KYBERNETIKA — VOLUME *52* (2016), NUMBER 1, PAGES 28–51

DIRECTIONAL QUANTILE REGRESSION IN OCTAVE (AND MATLAB)

Pavel Boček and Miroslav Šiman

Although many words have been written about two recent directional (regression) quantile concepts, their applications, and the algorithms for computing associated (regression) quantile regions, their software implementation is still not widely available, which, of course, severely hinders the dissemination of both methods. Wanting to partly fill in the gap here, we provide all the codes needed for computing and plotting the multivariate (regression) quantile regions in Octave and MATLAB, describe their use in detail, and explain their output with a few carefully designed examples.

Keywords: quantile regression, multivariate quantile, regression quantile, directional quantile, halfspace depth, regression depth, depth contour, Octave, MATLAB

Classification: 62-04, 65C60, 62H05, 62J99

1. INTRODUCTION

Two recent concepts of multivariate (regression) quantiles define directional (regression) quantiles as hyperplanes and (regression) quantile regions as intersections of certain induced directional (regression) quantile halfspaces. Those concepts have been elaborated in a long series of articles including [8, 9, 10, 16, 17, 18], and [23], and they have already found some practical applications in [15] and [24]; see also [4, 13], and [14] for other articles dealing with similar ideas. As the two concepts generalize the standard quantile regression of [12] and [11] to the multiple-response setup, they are very likely to once become as popular and widespread as their single-response predecessor.

Our primary aim is to equip practitioners with a collection of m-files (the moQuantile toolbox is available online at http://simu0292.utia.cas.cz/modQR) for Octave [7] and MATLAB [21] with the free SeDuMi [22] optimization toolbox so that they could employ directional multiple-output quantile regression on a daily basis. We did not have to start our work from scratch as [17] and [18] had already resolved the computational issues involved in both methods by means of parametric programming and experimentally implemented their algorithms in MATLAB. Their provisional codes are not officially available for download any more as they ceased to work in later versions of MATLAB (and they never worked in Octave). They nevertheless served as a starting point for our project. We have reworked them, corrected them, improved them, simplified them,

DOI: 10.14736/kyb-2016-1-0028

annotated them, adjusted them to the latest versions of Octave and MATLAB, supplemented them with illustrative demo examples as well as with the tools for processing their output, and now we describe them and provide them here with a topical tutorial to answer the growing demand of our research community. Furthermore, we have already translated them to R ([19]); see [2].

The single-response quantile regression has already become a widespread research tool and as such it has been implemented in common statistical and econometric software by means of general or special linear programming algorithms; see [11]. Nevertheless, our toolbox has not much in common with all those software solutions as it avoids the single-response case (where the parametric optimization using directions as parameters would make little sense) and employs it only internally for initializing the computation. As far as we know, the toolbox has no close software competitors in the general regression case with multivariate responses except for its R translation mentioned above.

Before we focus our attention on the codes, let us briefly review the properties of both directional (regression) quantile methods revealed in the aforementioned articles. Although the methods are conceptually different, both are motivated by the standard single-output quantile regression.

2. THEORY

For the sake of simplicity, we will further consider only the empirical case with n m-dimensional responses $\boldsymbol{Y}_i$ associated with p-dimensional regressors $\boldsymbol{X}_i = (1, \boldsymbol{Z}_i^\top)^\top$, $i = 1, \ldots, n > m + p - 1$, and with positive weights $w_i(\boldsymbol{Y}_i, \boldsymbol{Z}_i) > 0,\ i = 1, \ldots, n$. The location case (with $p = 1$) thus corresponds to empty vectors $\boldsymbol{Z}_i$, $i = 1, \ldots, n$. We will also assume that the random sample $(\boldsymbol{Y}_i^\top, \boldsymbol{Z}_i^\top)^\top \in \mathbb{R}^{m+p-1}, i = 1, \ldots, n$, comes from a continuous distribution. Then all the sample directional quantiles are uniquely defined and the algorithms of [17] and [18] should theoretically work fine with probability one for all but a finite number of quantile levels τ's. All the rare troublesome cases will be ignored in our exposition. In the location case with unit weights, they unfortunately include the popular choice $\tau = i/n,\ i = 0, 1, 2, \ldots, n$, as well (because these quantile levels would then lead to infinitely many directional τ-quantiles).

Both methods define the directional (regression) quantile hyperplane

$$\pi_{\tau\boldsymbol{u}} = \left\{(\boldsymbol{y}^\top, \boldsymbol{z}^\top)^\top \in \mathbb{R}^{m+p-1} : \boldsymbol{b}_{\tau\boldsymbol{u}}^\top \boldsymbol{y} - \boldsymbol{a}_{\tau\boldsymbol{u}}^\top \boldsymbol{x} = 0,\ \boldsymbol{x} = (1, \boldsymbol{z}^\top)^\top\right\}$$

and its upper directional (regression) quantile halfspace

$$\mathcal{H}_{\tau\boldsymbol{u}}^{+} = \left\{(\boldsymbol{y}^\top, \boldsymbol{z}^\top)^\top \in \mathbb{R}^{m+p-1} : \boldsymbol{b}_{\tau\boldsymbol{u}}^\top \boldsymbol{y} - \boldsymbol{a}_{\tau\boldsymbol{u}}^\top \boldsymbol{x} \geq 0,\ \boldsymbol{x} = (1, \boldsymbol{z}^\top)^\top\right\}$$

for any direction $\boldsymbol{u} \in \mathbb{R}^m \setminus \{\boldsymbol{0}\}$ and for any quantile level $\tau \in (0, 1)$ by means of (the solution to) the optimization problem

$$(\boldsymbol{a}_{\tau\boldsymbol{u}}^\top, \boldsymbol{b}_{\tau\boldsymbol{u}}^\top)^\top = \operatorname*{argmin}_{(\boldsymbol{a}^\top, \boldsymbol{b}^\top)^\top} \sum_{i=1}^{n} w_i \rho_\tau(\boldsymbol{b}^\top \boldsymbol{Y}_i - \boldsymbol{a}^\top \boldsymbol{X}_i)$$

subject to a method-specific constraint ($\boldsymbol{b}^\top \boldsymbol{u} = 1$ for Method 1 of [9] and $\boldsymbol{b} = \boldsymbol{u}$ for Method 2 of [16]) where $\rho_\tau(x) = x\big(\tau - \mathrm{I}(x < 0)\big)$ is the well-known quantile check

function and

$$\Psi_{\tau \boldsymbol{u}} = \sum_{i=1}^{n} w_i \rho_\tau(\boldsymbol{b}_{\tau \boldsymbol{u}}^\top \boldsymbol{Y}_i - \boldsymbol{a}_{\tau \boldsymbol{u}}^\top \boldsymbol{X}_i)$$

denotes the optimal value of the objective function computed from the residuals $r_{\tau \boldsymbol{u},i} = \boldsymbol{b}_{\tau \boldsymbol{u}}^\top \boldsymbol{Y}_i - \boldsymbol{a}_{\tau \boldsymbol{u}}^\top \boldsymbol{X}_i$, $i = 1, \ldots, n$. Typically, $w_i = 1$, $i = 1, \ldots, n$. Nevertheless, integer weights may be useful for handling multiple identical observations, and kernel weights asymptotically shrinking to zero lead to the local constant multivariate quantile regression of [8] for $p = 1$. The weighted case can be transformed to the unweighted one by substitutions $\boldsymbol{Y}_i := w_i \boldsymbol{Y}_i$ and $\boldsymbol{X}_i := w_i \boldsymbol{X}_i$ because the computation does not employ any special information about the first coordinate of $\boldsymbol{X}_i$'s. The transformation changes neither the quantile hyperplane coefficients nor the optimal value of the objective function, which is why we consider only unit weights hereinafter.

Next we can consider the finite set Π_τ of all directional (regression) τ-quantile hyperplanes passing through exactly $m + p - 1$ observations,

$$\Pi_\tau = \big\{\pi_{\tau \boldsymbol{u}} : \boldsymbol{u} \in \mathbb{R}^m,\ \|\boldsymbol{u}\| = 1,\ \pi_{\tau \boldsymbol{u}} \text{ contains } m + p - 1 \text{ observations}\big\},$$

and use it to define at least two meaningful, convex, and polyhedral (regression) τ-quantile regions, say exact ($\mathcal{R}_\tau^E$) and approximate ($\mathcal{R}_\tau^A$):

$$\mathcal{R}_\tau^E = \cap_{\pi_{\tau \boldsymbol{u}} \in \Pi_\tau} \mathcal{H}_{\tau \boldsymbol{u}}^+ \quad \text{and} \quad \mathcal{R}_\tau^A = \text{convhull}\{(\boldsymbol{Y}_i^\top, \boldsymbol{Z}_i^\top)^\top \in \mathbb{R}^{m+p-1} : (\boldsymbol{Y}_i^\top, \boldsymbol{Z}_i^\top)^\top \in \mathcal{R}_\tau^E\}.$$

In other words, $\mathcal{R}_\tau^E$ is defined as the intersection of all upper directional (regression) τ-quantile halfspaces containing $m + p - 1$ observations in the bordering directional (regression) quantile hyperplanes, and $\mathcal{R}_\tau^A$ is the convex hull of all the observations belonging to $\mathcal{R}_\tau^E$. The borders of $\mathcal{R}_\tau^E$ and $\mathcal{R}_\tau^A$ are sometimes called (exact) and approximate τ-quantile contours, respectively. The sets Π_τ, $\mathcal{R}_\tau^E$, and $\mathcal{R}_\tau^A$ do not depend on the directional quantile method used. If $\mathcal{R}_\tau^E$ is difficult to obtain due to the very large cardinality of Π_τ, then $\mathcal{R}_\tau^A$ can be used as its approximation because we can know the observations in $\mathcal{R}_\tau^E$ even without the information about its vertices and facets. It is also good to know that $\mathcal{R}_\tau^E$ must be non-empty (and thus contain at least one point of $\mathbb{R}^{m+p-1}$) for any $\tau \leq 1/(m + p)$. The $\boldsymbol{z}_0$-cuts of these regions for various $\boldsymbol{z}_0 \in \mathbb{R}^{p-1}$,

$$\mathcal{R}_\tau^E(\boldsymbol{z}_0) = \{\boldsymbol{y} \in \mathbb{R}^m : (\boldsymbol{y}^\top, \boldsymbol{z}_0^\top)^\top \in \mathcal{R}_\tau^E\} \quad \text{and} \quad \mathcal{R}_\tau^A(\boldsymbol{z}_0) = \{\boldsymbol{y} \in \mathbb{R}^m : (\boldsymbol{y}^\top, \boldsymbol{z}_0^\top)^\top \in \mathcal{R}_\tau^A\},$$

prove especially useful for detecting heteroscedasticity in a general regression case.

In the location case with originally unit weights, Π_τ consists of all those hyperplanes passing through exactly m observations that cut off at most $\lfloor n\tau \rfloor$ and at least $\lceil n\tau \rceil - m$ points from the data cloud. Consequently, the exact regions $\mathcal{R}_\tau^E$ must then coincide with the corresponding halfspace depth regions and thus have all their properties, see, e. g., [3, 6], and [20]. Their sure convexity makes them substantially different from density contours in the case of multimodal distributions that may easily arise even from very simple mixtures, see, e. g., [5]. We can also conclude that Π_τ then always contains enough material for computing m neighbouring exact contours simultaneously if $(m - 1)/n < \tau < 0.5$. This may simplify or speed up the computation of $\mathcal{R}_\tau^E$ and $\mathcal{R}_\tau^A$ for $p = 1$.

How do the methods differ? Mainly in the optimization constraint and its consequences. Method 1, introduced to the large statistical community in [9], uses the constraint $\boldsymbol{b}^\top \boldsymbol{u} = 1$ that leads to the scalar Lagrange multiplier $\lambda_{\tau\boldsymbol{u}}$ equal to $\Psi_{\tau\boldsymbol{u}}$. On the other hand, Method 2 of [16] employs the constraint $\boldsymbol{b} = \boldsymbol{u}$, which results in the Lagrange multiplier vector $\mu^{\boldsymbol{b}}_{\tau\boldsymbol{u}}$ linked to $\Psi_{\tau\boldsymbol{u}}$ through the formula $\Psi_{\tau\boldsymbol{u}} = \mu^{\boldsymbol{b}\top}_{\tau\boldsymbol{u}}\boldsymbol{u}$. But it is still more or less the ordinary τ-quantile regression of all projections $\mathbf{u}^\top \boldsymbol{Y}_i$'s on $\boldsymbol{X}_i$'s.

The optimization problems involved in the two methods can be solved for all nonzero $\boldsymbol{u}$'s in $\mathbb{R}^m$ by means of parametric linear programming. It turns out that the space $\mathbb{R}^m \setminus \{\mathbf{0}\}$ of all directions can be segmented into blunt polyhedral cones where the observations with zero residuals do not change and the solution has a simple form. In each cone, this technique also produces the Lagrange multipliers (or, dual variables) associated with residuals. The dual variables corresponding to positive and negative residuals are boring as they always equal $-\tau$ and $1-\tau$, respectively. On the other hand, the Lagrange multiplier vector $\mu^{\boldsymbol{r}_0}_{\tau\boldsymbol{u}}$ associated with zero residuals must have all its data-dependent coordinates in $(-\tau, 1-\tau)$. They might be used not only like the rank scores in the standard quantile regression, but also for defining halfspace depth of individual observations as in (5.1) of [16].

As for Method 1, there must exist a finite conic segmentation $\Gamma(\tau) = \{C_q(\tau) : q = 1, \dots N_\tau\}$ of $\mathbb{R}^m$ such that each $C_q(\tau)$ is a non-degenerate closed convex polyhedral cone, and $\boldsymbol{b}_{\tau\boldsymbol{u}} = \boldsymbol{b}_{q,\tau}/d_{q,\tau}(\boldsymbol{u})$, $\boldsymbol{a}_{\tau\boldsymbol{u}} = \boldsymbol{a}_{q,\tau}/d_{q,\tau}(\boldsymbol{u})$, $\lambda_{\tau\boldsymbol{u}} = \lambda_{q,\tau}/d_{q,\tau}(\boldsymbol{u})$, $(\Psi_{\tau\boldsymbol{u}} = \lambda_{\tau\boldsymbol{u}})$, and $\mu^{\boldsymbol{r}_0}_{\tau\boldsymbol{u}} = \mathbb{V}_{q,\tau}\boldsymbol{u}/d_{q,\tau}(\boldsymbol{u})$ for any $\mathbf{0} \neq \boldsymbol{u} \in C_q(\tau)$ where $d_{q,\tau}(\boldsymbol{u}) = \boldsymbol{b}^\top_{q,\tau}\mathbf{u}$ and $\boldsymbol{b}_{q,\tau} \in \mathbb{R}^m$, $\boldsymbol{a}_{q,\tau} \in \mathbb{R}^p$, $\lambda_{q,\tau} \in \mathbb{R}$, and $\mathbb{V}_{q,\tau} \in \mathbb{R}_{(m+p-1)\times m}$ are constant objects up to their possible dependence on τ and q. In other words, all directions $\boldsymbol{u}$ inside $C_q(\tau)$ lead to the same hyperplane coming through $m+p-1$ observations, up to the multiplicative factor used for scaling its coefficients.

As for Method 2, there must exist a finite conic segmentation $\Gamma(\tau) = \{C_q(\tau) : q = 1, \dots, N_\tau\}$ of $\mathbb{R}^m$ such that each $C_q(\tau)$ is a non-degenerate closed convex polyhedral cone and $\boldsymbol{b}_{\tau\boldsymbol{u}} = \boldsymbol{u}$, $\boldsymbol{a}_{\tau\boldsymbol{u}} = \mathbb{A}_{q,\tau}\boldsymbol{u}$, $\mu^{\boldsymbol{b}}_{\tau\boldsymbol{u}} = \mu^{\boldsymbol{b}}_{q,\tau}$, $(\Psi_{\tau\boldsymbol{u}} = \mu^{\boldsymbol{b}\top}_{q,\tau}\boldsymbol{u})$, and $\mu^{\boldsymbol{r}_0}_{\tau\boldsymbol{u}} = \mu^{\boldsymbol{r}_0}_{q,\tau}$ for any $\mathbf{u} \in C_q(\tau)$ where $\mathbb{A}_{q,\tau} \in \mathbb{R}_{p\times m}$, $\mu^{\boldsymbol{b}}_{q,\tau} \in \mathbb{R}^m$, and $\mu^{\boldsymbol{r}_0}_{q,\tau} \in \mathbb{R}^p$ are $\boldsymbol{u}$-independent entities in each $C_q(\tau)$ though they may vary with τ and q. In other words, each interior direction $\boldsymbol{u}$ of $C_q(\tau)$ corresponds to a different hyperplane passing through the same p observations. Each τ-quantile hyperplane passing through $m+p-1$ observations must thus correspond to a vertex direction of some $C_q(\tau)$. Such directions may also correspond to τ-quantile hyperplanes having some of their coefficients zero and passing through less than $m+p-1$ observations. Two adjacent cones of $\Gamma(\tau)$ need not always differ in the p observations fitted by the hyperplanes associated with their interior directions, but sometimes only in the sign of one of the quantile hyperplane regression coefficients.

The segmentations $\Gamma(\tau)$'s can be used for exact and efficient computation of many useful statistics defined by means of projection pursuit such as projection depth, skewness, kurtosis, and many other statistics based on the directional multipliers and quantile hyperplane coefficients presented here; see [23]. As $\Gamma(\tau) = -\Gamma(1-\tau)$ and $\pi_{\tau(-\boldsymbol{u})} = \pi_{(1-\tau)(\boldsymbol{u})}$, we allow only $\tau \in [0, 0.5]$ hereinafter.

Before proceeding further, we should finally clarify our illuminating notation. That is to say that we consistently use the same symbols for analogous but different entities figuring in both methods ($\boldsymbol{b}_{\tau\boldsymbol{u}}$, $\boldsymbol{a}_{\tau\boldsymbol{u}}$, $\Psi_{\tau\boldsymbol{u}}$, $\mu^{\boldsymbol{r}_0}_{\tau\boldsymbol{u}}$, $\Gamma(\tau)$, N_τ, etc.). It should be always clear

which method they belong to unless it is irrelevant to the statement made about them. In any case, we would like to point out that both the conic segmentation $\Gamma(\tau)$ and the number of its conic segments N_τ also depend on the method employed.

Our toolbox primarily contains method-specific functions for computing all the important information about all the cones in $\Gamma(\tau)$ that then makes it possible to calculate the optimal solution for any given $\boldsymbol{u}$. These functions are complemented with some auxiliary codes, instructional examples, and also with the function for computing the τ-quantile contours. Let us have a close look at it.

3. IMPLEMENTATION

3.1. Description of the toolbox

The toolbox consists of

- one m-file for computing the (regression) quantile contours (*evalContour.m*);
- two method-specific m-files for solving the parametric programming problems involved in Methods 1 and 2 (*compContourM1u.m* and *compContourM2u.m*), with:
  - two method-specific auxiliary m-files for generating their default input structure CTechST driving the computation (*getCTechSTM1u.m* and *getCTechSTM2u.m*),
  - two method-specific m-files generating the output in the structure CharST (*getCharSTM1u.m* and *getCharSTM2u.m*),
  - two method-specific auxiliary m-files for finding their initial solutions (*findOptimalBasisM1FromScratch.m* and *findOptimalBasisM2FromScratch.m*),
  - two auxiliary m-files for testing their input for correctness (*checkArray.m* and *checkCTechSTu.m*),
  - five auxiliary m-files for manipulating internal auxiliary lists (*addItem.m*, *addRow.m*, *delItem.m*, *findItem.m*, and *findRow.m*);
- five m-files with instructional examples (*ExampleA.m*, *ExampleB.m*, *ExampleC.m*, *ExampleD.m*, and *ExampleE.m*).

Generally, the method related to a method-specific m-file is indicated with M1 or M2 in its filename, and Mi is used for both M1 and M2. The auxiliary files are not supposed to be modified by the user.

All the codes should work both in Octave and MATLAB. Nevertheless, their outcomes can vary because of the different initialization of the random number generators there, which may affect the simulated input data (as in *ExampleA.m* to *ExampleE.m*) or the starting cone(s), for example. The same input should nevertheless lead to the same sets of distinct quantile hyperplanes. The examples *ExampleA.m* to *ExampleE.m* have been tailored to Octave and thus it is their Octave output that is discussed further below.

All the main codes are richly annotated and should be comprehensible with the aid of [17] and [18]. We strongly suggest the reader to study both of the technical articles

before reading further and trying to understand the codes because the files *compContourM1u.m* and *compContourM2u.m* respectively follow the algorithms described there to the last detail. The function *evalContour.m* mainly contains the vertex enumeration for identifying contour vertices, also mentioned in the two articles.

All method-specific codes can be studied side by side because they have been intentionally written in a way highlighting their similarities.

Although we cannot repeat here all the technical details, we can comment on the theoretical computational complexity of the algorithms behind Method 1 and Method 2 for fixed m and p. The computational complexity, say C_i, of finding the initial solution can be theoretically made as low as $O(n)$ almost surely under some regularity conditions according to Theorem 6.4 in [11]. The computational complexity of finding all the other solutions from the first one can then be made as low as $O(nN_\tau)$ in case of Method 1, and also in case of Method 2 if $m = 2$ or if $m > 2$ and only relatively few cones have more than $O(1)$ vertices; see [1] for the optimal complexity of the convex hull algorithm involved. Recall that N_τ denotes the number of cones in the conic segmentation $\Gamma(\tau)$ that is closely linked to the number of all distinct τ-quantile hyperplanes.

Nevertheless, our implementation of the toolbox is not optimal. For example, it uses inefficient SeDuMi solver for finding the initial solution because it is very reliable, accurate, and already available for both Octave and MATLAB. It negatively affects especially the computation of bivariate or extreme contours. The same reasons lead us to employ the convex hull algorithm whose theoretical computational complexity is unknown; see [1].

For the sake of clarity, we will further state predefined values in parentheses just after the variable or field name.

3.2. Solving the parametric programming problems

The parametric program involved in the ith-method can be solved by *compContourMiu.m* with the following header:

$$\text{COutST} = \text{compContourM}i\text{u(Tau, YMat, XMat, CTechST)},\ i = 1, 2.$$

3.2.1. Input

Naturally, Tau corresponds to τ and YMat $\in \mathbb{R}_{n\times m}$ (resp. XMat $\in \mathbb{R}_{n\times p}$) is a matrix containing $\boldsymbol{Y}_j^\top$ (resp. $\boldsymbol{X}_j^\top$) in its jth row, $j = 1, \ldots, n$. All these three parameters are numeric arrays. As far as the problem size is concerned, the program expects $\tau \in [0, 0.5]$, $n > m+p-1$, and $2 \le m \le 6$, and it should work reliably for problems characterized by the triples (m, n, p) up to $(2, 10000, 10)$, $(3, 500, 5)$, and $(4, 150, 3)$ where smaller values of n and p are also permitted. In general, the computation becomes prone to numerical errors and too time/memory consuming with increasing m, n, p, and τ.

The last input argument is a structure whose default value is generated by the auxiliary function getCTechSTMiu without any input. If some input argument to compContourMiu is provided, then all the preceding ones must be assigned as well. If the last input argument is missing, then the default structure is used. If the third argument is missing as well, then XMat is assumed to be the column unit vector of the right length.

There is no need to enter the last argument in most cases because the default setting should usually work well. But let us describe it in detail anyway. The structure CTechST contains some parameters controlling the computation. Its fields say the following:

ReportI (0): if some auxiliary information is reported on the screen to make the progess of the computation visible. Of course, it takes some execution time, especially for $m > 2$. What is then displayed? The value of τ (Tau) (but only if it was internally changed from a problematic input value leading to integer $n\tau$), the initial L_2-normed directional vector $\boldsymbol{u}_0$ used (U0Vec), the number of failures to find an accurate initial solution by the SeDuMi solver (NNotFound), the number of found accurate initial solutions with an inconvenient number of virtually zero coordinates (NBad), and also the width of each layer of the breadth-first search algorithm if it is employed.

OutSaveI (0): if the output is saved in file(s) on the disk. If not, then the output is lost except for the results stored in COutST, which is especially useful for large problems or simulations.

D2SpecI (1): if all the cones are passed through counter-clockwise when $m = 2$. Otherwise, the computation runs in the same way as for $m > 2$. The default setting should be preferred in most cases as it avoids some possible sources of errors.

BriefOutputI (1): if brief or verbose output information is computed. It always includes ConeID (the identification number of the cone), Nu (the number of negative residuals corresponding to all the directions in the interior of the cone), and UVec (the nonzero vertex or other vector $\boldsymbol{u}$ of the cone that is L_2-normalized if both $m = 2$ and CTechST.D2SpecI $= 1$, and L_∞-normalized otherwise). In general, the output rows depend both on the method and on the value of BriefOutputI:

Method 1:
1: [ConeID, Nu, UVec$^\top$, BDVec$^\top$, ADVec$^\top$, LambdaD] $\in \mathbb{R}^\Delta$,
$\Delta = 1 + 1 + m + m + p + 1$,
0: [ConeID, Nu, UVec$^\top$, BDVec$^\top$, ADVec$^\top$, LambdaD, vec(VUMat)$^\top$, IZ$^\top$] $\in \mathbb{R}^\Delta$,
$\Delta = 1 + 1 + m + m + p + 1 + (m + p - 1)m + (m + p - 1)$,
where BDVec, ADVec, LambdaD, and VUMat respectively stand for $\boldsymbol{b}_{q,\tau}$, $\boldsymbol{a}_{q,\tau}$, $\lambda_{q,\tau}$, and $\mathbb{V}_{q,\tau}$ from Section 2, for the q corresponding to ConeID.

Method 2:
1: [ConeID, Nu, UVec$^\top$, vec(ACOMat)$^\top$, MuBRow] $\in \mathbb{R}^\Delta$,
$\Delta = 1 + 1 + m + pm + m$,
0: [ConeID, Nu, UVec$^\top$, vec(ACOMat)$^\top$, MuBRow, MuR0Row, IZ$^\top$] $\in \mathbb{R}^\Delta$,
$\Delta = 1 + 1 + m + pm + m + p + p$,
where ACOMat, MuBRow, and MuR0Row respectively stand for $\mathbb{A}_{q,\tau}$, $\mu_{q,\tau}^{\boldsymbol{b}\top}$, and $\mu_{q,\tau}^{\boldsymbol{r}_0\top}$ from Section 2, for the q corresponding to ConeID.

As for vector IZ (or $\boldsymbol{I}_Z$ in the technical articles), it contains indices of the original observations with zero residuals. Its coordinates match those of MuR0Row and thus link the multipliers (or zero residuals) with concrete data points.

Even the output for BriefOutputI = 1 provides the information sufficient for computing the contours as well as $\boldsymbol{b}_{\tau\boldsymbol{u}}$, $\boldsymbol{a}_{\tau\boldsymbol{u}}$, $\Psi_{\tau\boldsymbol{u}}$, and $\lambda_{\tau\boldsymbol{u}}$ (Method 1) or $\mu^{\boldsymbol{b}}_{\tau\boldsymbol{u}}$ (Method 2). If one also wants to know $\mu^{\boldsymbol{r}_0}_{\tau\boldsymbol{u}}$ or all the observations on $\pi_{\tau\boldsymbol{u}}$ inside each cone, then he or she should set BriefOutputI = 0 and OutSaveI = 1 before the computation in order to obtain the output file(s) with the required information.

Finally, we should probably point out a few not-so-obvious facts.

Firstly, both the codes work with the objective function presented here that is n-times higher than the sample version of the population objective function in the definition of directional quantiles of [9] and [16]. Consequently, all the multipliers produced by the codes are n-times higher than those resulting from the original definition.

Secondly, the set of all output vectors $\boldsymbol{u}$'s generally consists not only of all the cone vertices, but also of some additional directions pointing to the vertices of the intersections of the cones with the artificial bounding box $[-1, 1]^m$ (and possibly with the orthants). Such redundant vertices only contaminate the output and their useless rows should be deleted from it before its further analysis.

Thirdly, the same cone may theoretically appear in the output more than once under different ConeID identifiers, unless $m = 2$ and CTechST.D2SpecI = 1.

CubRegWiseI (1): if the directional space is divided into orthants investigated separately, which splits the problem to 2^m smaller ones. This is helpful for solving large problems with $m > 2$ although it also generates some artificial cones with a facet in the orthant borders and may slow down the computation in some cases.

ArchAllFI (1): if the internal archive should contain all the past cone facet identifiers or only those from the last few layers. The latter way may be faster and less memory demanding but the former choice makes the computation more likely to terminate successfully. This is why storing all facet identifiers is highly recommended and even enforced by the program for $m > 3$.

(only for Method 2) SkipRedI (0): if the information from some cones is ignored, namely from the cones whose all facets are already known except for those artificially induced. This skipping saves some time and space, and it still almost surely preserves all vertices and information relevant to the contour computation. Therefore, it might be useful for solving very large problems. On the other hand, it makes the information in COutST regarding the inner points unreliable.

OutFilePrefS ('DQOutputM*i*_'): what is the prefix of possible output file names. Their suffix is always .dqo. And in between, there is the number of the file padded with zeros to form 6 digits. The default name of the first output file resulting from Method 1 is thus DQOutputM1_000001.dqo.

SeDuMiPathS (''): the path to the SeDuMi directory. It is employed only when SeDuMi cannot be found by the software.

The fields CubRegWiseI, ArchAllFI, and possibly SkipRedI are relevant only if the breadth-first search algorithm is used. They should be changed only when the default setting of CTechST fails, i. e., in case of very large problems beyond dimension two.

See directly *getCTechSTMiu.m* for further details.

3.2.2. Fixed output

It still remains to describe the output structure COutST. Its fields contain useful summary statistics regarding the computation:

CharST: a user-defined output structure.

CTechSTMsgS (''): '' or a self-explaining warning message regarding CTechST.

ProbSizeMsgS (''): '' or a self-explaining warning message regarding the problem size.

CompErrMsgS (''): '' or a self-explaining error message regarding the computation.

TauMsgS (''): '' or a self-explaining warning message regarding the τ.

PosVec ($\mathbf{0}_n$): the vector of length n indicating the positions of individual observations with respect to the (exact) τ-quantile region. PosVec(j) = 0/1/2 if the jth observation lies in the interior/border/exterior of that region, $j = 1, \ldots, n$. This information is reliable only after a successfully finished computation. If CTechST.SkipRedI = 1, then PosVec correctly identifies only all the outer observations.

NDQFiles (0): the number of output files.

NumB (0): the number of (not necessarily different) optimal bases considered.

MaxLWidth (0): the maximum width of one layer.

NIniNone (0): the number of SeDuMi failures to find an accurate initial solution.

NIniBad (0): the number of found accurate initial solutions with an inconvenient number of virtually zero coordinates.

NSkipCone (0): the number of skipped (almost degenerate) cones whose facet vertices could not be evaluated because no interior points sufficiently inside the cones (needed for the vertex enumeration by means of convhulln) could be found.

If CTechST.CubRegWiseI = 1, then the last four fields are calculated over all the individual orthants.

Furthermore, if CTechST.OutSaveI = 1, then the output files (with rows determined by CTechST.BriefOutputI described above) are always saved to the working directory.

3.2.3. User-defined output

The structure CharST is computed inside compContourM*i*u by means of getCharSTM*i*u with the following header:

CharST = getCharSTM*i*u(Tau, N, M, P, BriefDQMat, CharST, IsFirst).

Naturally, Tau, N, M, and P stand for τ, n, m, and p, respectively. The function is first called with BriefDQMat = [], CharST = [], and IsFirst = 1 to initialize CharST, and then repeatedly with IsFirst = 0 for each (potential) output file content in BriefDQMat to update CharST. (That is to say that BriefDQMat then successively contains the rows of potential individual output file(s) corresponding to CTechST.BriefOutputI = 1). This function thus makes it possible to obtain all required information during the run of compContourM*i*u without storing anything on the disk, i. e., without using the external output files at all (which might otherwise occupy even many gigabytes of the hard disk space in case of large multi-dimensional problems).

We provide a non-trivial suggestion for getCharSTM*i*u as a guidance for the users to create their own modifications. By default, the output structure CharST has the following fields:

NUESkip (0): the number of (skipped) hyperplanes corresponding to the directions artificially induced by the $[-1, 1]^m$ bounding box.

NAZSkip (0): the number of (skipped) hyperplanes (not counted in NUESkip) with at least one coordinate of $\boldsymbol{a}_{\tau\boldsymbol{u}}$ zero.

NBZSkip (0): the number of (skipped) hyperplanes (not counted in NUESkip) with at least one coordinate of $\boldsymbol{b}_{\tau\boldsymbol{u}}$ zero.

(if $m \leq 4$) HypMat ([]): the matrix with $(m + p)$ columns where only the distinct hyperplane coefficients $(\boldsymbol{b}_{\tau\boldsymbol{u}}^\top, \boldsymbol{a}_{\tau\boldsymbol{u}}^\top)^\top$ not contributing to NUESkip, NAZSkip, and NBZSkip are stored in rows after being normalized with $\|\boldsymbol{b}_{\tau\boldsymbol{u}}\|$ for Method 1, rounded, and sorted lexicographically.

CharMaxMat (all elements -Inf): the matrix with maxima of certain directional statistics over all the hyperplanes in HypMat (before the normalization of their coefficients) and corresponding cone vertices. Each row contains one such maximum in the last coordinate and one of the directions where it is attained in the preceding ones.

CharMinMat (all elements Inf): the matrix containing minima of certain directional statistics over all the hyperplanes in HypMat (before the normalization of their coefficients) and corresponding cone vertices. Each row contains one such minimum in the last coordinate and one of the directions where it is attained in the preceding ones.

Concerning Method 1:

$$\begin{array}{c} \text{CharMaxMat} = \\ \begin{pmatrix} \boldsymbol{u}^\top & \max \|\boldsymbol{b}_{\tau\boldsymbol{u}}\| \\ \boldsymbol{u}^\top & \max \Psi_{\tau\boldsymbol{u}} \\ \boldsymbol{u}^\top & \max \left(\Psi_{\tau\boldsymbol{u}}/\|\boldsymbol{b}_{\tau\boldsymbol{u}}\|\right) \\ \boldsymbol{u}^\top & \max \|(a^{(2)}_{\tau\boldsymbol{u}}, \ldots, a^{(p)}_{\tau\boldsymbol{u}})^\top\| \\ \boldsymbol{u}^\top & \max \left(\|(a^{(2)}_{\tau\boldsymbol{u}}, \ldots, a^{(p)}_{\tau\boldsymbol{u}})^\top\|/\|\boldsymbol{b}_{\tau\boldsymbol{u}}\|\right) \\ \boldsymbol{u}^\top & \max |a^{(2)}_{\tau\boldsymbol{u}}| \\ \boldsymbol{u}^\top & \max \left(|a^{(2)}_{\tau\boldsymbol{u}}|/\|\boldsymbol{b}_{\tau\boldsymbol{u}}\|\right) \\ \ldots & \ldots \\ \boldsymbol{u}^\top & \max |a^{(p)}_{\tau\boldsymbol{u}}| \\ \boldsymbol{u}^\top & \max \left(|a^{(p)}_{\tau\boldsymbol{u}}|/\|\boldsymbol{b}_{\tau\boldsymbol{u}}\|\right) \end{pmatrix} \end{array} \quad \begin{array}{c} \text{CharMinMat} = \\ \begin{pmatrix} \boldsymbol{u}^\top & \min \|\boldsymbol{b}_{\tau\boldsymbol{u}}\| \\ \boldsymbol{u}^\top & \min \Psi_{\tau\boldsymbol{u}} \\ \boldsymbol{u}^\top & \min \left(\Psi_{\tau\boldsymbol{u}}/\|\boldsymbol{b}_{\tau\boldsymbol{u}}\|\right) \\ \boldsymbol{u}^\top & \min \|(a^{(2)}_{\tau\boldsymbol{u}}, \ldots, a^{(p)}_{\tau\boldsymbol{u}})^\top\| \\ \boldsymbol{u}^\top & \min \left(\|(a^{(2)}_{\tau\boldsymbol{u}}, \ldots, a^{(p)}_{\tau\boldsymbol{u}})^\top\|/\|\boldsymbol{b}_{\tau\boldsymbol{u}}\|\right) \end{pmatrix} \end{array}$$

and Method 2:

$$\begin{array}{c} \text{CharMaxMat} = \\ \begin{pmatrix} \boldsymbol{u}^\top & \max \Psi_{\tau\boldsymbol{u}} \\ \boldsymbol{u}^\top & \max \|\mu^{\boldsymbol{b}}_{\tau\boldsymbol{u}}\| \\ \boldsymbol{u}^\top & \max \|(a^{(2)}_{\tau\boldsymbol{u}}, \ldots, a^{(p)}_{\tau\boldsymbol{u}})^\top\| \\ \boldsymbol{u}^\top & \max |a^{(2)}_{\tau\boldsymbol{u}}| \\ \ldots & \ldots \\ \boldsymbol{u}^\top & \max |a^{(p)}_{\tau\boldsymbol{u}}| \end{pmatrix} \end{array} \quad \begin{array}{c} \text{CharMinMat} = \\ \begin{pmatrix} \boldsymbol{u}^\top & \min \Psi_{\tau\boldsymbol{u}} \\ \boldsymbol{u}^\top & \min \|\mu^{\boldsymbol{b}}_{\tau\boldsymbol{u}}\| \\ \boldsymbol{u}^\top & \min \|(a^{(2)}_{\tau\boldsymbol{u}}, \ldots, a^{(p)}_{\tau\boldsymbol{u}})^\top\| \end{pmatrix} \end{array}$$

where $\boldsymbol{u}$ in each matrix row stands for one of the directions where the corresponding maximum or minimum is attained. The last rows are included only if $p \geq 2$. Ordinary users interested solely in the quantile contours will need only HypMat. They need not change the default functions if they do not intend to experiment with five- or six-dimensional responses.

3.3. Computing the contours

If AAMat is a matrix with two to six columns and BBVec is a vector with the same number of rows, then the set of inequalities AAMat*$\boldsymbol{z} \leq$ BBVec defines a convex polyhedral region whose approximate volume, vertices, and facets can be computed by

CST = evalContour(AAMat, BBVec) or CST = evalContour(AAMat, BBVec, IPVec)

where IPVec is a point known to be well in the interior of the region. It is always beneficial to enter a known point IPVec as the last argument, especially in terms of speed.

The fields of structure CST contain useful summary statistics regarding the region:

Status (0): the indicator of problems during the computation:

0: no problem,

2: the contour seems virtually empty,

3: a (SeDuMi) numerical failure,

4: the number of input parameters is too low,

5: AAMat is not a numerical matrix with two to six columns,

6: BBVec is not a numerical column vector of the same length as the first column of AAMat,

7: IPVec is not a numerical column vector of the same length as the first row of AAMat.

TVVMat ([]): the lexicographically sorted matrix with clearly distinct contour vertices in rows.

TKKMat ([]): the matrix with clearly distinct elementary facets of the contour in rows. Each facet is described with the indices of its vertices, referring to the rows of TVVMat.

Vol (NaN): the (approximate) volume of the contour.

NumF (NaN): the number of clearly distinct contour facets.

NumV (NaN): the number of clearly distinct contour vertices.

3.4. Installation instructions

1. Install Octave 3.5/MATLAB 8.1 (or higher versions). Both the programs already contain the important function convhulln employed in our codes.

2. Install the latest (free) SeDuMi optimization toolbox for both Octave and MATLAB, available online at `http://sedumi.ie.lehigh.edu/downloads`.

3. Unzip the toolbox presented here directly into your working directory or into another directory in the search path of your Octave/MATLAB installation.

4. EXAMPLES

We also include five examples, represented by the codes *ExampleA.m* to *ExampleE.m*. They should guide the reader through the most common applications. They focus on the essential functionality of the main programs and avoid unnecessary or fancy features, which makes them short and easy to read. Their graphical output is stored into their parent directory. This section discusses their output in Octave. Their output in MATLAB would be different because the random number generator there produces different simulated input data.

Example A illustrates the use of *compContourMiu.m* and *evalContour.m* in a regression context with $n = 7$ bivariate responses and one nontrivial regressor ($n = 7$, $m = 2$, $p = 2$, and $\tau = 0.20$). Any statistical analysis of such a small data set is meaningless and we include it only because its output is very short. The text output displayed by Octave

is presented here for *compContourM1u.m* and *compContourM2u.m*, always followed by that of *evalContour.m*:

```
Method No 1:

COutST =
 scalar structure containing the fields:
  CTechSTMsgS =
  ProbSizeMsgS =
  CompErrMsgS =
  TauMsgS =
  CharST =
   scalar structure containing the fields:
    CharMaxMat =
     -0.837374420  0.546629740  2.895248060
     -0.659121770  0.752036230  1.105585410
     -0.540682670  0.841226630  1.094840420
     -0.837374420  0.546629740  3.327051810
      0.986986650  0.160802210  2.064405560
     -0.837374420  0.546629740  3.327051810
      0.986986650  0.160802210  2.064405560
    CharMinMat =
     -0.540682670  0.841226630  1.000039340
     -0.942461970 -0.334313390  0.372737430
     -0.942461970 -0.334313390  0.308246970
      0.732940810 -0.680292410  0.461377320
      0.629025890 -0.777384350  0.349268630
    NUESkip = 0
    NAZSkip = 0
    NBZSkip = 0
    HypMat =
     -8.93050240e-01  4.49956970e-01 -3.39140000e-04  8.74947530e-01
     -8.02212740e-01 -5.97038290e-01 -5.84490000e-04  1.14914223e+00
     -5.91437940e-01 -8.06350520e-01 -5.33270000e-04  9.80044430e-01
     -5.48122800e-01  8.36397870e-01 -5.42734580e-01  1.29187840e+00
     -4.35098120e-01 -9.00383040e-01 -4.78160000e-04  8.32409880e-01
      2.62337890e-01  9.64976080e-01 -8.61964480e-01 -1.67005702e+00
      6.35270430e-01  7.72289760e-01 -3.34994120e-01 -1.41359070e+00
      8.46269490e-01  5.32755060e-01 -7.53303400e-01 -2.06440556e+00
      9.74871140e-01  2.22769540e-01 -6.10782640e-01 -1.15067004e+00
      9.99348660e-01 -3.60868200e-02 -4.52277280e-01 -3.49268630e-01
  PosVec =
    2
    1
    0
    2
```

```
    1
    1
    2
  NDQFiles = 1
  NumB = 11
  MaxLWidth = 1
  NIniNone = 0
  NIniBad = 0
  NSkipCone = 0
CST =
 scalar structure containing the fields:
  Status = 0
  TVVMat =
   -0.657944300 -0.205188100  0.566423300
   -0.583601100 -0.538160800  0.319304900
   -0.307763400  0.485896000 -0.364132000
    0.988159900  0.308462700 -0.849585000
  TKKMat =
    3  4  1
    4  2  1
    2  3  1
    3  2  4
  Vol = 0.0907320785783836
  NumF = 4
  NumV = 4
```

and

```
Method No 2:

COutST =
 scalar structure containing the fields:
  CTechSTMsgS =
  ProbSizeMsgS =
  CompErrMsgS =
  TauMsgS =
  CharST =
   scalar structure containing the fields:
    CharMaxMat =
     -0.548122800  0.836397870  1.094840420
     -0.548122800  0.836397870  1.105585410
      0.846269490  0.532755060  2.064405560
      0.846269490  0.532755060  2.064405560
    CharMinMat =
     -0.5914379401 -0.8063505200  0.3082469700
     -0.8022127400 -0.5970382900  0.3727374300
```

```
      0.9993486600 -0.0360868200  0.3492686300
    NUESkip = 0
    NAZSkip = 4
    NBZSkip = 8
    HypMat =
     -8.93050240e-01  4.49956970e-01 -3.39140000e-04  8.74947530e-01
     -8.02212740e-01 -5.97038290e-01 -5.84490000e-04  1.14914223e+00
     -5.91437940e-01 -8.06350520e-01 -5.33270000e-04  9.80044430e-01
     -5.48122800e-01  8.36397870e-01 -5.42734580e-01  1.29187840e+00
     -4.35098120e-01 -9.00383040e-01 -4.78160000e-04  8.32409880e-01
      2.62337890e-01  9.64976080e-01 -8.61964480e-01 -1.67005702e+00
      6.35270430e-01  7.72289760e-01 -3.34994120e-01 -1.41359070e+00
      8.46269490e-01  5.32755060e-01 -7.53303400e-01 -2.06440556e+00
      9.74871140e-01  2.22769540e-01 -6.10782640e-01 -1.15067004e+00
      9.99348660e-01 -3.60868200e-02 -4.52277280e-01 -3.49268630e-01
  PosVec =
    2
    1
    0
    2
    1
    1
    2
  NDQFiles = 1
  NumB = 17
  MaxLWidth = 1
  NIniNone = 0
  NIniBad = 0
  NSkipCone = 0
CST =
 scalar structure containing the fields:
  Status = 0
  TVVMat =
   -0.657944300 -0.205188100  0.566423300
   -0.583601100 -0.538160800  0.319304900
   -0.307763400  0.485896000 -0.364132000
    0.988159900  0.308462700 -0.849585000
  TKKMat =
    3  4  1
    4  2  1
    2  3  1
    3  2  4
  Vol = 0.0907320785783836
  NumF = 4
  NumV = 4
```

It is immediately apparent from COutST in both cases that:

- there are no warning and error messages (CTechSTMsgS, ProbSizeMsgS, CompErrMsgS, TauMsgS). The input was therefore correct and the computation probably ended successfully.
- there would be only one output file (NDQFiles).
- there have been no problems with finding the initial solution(s) starting the algorithm (NIniNone, NIniBad).
- the analyzed problem was very small in size (NumB, MaxLWidth).
- no (virtually) degenerate cones have been encountered during the computation (NSKipCone), which increases the reliability of produced results.
- there are three observations outside the quantile region, three in its boundary, and the third one lies in its interior (PosVec).

The (sub)structure CharST of COutST also provides useful information:

- that ten distinct quantile hyperplanes with $m + p - 1$ observations have been found. Their coefficient vectors $(\boldsymbol{b}_{\tau\boldsymbol{u}}^\top, \boldsymbol{a}_{\tau\boldsymbol{u}}^\top)^\top$ are normalized with $\|\boldsymbol{b}_{\tau\boldsymbol{u}}\|$ if Method 1 is used, rounded, sorted lexicographically, and finally recorded in the rows of HypMat. The last row thus roughly corresponds to the hyperplane $0.999y_1 - 0.036y_2 + 0.452 + 0.349x = 0$.
- that Method 1 results in no misleading hyperplanes (NUESkip) or potentially problematic hyperplanes with zero coefficients (NAZSkip, NBZSkip) to exlude from HypMat. Method 2 may lead to nonzero NAZSkip or NBZSkip in principle and thus there is no need to worry about that.
- the maxima (CharMaxMat) and minima (CharMinMat) of some statistics over all the hyperplanes of HypMat (before the normalization with $\|\boldsymbol{b}_{\tau\boldsymbol{u}}\|$ used for Method 1) and corresponding cone vertices. For example, $\max \|\boldsymbol{b}_{\tau\boldsymbol{u}}\| \doteq 2.895$ in Method 1 corresponds (e. g.) to $\boldsymbol{u} \doteq (-0.837, 0.547)^\top$. Such statistics might be useful for statistical inference according to [9] and [16].

Note that the output in HypMat, CharMaxMat, and CharMinMat is rounded. It is also worth pointing out that the last columns of CharMaxMat or CharMinMat contain some quantities common to both methods. It follows from the fact that both Method 1 and Method 2 should lead to the same HypMat.

Finally, the output structure CST of *evalContour.m* reveals some information about the resulting (regression) quantile region such as

- that it has 4 vertices (NumV), rounded, lexicographically sorted, and stored in the rows of TVVMat. The last vertex is thus roughly $(0.988, 0.308, -0.850)^\top$.
- that it has 4 elementary facets (NumF), stored in the rows of TKKMat. The last facet is thus defined by its corner vertices in the last three rows of TVVMat.

- that its (approximate) volume is equal to 0.09.

The code of *ExampleA.m* also shows how HypMat and PosVec might be used to find the input parameters AAMat, BBVec, and IPVec of *evalContour.m* and how the regression quantile contour can be plotted; see Figure 1 for the graphical output.

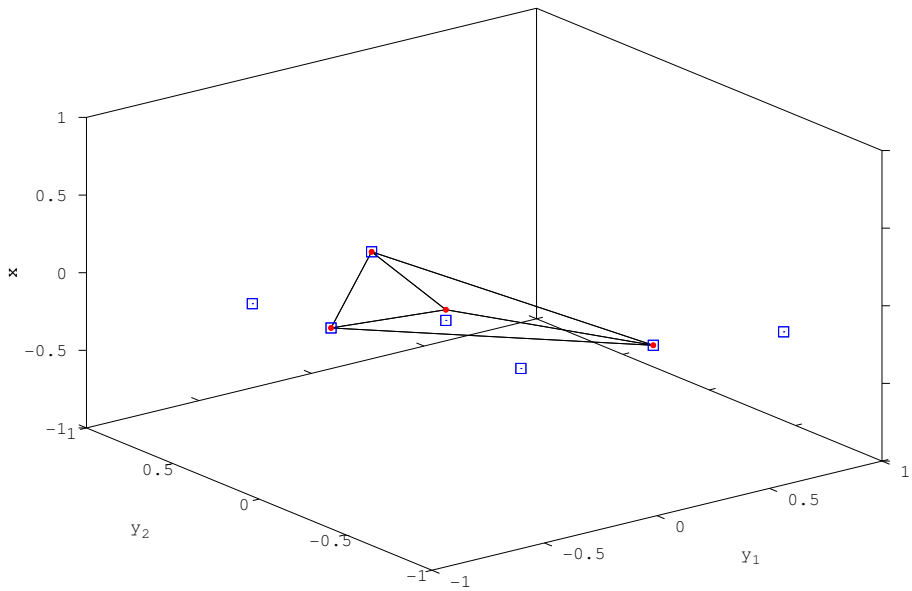

Fig. 1. (Octave output) This figure presents the regression quantile region with four vertices (red circles) that can be obtained by both Method 1 and Method 2 for $n = 7$, $m = 2$, $p = 2$, and $\tau = 0.2$ from n observations (blue squares) driven by the model where $(Y_1, Y_2)^\top \sim U([-1, 1]^2)$ is independent of $X \sim U([-1, 1])$. See *ExampleA.m* for all the technical details.

As both Method 1 and Method 2 should lead to the same graphical output, we employ only the former in the subsequent examples. The latter would be used if we changed compContourM1u to compContourM2u and getCTechSTM1u to getCTechSTM2u everywhere in the codes.

Example B (with *ExampleB.m*) should teach the reader how a bunch of location quantile contours can be computed efficiently from the innermost one outwards, i. e., from the highest τ to the lowest. For the sake of simplicity, three τ-contours, $\tau \in \{0.3579, 0.1357, 0.0135\}$, are computed from $n = 1357$ independent bivariate observations with unit weights (and uniformly distributed in $[-1, 1]^2$) in three different ways leading to the same graphical output; see Figure 2.

Firstly, each contour is computed from the whole dataset in a straightforward way.

Secondly, when one contour is computed, then all its n_{Int} original interior points are replaced with their average as a single point with weight n_{Int} and the next contour is

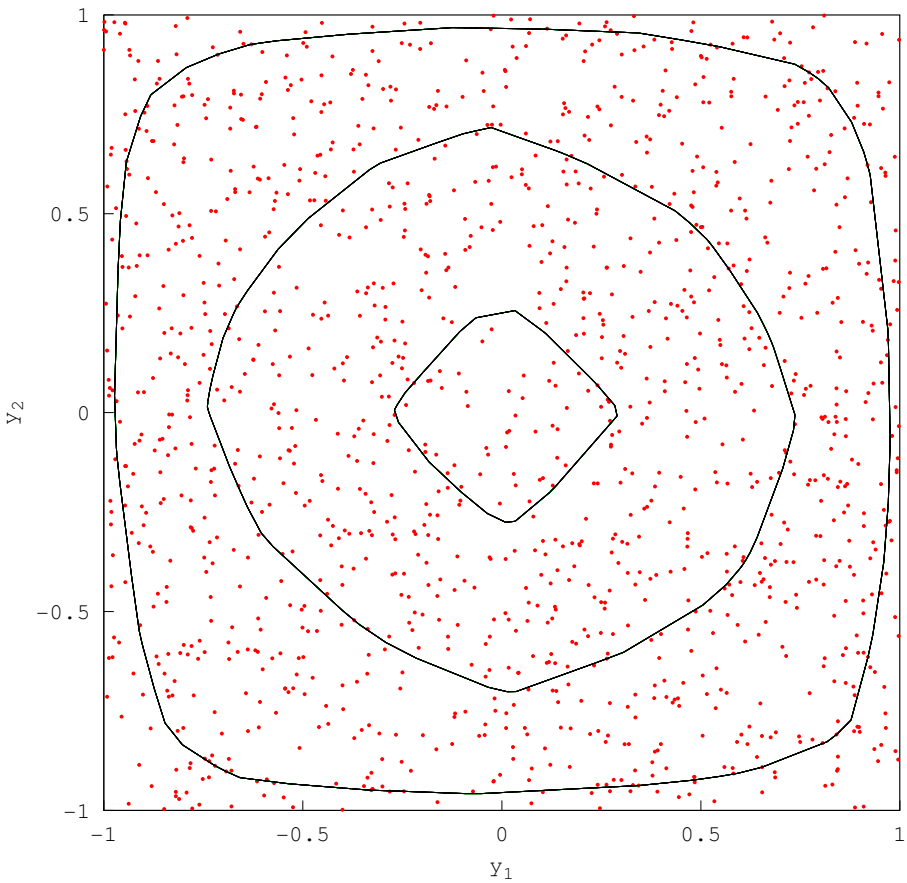

Fig. 2. (Octave output) This figure plots three 2D (green) location τ-quantile contours, $\tau = 0.3579, 0.1357$, and 0.0135, obtained from $n = 1357$ (red) bivariate independent random points coming from the model $(Y_1, Y_2)^\top \sim U([-1, 1]^2)$. They were computed in three ways leading to the same graphical output: (a) directly from the original data, (b) by using weights and replacing the observations surely in the interior of the computed contour with a single pseudo-observation, and (c) by changing τ and deleting the observations surely in the interior of the computed contour. The last two tricks are discussed in detail in the text and based on computing the contours from the innermost one outwards. They might lead to substantial speed benefits in cases of very large data sets and/or m-variate responses, $m \geq 3$. See *ExampleB.m* for all the technical details.

computed from such a reduced dataset for the other τ in turn. This approach should lead to the same hyperplane coefficients as the first one with Method 2 because it keeps the gradient conditions unchanged. We are grateful to Prof. Roger Koenker for kindly pointing us to this possibility.

Thirdly, when one contour is computed from n_c observations, then all its n_{Int} interior points are deleted and the remaining points are used for computing the other contour in turn, but with corresponding τ modified to $\tau_{\mathrm{mod}} = \tau n/(n_c - n_{\mathrm{Int}})$. The procedure starts with $n = n_c$ and proceeds recursively. It still leads to the same contours as the first approach, but it changes the information provided with them.

We think that these tricks might be found useful especially when $m > 2$, the dataset is large, and the contours are numerous. Unfortunately, they are guaranteed to work correctly only in the location case where the quantile contours are necessarily nested and no quantile crossing occurs.

Example C, associated with *ExampleC.m* and Figure 3, demonstrates the use of affine equivariance for computing a τ-quantile contour. It is computed for $\tau = 0.20$ from $n = 21$ trivariate independent random points uniformly distributed in $[0, 1]^3$ in two equivalent ways: (a) directly from the original data, and (b) from the data standardized to the $[-1, 1]^3$ cube by scaling and shifting the resulting vertices in line with the rules of affine equivariance. The latter approach can be analogously extended even to the regression case. It decreases the probability of numerical errors because our codes are tailored to datasets of such range, and as such it is heartily recommended. The standardization might also be done by means of some robust estimates of the center and the variance matrix, for example, or in any other meaningful way.

Example D might be found useful by those wishing to implement parametric multivariate quantile regression. It computes and displays x_0-cuts (i. e., $\boldsymbol{z}_0$-cuts for $\boldsymbol{z}_0 = (x_0, x_0^2)^\top$), $x_0 = -0.8, -0.6, \ldots, 0.8$, of the parametric regression τ-quantile region, $\tau = 0.35$, obtained for regressors 1, X, and X^2 from $n = 9\,999$ bivariate random points coming from the model $(Y_1, Y_2)^\top = (X, X^2)^\top + \boldsymbol{\varepsilon}$ with independent $X \sim U([-1, 1])$ and $\boldsymbol{\varepsilon} \sim U([-1, 1]^2)$. See Figure 4 and *ExampleD.m*.

Example E concludes our exposition with local(ly) constant (nonparametric) regression τ-quantile cuts obtained for the same τ and x_0's and from the same dataset as in Example D with the aid of normal kernel weights with bandwidth 0.4. Note that here we have to compute a τ-quantile region for each x_0 while the parametric regression in Example D requires only one τ-quantile region for all x_0's. The nonparametric cuts are slightly prolonged due to the relatively high bandwidth and owing to the local constant character of the regression. See Figure 5 and *ExampleE.m*.

5. COMMON PROBLEMS

At the end, we would like to list a few common problems and the ways to overcome them:

bad data the data points are in a bad position (e. g., they are not in general position). This often happens when rounded or inherently discrete-valued observations are processed, when dummy variables are used in multivariate quantile regression models, or when a bad random number generator is used for simulating the obser-

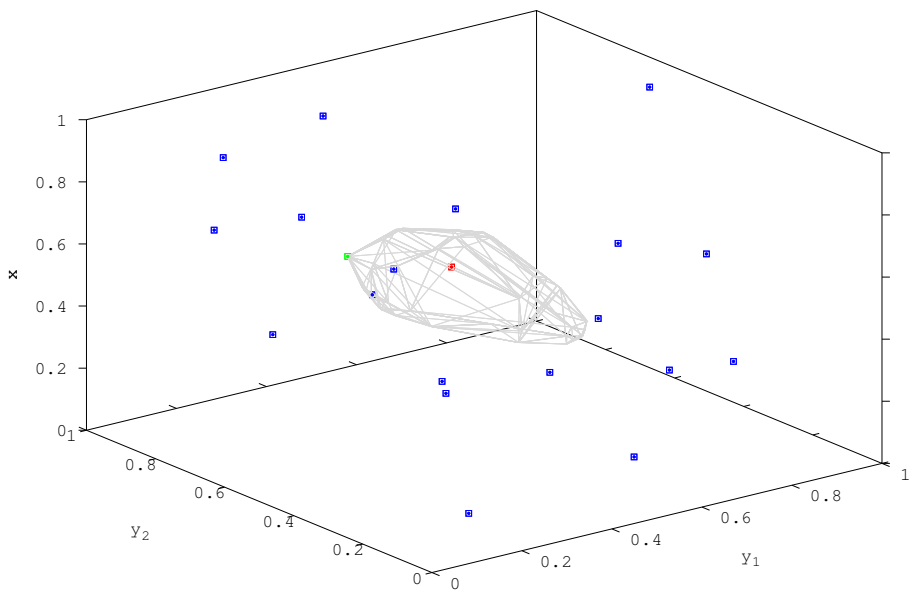

Fig. 3. (Octave output) This figure displays one 3D (triangulated) location τ-quantile contour, $\tau = 0.20$, obtained from $n = 21$ trivariate independent random points coming from the model $(Y_1, Y_2, Y_3)^\top \sim U([0,1]^3)$. It was computed in two ways leading to the same graphical output: (a) directly from the original data, and (b) from standardized data by means of affine equivariance. Both ways correctly lead to the same results and rightly indicate (red) interior, (green) bordery and (blue) exterior points of the τ-quantile region that are marked with squares in the former case and with circles in the latter. See *ExampleC.m* for all the technical details.

vations. Such a problem can sometimes be prevented by perturbing the data with some random noise of a reasonably small magnitude before the computation. Furthermore, a few identical observations may be replaced with one of them, weighted by the total number of its occurrences.

bad tau the τ leads to problems. The computation may fail for a finite number of problematic τ's, e. g., if τn is an integer in the location case. There the τ's correspond to the cases where the sample quantiles are not uniquely defined. This situation may happen easily for τ's in a fractional form or with only a few decimal digits. It may also occur unexpectedly when the number of observations automatically changes during the computation of several contours such as in Example B. If this problem (almost) arises, then it can be fixed by perturbing τ with a tiny number in the right direction. Similar strategy is also adopted by the codes *compContourMiu.m*, but only in the location case and with a warning output message

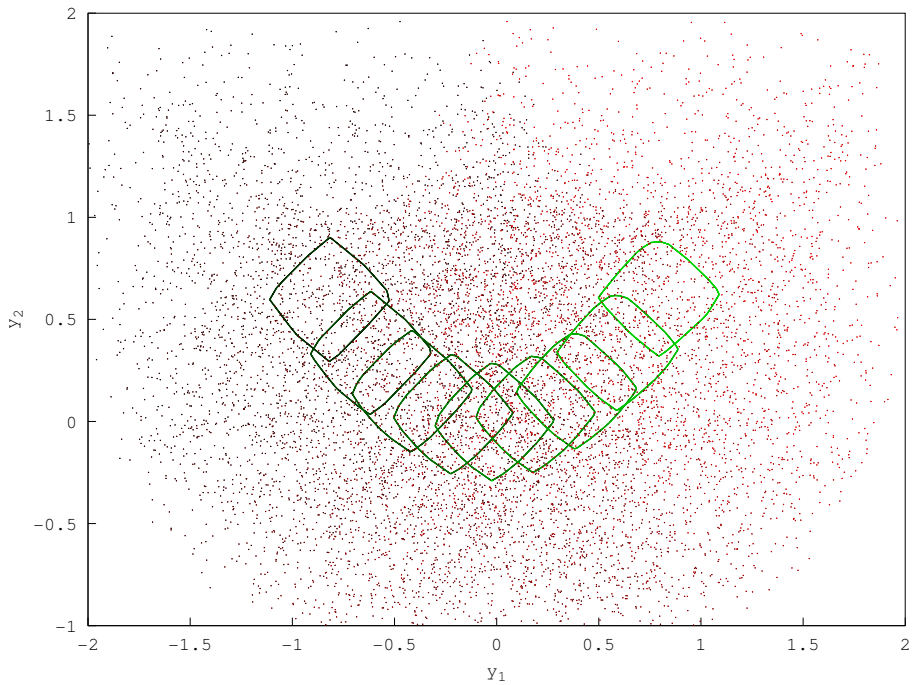

Fig. 4. (Octave output) Given $\tau = 0.35$ and $x_0 = -0.8, -0.6, \ldots, 0.8$, this figure shows (green) x_0-cuts of the parametric regression τ-quantile region obtained for regressors 1, X, and X^2 from $n = 9\,999$ (red) bivariate random points coming from the model $(Y_1, Y_2)^\top = (X, X^2)^\top + \boldsymbol{\varepsilon}$ with independent $X \sim U([-1, 1])$ and $\boldsymbol{\varepsilon} \sim U([-1, 1]^2)$. The cuts lighten with increasing x_0 and the points darken with decreasing regressor values. See *ExampleD.m* for all the technical details.

explaining it. If the perturbation is small enough, then the quantile contours may remain the same though the other output usually changes. This might confuse some users if they did not notice the automatic perturbation.

bad scale if the data set far exceeds the unit (hyper)sphere, then the computation may fail easily because the programs have been optimized for individual responses and regressors drawn from the interval $[-1, 1]$. This problem can be avoided by changing the units of the observations or by employing affine equivariance as in Example C.

Similar problems often arise when non-uniform weights are used to transform properly scaled data before the computation, for example when the local(ly) constant regression quantiles are computed. Then the weights should be scaled conveniently and the observations with virtually zero weights should be excluded from the com-

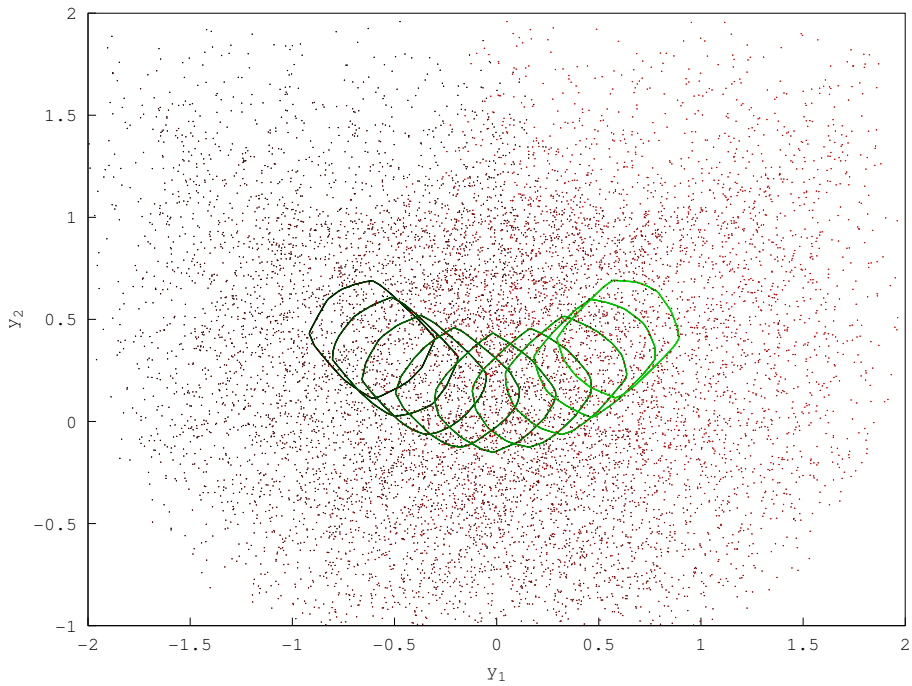

Fig. 5. (Octave output) This figure uses the same settings as Example D and Figure 4. It displays x_0-cuts through the local constant (i. e., nonparametric) regression τ-quantile regions obtained for each x_0 with the aid of normal kernel weights corresponding to bandwidth 0.4. See *ExampleE.m* for all the technical details.

putation. This would have been necessary in Example E if we had decided to use much smaller bandwidth than 0.4.

bad expectations the computation and its output may differ from our expectations. For example, not all people immediately realize that the computation may take hours or fail even for moderately sized data sets in $\mathbb{R}^3$, that HypMat is not always present in COutST.CharST by default, that the computed quantile contour may be empty, or that the regression quantile contours need not be nested or bounded.

bad interpretation the results are interpreted erroneously. The quantile level τ is not directly linked to the probability content of the τ-quantile regions. The parametric multivariate quantile regression models should also include as regressors all the variables influencing the dispersion of response coordinates. And even if such models are miraculously specified correctly, there is still no reason to interpret the cuts of the resulting regression quantile regions as conditional multivariate quantiles, maybe except for some very special cases. Such an interpretation would

be somehow justified only if the nonparametric regression quantiles of [8] were used, as in Example E.

As you have probably realized, τ denotes the quantile level and n stands for the number of observations everywhere in this section.

ACKNOWLEDGMENTS

The research work of Pavel Boček and Miroslav Šiman was supported by the grant GA14-07234S from the Czech Science Foundation. Miroslav Šiman would also like to thank Davy Paindaveine, Marc Hallin, Claude Adan, Nancy de Munck, and Romy Genin for all the good they did for him (and for all the good he could learn from them) during his stay at Université Libre de Bruxelles.

(Received June 22, 2015)

REFERENCES

[1] C. B. Barber and H. Huhdanpaa: The quickhull algorithm for convex hulls. ACM Trans. Math. Software *22* (1996), 469–483. DOI:10.1145/235815.235821

[2] P. Boček and M. Šiman: Directional quantile regression in R. Submitted, 2016.

[3] Z. Chen and D. E. Tyler: On the behavior of Tukey's depth and median under symmetric stable distributions. J. Statist. Planning Inference *122* (2004), 111–124. DOI:10.1016/j.jspi.2003.06.017

[4] Y. Cheng and J. G. De Gooijer: On the uth geometric conditional quantile. J. Statist. Planning Inference *137* (2007), 1914–1930. DOI:10.1016/j.jspi.2006.02.014

[5] Š. Došlá: Conditions for bimodality and multimodality of a mixture of two unimodal densities. Kybernetika *45* (2009), 279–292.

[6] S. Dutta, A. K. Ghosh, and P. Chaudhuri: Some intriguing properties of Tukey's half-space depth. Bernoulli *17* (2011), 1420–1434. DOI:10.3150/10-bej322

[7] J. W. Eaton, D. Bateman, and S. Hauberg: GNU Octave Version 3.0.1 Manual: A High-Level Interactive Language for Numerical Computations. CreateSpace Independent Publishing Platform, 2009.

[8] M. Hallin, Z. Lu, D. Paindaveine, and M. Šiman: Local bilinear multiple-output quantile/depth regression. Bernoulli *21* (2015), 1435–1466. DOI:10.3150/14-bej610

[9] M. Hallin, D. Paindaveine, and M. Šiman: Multivariate quantiles and multiple-output regression quantiles: From L_1 optimization to halfspace depth. The Ann. Statist. *38* (2010), 635–669. DOI:10.1214/09-aos723

[10] M. Hallin, D. Paindaveine, and M. Šiman: Rejoinder. The Ann. Statist. *38* (2010), 694–703. DOI:10.1214/09-aos723rej

[11] R. Koenker: Quantile Regression. Cambridge University Press, New York 2005. DOI:10.1017/cbo9780511754098

[12] R. Koenker and G. J. Bassett: Regression quantiles. Econometrica *46* (1978), 33–50. DOI:10.2307/1913643

[13] V. Koltchinskii: M-estimation, convexity and quantiles. The Ann. Statist. *25* (1997), 435–477. DOI:10.1214/aos/1031833659

[14] L. Kong and I. Mizera: Quantile tomography: Using quantiles with multivariate data. Statist. Sinica *22* (2012), 1589–1610. DOI:10.5705/ss.2010.224

[15] I. W. McKeague, S. López-Pintado, M. Hallin, and M. Šiman: Analyzing growth trajectories. J. Developmental Origins of Health and Disease *2* (2011), 322–329. DOI:10.1017/s2040174411000572

[16] D. Paindaveine and M. Šiman: On directional multiple-output quantile regression. J. Multivariate Anal. *102* (2011), 193–212. DOI:10.1016/j.jmva.2010.08.004

[17] D. Paindaveine and M. Šiman: Computing multiple-output regression quantile regions. Comput. Statist. Data Anal. *56* (2012), 840–853. DOI:10.1016/j.csda.2010.11.014

[18] D. Paindaveine and M. Šiman: Computing multiple-output regression quantile regions from projection quantiles. Computat. Statist. *27* (2012), 29–49. DOI:10.1007/s00180-011-0231-y

[19] R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna 2008.

[20] P. J. Rousseeuw and I. Ruts: The depth function of a population distribution. Metrika *49* (1999), 213–244.

[21] The MathWorks, Inc.: MATLAB. Natick, Massachusetts 2013.

[22] J. F. Sturm : Using SeDuMi 1.02, a MATLAB Toolbox for Optimization over Symmetric Cones Optimization Methods and Software *11–12* (1999), 625–653. http://sedumi.ie.lehigh.edu/downloads

[23] M. Šiman: On exact computation of some statistics based on projection pursuit in a general regression context. Comm. Statist. – Simul. Comput. *40* (2011), 948–956. DOI:10.1080/03610918.2011.560730

[24] M. Šiman: Precision index in the multivariate context. Comm. Statist. – Theory and Methods *43* (2014), 377–387. DOI:10.1080/03610926.2012.661509

Pavel Boček, Institute of Information Theory and Automation, The Czech Academy of Sciences, Pod Vodárenskou věží 4, Praha 8. Czech Republic.
e-mail: bocek@utia.cas.cz

Miroslav Šiman, Institute of Information Theory and Automation, The Czech Academy of Sciences, Pod Vodárenskou věží 4, Praha 8. Czech Republic.