 4.7M$ hypothesis tests, using Wald-type test. We used the 'kernel' method (Powell, 1991) to estimate the density functions in the asymptotic variance-covariance matrix. The 'kernel' works well for both i.i.d. or weakly dependent data (Kato, 2012). The test is implemented in 'summary.rq' function from the R package 'Quantreg'. Then we compare our proposed algorithm with 'Quantreg' under the same condition. Though there is not a specified order of gene expressions in the dataset, we will run both 'adding' and 'alternating' methods just for illustration purposes. The resulting CPU times (excluding the time required to load the data) are reported in Figure 6.

In Figure 6, the total running time of our adaptive algorithm contains two parts: the darker blue/green bars are the runtime for estimating coefficients, while the stacked lighter blue/green bars are the runtime for calculating



the test statistics. Runtime for estimating coefficients in both 'adding' method and 'alternating' method decreases when the quantile level goes from center ($\tau = 0.50$) to more extreme values ($\tau = 0.10$). 'Adding' method is slightly better than 'alternating' method, since there is no natural order to bring more advantages to alternate variables. The adaptive methods consume 65%–75% less time than the same 'kernel' test in 'Quantreg' package. Furthermore, it is easy to use parallel computing on the 'adding' method, which makes it possible to test all 4.7M gene expressions in 15 minutes on a specified quantile level on a single i9-7900X CPU with 32GB memory.

6. Discussion. In this article, we present a modified version of the simplex algorithm in linear programming for a set of quantile regressions that share a large partition of design matrices. Our algorithm reduces the fitting time by taking advantage of fitting results from previous regressions. Such a warm start can reduce $90\% - 95\%$ iterations compared with standard linear programming. The proposed algorithm fits a partial regression only with a shared design matrix first and adds the additional covariates into the design matrix to update the estimating result. When the unique partition of design matrices in two adjacent regressions has great similarity, one could fit a complete regression first and change the unique variables from the previous estimating result. Both our simulation and real data results show that our adaptive algorithms take significantly less time than regular linear programming. We expect these algorithms to have better performance if implemented in a low-level programming language. The utility of the adaptive algorithm is not limited to association testing. It can also be used in many fields that require fitting large amounts of similar regressions, such as quantile random forests. In addition, the idea that we think up in this paper shows a possible way to optimize the estimating procedure in quantile analysis.

APPENDIX A: SEQUENTIAL ADAPTIVE SIMPLEX ALGORITHM (SA-SIMPLEX)

If $\boldsymbol{X}_i^{(\ell)}$ and $\boldsymbol{X}_i^{(\ell')}$ are similar, it is reasonable to expect that the two optimal solutions of the ℓ-th and ℓ'-th regressions are close. In particular, after obtaining an optimal solution to one problem, we aim to use the solution to obtain a better initial solution or tableau for the next problem.

Briefly speaking, fixing the set of basic variables as the optimal basic variables of the previous problem, we update the tableau using the new data set. We first update the nonbasic variables for the new problem, and then update the solution, including basic variables, by introducing an auxiliary LP problem. Finally, we conduct simplex pivoting steps to achieve the optimal solution.

The linear programming problem for the ℓ-th regression in (3.1) can be written as

$$\min_{\boldsymbol{\eta}} \ \boldsymbol{c}^\top \boldsymbol{\eta}, \quad \text{s.t. } \boldsymbol{A}^{(\ell)}\boldsymbol{\eta} = \boldsymbol{Y},\ \boldsymbol{\eta} \geq 0,\ \ell = 1, \ldots, m, \tag{A.1}$$

where

$$\boldsymbol{A}^{(\ell)}_{n\times(2p+2q+2n)} = \left(\boldsymbol{X}^{(\ell)}_{n\times p} - \boldsymbol{X}^{(\ell)}_{n\times p}\ \boldsymbol{Z}_{n\times q} - \boldsymbol{Z}_{n\times q}\ \boldsymbol{I}_n - \boldsymbol{I}_n\right)$$

is the constrain matrix with $\boldsymbol{X}^{(\ell)}$ and $\boldsymbol{Z}$, $\boldsymbol{\eta} = (\boldsymbol{\beta}^{+\top}_{\ell,\tau}, \boldsymbol{\beta}^{-\top}_{\ell,\tau}, \boldsymbol{\alpha}^{+}_{\ell,\tau}, \boldsymbol{\alpha}^{-}_{\ell,\tau}, \boldsymbol{u}^\top, \boldsymbol{v}^\top)^\top$ is the vector of parameters, and

$$\boldsymbol{c}^\top = (\underbrace{0, 0, \ldots, 0}_{2p+2q\ 0's}, \underbrace{\tau, \tau, \ldots, \tau}_{n\ \tau's}, \underbrace{1-\tau, 1-\tau, \ldots, 1-\tau}_{n\ (1-\tau)'s})$$

is the objective vector.

Note that for the m regressions/optimizations, they only differ in the first $2p$ columns of the constraint matrix $\boldsymbol{A}^{(\ell)}$. Following the framework of the simplex method outlined in the preceding section, we construct the tabular of the ℓ-th regression at its optimal solution.

	η_1 $\cdots$ η_p	η_{p+1} $\cdots$ η_{2p}	η_{2p+1} $\cdots$ $\cdots$ η_{2p+2q}	$\eta_{2p+2q+1}$ $\cdots$ $\eta_{2p+2q+2n}$	
	β_1^+ $\cdots$ β_p^+	β_1^- $\cdots$ β_p^-	α_1^+ $\cdots$ $\cdots$ α_q^-	u_1 $\cdots$ v_n	
η_{B_1}					b_1
$\vdots$					$\vdots$
η_{B_i}	$\boldsymbol{\Gamma}_{\beta^+}$	$\boldsymbol{\Gamma}_{\beta^-}$	$\boldsymbol{\Gamma}_{\alpha^+}$ $\boldsymbol{\Gamma}_{\alpha^-}$	$\boldsymbol{\Gamma}_{uv}$	b_i
$\vdots$					$\vdots$
η_{B_n}					b_n
r	r_1 $\cdots$ r_p	r_{p+1} $\cdots$ r_{2p}	r_{2p+1} $\cdots$ $\cdots$ r_{2p+2q}	$r_{2p+2q+1}$ $\cdots$ $r_{2p+2q+2n}$	$-\zeta$

Table 7: Example, $T(B^{(\ell)})$ at the optimal solution

Suppose that $\boldsymbol{X}^{(\ell+1)}$ is the nearest neighbor to $\boldsymbol{X}^{(\ell)}$, we outline an algorithm to find an efficient warm-start tabular of the $(\ell+1)$-th regression from the optimized tabular of the ℓ-th regression as shown in Table 7.

We first introduce some notations. Let $\mathcal{K}_B^{(\ell)} = \{1, ..., 2p\} \cap B^{(\ell)}$, and $\mathcal{K}_D^{(\ell)} = \{1, ..., 2p\} \cap D^{(\ell)}$. Here $\mathcal{K}_B^{(\ell)}$ and $\mathcal{K}_D^{(\ell)}$ represent the basic and non-basic columns, respectively, among the first $2p$ columns of $\boldsymbol{\Gamma}$ in Table 7. Consequently, we let $\boldsymbol{A}^{(\ell+1)}[\mathcal{K}_B^{(\ell)}]$, $\boldsymbol{A}^{(\ell+1)}[\mathcal{K}_D^{(\ell)}]$, $T[\mathcal{K}_B^{(\ell)}]$, $T[\mathcal{K}_{\mathcal{D}}^{(\ell)}]$ denote the corresponding columns in $\boldsymbol{A}^{(\ell+1)}$ and current tableau T, and let $\boldsymbol{\eta}_{\mathcal{K}_B^{(\ell)}}$ be the corresponding elements in $\boldsymbol{\eta}$. We point out that the sizes of $\mathcal{K}_B^{(\ell)}$ and $\mathcal{K}_D^{(\ell)}$ are both p.

We then outline our algorithm below. The algorithm runs in four steps. In Step 1, we ensure all y_i's to be positive. In Step 2, we update the nonbasic variables using the new dataset. In Step 3, we update the basic variables by introducing an auxiliary LP problem. In Step 4, we update the solution by the simplex method to achieve an optimal solution to the LP problem.

Step 1: To ensure all $y_i's$ to be positive, we multiply the rows in $\boldsymbol{A}^{(\ell)}$ and $\boldsymbol{A}^{(\ell+1)}$ where $y_i < 0$ by -1, which results in new $\boldsymbol{A}^{(\ell)}$ and $\boldsymbol{A}^{(\ell+1)}$.

Step 2: This step updates the nonbasic variables using the new dataset. We incorporate $\boldsymbol{A}^{(\ell+1)}[\mathcal{K}_D^{(\ell)}]$ to update the corresponding columns in the tableau $T[\mathcal{K}_D^{(\ell)}]$ in this step. Since the corresponding columns do not overlap with any basic variables of the optimal solution to the previous problem, $T[\mathcal{K}_B^{(l)}]$ remains the same at this step. By (2.5), we only update corresponding columns in $\boldsymbol{\Gamma}$ in Table 7. Specifically, we replace $T[\mathcal{K}_D^{(\ell)}]$ by $\left\{\boldsymbol{A}^{(\ell)}[B^{(\ell)}]\right\}^{-1}\boldsymbol{A}^{(\ell+1)}[\mathcal{K}_D^{(\ell)}]$.

By the example mentioned previously, without loss of generality, we assume $\{\beta_1^+, ..., \beta_p^+\}$ are nonbasic variables, and $\{\beta_1^-, ..., \beta_p^-\}$ are basic variables in the previous model (A.1) (i.e. $\mathcal{K}_D = \{1, \ldots, p\}$, $\mathcal{K}_B = \{p+1, \ldots, 2p\}$). Table 8 illustrates the updated tableau after Step 2.

	η_1	$\cdots$	η_p	η_{p+1}	$\cdots$	η_{2p}	η_{2p+1}	$\cdots$	$\cdots$	η_{2p+2q}	$\eta_{2p+2q+1}$	$\cdots$	$\eta_{2p+2q+2n}$	
	β_1^+	$\cdots$	β_p^+	β_1^-	$\cdots$	β_p^-	α_1^+	$\cdots$	$\cdots$	α_q^-	u_1	$\cdots$	v_n	
η_{B_1}														b_1
$\vdots$	$\left\{\boldsymbol{A}^{(\ell)}[B^{(\ell)}]\right\}^{-1}$			$\boldsymbol{\Gamma}_{\beta^-}$			$\boldsymbol{\Gamma}_{\alpha^+}$			$\boldsymbol{\Gamma}_{\alpha^-}$	$\boldsymbol{\Gamma}_{uv}$			$\vdots$
η_{B_i}	$\boldsymbol{A}^{(\ell+1)}[\mathcal{K}_D^{(\ell)}]$													b_i
$\vdots$														$\vdots$
η_{B_n}														b_n
r	r_1	$\cdots$	r_p	r_{p+1}	$\cdots$	r_{2p}	r_{2p+1}	$\cdots$	$\cdots$	r_{2p+2q}	$r_{2p+2q+1}$	$\cdots$	$r_{2p+2q+2n}$	$-\zeta$

Table 8: Example, Step 2

Step 3: This step introduces an auxiliary problem to update the solution, including basic variables using the new dataset. In this step, we incorporate $\boldsymbol{A}^{(\ell+1)}[\mathcal{K}_B^{(\ell)}]$ to update the columns corresponding to basic variables in the tableau. However, updating basic variables is more challenging than updating nonbasic variables as in the previous step because we need to maintain the equalities in (2.5). To address the challenge, we first duplicate the basic variables and the corresponding columns of the $\boldsymbol{\Gamma}$ part (i.e., $T[\mathcal{K}_B^{(\ell)}]$ part). We then let these duplicated variables still be basic, and let the original variables be nonbasic. Then, we update $T[\mathcal{K}_B^{(\ell)}]$ by new data. Since the original basic variables are now nonbasic, after updating $T[\mathcal{K}_B^{(\ell)}]$, the equalities in (2.5) still hold. Then, we introduce an auxiliary LP problem, of which, at the optimal solution, all duplicated variables become zeros and thus nonbasic. In the last step, we eliminate the duplicated variables, and take the corresponding solution as a warm start to achieve the optimal solution by running the simplex method.

In particular, we add p new columns to the right of the tableau in Table 8 by duplicating $T[\mathcal{K}_B^{(\ell)}]$ and denote these variables as $\boldsymbol{\eta}^*_{\mathcal{K}_B^{(\ell)}}$, and we replace $T[\mathcal{K}_B^{(\ell)}]$ by $\left\{\boldsymbol{A}^{(\ell)}[B^{(\ell)}]\right\}^{-1}\boldsymbol{A}^{(\ell+1)}[\mathcal{K}_B^{(\ell)}]$. As we discussed above, we let $\boldsymbol{\eta}^*_{\mathcal{K}_B^{(\ell)}}$ be basic and let original basic variables be nonbasic. Thus, we replace the last row of the tableau ($\boldsymbol{r}$ and $-\zeta$ parts) by letting

$$\boldsymbol{r} = \big(\boldsymbol{c}^\top - \boldsymbol{c}^\top_{B^{(\ell)}}\left\{\boldsymbol{A}^{(\ell)}[B^{(\ell)}]\right\}^{-1}\boldsymbol{A}^{(\ell+1)}, \underbrace{0,\ldots,0}_{p\ 0's}\big),$$

$$-\zeta = -\boldsymbol{c}^\top_{B^{(\ell)}}\left\{\boldsymbol{A}^{(\ell)}[B^{(\ell)}]\right\}^{-1}|\boldsymbol{Y}|,$$

where $|\boldsymbol{Y}|$ denotes the vector of element-wise absolute values of $\boldsymbol{Y}$.

Next, we convert $\boldsymbol{\eta}^*_{\mathcal{K}_B^{(\ell)}}$ into nonbasic variables (and thus p variables in the first $2p+2q+2n$ columns of the tableau becomes basic). After that, we get an updated feasible solution to the new problem, and we may simply eliminate the columns corresponding to $\boldsymbol{\eta}^*_{\mathcal{K}_B^{(\ell)}}$ in the tableau. In particular, in order to make variables $\boldsymbol{\eta}^*_{\mathcal{K}_B^{(\ell)}}$ nonbasic, we consider the following LP problem, which minimizes the sum of variables $\boldsymbol{\eta}^*_{\mathcal{K}_B^{(\ell)}}$:

$$\text{(A.2)}\qquad \min_{\boldsymbol{\eta}^*_{\mathcal{K}_B^{(\ell)}},\boldsymbol{\eta}} \mathbf{1}_k^\top\boldsymbol{\eta}^*_{\mathcal{K}_B^{(\ell)}},\quad \text{s.t. } \left(\boldsymbol{A}^{(\ell+1)}, \boldsymbol{A}^{(\ell)}[\mathcal{K}_B^{(\ell)}]\right)\begin{pmatrix}\boldsymbol{\eta}\\ \boldsymbol{\eta}^*_{\mathcal{K}_B^{(\ell)}}\end{pmatrix} = \boldsymbol{Y},\ \boldsymbol{\eta}\geq 0,\ \boldsymbol{\eta}^*_{\mathcal{K}_B^{(\ell)}}\geq 0.$$

Since $\boldsymbol{\eta}^*_{\mathcal{K}_B^{(\ell)}}$ is non-negative and there exists a feasible solution $\boldsymbol{\eta}$ such that $\boldsymbol{A}^{(\ell+1)}\boldsymbol{\eta} = \boldsymbol{Y}$, we have that $\boldsymbol{\eta}^*_{\mathcal{K}_B^{(\ell)}} = 0$ at the optimal solution. Thus, at the optimal solution to the problem above, variables $\boldsymbol{\eta}^*_{\mathcal{K}_B^{(\ell)}}$ become nonbasic.

To include the auxiliary problem (A.2) into the tableau, we add a new row $\boldsymbol{r}^*$ to the bottom of the tableau as shown in Table 9:

$$\text{(A.3)}\qquad \boldsymbol{r}^* = \boldsymbol{c}^{*\top} - \boldsymbol{c}^{*\top}_{B^*}\left\{\boldsymbol{A}^{(\ell)}[B^{(\ell)}]\right\}^{-1}\left(\boldsymbol{A}^{(\ell+1)}, \boldsymbol{A}^{(\ell)}[\mathcal{K}_B^{(\ell)}]\right),$$

where $\boldsymbol{c}^{*\top} = (\underbrace{0,0,\ldots,0}_{2p+2q+2n\ 0's}, \underbrace{1,1,\ldots,1}_{p\ 1's})$,

$$B^* = B^{(\ell)}\backslash\mathcal{K}_B^{(\ell)} \cup \{\underbrace{(2p+2q+2n)+1,\ldots,(2p+2q+2n)+p}_{p\text{ variables}}\}.$$

After this step, following the example we discussed in Step 2, the tableau now becomes

	η_1 $\cdots$ η_p	η_{p+1} $\cdots$ η_{2p}	η_{2p+1} $\cdots$ $\cdots$	η_{2p+2q} $\cdots$ $\eta_{2p+2q+2n}$	η^*_{p+1} $\cdots$ η^*_{2p}	
	β^+_1 $\cdots$ β^+_p	β^-_1 $\cdots$ β^-_p	α^+_1 $\cdots$ $\cdots$	α^-_q $\cdots$ $\cdots$	β^{*-}_1 $\cdots$ β^{*-}_p	
η_{B_1}						b_1
$\vdots$	$\{\boldsymbol{A}^{(\ell)}[B^{(\ell)}]\}^{-1}$	$\{\boldsymbol{A}^{(\ell)}[B^{(\ell)}]\}^{-1}$	$\boldsymbol{\Gamma}_{\alpha^+}$	$\boldsymbol{\Gamma}_{\alpha^-}$ $\boldsymbol{\Gamma}_{uv}$	$\boldsymbol{\Gamma}_{\beta^-}$	$\vdots$
η_{B_i}	$\boldsymbol{A}^{(\ell+1)}[\mathcal{K}^{(\ell)}_D]$	$\boldsymbol{A}^{(\ell+1)}[\mathcal{K}^{(\ell)}_B]$				b_i
$\vdots$						$\vdots$
η_{B_n}						b_n
r	$\boldsymbol{c}^\top - \boldsymbol{c}^\top_{B^{(\ell)}}\{\boldsymbol{A}^{(\ell)}[B^{(\ell)}]\}^{-1}\boldsymbol{A}^{(\ell+1)}$				0 $\cdots$ 0	$-\zeta$
r^*	$\boldsymbol{c}^{*\top} - \boldsymbol{c}^{*\top}_{B^*}\{\boldsymbol{A}^{(\ell)}[B^{(\ell)}]\}^{-1}\left(\boldsymbol{A}^{(\ell+1)}, \boldsymbol{A}^{(\ell)}[\mathcal{K}^{(\ell)}_B]\right)$					

Table 9: Example, Step 3

where $-\zeta = -\boldsymbol{c}^\top_{B^{(\ell)}}\{\boldsymbol{A}^{(\ell)}[B^{(\ell)}]\}^{-1}|\boldsymbol{Y}|$.

Finally, we solve sub-problem (A.2) while conducting the pivot for the whole tableau (including both of the two rows at the bottom). We start from the basic variables index set B^* in (A.3), and conduct pivot based on the last row ($\boldsymbol{r}^*$), where the pivoting rule is described in Section 2.3. After getting the solution, $\boldsymbol{\eta}^*_{\mathcal{K}^{(\ell)}_B}$ become nonbasic variables, so we delete $\boldsymbol{\eta}^*_{\mathcal{K}^{(\ell)}_B}$ and $\boldsymbol{r}^*$ from the table.

Step 4: We achieve an optimal solution to the problem of interest in this step. We start to pivot from the tableau generated after Step 3 until reaching an optimal solution. Consequently, given an optimal solution $\hat{\boldsymbol{\eta}} = (\boldsymbol{\beta}^{+\top}_{\ell,\tau}, \boldsymbol{\beta}^{-\top}_{\ell,\tau}, \boldsymbol{\alpha}^+_{\ell,\tau}, \boldsymbol{\alpha}^-_{\ell,\tau}, \boldsymbol{u}^\top, \boldsymbol{v}^\top)^\top$, we obtain estimators $\hat{\boldsymbol{\beta}}_{\ell,\tau} = \boldsymbol{\beta}^+_{\ell,\tau} - \boldsymbol{\beta}^-_{\ell,\tau}, \hat{\boldsymbol{\alpha}}_{\ell,\tau} = \boldsymbol{\alpha}^+_{\ell,\tau} - \boldsymbol{\alpha}^-_{\ell,\tau}$.

REFERENCES

BARRODALE, I. and ROBERTS, F. (1974). Solution of an overdetermined system of equations in the l_1 norm. *Communications of the ACM* **17** 319–320.

BERTSIMAS, D. and TSITSIKLIS, J. N. (1997). *Introduction to linear optimization* **6**. Athena Scientific Belmont, MA.

CADE, B. S. and NOON, B. R. (2003). A gentle introduction to quantile regression for ecologists. *Frontiers in Ecology and the Environment* **1** 412–420.

FRISCH, R. (1956). La résolution des problèmes de programme linéaire par la méthode du potentiel logarithmique. *Cahiers du Seminaire D'Econometrie* **4** 7–23.

IORIO, F., KNIJNENBURG, T. A., VIS, D. J., BIGNELL, G. R., MENDEN, M. P., SCHUBERT, M., ABEN, N., GONÇALVES, E., BARTHORPE, S., LIGHTFOOT, H. et al. (2016). A landscape of pharmacogenomic interactions in cancer. *Cell* **166** 740–754.

KATO, K. (2012). Asymptotic normality of Powells kernel estimator. *Annals of the Institute of Statistical Mathematics* **64** 255–273.

KOENKER, R. (2013). Quantreg: Quantile Regression. R package version 5.05. *R Foundation for Statistical Computing: Vienna*. Available at: http://CRAN.R-project.org/package=quantreg.

KOENKER, R. (2019). Quantile Regression. http://www.econ.uiuc.edu/~roger/research/rq/rq.html. [Online; accessed 27-May-2019].

KOENKER, R. and BASSETT, G. J. (1978). Regression quantiles. *Econometrica: Journal of the Econometric Society* **46** 33–50.

KOENKER, R. W. and D'OREY, V. (1987). Algorithm AS 229: Computing regression quantiles. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* **36** 383–393.

KOENKER, R. and D'OREY, V. (1994). Remark AS R92: A remark on algorithm AS 229: Computing dual regression quantiles and regression rank scores. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* **43** 410–414.

KORSUNSKY, I., MILLARD, N., FAN, J., SLOWIKOWSKI, K., ZHANG, F., WEI, K., BAGLAENKO, Y., BRENNER, M., LOH, P.-R. and RAYCHAUDHURI, S. (2019). Fast, sensitive and accurate integration of single-cell data with Harmony. *Nature methods* **16** 1289–1296.

LING, W., ZHAO, N., PLANTINGA, A. M., LAUNER, L., FODOR, A. A., MEYER, K. A. and WU, M. (2021). A zero-inflated quantile approach (ZINQ) for powerful and robust microbiome data analysis.

LIPPERT, C., LISTGARTEN, J., LIU, Y., KADIE, C. M., DAVIDSON, R. I. and HECKERMAN, D. (2011). FaST linear mixed models for genome-wide association studies. *Nature methods* **8** 833–835.

MACKE, J. H., BERENS, P., ECKER, A. S., TOLIAS, A. S. and BETHGE, M. (2009). Generating spike trains with specified correlation coefficients. *Neural Computation* **21** 397–423.

MBATCHOU, J., BARNARD, L., BACKMAN, J., MARCKETTA, A., KOSMICKI, J. A., ZIYATDINOV, A., BENNER, C., ODUSHLAINE, C., BARBER, M., BOUTKOV, B. et al. (2021). Computationally efficient whole-genome regression for quantitative and binary traits. *Nature genetics* **53** 1097–1103.

MOSTELLER, F. and TUKEY, J. W. (1977). *Data Analysis and Regression: A Second Course in Statistics*. *Addison-Wesley series in behavioral science*. Addison-Wesley Publishing Company.

PORTNOY, S., KOENKER, R. et al. (1997). The Gaussian hare and the Laplacian tortoise: computability of squared-error versus absolute-error estimators. *Statistical Science* **12** 279–300.

POWELL, J. L. (1991). Estimation of monotonic regression models under quantile restrictions. In *Nonparametric and Semiparametric Methods in Econometrics and Statistics* 14, 357–384. Cambridge University Press.

SHABALIN, A. A. (2012). Matrix eQTL: ultra fast eQTL analysis via large matrix operations. *Bioinformatics* **28** 1353–1358.

SONG, X., LI, G., ZHOU, Z., WANG, X., IONITA-LAZA, I. and WEI, Y. (2017). QRank: a novel quantile regression tool for eQTL discovery. *Bioinformatics* **33** 2123–2130.

VANDERBEI, R. J. et al. (2015). *Linear Programming: Foundations and Extensions*. Springer.

WANG, T., IONITA-LAZA, I. and WEI, Y. (2022). Integrated Quantile RAnk Test (iQRAT) for gene-level associations. *The Annals of Applied Statistics* **16** 1423–1444.

WILLIAMS, P. T. (2012). Quantile-specific penetrance of genes affecting lipoproteins, adiposity and height. *PloS One* **7** e28764.

ZHANG, W., WEI, Y., ZHANG, D. and XU, E. Y. (2020). ZIAQ: a quantile regression method for differential expression analysis of single-cell RNA-seq data. *Bioinformatics* **36** 3124–3130.