Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computational Statistics & Data Analysis 50 (2006) 5 – 20

COMPUTATIONAL STATISTICS & DATA ANALYSIS

www.elsevier.com/locate/csda

Computation of Huber's M-estimates for a block-angular regression problem☆

Xiao-Wen Chang*

School of Computer Science, McGill University, 3480 University Street, Montreal, Quebec, Canada H3A 2A7

Available online 23 August 2004

Abstract

Huber's M-estimation technique is applied to a block-angular regression problem, which may arise from some applications. A recursive, modified Newton approach to computing the estimates is presented. The structure of the problem is exploited to make the algorithm efficient. It is shown how to efficiently compute a descent search direction by using updating/downdating techniques for matrix factorizations. Numerical test results suggest that the proposed approach is effective.

Keywords: Block angular matrix; Huber's M-estimator; Least-squares estimator; Recursive estimators; Newton's method

☆ This research was supported by NSERC of Canada Grant RGPIN217191-03, FQRNT of Quebec Grant 2001-NC-66487, and NSERC-GEOIDE Network Project ENV#14.

* Tel.: +1-514-398-8259; fax: +1-514-398-3883.

E-mail address: chang@cs.mcgill.ca (X.-W. Chang).

URL: http://www.cs.mcgill.ca/∼chang.

0167-9473/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.csda.2004.07.019

1. Problem

In this paper, we consider the following block-angular linear model:

$$\begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{k-1} \\ \boldsymbol{y}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}_1 & & & & \boldsymbol{Z}_1 \\ & \ddots & & & \vdots \\ & & \boldsymbol{X}_{k-1} & & \boldsymbol{Z}_{k-1} \\ & & & \boldsymbol{X}_k & \boldsymbol{Z}_k \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_1 \\ \vdots \\ \boldsymbol{\beta}_{k-1} \\ \boldsymbol{\beta}_k \\ \boldsymbol{\gamma} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\epsilon}_1 \\ \vdots \\ \boldsymbol{\epsilon}_{k-1} \\ \boldsymbol{\epsilon}_k \end{bmatrix}, \quad k = 1, 2, \ldots, \tag{1}$$

or equivalently

$$\boldsymbol{y}_{[k]} = \boldsymbol{X}_{[k]}\boldsymbol{\beta}_{[k]} + \boldsymbol{\epsilon}_{[k]}, \quad k = 1, 2, \ldots,$$

where for $j=1, 2, \ldots, \boldsymbol{y}_j \in \mathscr{R}^{n_j}$ is the measurement vector obtained at (usually time) step j, $\boldsymbol{X}_j \in \mathscr{R}^{n_j \times p_j}$, $\boldsymbol{Z}_j \in \mathscr{R}^{n_j \times p_0}$, $\boldsymbol{\beta}_j \in \mathscr{R}^{pj}$ and $\gamma \in \mathscr{R}^{p0}$ are parameter vectors, $\boldsymbol{\epsilon}_j$ is a random error vector, and for $k=1, 2, \ldots$, the block-angular matrix $\boldsymbol{X}_{[k]}$ has full column rank. Note that vector $\boldsymbol{\gamma}$ does not change with time. This model arises from the global positioning system (GPS) (see, e.g., Chang and Paige, 2003), where $\boldsymbol{\beta}_k$ includes the coordinates of the relative position of a roving GPS receiver to a stationary receiver and the receiver clock error at time step k, and $\boldsymbol{\gamma}$ is the so-called ambiguity vector which is invariant with time. It may also arise in other applications, such as photogrammetry and geodetic survey (see, e.g., Björck, 1996, Section 6.3 and references therein).

In the ideal situation, the mean $\mathscr{E}\{\boldsymbol{\epsilon}_{[k]}\} = \mathbf{0}$ and the covariance matrix $\text{cov}\{\boldsymbol{\epsilon}_{\lfloor k \rfloor}\} = \sigma^2\mathbf{I}$. Then the least squares (LS) estimation for model (1) gives the best linear unbiased estimator of $\boldsymbol{\beta}_{[k]}$. For a fixed k, there are a few algorithms available for computing the LS estimate of $\boldsymbol{\beta}_{[k]}$ (see, e.g., Cox, 1990; Golub et al., 1979; Golub and Plemmons, 1980). For an incremental k, a recursive version of the algorithm proposed in Golub et al. (1979) (see also Björck, 1996, Section 6.3) can easily be developed. But sometimes the measurements are contaminated, leading to outliers. It is well known that the LS estimator is sensitive to large outliers. A typical approach to handle this estimation problem is to use robust M-estimation techniques (see Huber, 1981). In this paper, we would like to apply the well-known Huber's M-estimation technique. We will assume that there are at most a few outliers at each time step, as is the case in GPS. Our goal is to present an efficient recursive algorithm to compute Huber's M-estimates by exploiting the structures of (1). To our knowledge, there has not been any work on the computation of Huber's M-estimates for this block-angular model.

This paper is organized as follows. In Section 2, we introduce Huber's M-estimation for a general linear model and give a general framework for Newton's method with an exact line search to solve the problem. A new strategy is proposed to handle a possible singularity problem occurring in Newton's method. In Section 3, we present a recursive Newton method to compute Huber's M-estimates for model (1). We show how to use updating/downdating techniques for matrix factorizations to compute a Newton direction in an efficient way. To simplify the computation and reduce the computational cost and memory requirement, we propose to compute a modified Newton direction in each iteration step. In Section 4, we give numerical test results to illustrate the performance of our algorithms. Finally a summary is given in Section 5.

Throughout this paper we use bold lowercase letters for vectors and bold uppercase letters for matrices.

2. Introduction to Huber's M-estimation

In this section, we first describe Huber's M-estimation problem for a general linear regression problem, and then introduce a general framework for Newton's method with a line search to solve the problem. We discuss advantages and disadvantages of different strategies dealing with a singularity problem which may happen in Newton's method and propose a new strategy which is suitable for solving our estimation problem.

Suppose we have a linear model

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{2}$$

where $\boldsymbol{y} = [y_1, \ldots, y_n]^{\mathrm{T}} \in \mathscr{R}^n$, $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]^{\mathrm{T}} \in \mathscr{R}^{n\times p}$, $\boldsymbol{\beta} \in \mathscr{R}^p$ and $\boldsymbol{\epsilon} \in \mathscr{R}^n$. Write $\boldsymbol{r}(\boldsymbol{\beta}) \equiv \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}$, with ith element $r_i(\boldsymbol{\beta})$. Huber's M-estimation problem is the following optimization problem:

$$\min_{\boldsymbol{\beta}} \{F(\boldsymbol{\beta}) \equiv \sum_{i=1}^{n} \rho(r_i(\boldsymbol{\beta}))\}, \tag{3}$$

with the nonnegative, convex, piecewise function (it is linear, then quadratic, then linear)

$$\rho(t) \equiv \begin{cases} \frac{1}{2}t^2, & |t| \leqslant c, \\ c|t| - \frac{1}{2}c^2, & |t| > c, \end{cases} \tag{4}$$

defined for some tuning constant $c > 0$. $\rho(t)$ in the form of (4) is called the Huber function (here we have taken the scaling factor in the original Huber objective function to be 1, as in some literature, see, e.g., Antoch and Ekblom, 1995; Madsen and Nielsen, 1990; O'Leary, 1990). Note that Huber's M-estimation is a mixed l_2 and l_1 minimization problem. There are several other well-known functions such as the Fair, Talwar, Tukey, and Welsh functions, but the Huber function is probably the most popular one.

We see from (4) that $\rho(t)$ is a continuous function, with continuous and nondecreasing first derivative, since

$$\rho'(t) = \begin{cases} t, & |t| \leqslant c, \\ c\,\mathrm{sign(t)}, & |t| > c, \end{cases} \qquad \rho''(t) = \begin{cases} 1, & |t| \leqslant c, \\ 0, & |t| > c. \end{cases} \tag{5}$$

Strictly speaking, $\rho(t)$ has only a right (left) second derivative at $t = -c$ $(t = c)$. But from a practical point of view, there is no harm in defining $\rho''(\pm c) = 1$.

The active index set and nonactive set at $\boldsymbol{\beta} \in \mathscr{R}^n$ are, respectively, defined by

$$\nu(\boldsymbol{\beta}) \equiv \{i : |r_i(\boldsymbol{\beta})| \leqslant c\}, \quad \bar{\nu}(\boldsymbol{\beta}) \equiv \{i : |r_i(\boldsymbol{\beta})| > c\}.$$

The active matrix $\boldsymbol{X}^{\nu}$ of $\boldsymbol{X}$ at $\boldsymbol{\beta}$ is defined to be the remaining matrix of $\boldsymbol{X}$ after the rows whose indexes in $\bar{\nu}(\boldsymbol{\beta})$ are deleted. Define the sign vector

$$\boldsymbol{s}(\boldsymbol{\beta}) \equiv [s_1(\boldsymbol{\beta}), \ldots, s_n(\boldsymbol{\beta})]^{\mathrm{T}}, \quad s_i(\boldsymbol{\beta}) \equiv \begin{cases} -1, & r_i(\boldsymbol{\beta}) < -c, \\ 0, & |r_i(\boldsymbol{\beta})| \leqslant c, \\ 1, & r_i(\boldsymbol{\beta}) > c \end{cases}$$

and the weight matrix

$$\boldsymbol{W}(\boldsymbol{\beta}) \equiv \mathrm{diag}(w_1(\boldsymbol{\beta}), \ldots, w_n(\boldsymbol{\beta})), \quad w_i(\boldsymbol{\beta}) \equiv 1 - s_i^2(\boldsymbol{\beta}) = \rho''(r_i(\boldsymbol{\beta})).$$

The function $F(\boldsymbol{\beta})$ in (3) can then be written

$$F(\boldsymbol{\beta}) = \tfrac{1}{2}\boldsymbol{r}^{\mathrm{T}}(\boldsymbol{\beta})W(\boldsymbol{\beta})\boldsymbol{r}(\boldsymbol{\beta}) + c\boldsymbol{s}^{\mathrm{T}}(\boldsymbol{\beta})\left[\boldsymbol{r}(\boldsymbol{\beta}) - \tfrac{1}{2}c\boldsymbol{s}(\boldsymbol{\beta})\right]. \tag{6}$$

Since $\partial\boldsymbol{r}(\boldsymbol{\beta})^{\mathrm{T}}/\partial\boldsymbol{\beta} = -\boldsymbol{X}^{\mathrm{T}}$, differentiating (6) gives the gradient of $F(\boldsymbol{\beta})$

$$F'(\boldsymbol{\beta}) \equiv \frac{\partial F(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -\boldsymbol{X}^{\mathrm{T}}[\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{r}(\boldsymbol{\beta}) + c\boldsymbol{s}(\boldsymbol{\beta})] = -\sum_{i\in\nu(\boldsymbol{\beta})} \boldsymbol{x}_i r_i(\boldsymbol{\beta}) - c\sum_{i\in\bar{\nu}(\boldsymbol{\beta})} \boldsymbol{x}_i s_i(\boldsymbol{\beta}).$$

The symmetric nonnegative definite Hessian matrix is given by

$$F''(\boldsymbol{\beta}) \equiv \frac{\partial^2 F(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^{\mathrm{T}}} = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{X} = \sum_{i\in\nu(\boldsymbol{\beta})} \boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}.$$

A general framework for the Newton method with a line search for solving (3) can be described as follows:

```
Given an initial estimate $\boldsymbol{\beta}$,
repeat until convergence:
    solve the following equation for the search direction $\boldsymbol{h}$:
        $F''(\boldsymbol{\beta})\boldsymbol{h} = -F'(\boldsymbol{\beta}), \quad \text{or} \quad \boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{X}\boldsymbol{h} = \boldsymbol{X}^{\mathrm{T}}[\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{r}(\boldsymbol{\beta}) + c\boldsymbol{s}(\boldsymbol{\beta})],$        (7)
    perform a line search and update $\boldsymbol{\beta} := \boldsymbol{\beta} + \hat{\alpha}\boldsymbol{h}$.
```

In the following we give some remarks about this general scheme.

Usually the LS estimate $\boldsymbol{\beta}_{\mathrm{LS}}$ for model (2) is taken to be the initial estimate, so that if $\bar{\nu}(\boldsymbol{\beta}_{\mathrm{LS}}) = \phi$ (no outliers),

$$\boldsymbol{W}(\boldsymbol{\beta}_{\mathrm{LS}}) = \boldsymbol{I}_n, \quad \boldsymbol{s}(\boldsymbol{\beta}_{\mathrm{LS}}) = \boldsymbol{0}, \quad \boldsymbol{X}^{\mathrm{T}}[\boldsymbol{W}(\boldsymbol{\beta}_{\mathrm{LS}})\boldsymbol{r}(\boldsymbol{\beta}_{\mathrm{LS}}) + c\boldsymbol{s}(\boldsymbol{\beta}_{\mathrm{LS}})] = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{r}(\boldsymbol{\beta}_{\mathrm{LS}}) = \boldsymbol{0}.$$

Notice that $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{r}(\boldsymbol{\beta}_{\mathrm{LS}}) = \boldsymbol{0}$ are the normal equations and from (7) we see $\boldsymbol{h} = \boldsymbol{0}$. Thus $\boldsymbol{\beta}_{\mathrm{LS}}$ solves (3).

Any $\boldsymbol{h}$ satisfying (7) with a *nonzero* $F'(\boldsymbol{\beta})$ is a *strictly* descent direction for the functional F at $\boldsymbol{\beta}$, since

$$\boldsymbol{h}^{\mathrm{T}}F'(\boldsymbol{\beta}) = -\boldsymbol{h}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{X}\boldsymbol{h} = -\|W(\boldsymbol{\beta})\boldsymbol{X}\boldsymbol{h}\|_2^2 < 0. \tag{8}$$

It can be shown that there is a minimizer $\boldsymbol{\beta}$ of $F(\boldsymbol{\beta})$ such that $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{X}$ is nonsingular (see, e.g., Osborne, 1985, p. 272). When $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{X}$ is nonsingular, it is positive definite and the Cholesky factorization of $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{X}$ can be used to solve (7). The Cholesky factor can be obtained from the QR factorization of the active matrix $\boldsymbol{X}^{v}$ of $\boldsymbol{X}$ at $\boldsymbol{\beta}$ due to $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{X}=(\boldsymbol{X}^{v})^{\mathrm{T}}\boldsymbol{X}^{v}$. Since the active matrices for two consecutive iterates usually differ by just a few rows, updating/downdating techniques for matrix factorizations (see, e.g., Björck, 1996, Chapter 3) should be used for efficiency.

But if during the iterative process there are more than $n-p$ large residuals beyond the tuning constant c, so that more than $n-p$ diagonal elements of $\boldsymbol{W}(\boldsymbol{\beta})$ are zero, then $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{X}$ is singular, i.e., the active matrix $\boldsymbol{X}^{v}$ at $\boldsymbol{\beta}$ does not have full column rank. There are several strategies for handling this problem, see below.

In the so-called H-algorithm, Huber and Dutter (1974) used $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ instead of $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{X}$. This approach has another advantage: only one matrix factorization is needed for solving the linear equations (see (7)) to obtain the search directions. The W-algorithm, which is also called the iteratively reweighted LS algorithm (see, e.g., Huber, 1981; Osborne, 1985, Sections 5.4, 5.6), replaces $\boldsymbol{W}(\boldsymbol{\beta})$ by $\mathrm{diag}(d_1,\ldots,d_n)$ with $d_i=1$ if $|r_i|\leqslant c$ or $d_i=c/|r_i(\boldsymbol{\beta})|$ if $|r_i|>c$. This approach is often used by practical people. But both algorithms may have slow convergence.

In the algorithm given by Madsen and Nielsen (1990), later modified by Chen and Pinar (1998), if the equation in (7) is consistent (i.e., it has solutions), $\boldsymbol{h}$ is taken to be a basic solution or a minimum 2-norm solution, otherwise $\boldsymbol{h}$ is taken to be the solution of

$$\boldsymbol{H}\boldsymbol{h}=\boldsymbol{X}^{\mathrm{T}}[\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{r}(\boldsymbol{\beta})+c\boldsymbol{s}(\boldsymbol{\beta})],$$

where $\boldsymbol{H}$ is chosen to be a positive definite matrix, e.g., $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{X}+\delta\boldsymbol{I}$ with a small constant δ. The (modified) Madsen and Nielsen algorithm has finite convergence and is quite efficient as reported in Madsen and Nielsen (1990). But checking consistency is numerically difficult and also the computational cost is too large if the matrix $\boldsymbol{X}$ is large or structured, because a pivoting strategy in the matrix factorizations has to be used.

In Antoch and Ekblom (1995), if $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{X}$ is found to be singular, then it is replaced by $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{X}+\delta\boldsymbol{I}$. The shortcoming with this approach (and the previous approach as well) is that updating/downdating of the matrix factorization is expensive when the matrix becomes a singular matrix from a nonsingular matrix, or *vice versa*, since it is expensive to go from the factorization of $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{X}$ to $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{X}+\delta\boldsymbol{I}$, or *vice versa*.

In O'Leary (1990), the strategy is to use a very large tuning constant at the beginning, then gradually decrease the value of the tuning constant to the desired value over the first four steps of the iteration. This strategy avoids the pivoting issue and is useful in the cases where the matrix $\boldsymbol{X}$ is large, sparse and structured, like $\boldsymbol{X}_{[k]}$ in (1). But no implementation details for this strategy were given in O'Leary (1990), nor was a guarantee given that this strategy would always work.

In this paper, we use the following strategy to handle this singularity problem. If the active matrix $\boldsymbol{X}^{v}$ is not of full column rank, we choose the row vector from those $\boldsymbol{x}_i^{\mathrm{T}}$ with $i\in\bar{v}(\boldsymbol{\beta})$ which corresponds to the smallest residual in magnitude and add this row vector to $\boldsymbol{X}^{v}$. We continue this process until the updated $\boldsymbol{X}^{v}$ has full column rank. In practice for a general matrix $\boldsymbol{X}$, when the updated $\boldsymbol{X}^{v}$ becomes square, it is usually nonsingular. Then we solve (cf. (7))

$$(\boldsymbol{X}^{v})^{\mathrm{T}}\boldsymbol{X}^{v}\boldsymbol{h}=\boldsymbol{X}^{\mathrm{T}}[\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{r}(\boldsymbol{\beta})+c\boldsymbol{s}(\boldsymbol{\beta})]$$

for the search direction $\boldsymbol{h}$. Notice that we did not change the actual weight matrix $\boldsymbol{W}(\boldsymbol{\beta})$ and the sign vector $\boldsymbol{s}(\boldsymbol{\beta})$ because we did not change the tuning constant. It is easy to observe that the resulting $\boldsymbol{h}$ is a descent direction.

The step length $\hat{\alpha}$ in the line search can be found exactly in theory. Write

$$\phi(\alpha) \equiv F(\boldsymbol{\beta} + \alpha\boldsymbol{h}) = \sum_{i=1}^{n} \rho(y_i - \boldsymbol{x}_i^{\mathrm{T}}(\boldsymbol{\beta} + \alpha\boldsymbol{h})).$$

Since $\phi(\alpha)$ is the sum of nonnegative, convex, piecewise defined functions of α, with each piece being either quadratic or linear, see (3)–(5), it too is a nonnegative, convex, piecewise defined function with each piece being quadratic (possibly linear). Therefore $\phi'(\alpha)$ must be piecewise, with each piece linear (possibly constant), and we can find exactly a minimizer $\hat{\alpha}$ of $\phi(\alpha)$, i.e.,

$$\hat{\alpha} = \arg \min_{\alpha} \phi(\alpha).$$

This $\hat{\alpha}$ is a zero of $\phi'(\alpha)$. We can also see that since $\boldsymbol{h}$ is a strict descent direction for F at $\boldsymbol{\beta}$, $\phi'(0) < 0$, and since $\phi(\alpha)$ is convex, $\phi'(\alpha)$ is increasing. Since $\boldsymbol{Xh} \neq 0$, $\phi(\alpha) \to \infty$ as $\alpha \to \infty$. So $\hat{\alpha}$ must be positive and finite. For the efficient computation of $\hat{\alpha}$, see Madsen and Nielsen (1990).

Finally we would like to mention that recursively computing the M-estimates for the general linear model (2) has been considered in Antoch and Ekblom (1995), which investigates how to use the M-estimate for (2) to compute the new M-estimate when one more equation is added to (2).

3. Finding Huber’s M-estimates for the block-angular model

The main cost in computing the Huber M-estimate for the general linear model (2) is the cost of solving (7), or the cost of finding the Cholesky factor of $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{\beta})\boldsymbol{X}$, in each iteration. In the block-angular model (1), the matrices $\boldsymbol{X}_{[k]}$ have a special structure. We will show in this section how to use this structure to efficiently compute the Cholesky factors during the iterations. We first discuss the computation from iteration step i to iteration step $i+1$ at time step k in Section 3.1, then discuss the computation from time step k to time step $k+1$ in Section 3.2. During the iteration process at each time step, we may need to update or downdate the active matrices of previous time steps. This makes the computation very complicated. In order to overcome this, we propose to compute a modified Newton direction in Section 3.3.

3.1. *Computation from iteration step i to iteration step $i+1$*

Since $\boldsymbol{X}_{[k]}$ has full column rank, all $\boldsymbol{X}_j$, $j = 1, \ldots, k$ must have full column rank. There is a minimizer $\boldsymbol{\beta}_{[k|k]}$ such that the corresponding active matrix of $\boldsymbol{X}_{[k]}$ has full column rank, so the corresponding active matrix $\boldsymbol{X}_j^{\nu}$ of $\boldsymbol{X}_j$ must have full column rank, $j = 1, \ldots, k$. In our algorithm we always assume that the active matrices $\boldsymbol{X}_j^{\nu}$ during the iterations have full column rank. This can be ensured by using the strategy we proposed in Section 2.

Suppose at time step k, the iterate $\boldsymbol{\beta}^{(i)}_{[k|k]}$ and the active matrix $\boldsymbol{X}^{(i)}_{[k|k]}$ of $\boldsymbol{X}_{[k]}$ at iteration step i are, respectively, denoted by

$$\boldsymbol{\beta}^{(i)}_{[k|k]} \equiv \begin{bmatrix} \boldsymbol{\beta}^{(i)}_{1|k} \\ \vdots \\ \boldsymbol{\beta}^{(i)}_{k|k} \\ \gamma^{(i)}_k \end{bmatrix}, \quad \boldsymbol{X}^{(i)}_{[k|k]} \equiv \begin{bmatrix} \boldsymbol{X}^{(i)}_{1|k} & & & \boldsymbol{Z}^{(i)}_{1|k} \\ & \ddots & & \vdots \\ & & \boldsymbol{X}^{(i)}_{k|k} & \boldsymbol{Z}^{(i)}_{k|k} \end{bmatrix}.$$

Suppose $[\boldsymbol{X}^{(i)}_{j|k}, \boldsymbol{Z}^{(i)}_{j|k}]$ has the orthogonal factorization:

$$\begin{bmatrix} (\boldsymbol{Q}^{(i)}_{j|k})^{\mathrm{T}} \\ (\bar{\boldsymbol{Q}}^{(i)}_{j|k})^{\mathrm{T}} \end{bmatrix} [\boldsymbol{X}^{(i)}_{j|k} \quad \boldsymbol{Z}^{(i)}_{j|k}] = \begin{bmatrix} \boldsymbol{R}^{(i)}_{j|k} & \hat{\boldsymbol{R}}^{(i)}_{j|k} \\ \boldsymbol{0} & \bar{\boldsymbol{R}}^{(i)}_{j|k} \end{bmatrix}, \quad j = 1, \ldots, k, \tag{9}$$

where $[\boldsymbol{Q}^{(i)}_{j|k}, \bar{\boldsymbol{Q}}^{(i)}_{j|k}]$ is orthogonal, $\boldsymbol{R}^{(i)}_{j|k}$ is nonsingular upper triangular. Here the structure of $\bar{\boldsymbol{R}}^{(i)}_{j|k}$ depends on the dimensions $n_j \times (p_j + p_0)$ of $[\boldsymbol{X}_j, \quad \boldsymbol{Z}_j]$. In the equivalent of (9) for $[\boldsymbol{X}_j, \quad \boldsymbol{Z}_j]$ (see (16)) if $n_j \leqslant p_j + p_0$, $\bar{\boldsymbol{R}}_j$ is a full matrix, then here $\bar{\boldsymbol{R}}^{(i)}_{j|k}$ is left as a full matrix too, i.e., we do not make it have any special structure; otherwise $\bar{\boldsymbol{R}}_j$ is upper triangular, so we make $\bar{\boldsymbol{R}}^{(i)}_{j|k}$ to be upper triangular or upper trapezoidal. Here the Q-factor will be formed and stored for later updating/downdating use. Then we have

$$\begin{bmatrix} (\boldsymbol{Q}^{(i)}_{1|k})^{\mathrm{T}} & & \\ & \ddots & \\ & & (\boldsymbol{Q}^{(i)}_{k|k})^{\mathrm{T}} \\ (\bar{\boldsymbol{Q}}^{(i)}_{1|k})^{\mathrm{T}} & & \\ & \ddots & \\ & & (\bar{\boldsymbol{Q}}^{(i)}_{k|k})^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \boldsymbol{X}^{(i)}_{1|k} & & & \boldsymbol{Z}^{(i)}_{1|k} \\ & \ddots & & \vdots \\ & & \boldsymbol{X}^{(i)}_{k|k} & \boldsymbol{Z}^{(i)}_{k|k} \end{bmatrix}$$

$$= \begin{bmatrix} \boldsymbol{R}^{(i)}_{1|k} & & & \hat{\boldsymbol{R}}^{(i)}_{1|k} \\ & \ddots & & \vdots \\ & & \boldsymbol{R}^{(i)}_{k|k} & \hat{\boldsymbol{R}}^{(i)}_{k|k} \\ & & & \bar{\boldsymbol{R}}^{(i)}_{1|k} \\ & & & \vdots \\ & & & \bar{\boldsymbol{R}}^{(i)}_{k|k} \end{bmatrix}. \tag{10}$$

Suppose we have the following QR factorization (remember the matrix has full column rank):

$$(\tilde{\boldsymbol{Q}}^{(i)}_{[k]})^{\mathrm{T}} \begin{bmatrix} \bar{\boldsymbol{R}}^{(i)}_{1|k} \\ \vdots \\ \bar{\boldsymbol{R}}^{(i)}_{k|k} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{R}}^{(i)}_{[k]} \\ \boldsymbol{0} \end{bmatrix}, \quad \tilde{\boldsymbol{Q}}^{(i)}_{[k]} \text{ orthogonal}, \quad \tilde{\boldsymbol{R}}^{(i)}_{[k]} \text{ nonsingular upper triangular.} \tag{11}$$

Here the Q-factor is not formed or stored because it may be too large when k becomes large. If the condition number of $\tilde{\boldsymbol{R}}^{(i)}_{[k]}$ is known to be acceptable, then for efficiency we may compute the R-factor $\tilde{\boldsymbol{R}}^{(i)}_{[k]}$ by the Cholesky factorization:

$$\sum_{j=1}^{k} (\bar{\boldsymbol{R}}^{(i)}_{j|k})^{\mathrm{T}} \bar{\boldsymbol{R}}^{(i)}_{j|k} = (\tilde{\boldsymbol{R}}^{(i)}_{[k]})^{\mathrm{T}} \tilde{\boldsymbol{R}}^{(i)}_{[k]}, \tag{12}$$

where the structure of $\bar{\boldsymbol{R}}^{(i)}_{j|k}$ should be used in forming $(\bar{\boldsymbol{R}}^{(i)}_{j|k})^{\mathrm{T}} \bar{\boldsymbol{R}}^{(i)}_{j|k}$. Even when the matrices $\bar{\boldsymbol{R}}^{(i)}_{j|k}$ in (11) are upper triangular, (12) is more efficient in floating point operations than (11) when $k \geqslant 3$.

Then from (10) and (11) there is an orthogonal matrix $[\boldsymbol{Q}^{(i)}_{[k|k]}, \bar{\boldsymbol{Q}}^{(i)}_{[k|k]}]$ such that

$$\begin{bmatrix} (\boldsymbol{Q}^{(i)}_{[k|k]})^{\mathrm{T}} \\ (\bar{\boldsymbol{Q}}^{(i)}_{[k|k]})^{\mathrm{T}} \end{bmatrix} \boldsymbol{X}^{(i)}_{[k|k]} = \begin{bmatrix} \boldsymbol{R}^{(i)}_{1|k} & & \hat{\boldsymbol{R}}^{(i)}_{1|k} \\ & \ddots & \vdots \\ & \boldsymbol{R}^{(i)}_{k|k} & \hat{\boldsymbol{R}}^{(i)}_{k|k} \\ & & \tilde{\boldsymbol{R}}^{(i)}_{[k]} \\ & & \boldsymbol{0} \end{bmatrix}. \tag{13}$$

Note that we only compute the R-factor here. Thus we have

$$\begin{aligned} &\boldsymbol{X}^{\mathrm{T}}_{[k]} \boldsymbol{W}(\boldsymbol{\beta}^{(i)}_{[k|k]}) \boldsymbol{X}_{[k]} = (\boldsymbol{X}^{(i)}_{[k|k]})^{\mathrm{T}} \boldsymbol{X}^{(i)}_{[k|k]} \\ &= \begin{bmatrix} (\boldsymbol{R}^{(i)}_{1|k})^{\mathrm{T}} & & & \\ & \ddots & & \\ & & (\boldsymbol{R}^{(i)}_{k|k})^{\mathrm{T}} & \\ (\hat{\boldsymbol{R}}^{(i)}_{1|k})^{\mathrm{T}} & \cdots & (\hat{\boldsymbol{R}}^{(i)}_{k|k})^{\mathrm{T}} & (\tilde{\boldsymbol{R}}^{(i)}_{[k]})^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \boldsymbol{R}^{(i)}_{1|k} & & \hat{\boldsymbol{R}}^{(i)}_{1|k} \\ & \ddots & \vdots \\ & \boldsymbol{R}^{(i)}_{k|k} & \hat{\boldsymbol{R}}^{(i)}_{k|k} \\ & & \tilde{\boldsymbol{R}}^{(i)}_{[k]} \end{bmatrix}. \end{aligned}$$

With this factorization, we then solve the corresponding linear system (cf. (7)) to find the search direction. After the line search, we obtain the new iterate $\boldsymbol{\beta}^{(i+1)}_{[k|k]}$. Then we start iteration step $i+1$.

At iteration step $i+1$, we use $\boldsymbol{\beta}^{(i+1)}_{[k|k]}$ to find the corresponding active matrix $\boldsymbol{X}^{(i+1)}_{[k|k]}$. From the orthogonal factorization of $[\boldsymbol{X}^{(i)}_{j|k}, \boldsymbol{Z}^{(i)}_{j|k}]$ (see (9)) we can use standard QR updating/downdating techniques to compute the orthogonal factorization of the new active matrix $[\boldsymbol{X}^{(i+1)}_{j|k}, \boldsymbol{Z}^{(i+1)}_{j|k}]$, which is usually different from $[\boldsymbol{X}^{(i)}_{j|k}, \boldsymbol{Z}^{(i)}_{j|k}]$ by at most a few rows, at least when the iterate is close to the solution. We mentioned before that if $n_j \leqslant p_j + p_0$ then $\bar{\boldsymbol{R}}^{(i)}_{j|k}$ is a full matrix, so the goal of the updating/downdating is only to keep the upper triangular structure of $\boldsymbol{R}^{(i)}_{j|k}$. When $n_j > p_j + p_0$, $\bar{\boldsymbol{R}}^{(i)}_{j|k}$ is trapezoidal or upper triangular, and this structure needs to be kept in the updating/downdating process. After the above updating/downdating process, we can compute the version of (12) for $i+1$. Since we assumed that there are at most a few outliers in each time step, often the active matrices $[\boldsymbol{X}^{(i)}_{j|k}, \boldsymbol{Z}^{(i)}_{j|k}]$

do not change for $j < k$, so we seldom need to perform the above updating/downdating process for the orthogonal factorization (9) for $j < k$. Therefore most terms in the summation in (12) usually do not change, and the version of (12) for $i + 1$ can efficiently be computed.

3.2. Computation from time step k to time step $k + 1$

Suppose at the end of time step k, we have converged to Huber's M-estimate

$$\boldsymbol{\beta}_{[k|k]} = [\boldsymbol{\beta}_{1|k}^{\mathrm{T}}, \ldots, \boldsymbol{\beta}_{k|k}^{\mathrm{T}}, \boldsymbol{\gamma}_k^{\mathrm{T}}]^{\mathrm{T}}.$$

Then we can obtain the full column rank active matrix at $\boldsymbol{\beta}_{[k|k]}$:

$$\boldsymbol{X}_{[k|k]} \equiv \begin{bmatrix} \boldsymbol{X}_{1|k} & & & \boldsymbol{Z}_{1|k} \\ & \ddots & & \vdots \\ & & \boldsymbol{X}_{k|k} & \boldsymbol{Z}_{k|k} \end{bmatrix}. \tag{14}$$

Using the results obtained in the previous iteration step, we compute the R-factor of $\boldsymbol{X}_{[k|k]}$ by the techniques we have already mentioned in the last paragraph of Section 3.1 and we have (cf. (13))

$$\begin{bmatrix} \boldsymbol{Q}_{[k|k]}^{\mathrm{T}} \\ \bar{\boldsymbol{Q}}_{[k|k]}^{\mathrm{T}} \end{bmatrix} \boldsymbol{X}_{[k|k]} = \begin{bmatrix} \boldsymbol{R}_{1|k} & & & \hat{\boldsymbol{R}}_{1|k} \\ & \ddots & & \vdots \\ & & \boldsymbol{R}_{k|k} & \hat{\boldsymbol{R}}_{k|k} \\ & & & \tilde{\boldsymbol{R}}_{[k]} \\ & & & \mathbf{0} \end{bmatrix}. \tag{15}$$

At time step $k + 1$, let the initial estimate for $\boldsymbol{\beta}_{[k+1]}$ be denoted by

$$\boldsymbol{\beta}_{[k+1|k+1]}^{(0)} \equiv [(\boldsymbol{\beta}_{1|k+1}^{(0)})^{\mathrm{T}}, \ldots, (\boldsymbol{\beta}_{k|k+1}^{(0)})^{\mathrm{T}}, (\boldsymbol{\beta}_{k+1|k+1}^{(0)})^{\mathrm{T}}, (\boldsymbol{\gamma}_{k+1}^{(0)})^{\mathrm{T}}]^{\mathrm{T}}.$$

Naturally we take

$$\boldsymbol{\beta}_{j|k+1}^{(0)} = \boldsymbol{\beta}_{j|k}, \quad j = 1, \ldots, k, \qquad \boldsymbol{\gamma}_{k+1}^{(0)} = \boldsymbol{\gamma}_k.$$

For the initial estimate of $\boldsymbol{\beta}_{k+1}$, we take

$$\boldsymbol{\beta}_{k+1|k+1}^{(0)} = \arg\min_{\boldsymbol{\beta}_{k+1}} \|(\boldsymbol{y}_{k+1} - \boldsymbol{Z}_{k+1}\boldsymbol{\gamma}_k) - \boldsymbol{X}_{k+1}\boldsymbol{\beta}_{k+1}\|_2.$$

Compute the orthogonal factorization of $[\boldsymbol{X}_{k+1}, \boldsymbol{Z}_{k+1}] \in \mathscr{R}^{n_{k+1}\times(p_{k+1}+p_0)}$:

$$\begin{bmatrix} \boldsymbol{Q}_{k+1}^{\mathrm{T}} \\ \bar{\boldsymbol{Q}}_{k+1}^{\mathrm{T}} \end{bmatrix} [\boldsymbol{X}_{k+1} \quad \boldsymbol{Z}_{k+1}] = \begin{bmatrix} \boldsymbol{R}_{k+1} & \hat{\boldsymbol{R}}_{k+1} \\ 0 & \bar{\boldsymbol{R}}_{k+1} \end{bmatrix}, \tag{16}$$

where $[\boldsymbol{Q}_{k+1}, \bar{\boldsymbol{Q}}_{k+1}]$ is orthogonal, $\boldsymbol{R}_{k+1}$ is nonsingular upper triangular, and $(n_{k+1} - p_{k+1}) \times p_0$ $\bar{\boldsymbol{R}}_{k+1}$ is a full matrix if $n_{k+1} - p_{k+1} \leqslant p_0$, or upper triangular otherwise. From this we can easily obtain $\boldsymbol{\beta}_{k+1|k+1}^{(0)}$. Notice that if we only wanted to obtain $\boldsymbol{\beta}_{k+1|k+1}^{(0)}$, we would only need to compute the QR factorization of $\boldsymbol{X}_{k+1}$. But for later updating/downdating uses, we compute (16). After finding $\boldsymbol{\beta}^{(0)}_{k+1|k+1}$, we can obtain the active matrix

$$\boldsymbol{X}^{(0)}_{[k+1|k+1]} \equiv \begin{bmatrix} \boldsymbol{X}_{1|k} & & & & \boldsymbol{Z}_{1|k} \\ & \ddots & & & \vdots \\ & & \boldsymbol{X}_{k|k} & & \boldsymbol{Z}_{k|k} \\ & & & \boldsymbol{X}^{(0)}_{k+1|k+1} & \boldsymbol{Z}^{(0)}_{k+1|k+1} \end{bmatrix},$$

where $[\boldsymbol{X}^{(0)}_{k+1|k+1}, \boldsymbol{Z}^{(0)}_{k+1|k+1}]$ is the active matrix of $[\boldsymbol{X}_{k+1}, \boldsymbol{Z}_{k+1}]$ for the measurement equations obtained at time step $k+1$. Applying QR downdating techniques to (16), we obtain the orthogonal factorization of $[\boldsymbol{X}^{(0)}_{k+1|k+1}, \boldsymbol{Z}^{(0)}_{k+1|k+1}]$:

$$\begin{bmatrix} (\boldsymbol{Q}^{(0)}_{k+1|k+1})^{\mathrm{T}} \\ (\bar{\boldsymbol{Q}}^{(0)}_{k+1|k+1})^{\mathrm{T}} \end{bmatrix} [\boldsymbol{X}^{(0)}_{k+1|k+1}, \boldsymbol{Z}^{(0)}_{k+1|k+1}] = \begin{bmatrix} \boldsymbol{R}^{(0)}_{k+1|k+1} & \hat{\boldsymbol{R}}^{(0)}_{k+1|k+1} \\ \boldsymbol{0} & \bar{\boldsymbol{R}}^{(0)}_{k+1|k+1} \end{bmatrix}. \quad (17)$$

Then we have

$$\begin{bmatrix} \boldsymbol{Q}^{\mathrm{T}}_{[k|k]} & \\ & (\boldsymbol{Q}^{(0)}_{k+1|k+1})^{\mathrm{T}} \\ \bar{\boldsymbol{Q}}^{\mathrm{T}}_{[k|k]} & \\ & (\bar{\boldsymbol{Q}}^{(0)}_{k+1|k+1})^{\mathrm{T}} \end{bmatrix} \boldsymbol{X}^{(0)}_{[k+1|k+1]}$$

$$= \begin{bmatrix} \boldsymbol{R}_{1|k} & & & & \hat{\boldsymbol{R}}_{1|k} \\ & \ddots & & & \vdots \\ & & \boldsymbol{R}_{k|k} & & \hat{\boldsymbol{R}}_{k|k} \\ & & & \boldsymbol{R}^{(0)}_{k+1|k+1} & \hat{\boldsymbol{R}}^{(0)}_{k+1|k+1} \\ & & & & \tilde{\boldsymbol{R}}_{[k]} \\ & & & & \boldsymbol{0} \\ & & & & \bar{\boldsymbol{R}}^{(0)}_{k+1|k+1} \end{bmatrix}. \quad (18)$$

The next step is to compute the QR factorization:

$$(\tilde{\boldsymbol{Q}}^{(0)}_{[k+1]})^{\mathrm{T}} \begin{bmatrix} \tilde{\boldsymbol{R}}_{[k]} \\ \bar{\boldsymbol{R}}^{(0)}_{k+1|k+1} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{R}}^{(0)}_{[k+1]} \\ \boldsymbol{0} \end{bmatrix}. \quad (19)$$

Here again the Q-factor does not need to be stored. From (18) and (19), we see that there exists an orthogonal matrix $[\boldsymbol{Q}^{(0)}_{[k+1|k+1]}, \bar{\boldsymbol{Q}}^{(0)}_{[k+1|k+1]}]$ such that

$$\begin{bmatrix} (\boldsymbol{Q}^{(0)}_{[k+1|k+1]})^{\mathrm{T}} \\ (\bar{\boldsymbol{Q}}^{(0)}_{[k+1|k+1]})^{\mathrm{T}} \end{bmatrix} \boldsymbol{X}^{(0)}_{[k+1|k+1]} = \begin{bmatrix} \boldsymbol{R}_{1|k} & & & & \hat{\boldsymbol{R}}_{1|k} \\ & \ddots & & & \vdots \\ & & \boldsymbol{R}_{k|k} & & \hat{\boldsymbol{R}}_{k|k} \\ & & & \boldsymbol{R}^{(0)}_{k+1|k+1} & \hat{\boldsymbol{R}}^{(0)}_{k+1|k+1} \\ & & & & \tilde{\boldsymbol{R}}^{(0)}_{[k+1]} \\ & & & & \boldsymbol{0} \end{bmatrix},$$

leading to

$$
\begin{aligned}
&\boldsymbol{X}_{[k+1]}^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{\beta}^{(0)}_{[k+1|k+1]})\boldsymbol{X}_{[k+1]}\\
&\quad=\begin{bmatrix}
\boldsymbol{R}_{1|k}^{\mathrm{T}} & & & & \\
 & \ddots & & & \\
 & & \boldsymbol{R}_{k|k}^{\mathrm{T}} & & \\
 & & & (\boldsymbol{R}^{(0)}_{k+1|k+1})^{\mathrm{T}} & \\
\hat{\boldsymbol{R}}_{1|k}^{\mathrm{T}} & \cdots & \hat{\boldsymbol{R}}_{k|k}^{\mathrm{T}} & (\hat{\boldsymbol{R}}^{(0)}_{k+1|k+1})^{\mathrm{T}} & (\tilde{\boldsymbol{R}}^{(0)}_{[k+1]})^{\mathrm{T}}
\end{bmatrix}\\
&\quad\times\begin{bmatrix}
\boldsymbol{R}_{1|k} & & & & \hat{\boldsymbol{R}}_{1|k}\\
 & \ddots & & & \vdots\\
 & & \boldsymbol{R}_{k|k} & & \hat{\boldsymbol{R}}_{k|k}\\
 & & & \boldsymbol{R}^{(0)}_{k+1|k+1} & \hat{\boldsymbol{R}}^{(0)}_{k+1|k+1}\\
 & & & & \tilde{\boldsymbol{R}}^{(0)}_{[k+1]}
\end{bmatrix}.
\end{aligned}
$$

Now we can continue the iterations at time step $k+1$.

3.3. A modified Newton direction

From Section 3.1 we observe that in each iteration at time step k, the active matrix of $[\boldsymbol{X}_j, \boldsymbol{Z}_j]$ for any $j \leqslant k$ may need to be updated, and thus also its QR factorization (see (9)). The worst case is that at each iteration step each term on the left-hand side of (12) needs to be re-computed in computing the Cholesky factorization. This rarely happens, but the possibility of such a scenario makes the implementation complicated. Also we have to keep the Q-factors of the QR factorizations (9) for updating use. This may cause storage problem when k is large.

One modification is that we change the definition of the coefficient matrix of (7) to make the updating/downdating of its factorization much easier. Specifically, at the end of each time step, we freeze the active matrix corresponding to the measurement equations obtained at the current time step, i.e., this active matrix will not be updated any more in later time steps. Thus at time step $k+1$, the 'active' matrix $\boldsymbol{X}_{[k|k]}$ of $\boldsymbol{X}_{[k]}$, which was formed by all 'active' matrices obtained at each time step up to and including time step k, will not be updated, so that $\boldsymbol{R}_{j|k}$ and $\hat{\boldsymbol{R}}_{j|k}$ $(j=1,\ldots,k)$ and $\tilde{\boldsymbol{R}}_{[k]}$ in (18) will not be updated, and only the QR factorization (17) corresponding to the measurement equations obtained at time step $k+1$ needs updating and the QR factorization (19) needs re-computing during the iterations. Thus we do not need to hold the Q-factor of the QR factorization (9) of any previous time steps in memory. The above procedure changes the definition of the coefficient matrix of (7). But we would like to emphasize that the right-hand side of (7), which is determined by the given tuning constant and the current iterate, is still defined as before. This will ensure that $\boldsymbol{h}$ obtained by solving (7) with the redefined symmetric positive definite coefficient matrix is still a descent direction, which is now a modified Newton direction.

4. Numerical tests

In this section, we give some numerical test results to illustrate the performance of our algorithms. All computations were implemented using MATLAB.

We generated the test problems as follows. For $k = 1, 2, \ldots, 100$, we took

$$\begin{aligned} &\boldsymbol{X}_k : 20 \times 4 \text{ random matrix}, \quad \boldsymbol{Z}_k : 20 \times 10 \text{ random matrix}, \\ &\boldsymbol{\beta}_k = [1, \ldots, 1]^{\mathrm{T}} \in \mathscr{R}^4, \quad \gamma = [1, \ldots, 1]^{\mathrm{T}} \in \mathscr{R}^{10}, \end{aligned} \tag{20}$$

where the random matrices were generated by the MATLAB function `randn`. The measurement vector $\boldsymbol{y}_k$ was constructed first by

$$\boldsymbol{y}_k = \boldsymbol{X}_k \boldsymbol{\beta}_k + \boldsymbol{Z}_k \gamma + \boldsymbol{\epsilon}_k,$$

where the noise vector $\boldsymbol{\epsilon}_k$ was generated by `randn` multiplied by $\sigma = 0.01$, i.e., $\boldsymbol{\epsilon}_k \sim \sigma N(\mathbf{0}, \boldsymbol{I})$. Then two outliers $20\sigma N(0,1)$ were added to two elements of $\boldsymbol{y}_k$ which were randomly chosen for each k. The tuning constant c was chosen to be 1.5σ. We performed 1000 simulation runs, where each run had 100 time steps. The termination criterion for both the Newton method and the modified Newton method at time step k is $\|\boldsymbol{\beta}_{[k|k]}^{(i+1)} - \boldsymbol{\beta}_{[k|k]}^{(i)}\|_2 / \|\boldsymbol{\beta}_{[k|k]}^{(i)}\|_2 \leqslant 10^{-5}$, where i is the iteration step. For comparison, we also computed the LS estimate of $\boldsymbol{\beta}_k$ at time step k by a recursive LS method, which was easily developed based on Golub et al., 1979.

Fig. 1 displays the absolute errors (in 2-norm) in Huber's M-estimate (by the Newton method) and the LS estimate of $\boldsymbol{\beta}_k$ obtained at time step k (for $k = 1, \ldots, 100$) for one sample data (note the true value of $\boldsymbol{\beta}_k$ is given in (20)). It also gives the results for the LS estimates without the added outliers in the measurement vectors. Fig. 2 gives the *average* errors for 1000 replications. We can see that when there are outliers, Huber's M-estimate is

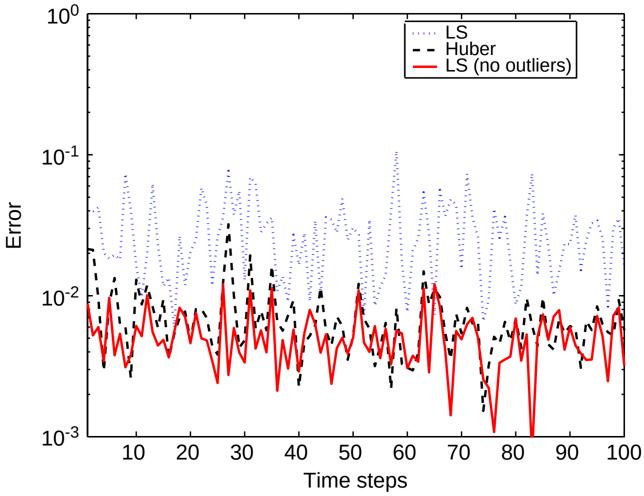

Fig. 1. Errors in the estimates of $\boldsymbol{\beta}_k$ at time step k (1 sample).

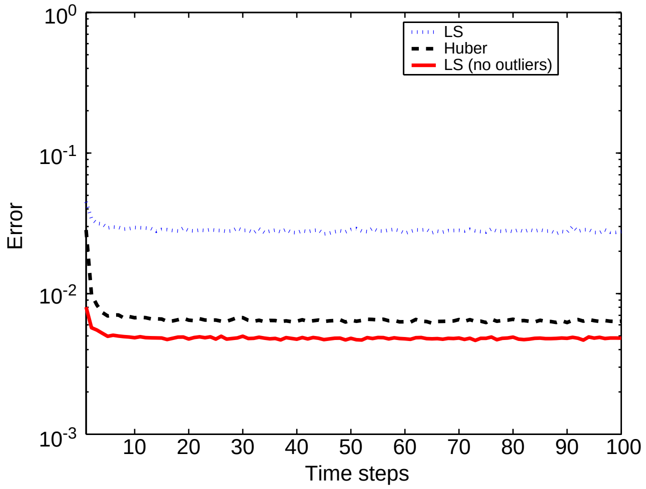

Fig. 2. Average errors in the estimates of $\boldsymbol{\beta}_k$ at time step k (1000 replications).

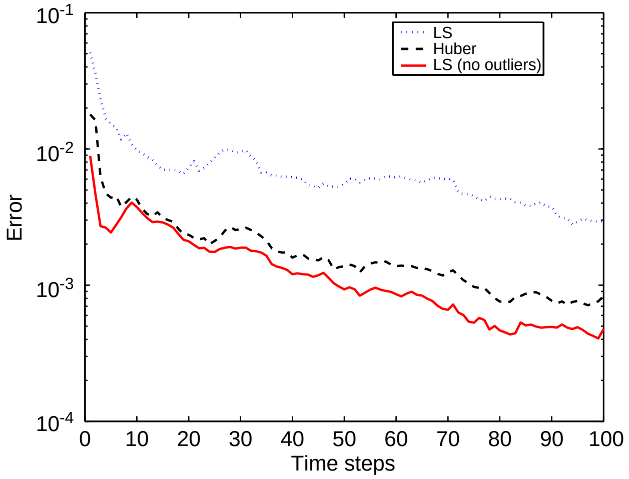

Fig. 3. Errors in the estimates of γ at time step k (1 sample).

more accurate than the corresponding LS estimate, and can significantly reduce the effect of outliers. In our simulations, we also tested different values of the outliers. The results indicated that the larger the outliers, the better Huber's M-estimates, compared with the LS estimates. From Figs. 1 and 2 we see that although Huber's M-estimate is not as good as the corresponding LS estimate without outliers, the former is a good approximation to the latter. The above observations can also be made from Figs. 3 and 4, which give errors in the three estimates for γ. But Fig. 3 shows that the errors for the three estimates of γ are much

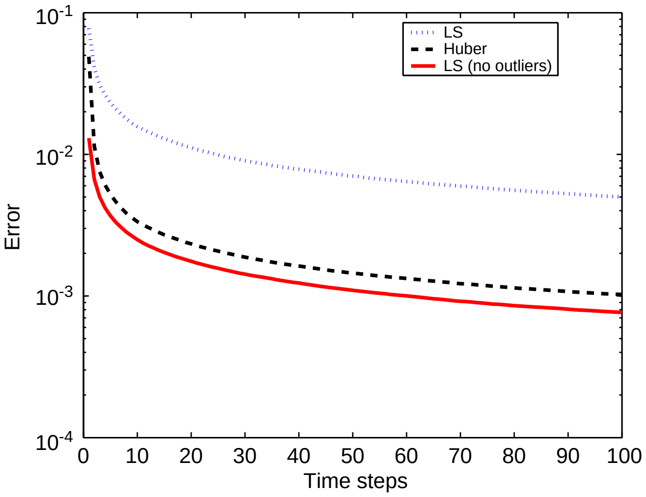

Fig. 4. Average errors in the estimates of γ at time step k (1000 replications).

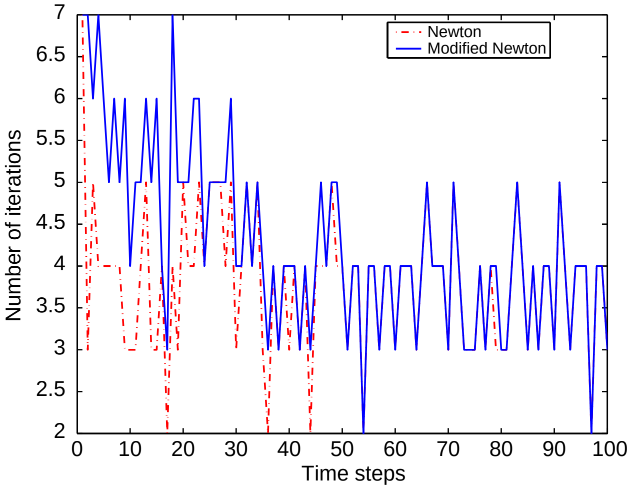

Fig. 5. Number of iterations (1 sample).

more smooth and tend to decrease with time step increasing. It is easy to understand this. Note that the vector γ is the same at each time step, while $\boldsymbol{\beta}_k$ are different for different time steps k. When more measurement equations are available, we can expect the estimates for γ to become more accurate. Similar observations to the above for the estimates of γ can be made for the estimates $\boldsymbol{\beta}_{j|k}$ $(k = j, j + 1, \ldots)$ of $\boldsymbol{\beta}_j$ for any fixed j.

Fig. 5 displays the number of iterations of the Newton method and its modified version for each time step for the one sample of data. Fig. 6 gives the average number of iterations of the

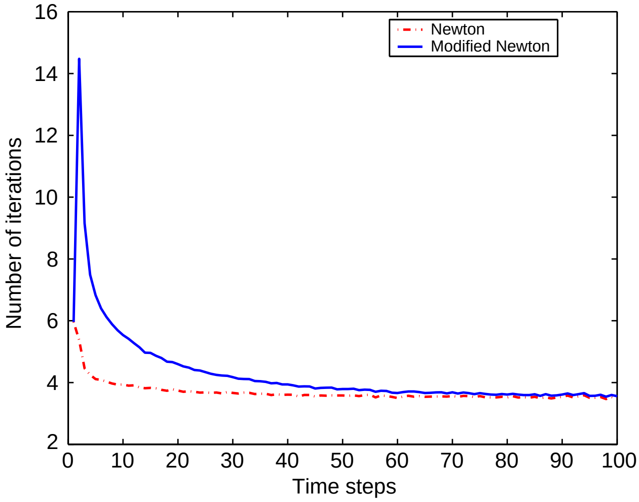

Fig. 6. Average number of iterations (1000 replications).

two methods for the 1000 replications. We see that for the initial few time steps, the modified method took more iterations (the two methods had the same number of iterations for the first time step), but later the difference between the two methods in terms of the number of iterations becomes smaller and smaller. After 50 time steps, there is almost no difference. From Section 3.3 we know that the modified Newton method has a few advantages over the Newton method. The former is much simpler, requires less storage, and also has the potential to cost much less in each iteration step. Figs. 5 and 6 suggest that the modified Newton method has no significant drawback, compared with the Newton method.

We also tested the case where the outliers were added to the same positions in the measurement vectors for all the time steps and observed no significant difference in the results compared with the previous case.

5. Summary

We considered applying Huber's M-estimation techniques to a block-angular linear model, which needs to be estimated recursively as more data accrues. The main cost of a Newton method is the computation of the Newton search direction. We showed how to take advantage of the structure of the block-angular matrices to efficiently compute this by using updating/downdating techniques for matrix factorizations. However, it is possible that factors from previous time steps might need updating/downdating. To reduce the computational burden and the storage requirement, we proposed to compute a modified Newton direction, which only needs updating/downdating of two simple QR factorizations involving the matrices at the current time step (see (17) and (19)). Numerical test results showed that the method with this modification took some extra iterations for only initial time steps, but later on it took almost the same number of iterations as the original method.

The test results also showed that Huber’s estimation can significantly reduce the influence of outliers for our estimation problem. In the future we would like to study the convergence of the modified Newton method and apply it to GPS.

Acknowledgements

The author would like to thank Ying Guo for carrying out the numerical simulations, and Chris Paige for his valuable comments and suggestions. The author is also grateful to two referees for their thoughtful and helpful suggestions which improved the paper.

References

Antoch, J., Ekblom, H., 1995. Recursive robust regression: computational aspect and comparison. Comput. Statist. Data Anal. 19, 115–234.

Björck, Å., 1996. Numerical Methods for Least Squares Problems, SIAM, Philadelphia.

Chang, X.-W., Paige, C.C., 2003. An algorithm for combined code and carrier phase based GPS positioning. BIT Numer. Math. 43, 915–927.

Chen, B., Pinar, M.C., 1998. On Newton’s method for Huber’s robust M-estimation problem in linear regression. BIT 38, 674–684.

Cox, M.G., 1990. The least-squares solution of linear equations with block-angular observation matrix. In: Cox, M.G., Hammarling, S.J. (Eds.), Reliable Numerical Computation. Oxford University Press, UK, pp. 227–240.

Golub, G.H., Luk, F.T., Pagano, M., 1979. A sparse least squares problem in photogrammetry. In: Gentleman, J.F. (Ed.), Proceedings of the Computer science and Statistics 12th Annual Symposium on the Interface. University of Waterloo, Canada, pp. 26–30.

Golub, G.H., Plemmons, R.J., 1980. Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition. Linear Algebra Appl. 34, 3–28.

Huber, P.J., 1981. Robust Statistics, Wiley, New York.

Huber, P.J., Dutter, R., 1974. Numerical solution of robust regression problem. In: Brushmann, G. (Ed.), COMPSTAT 1974 Proceedings of the Symposium on Computational Statistics. Physika Verlag, Vienna, pp. 165–172.

Madsen, K., Nielsen, H.B., 1990. Finite algorithms for robust linear regression. BIT 30, 682–699.

O’Leary, D.P., 1990. Robust regression computation using iteratively reweighted least squares. SIAM J. Matrix Anal. Appl. 11, 466–488.

Osborne, M.R., 1985. Finite Algorithms in Optimization and Data Analysis, Wiley, Toronto.