Huber's M-estimation in GPS Positioning: Computational Aspects

Xiao-Wen Chang and Ying Guo, *McGill University, Canada*

BIOGRAPHIES

Dr. Chang is Assistant Professor of Computer Science at McGill University. He holds a B.Sc and an M.Sc in Computational Mathematics from Nanjing University, China, and a Ph.D in Computer Science from McGill University. His main research area is scientific computing with particular emphasis on matrix computations and applications. He has been involved in GPS research since 1998 and his current interests include GPS positioning and integrity monitoring.

Ms. Guo is a System Administrator at McGill University, Network Communication Services. She holds a B.Sc and an M.Sc in physics from Wuhan University, China, and an M.Sc in computer science from McGill University. Her research interests include robust estimation and its application in GPS.

ABSTRACT

When GPS signal measurements have outliers, using least squares (LS) estimation will likely give poor position estimates. One of typical approaches to handling this problem is to use robust estimation techniques. In this paper, we study the computational issues of Huber's M-estimation applied to relative positioning. First for code based relative positioning, we use simulation results to show Newton's method usually converges faster than the iteratively reweighted least squares (IRLS) method, which is often used in geodesy for computing robust estimates of parameters. Then for code and carrier phase based relative positioning, we present a recursive modified Newton method to compute Huber's M-estimates of the positions. The structures of the model are exploited to make the method efficient. Numerical stability and storage issues are also taken into account in designing the numerical method. Simulation results are given to illustrate the effectiveness of the method.

1 INTRODUCTION

The typical approach for GPS positioning is the least squares (LS) estimation. But LS estimation is sensitive to outliers (unspecified large errors) in the measurements, thus it will likely give poor position estimates if the measurements have outliers. There are a few sources which can lead to outliers in the measurement data. One source is the satellite failure. The other sources include excessive multipath, ionospheric delay and diffraction, or the presence of interference or channel biases. In order to get good position estimates, the outliers have to be appropriately handled.

One approach is to apply the fault detection and identification procedure (see, e.g., [6] and [13]). This approach uses some detection & identification techniques to detect possible unspecified errors in the input data and isolate the faulty data. Then either the faulty data are discarded or the model for position estimation is adjusted to take the unspecified errors into account. But sometimes detection and identification by regular methods (see [6] for some typical methods based on LS residuals) are difficult, in particular when outliers appear in multiple observations. Furthermore, in the case of limited observation data available, the faulty data can not be simply discarded. Otherwise, the problem of rank deficiency may arise.

The other approach is to use robust estimation techniques. The aim of robust estimation is to reduce the influence of outliers on the parameter estimation. It automatically identifies outliers in the observations and give the corresponding observations less weight in estimation. Because of the good statistical properties and the relatively low computing effort, the M-estimation techniques, especially the famed Huber's M-estimation technique (see Huber [5]), is widely used in robust estimation. Using robust estimation techniques in GPS position estimation appears quite new, although the techniques have been used in geodesy for many years. Recently Yang, He and Xu in [16] pro-

posed an adaptive robust Kalman filter for position estimation, which is some kind of combination of an adaptive Kalman filter and Huber's M-estimation. To deal with the correlated double differenced measurements, the so-called dependent equivalent weight matrix based on Huber's weight function was used in their method (see [15] for more details). Their test results shows that the robust estimation can effectively resist the influence of the outliers. More recently Wieser and Brunner in [14] present a modified Danish method for short static positioning using double differenced carrier phase measurements. Since the standard Danish method is not applicable to correlated observations, it was modified to handle correlated double differenced phase observations. Their test results indicated that their method performs significantly better than the least squares method if unfavorable signal distortion occurs. Like many papers in robust M-estimation in geodesy, computer implementation of these methods were not adequately addresses in [16] and [14].

In this paper, we consider applying Huber's M-estimation technique to short baseline relative positioning by using the single differenced code and carrier phase measurements. We do not use any dynamic equations, since it is often difficult to get accurate dynamic equations in practice. Our main goal is to address the computer implementation issue. We will use code based short baseline relative positioning as an example to show Newton's method is usually faster than the iteratively reweighted least squares (IRLS) method, which is widely used in geodesy for robust M-estimation (see, e.g., [7, Sec 3.8.2], [8], [14]). For code and carrier phase based short baseline relative positioning, we will present a recursive modified Newton method for computing Huber's M-estimates. We make full use of the structure of the positioning model to make our method efficient. Numerical stability and storage requirement are also taken into account in designing the numerical method. Although our goal is not to show how significant Huber's M-estimation is to GPS positioning, we give some simulation results to demonstrate that Huber's M-estimation can give (much) better results than LS estimation when there are outliers.

The paper is organized as follows. In Section 2 we introduce Huber's M-estimation for a general linear model and discuss the strategies of handling a singularity problem. In Section 3 we introduce the mathematical models for short baseline relative positioning. In Section 4 we use simulation results to show Newton's method is faster than the IRLS method for Huber's M-estimation. In Section 5 for code and carrier phase based positioning we present a recursive modified Newton method for computing Huber's M-estimates, and give some simulation results. Finally we give a summary in Section 6.

Throughout this paper, we use bold lower case letters for vectors and bold upper case letters for matrices. The unit matrix will be denoted by $\boldsymbol{I}$ or sometimes by $\boldsymbol{I}_n$ if its dimension is n by n. The i-th column of the unit matrix is denoted by $\boldsymbol{e}_i$, while $\boldsymbol{e} \equiv (1, 1, \ldots, 1)^T$ (we use $\equiv$ to mean 'is defined to be'). We use the norm $\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^T\boldsymbol{x}}$ for vectors. $\mathcal{E}\{\cdot\}$ will denote the expected value, and $\mathrm{cov}\{\cdot\}$ will denote the covariance matrix, that is $\mathrm{cov}\{\boldsymbol{x}\} = \mathcal{E}\{(\boldsymbol{x} - \mathcal{E}\{\boldsymbol{x}\})(\boldsymbol{x} - \mathcal{E}\{\boldsymbol{x}\})^T\}$. $\boldsymbol{v} \sim \mathcal{N}(\bar{\boldsymbol{v}}, \boldsymbol{V})$ will indicate that $\boldsymbol{v}$ is a normally distributed random vector with expected value $\bar{\boldsymbol{v}}$ and covariance $\boldsymbol{V}$.

2 HUBER'S M-ESTIMATION FOR A GENERAL LINEAR MODEL

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

Suppose we have a general linear model

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{v}, \tag{1}$$

where $\boldsymbol{y} = [y_1, \ldots, y_m]^T \in \mathcal{R}^m$, $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m]^T \in \mathcal{R}^{m\times n}$, and $\mathrm{cov}\{\boldsymbol{v}\} = \sigma^2\boldsymbol{I}$ (σ is assumed to be be known). Write $\boldsymbol{r}(\boldsymbol{x}) \equiv \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}$, with ith element $r_i(\boldsymbol{x})$. Huber's M-estimation problem is the following optimization problem:

$$\min_{\boldsymbol{x}}\{F(\boldsymbol{x}) \equiv \sum_{i=1}^{m} \rho(r_i(\boldsymbol{x}))\}, \tag{2}$$

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

$$\rho(t) \equiv \begin{cases} \frac{1}{2}t^2, & |t| \le \gamma \\ \gamma|t| - \frac{1}{2}\gamma^2, & |t| > \gamma \end{cases} \tag{3}$$

defined for some tuning constant $\gamma > 0$. Often γ is chosen to be in the interval $[1.5\sigma, 2\sigma]$ (see Koch [7, Sec 3.8.3]). $\rho(t)$ in the form of (3) is called the Huber function. Note that Huber's M-estimation is a mixed l_2 and l_1 minimization problem. For M-estimation, 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 (3) that $\rho(t)$ is a continuous function, with continuous and nondecreasing first derivative, since

$$\rho'(t) = \begin{cases} t, & |t| \le \gamma, \\ \gamma\,\mathrm{sign}(t), & |t| > \gamma. \end{cases} \qquad \rho''(t) = \begin{cases} 1, & |t| \le \gamma, \\ 0, & |t| > \gamma. \end{cases} \tag{4}$$

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

The active index set and inactive set at $\boldsymbol{x} \in \mathcal{R}^n$ are respectively defined by

$$\nu(\boldsymbol{x}) \equiv \{i : |r_i(\boldsymbol{x})| \le \gamma\}, \quad \bar{\nu}(\boldsymbol{x}) \equiv \{i : |r_i(\boldsymbol{x})| > \gamma\}.$$

If $i \in \nu(\boldsymbol{x})$, we say that the i-th equation of the model (1) is active at $\boldsymbol{x}$, otherwise we say it is inactive at $\boldsymbol{x}$. The active matrix $\boldsymbol{A}^\nu(\boldsymbol{x})$ of $\boldsymbol{A}$ at $\boldsymbol{x}$ is defined to be the matrix formed by the rows of $\boldsymbol{A}$ corresponding to the active equations at $\boldsymbol{x}$. Sometimes for simplicity we just use $\boldsymbol{A}^\nu$ instead of $\boldsymbol{A}^\nu(\boldsymbol{x})$ if there is no any confusion. Define the sign vector

$$\boldsymbol{s}(\boldsymbol{x}) \equiv \begin{bmatrix} s_1(\boldsymbol{x}) \\ \vdots \\ s_m(\boldsymbol{x}) \end{bmatrix}, \quad s_i(\boldsymbol{x}) \equiv \begin{cases} -1, & r_i(\boldsymbol{x}) < -\gamma, \\ 0, & |r_i(\boldsymbol{x})| \le \gamma, \\ 1, & r_i(\boldsymbol{x}) > \gamma, \end{cases}$$

and the weight matrix

$$\begin{aligned} \boldsymbol{W}(\boldsymbol{x}) &\equiv \mathrm{diag}(w_1(\boldsymbol{x}), \ldots, w_m(\boldsymbol{x})), \\ w_i(\boldsymbol{x}) &\equiv 1 - s_i^2(\boldsymbol{x}) = \rho''(r_i(\boldsymbol{x})). \end{aligned}$$

The objective function $F(\boldsymbol{x})$ in (2) can then be written

$$F(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{r}(\boldsymbol{x})^T W(\boldsymbol{x})\boldsymbol{r}(\boldsymbol{x}) + \gamma \boldsymbol{s}(\boldsymbol{x})^T[\boldsymbol{r}(\boldsymbol{x}) - \frac{1}{2}\gamma \boldsymbol{s}(\boldsymbol{x})]. \tag{5}$$

Since $\frac{\partial \boldsymbol{r}(\boldsymbol{x})^T}{\partial \boldsymbol{x}} = -\boldsymbol{A}^T$, differentiating (5) gives the gradient of $F(\boldsymbol{x})$

$$\begin{aligned} F'(\boldsymbol{x}) \equiv \frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{x}} &= -\boldsymbol{A}^T(\boldsymbol{W}(\boldsymbol{x})\boldsymbol{r}(\boldsymbol{x}) + \gamma \boldsymbol{s}(\boldsymbol{x})) \\ &= -\sum_{i \in \nu(\boldsymbol{x})} \boldsymbol{a}_i r_i(\boldsymbol{x}) - \gamma \sum_{i \in \bar{\nu}(\boldsymbol{x})} \boldsymbol{a}_i s_i(\boldsymbol{x}). \end{aligned} \tag{6}$$

The symmetric nonnegative definite Hessian matrix is given by

$$\begin{aligned} F''(\boldsymbol{x}) \equiv \frac{\partial^2 F(\boldsymbol{x})}{\partial \boldsymbol{x} \partial \boldsymbol{x}^T} &= \boldsymbol{A}^T \boldsymbol{W}(\boldsymbol{x}) \boldsymbol{A} \\ &= \sum_{i \in \nu(\boldsymbol{x})} \boldsymbol{a}_i \boldsymbol{a}_i^T = \boldsymbol{A}^\nu(\boldsymbol{x})^T \boldsymbol{A}^\nu(\boldsymbol{x}). \end{aligned}$$

A general framework for Newton's method with a line search for solving (2) can be described as follows:

Given an initial estimate $\boldsymbol{x}$
repeat until convergence:
find the search direction $\boldsymbol{h}$ by solving

$$F''(\boldsymbol{x})\boldsymbol{h} = -F'(\boldsymbol{x}),$$

$$\text{or} \quad \boldsymbol{A}^\nu(\boldsymbol{x})^T \boldsymbol{A}^\nu(\boldsymbol{x})\boldsymbol{h} = \boldsymbol{A}^T[\boldsymbol{W}(\boldsymbol{x})\boldsymbol{r}(\boldsymbol{x}) + \gamma \boldsymbol{s}(\boldsymbol{x})], \tag{7}$$

perform a line search and update $\boldsymbol{x} := \boldsymbol{x} + \hat{\alpha}\boldsymbol{h}$.

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

Usually the least squares estimate $\boldsymbol{x}_{LS}$ for the model (1) is taken to be the initial estimate, so that if $\bar{\nu}(\boldsymbol{x}_{LS}) = \phi$ (no outliers),

$$\begin{aligned} &\boldsymbol{W}(\boldsymbol{x}_{LS}) = \boldsymbol{I}_m, \quad \boldsymbol{s}(\boldsymbol{x}_{LS}) = \boldsymbol{0}, \\ &\boldsymbol{A}^T[\boldsymbol{W}(\boldsymbol{x}_{LS})\boldsymbol{r}(\boldsymbol{x}_{LS}) + \gamma \boldsymbol{s}(\boldsymbol{x}_{LS})] = \boldsymbol{A}^T \boldsymbol{r}(\boldsymbol{x}_{LS}) = \boldsymbol{0}. \end{aligned}$$

Notice that $\boldsymbol{A}^T\boldsymbol{r}(\boldsymbol{x}_{LS}) = \boldsymbol{0}$ are the normal equations and from (7) we see $\boldsymbol{h} = 0$. Thus $\boldsymbol{x}_{LS}$ solves (2). Any $\boldsymbol{h}$ satisfying (7) with a *nonzero* $F'(\boldsymbol{x})$ is a *strictly* descent direction for the functional F at $\boldsymbol{x}$, since

$$\boldsymbol{h}^T F'(\boldsymbol{x}) = -\boldsymbol{h}^T \boldsymbol{A}^\nu(\boldsymbol{x})^T \boldsymbol{A}^\nu(\boldsymbol{x})\boldsymbol{h} < 0. \tag{8}$$

It can be shown that there is a minimizer $\boldsymbol{x}$ of $F(\boldsymbol{x})$ such that $\boldsymbol{A}^\nu(\boldsymbol{x})^T\boldsymbol{A}^\nu(\boldsymbol{x})$ is nonsingular (see, e.g., Madsen and Nielsen [9]). When $\boldsymbol{A}^\nu(\boldsymbol{x})^T\boldsymbol{A}^\nu(\boldsymbol{x})$ is nonsingular, it is positive definite and the Cholesky factorization of $\boldsymbol{A}^\nu(\boldsymbol{x})^T\boldsymbol{A}^\nu(\boldsymbol{x})$ can be used to solve (7). Specifically speaking, if $\boldsymbol{A}^\nu(\boldsymbol{x})^T\boldsymbol{A}^\nu(\boldsymbol{x})$ has the Cholesky factorization (see e.g., Björck [2, Sec 2.2.2])

$$\boldsymbol{A}^\nu(\boldsymbol{x})^T\boldsymbol{A}^\nu(\boldsymbol{x}) = \boldsymbol{R}^T\boldsymbol{R}, \tag{9}$$

where $\boldsymbol{R}$ is upper triangular, then $\boldsymbol{h}$ can easily be obtained by solving two triangular systems

$$\boldsymbol{R}^T\tilde{\boldsymbol{h}} = \boldsymbol{A}^T[\boldsymbol{W}(\boldsymbol{x})\boldsymbol{r}(\boldsymbol{x}) + \gamma \boldsymbol{s}(\boldsymbol{x})], \quad \boldsymbol{R}\boldsymbol{h} = \tilde{\boldsymbol{h}}.$$

The Cholesky factor $\boldsymbol{R}$ of $\boldsymbol{A}^\nu(\boldsymbol{x})^T\boldsymbol{A}^\nu(\boldsymbol{x})$ can also be obtained from the QR factorization of $\boldsymbol{A}^\nu(\boldsymbol{x})$. In fact, if $\boldsymbol{A}^\nu(\boldsymbol{x})$ has the QR factorization (see e.g., [2, Sec 1.3])

$$\begin{aligned} &\boldsymbol{A}^\nu(\boldsymbol{x}) = [\boldsymbol{Q}, \bar{\boldsymbol{Q}}] \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{Q}\boldsymbol{R}, \\ &[\boldsymbol{Q}, \bar{\boldsymbol{Q}}] \ \text{orthogonal}, \quad \boldsymbol{R} \ \text{upper triangular}, \end{aligned} \tag{10}$$

then we have

$$\boldsymbol{A}^\nu(\boldsymbol{x})^T\boldsymbol{A}^\nu(\boldsymbol{x}) = \boldsymbol{R}^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{R} = \boldsymbol{R}^T\boldsymbol{R}. \tag{11}$$

Since forming $\boldsymbol{A}^\nu(\boldsymbol{x})^T\boldsymbol{A}^\nu(\boldsymbol{x})$ may lose information (see e.g., [2, p.44]), for numerically stability, we prefer to find $\boldsymbol{R}$ by the QR factorization (10). The main computational cost in each iteration is then that of computing the QR factorization (10). Since the active matrices for two consecutive iterates usually differ by just

a few rows, updating/downdating techniques for the QR factorization can be used to compute (10) during the iterations for efficiency. For details of the updating/downdating of the QR factorization, see, e.g., [2, Sec 3.2].

If during the iterative process there are more than $m - n$ large residuals beyond the tuning constant γ, then the number of rows of $\boldsymbol{A}^\nu(\boldsymbol{x})$ is smaller than the number of its columns, so $\boldsymbol{A}^\nu(\boldsymbol{x})^T\boldsymbol{A}^\nu(\boldsymbol{x})$ is singular. There are several strategies to handle this problem. For example, in Antoch and Ekblom [1], if $\boldsymbol{A}^\nu(\boldsymbol{x})^T\boldsymbol{A}^\nu(\boldsymbol{x})$ is found to be singular, then it is replaced by $\boldsymbol{A}^\nu(\boldsymbol{x})^T\boldsymbol{A}^\nu(\boldsymbol{x}) + \epsilon\boldsymbol{I}$. The shortcoming with this approach 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 $\boldsymbol{A}^\nu(\boldsymbol{x})^T\boldsymbol{A}^\nu(\boldsymbol{x})$ to $\boldsymbol{A}^\nu(\boldsymbol{x})^T\boldsymbol{A}^\nu(\boldsymbol{x}) + \epsilon\boldsymbol{I}$, or vice versa. In O'Leary [10], 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 4 steps of the iteration. But no implementation details for this strategy were given in [10], nor was a guarantee given that this strategy would always work.

In this paper we propose the following strategy to handle the singularity problem. If the active matrix $\boldsymbol{A}^\nu(\boldsymbol{x})$ is not of full column rank, we choose the row vector from those $\boldsymbol{a}_i^T$ with $i \in \bar{\nu}(\boldsymbol{x})$ according to some criterion and add this row vector to $\boldsymbol{A}^\nu(\boldsymbol{x})$. We continue this process until the updated $\boldsymbol{A}^\nu(\boldsymbol{x})$ has full column rank. For simplicity, the updated $\boldsymbol{A}^\nu(\boldsymbol{x})$ is still called an active matrix. For a general matrix $\boldsymbol{A}$ such as that in GPS code based relative positioning (see Section 4), a sensible criterion is that the chosen row vector corresponds to the smallest residual in magnitude. In practice, we may know when the updated $\boldsymbol{A}^\nu(\boldsymbol{x})$ will become full column rank. For GPS code based relative positioning, when the updated $\boldsymbol{A}^\nu(\boldsymbol{x})$ becomes square, we can reasonably assume that it is nonsingular. For GPS code and carrier phase based relative positioning, the matrix $\boldsymbol{A}$ has some special structure, and we will show how to choose row vectors to make the updated $\boldsymbol{A}^\nu(\boldsymbol{x})$ have full column rank in Section 5. After a full column rank $\boldsymbol{A}^\nu(\boldsymbol{x})$ is found, we solve (7) for the search direction $\boldsymbol{h}$. Note that although now $\boldsymbol{A}^\nu(\boldsymbol{x})$ may not be the true active matrix at $\boldsymbol{x}$ any more, we still have (8) and thus $\boldsymbol{h}$ is a descent direction.

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

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

Since $\phi(\alpha)$ is the sum of nonnegative, convex, piecewise defined functions of α, with each piece being either quadratic or linear, see (2)–(4), 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 $\phi'(\alpha)$. An efficient method for computing $\hat{\alpha}$ can be found in Madsen and Nielsen [9].

The commonly used method for solving the robust M-estimation problem in geodesy is the IRLS method (see, e.g., Koch [7]). Note that (7) can also be written

$$\boldsymbol{A}^T\boldsymbol{W}(\boldsymbol{x})\boldsymbol{A}\boldsymbol{h} = \boldsymbol{A}^T[\boldsymbol{W}(\boldsymbol{x})\boldsymbol{r}(\boldsymbol{x}) + \gamma\boldsymbol{s}(\boldsymbol{x})]. \qquad (12)$$

If we replace $\boldsymbol{W}(\boldsymbol{x})$ on the left hand side of (12) by $\boldsymbol{D}(\boldsymbol{x}) = \mathrm{diag}(d_1, \ldots, d_m)$ with $d_i = 1$ if $|r_i(\boldsymbol{x})| \le \gamma$ or $d_i = \gamma/|r_i(\boldsymbol{x})|$ if $|r_i(\boldsymbol{x})| > \gamma$, then we can easily verify that (12) can be written

$$\boldsymbol{A}^T\boldsymbol{D}(\boldsymbol{x})\boldsymbol{A}\boldsymbol{x}_{\mathrm{new}} = \boldsymbol{A}^T\boldsymbol{D}(\boldsymbol{x})\boldsymbol{y}, \qquad (13)$$

where $\boldsymbol{x}_{\mathrm{new}} = \boldsymbol{x} + \boldsymbol{h}$. The iterative method in which the iteration sequence defined by (13) (i.e., $\boldsymbol{x}_{\mathrm{new}}$ is a new iterate) is just the IRLS method for Huber's M-estimation. Choosing different $\boldsymbol{D}(\boldsymbol{x})$ in (13) will lead other robust M-estimation methods, such as the Danish method. Although it is easy to understand and implement the IRLS method, we cannot use updating/downdating techniques to factorize $\boldsymbol{A}^T\boldsymbol{D}(\boldsymbol{x})\boldsymbol{A}$ quickly to solve the linear systems (13) during the iterations. Furthermore usually the IRLS method has a linear convergence rate, while Newton's method has quadratic convergence rate (see [11, Sec 5.4 & Sec 5.6]).

3 MATHEMATICAL MODELS

In this paper we consider relative positioning based on code and carrier phase measurements from L1. Ideally when the distance between the stationary receiver and roving receivers is short, the signals received by the two receivers from the same satellite has almost the same ionospheric refraction and tropospheric refraction. Then the single difference code and carrier phase measurement equations for satellite i at epoch k can be written (cf. [4])

$$\rho_k^i = (\boldsymbol{e}_k^i)^T\boldsymbol{x}_k + c\delta t_k + \mu_k^i, \qquad (14)$$

$$\phi_k^i = (\boldsymbol{e}_k^i)^T\boldsymbol{x}_k + c\delta t_k + \lambda N^i + \nu_k^i, \qquad (15)$$

where all terms are in the units of meters, and

- ρ_k^i is the single differenced code measurement;

- ϕ_k^i is the single differenced carrier phase measurement;
- $\boldsymbol{x}_k$ is the base line vector pointing from the stationary receiver to the roving receiver;
- $\boldsymbol{e}_k^i$ is the unit vector pointing from the midpoint of the baseline to satellite i;
- c is the speed of light;
- δt_k is the single differenced receiver clock error;
- λ is the wavelength of L1 carrier, $\lambda \approx 19$ cm;
- N^i is the single differenced ambiguity including the single differenced initial phase of the receiver generated signals;
- μ_k^i is single differenced noise including multipath for the code measurement;
- ν_k^i is single differenced noise including multipath for the carrier phase measurement.

Suppose there are m visible satellites. Define

$$\boldsymbol{y}_k^\rho = \begin{bmatrix} \rho_k^1 \\ \vdots \\ \rho_k^m \end{bmatrix}, \quad \boldsymbol{y}_k^\phi = \begin{bmatrix} \phi_k^1 \\ \vdots \\ \phi_k^m \end{bmatrix}, \quad \boldsymbol{E}_k = \begin{bmatrix} (\boldsymbol{e}_k^1)^T \\ \vdots \\ (\boldsymbol{e}_k^m)^T \end{bmatrix},$$

$$\beta_k = c\delta t_k, \quad \boldsymbol{a} = \lambda \begin{bmatrix} N^1 \\ \vdots \\ N^m \end{bmatrix}, \quad \boldsymbol{v}_k^\rho = \begin{bmatrix} \mu_k^1 \\ \vdots \\ \mu_k^m \end{bmatrix}, \quad \boldsymbol{v}_k^\phi = \begin{bmatrix} \nu_k^1 \\ \vdots \\ \nu_k^m \end{bmatrix}.$$

Then from (14) and (15) we have

$$\boldsymbol{y}_k^\rho = \boldsymbol{E}_k \boldsymbol{x}_k + \beta_k \boldsymbol{e} + \boldsymbol{v}_k^\rho, \tag{16}$$

$$\boldsymbol{y}_k^\phi = \boldsymbol{E}_k \boldsymbol{x}_k + \beta_k \boldsymbol{e} + \boldsymbol{a} + \boldsymbol{v}_k^\phi, \tag{17}$$

where as in GPS literature, we assume

$$\boldsymbol{v}_k^\rho \sim \mathcal{N}(\boldsymbol{0}, \sigma_\rho^2 \boldsymbol{I}_m), \quad \boldsymbol{v}_k^\phi \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2 \boldsymbol{I}_m),$$

and $\boldsymbol{v}_k^\rho, \boldsymbol{v}_l^\rho, \boldsymbol{v}_k^\phi$, and $\boldsymbol{v}_l^\phi$ are uncorrelated to each other for any epochs k and l ($k \neq l$). Notice that in $\boldsymbol{E}_k$, $\boldsymbol{e}_k^i$ depends on the baseline vector $\boldsymbol{x}_k$. Once $\boldsymbol{x}_k$ is known, $\boldsymbol{E}_k$ will be known. In other words, the above measurement equations are nonlinear. But $\boldsymbol{E}_k$ is not sensitive to changes in $\boldsymbol{x}_k$, since the distance from a satellite to any of the two receivers is far larger than the baseline. So we can use an approximation to $\boldsymbol{x}_k$, say, our estimate of $\boldsymbol{x}_{k-1}$ at epoch $k-1$, to compute the approximation to $\boldsymbol{E}_k$. Usually this approximation is good enough. But if we want, after obtaining an estimate of $\boldsymbol{x}_k$ at epoch k, we can recompute $\boldsymbol{E}_k$, and do more iterations if necessary. For the first epoch, we set the initial estimate of $\boldsymbol{x}_1$ to be a zero vector and then compute the corresponding $\boldsymbol{E}_1$ (i.e., we take $\boldsymbol{e}_1^i$ to be the unit vector pointing from the stationary receiver to satellite i). Since $\boldsymbol{E}_1$ computed by this initial $\boldsymbol{x}_1$ is not very accurate, it is recomputed one more time when the new estimate of $\boldsymbol{x}_1$ is obtained from our computational method. So from now on, we just assume all $\boldsymbol{E}_k$ are known.

4 COMPUTATION FOR CODE BASED POSITIONING

In this section, we apply Newton's method to code based relative positioning directly. The purpose of this section is to use simulation results to show that for computing Huber's M-estimates Newton's method converges faster than the IRLS method, which is often used in geodesy, and demonstrate that Huber's estimation gives better position estimates than the LS estimation when there are outliers in the measurements.

All our computations were performed in MATLAB 6.5 on a Celeron PC running Windows XP. The 24 GPS satellite constellation data in YUMA ephemeris format for the week of June 30th to July 6th 1998 was used in the simulations. The roving receiver was assumed to be on board an aircraft circling horizontally with center directly above the reference station at a constant speed of $100\,m/s$. The baseline was 1 km. For each epoch, a set of code measurement data from 8 satellites were used. The time interval between two consecutive epochs was 1 second. At each epoch, the single differenced code measurements were constructed by

$$\boldsymbol{y}_k^\rho = \bar{\boldsymbol{y}}_k + c\delta t_k \boldsymbol{e} + \boldsymbol{v}_k^\rho + \boldsymbol{b}^\rho, \tag{18}$$

where the i-th component of $\bar{\boldsymbol{y}}_k$ is the difference of the true range between satellite i and the stationary receiver and the true range between satellite i and the roving receiver, δt_k is the difference between the clock offsets of the stationary receiver and the roving receiver, both which were modeled by white noise input to a second order Markov process based on [12, p.417], $\boldsymbol{v}_k^\rho \sim \mathcal{N}(\boldsymbol{0}, \sigma_\rho^2 \boldsymbol{I})$ with $\sigma_\rho = 1m$, and $\boldsymbol{b}^\rho$ is the outlier vector.

We used the IRLS method and Newton's method to compute Huber's M-estimates of the positions. For consistency, both methods used the same stopping criterion: the iteration process stops when the difference between the position estimates at two consecutive iteration steps is less than $0.1\,m$. Thus the position estimates computed with both methods have the same numerical accuracy. The tuning constant γ was set to be 1.5. We used 8 visible satellites and took the outlier vector $\boldsymbol{b}^\rho = [0, 12, 0, 0, 8, 0, 0, 0]^T$ for each epoch. In order to reduce random effect on the results, we

performed 100 simulation runs for the same satellite geometry, and each run had 200 epochs.

Figure 1 displays the *average* number of iterations for Newton's method and the IRLS method at each epoch. From this figure, we see indeed Newton's method converges much faster than the IRLS method.

Figure 2 shows the *average* position error in Huber's M-estimates at each epoch, which were computed by Newton's method, and the *average* position errors in the LS estimates which were computed by using the QR factorization method (see e.g., [2, Sec 2.4]). For comparison, Figure 2 also gives the average position errors in the LS estimates when the measurements do not have the outliers. We observe that Huber's M-estimation can reduce the effect of outliers—giving better position estimates than the LS estimation, although the estimates are not as good as the LS estimates without outliers.

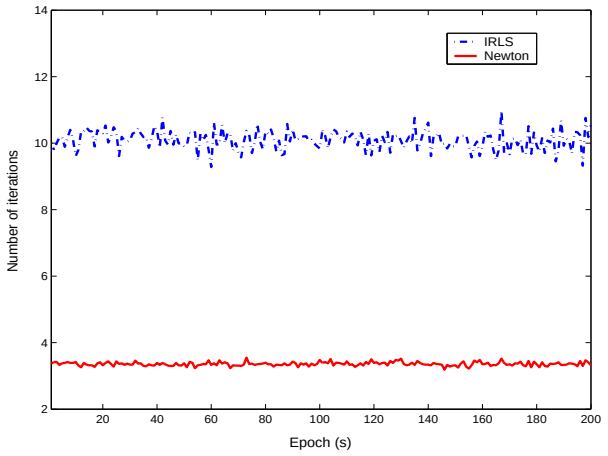

Figure 1: Average number of iterations

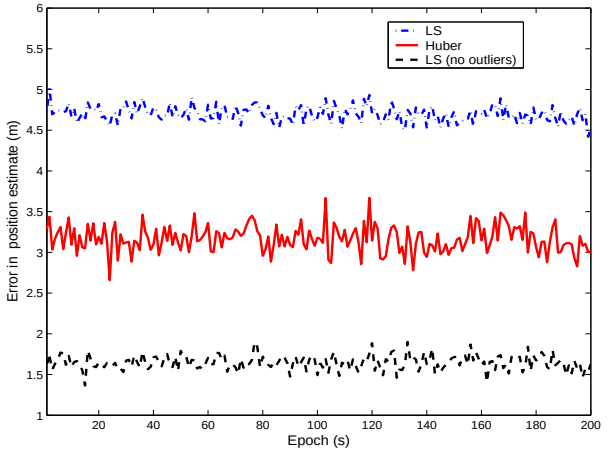

Figure 2: Average errors in the position estimates

5 COMPUTATION FOR CODE AND CARRIER PHASE BASED POSITIONING

5.1 Numerical method

In order to make the code noise vector have the same covariance matrix as the carrier phase noise vector, multiply (16) by $\sigma \equiv \sigma_\phi/\sigma_\rho$, and then combine it with (17), leading to

$$\begin{bmatrix} \boldsymbol{y}_k^\phi \\ \sigma \boldsymbol{y}_k^\rho \end{bmatrix} = \begin{bmatrix} \boldsymbol{e} & \boldsymbol{E}_k \\ \sigma \boldsymbol{e} & \sigma \boldsymbol{E}_k \end{bmatrix} \begin{bmatrix} \beta_k \\ \boldsymbol{x}_k \end{bmatrix} + \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{a} + \begin{bmatrix} \boldsymbol{v}_k^\phi \\ \sigma \boldsymbol{v}_k^\rho \end{bmatrix}, \quad (19)$$

where $\begin{bmatrix} \boldsymbol{v}_k^\phi \\ \sigma \boldsymbol{v}_k^\rho \end{bmatrix} \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2 \boldsymbol{I}_{2m})$. Define

$$\boldsymbol{y}_k \equiv \begin{bmatrix} \boldsymbol{y}_k^\phi \\ \sigma \boldsymbol{y}_k^\rho \end{bmatrix}, \quad \boldsymbol{B}_k \equiv \begin{bmatrix} \boldsymbol{e} & \boldsymbol{E}_k \\ \sigma \boldsymbol{e} & \sigma \boldsymbol{E}_k \end{bmatrix},$$
$$\boldsymbol{z}_k \equiv \begin{bmatrix} \beta_k \\ \boldsymbol{x}_k \end{bmatrix}, \quad \boldsymbol{C} \equiv \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{v}_k \equiv \begin{bmatrix} \boldsymbol{v}_k^\phi \\ \sigma \boldsymbol{v}_k^\rho \end{bmatrix}.$$

Then (19) can be rewritten

$$\boldsymbol{y}_k = \boldsymbol{B}_k \boldsymbol{z}_k + \boldsymbol{C}\boldsymbol{a} + \boldsymbol{v}_k, \quad \boldsymbol{v}_k \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2 \boldsymbol{I}_{2m}). \quad (20)$$

Combining these equations for $k = 1, 2, \ldots$ gives

$$\begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{k-1} \\ \boldsymbol{y}_k \end{bmatrix} = \left[\begin{array}{ccc|c} \boldsymbol{B}_1 & & & \boldsymbol{C} \\ & \ddots & & \vdots \\ & & \boldsymbol{B}_{k-1} & \boldsymbol{C} \\ & & & \boldsymbol{C} \end{array}\right] \left[\begin{array}{c} \boldsymbol{z}_1 \\ \vdots \\ \boldsymbol{z}_{k-1} \\ \boldsymbol{z}_k \\ \hline \boldsymbol{a} \end{array}\right] + \begin{bmatrix} \boldsymbol{v}_1 \\ \vdots \\ \boldsymbol{v}_{k-1} \\ \boldsymbol{v}_k \end{bmatrix}, \quad (21)$$

or equivalently (with obvious notation)

$$\boldsymbol{y}_{[k]} = [\boldsymbol{B}_{[k]}, \boldsymbol{C}_{[k]}] \begin{bmatrix} \boldsymbol{z}_{[k]} \\ \boldsymbol{a} \end{bmatrix} + \boldsymbol{v}_{[k]} \equiv \boldsymbol{A}_{[k]} \boldsymbol{x}_{[k]} + \boldsymbol{v}_{[k]}. \quad (22)$$

Suppose there are at least 4 visible satellites at any epoch. It is reasonable to assume that any 4 rows of $[\boldsymbol{e}, \boldsymbol{E}_j]$ for any j is nonsingular (this assumption will be used later). Of course $[\boldsymbol{e}, \boldsymbol{E}_j]$ will have full column rank. Then it is straightforward to show that $\boldsymbol{A}_{[k]}$ has full column rank.

At an estimate of $\boldsymbol{x}_{[k]}$, we can find the corresponding active matrix of $\boldsymbol{A}_{[k]}$. For convenience, we also call $[\boldsymbol{B}_j^\nu, \boldsymbol{C}^\nu]$ which is formed by the rows of $[\boldsymbol{B}_j, \boldsymbol{C}]$ corresponding to the active equations the active matrix of $[\boldsymbol{B}_j, \boldsymbol{C}]$ at the estimate, and call $\boldsymbol{B}_j^\nu$ the active matrix of $\boldsymbol{B}_j$ at the estimate.

For each epoch k, we could directly apply Newton's method to (22) to compute Huber's M-estimate of $\boldsymbol{x}_{[k]}$. But obviously this is not computationally efficient. At

epoch k, when we compute Huber's M-estimate of $\boldsymbol{x}_{[k]}$, we should use the results available at epoch $k-1$. Also the structure of $\boldsymbol{A}_{[k]}$ should be carefully exploited. In Chang [3], a recursive modified Newton method is proposed to compute Huber's M-estimates for a model more general than (21), which may arise from other applications. We would like to apply the techniques presented in [3] to the model (21), where $\boldsymbol{B}_j$ and $\boldsymbol{C}$ have some special structures.

From the general framework for Newton's method, we see that at epoch k the main cost of computing Huber's M-estimate is the cost of finding the R-factor of the QR factorization of the active matrix $\boldsymbol{A}_{[k]}^{\nu}$ (see (10)) in each iteration step. As we mentioned in Section 2, for computational efficiency, updating and downdating techniques of the QR factorization of the active matrix $\boldsymbol{A}_{[k]}^{\nu}$ have to be used. During the iterations at epoch k, a measurement equation at any previous epoch may become an active equation from an inactive equation or vice versa. This makes updating/downdating of the QR factorization of the active matrices much complicated. Furthermore the Q-factors of all $[\boldsymbol{B}_j, \boldsymbol{C}]$ have to be stored for updating and downdating use at later epochs. When k is large, this may cause a computer memory problem. Thus we would like to modify Newton's method. When we compute Huber's M-estimate of $\boldsymbol{x}_{[k]}$ at epoch k, we do not update the active matrix of $[\boldsymbol{B}_j, \boldsymbol{C}]$ at Huber's M-estimate of $\boldsymbol{x}_{[j]}$ obtained at epoch j for $j = 1, \ldots, k-1$, and only update the active matrix of $[\boldsymbol{B}_k, \boldsymbol{C}]$ at each iterate. In other words, in each iteration step at epoch k we will use a modified active matrix (whose first $k-1$ row blocks will not be changed during the iterations) to replace the true active matrix $\boldsymbol{A}_{[k]}^{\nu}$ at the iterate. Then updating/downdating of the QR factorization of the modified active matrix will be much simpler than that of the actual active matrix and may cost much less. The details of the computation will be given later. With the above modification we can easily show that we will still obtain a descent direction at iterates (unless the convergence has been reached), since we actually use a new symmetric positive definite matrix to replace the left hand side of (7) without changing the definition of the right hand side of (7).

For later use, here we describe the computation of the QR factorization of $[\boldsymbol{B}_k, \boldsymbol{C}]$. Notice that $[\boldsymbol{B}_k, \boldsymbol{C}]$ has a special structure, which should be used in the computation for efficiency. Define the orthogonal matrix

$$\boldsymbol{G} \equiv \begin{bmatrix} c\boldsymbol{I} & -s\boldsymbol{I} \\ s\boldsymbol{I} & c\boldsymbol{I} \end{bmatrix},$$

where

$$\gamma \equiv \sqrt{1+\sigma^2}, \quad c \equiv 1/\gamma, \quad s \equiv \sigma/\gamma.$$

Then multiplying $[\boldsymbol{B}_k, \boldsymbol{C}]$ by $\boldsymbol{G}^T$ from the left gives

$$\boldsymbol{G}^T \left[\boldsymbol{B}_k \mid \boldsymbol{C}\right] = \left[\begin{array}{cc|c} c\boldsymbol{I} & s\boldsymbol{I} & \\ -s\boldsymbol{I} & c\boldsymbol{I} & \end{array}\right] \left[\begin{array}{cc|c} \boldsymbol{e} & \boldsymbol{E}_k & \boldsymbol{I} \\ \sigma\boldsymbol{e} & \sigma\boldsymbol{E}_k & \boldsymbol{0} \end{array}\right] = \left[\begin{array}{cc|c} \gamma\boldsymbol{e} & \gamma\boldsymbol{E}_k & c\boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} & -s\boldsymbol{I} \end{array}\right]. \tag{23}$$

Use the Householder transformations (see e.g., [2, Sec 2.3]) to compute the QR factorization:

$$\boldsymbol{H}_k^T \left[\begin{array}{cc|c} \gamma\boldsymbol{e} & \gamma\boldsymbol{E}_k & c\boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} & -s\boldsymbol{I} \end{array}\right] = \left[\begin{array}{cc|c} \boldsymbol{R}_k & & \hat{\boldsymbol{R}}_k \\ \boldsymbol{0} & & \bar{\boldsymbol{R}}_k \end{array}\right], \tag{24}$$

where $\boldsymbol{H}_k \in \mathcal{R}^{2m \times 2m}$ is the product of $m+4$ Householder transformations, $\boldsymbol{R}_k$ is a 4×4 nonsingular upper triangular matrix, and $\bar{\boldsymbol{R}}_k$ is a $(2m-4) \times m$ upper triangular matrix. Here the Householder transformations can be implemented to take advantage of the the structure of the matrix. Combining (23) and (24) gives the QR factorization of $[\boldsymbol{B}_k, \boldsymbol{C}]$:

$$\left[\begin{array}{c} \boldsymbol{Q}_k^T \\ \hline \bar{\boldsymbol{Q}}_k^T \end{array}\right] \left[\boldsymbol{B}_k \mid \boldsymbol{C}\right] = \left[\begin{array}{cc|c} \boldsymbol{R}_k & & \hat{\boldsymbol{R}}_k \\ \hline \boldsymbol{0} & & \bar{\boldsymbol{R}}_k \end{array}\right], \quad [\boldsymbol{Q}_k, \bar{\boldsymbol{Q}}_k] \equiv \boldsymbol{G}\boldsymbol{H}_k, \tag{25}$$

where $[\boldsymbol{Q}_k, \bar{\boldsymbol{Q}}_k]$ is orthogonal, and $\boldsymbol{Q}_k^T$ and $[\boldsymbol{R}_k, \hat{\boldsymbol{R}}_k]$ have the same number of rows. Here the Q-factor $[\boldsymbol{Q}_k, \bar{\boldsymbol{Q}}_k]$ needs to be formed and stored. It will be used during the iteration process at epoch k. But after epoch k, it will not be used any more and can be discarded.

In the following we will discuss the computation at epoch k.

First we consider the initial case $k = 1$. Note that the model is

$$\begin{bmatrix} \boldsymbol{y}_1^{\phi} \\ \sigma\boldsymbol{y}_1^{\rho} \end{bmatrix} = \begin{bmatrix} \boldsymbol{e} & \boldsymbol{E}_1 & \boldsymbol{I} \\ \sigma\boldsymbol{e} & \sigma\boldsymbol{E}_1 & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \boldsymbol{x}_1 \\ \boldsymbol{a} \end{bmatrix} + \begin{bmatrix} \boldsymbol{v}_1^{\phi} \\ \sigma\boldsymbol{v}_1^{\rho} \end{bmatrix},$$

$$\text{or} \quad \boldsymbol{y}_1 = [\boldsymbol{B}_1, \boldsymbol{C}] \begin{bmatrix} \boldsymbol{z}_1 \\ \boldsymbol{a} \end{bmatrix} + \boldsymbol{v}_{[1]} = \boldsymbol{A}_{[1]}\boldsymbol{x}_{[1]} + \boldsymbol{v}_{[1]}.$$

We can apply Newton's method given in Section 2 to the above model directly to compute Huber's M-estimate, with the initial estimate taken to be the LS estimate, which can be computed by the QR factorization of $[\boldsymbol{B}_1, \boldsymbol{C}]$ (cf. (25)). From the structure of the matrix $\boldsymbol{C}$, we observe that Huber's M-estimation cannot reduce the influence of outliers in carrier phase measurements, since outliers will be absorbed into the unknown ambiguities, i.e., we cannot distinguish outliers and ambiguities (This is also true for $k > 1$ if all carrier phase measurement equations at different epochs have the same outlier vector). So the residuals for the carrier phase measurement equations at an

iterate are likely small and all the carrier phase measurement equations are likely active. Also in theory there is Huber's estimate at which the active matrix of $\boldsymbol{A}_{[1]}$ has full column rank (see the statement after equation (8)), so all carrier phase measurement equations must be active at the estimate, otherwise from the structure of $\boldsymbol{C}$ we observe that the active matrix will have at least one zero column. In our implementation, we force the active matrix at any iterate to include all the m row vectors corresponding to the m carrier phase measurement equations. If the number of rows of an active matrix is less than $m+4$ (the number of its columns), then the active matrix does not have full column rank and we use the strategy proposed in Section 2 to handle this singularity problem. Specifically we add to the active matrix the row vectors corresponding to the code measurement equations which are not active and have the smallest residuals such that the new active matrix (which is actually not true active matrix) becomes square, then it is easy to show it is nonsingular under our earlier assumption that any 4 rows of $[\boldsymbol{e}, \boldsymbol{E}_j]$ is nonsingular.

Now we consider the general case $k > 1$. Let Huber's M-estimate of $\boldsymbol{x}_{[j]}$ at any epoch j be denoted by $\boldsymbol{x}_{[j|j]} = [\boldsymbol{z}_{1|j}^T, \ldots, \boldsymbol{z}_{j|j}^T, \boldsymbol{a}_j^T]^T$ and let the active matrix of $[\boldsymbol{B}_j, \boldsymbol{C}]$ at $\boldsymbol{x}_{[j|j]}$ by $[\boldsymbol{B}_{j|j}, \boldsymbol{C}_j]$. At the end of epoch $k-1$, we obtain

$$\begin{aligned}\boldsymbol{A}_{[k-1|k-1]} &\equiv [\boldsymbol{B}_{[k-1|k-1]}, \boldsymbol{C}_{[k-1|k-1]}] \\ &\equiv \begin{bmatrix} \boldsymbol{B}_{1|1} & & & \boldsymbol{C}_1 \\ & \ddots & & \vdots \\ & & \boldsymbol{B}_{k-1|k-1} & \boldsymbol{C}_{k-1} \end{bmatrix}, \end{aligned} \tag{26}$$

where we assume that $\boldsymbol{B}_{j|j}$ (for $j = 1, \ldots, k-1$) has full column rank (we will discuss how to ensure this later), so $\boldsymbol{A}_{[k-1|k-1]}$ has full column rank. For computational efficiency, we will use $\boldsymbol{A}_{[k-1|k-1]}$ in lieu of true active matrices of $\boldsymbol{A}_{[k-1]}$ at later iterates for computing descent directions, and will not update $\boldsymbol{A}_{[k-1|k-1]}$ any more.

Suppose at the end of epoch $k-1$, we have obtained the QR factorization of $\boldsymbol{A}_{[k-1|k-1]}$:

$$\begin{aligned}&\begin{bmatrix} \boldsymbol{Q}_{[k-1|k-1]}^T \\ \hline \bar{\boldsymbol{Q}}_{[k-1|k-1]}^T \end{bmatrix} \boldsymbol{A}_{[k-1|k-1]} \\ &= \begin{bmatrix} \boldsymbol{R}_{1|1} & & & \hat{\boldsymbol{R}}_{1|1} \\ & \ddots & & \vdots \\ & & \boldsymbol{R}_{k-1|k-1} & \hat{\boldsymbol{R}}_{k-1|k-1} \\ \hline & & & \tilde{\boldsymbol{R}}_{[k-1]} \\ & & & \boldsymbol{0} \end{bmatrix}, \end{aligned} \tag{27}$$

where $[\boldsymbol{Q}_{[k-1|k-1]}, \bar{\boldsymbol{Q}}_{[k-1|k-1]}]$ is orthogonal, and $\boldsymbol{R}_{j|j}$ for $j = 1, \ldots, k-1$ and $\tilde{\boldsymbol{R}}_{[k-1]}$ are nonsingular upper triangular. Here the Q-factor $[\boldsymbol{Q}_{[k-1|k-1]}, \bar{\boldsymbol{Q}}_{[k-1|k-1]}]$ is not used in computation and so is not formed and stored.

At epoch k, let the initial estimate for $\boldsymbol{x}_{[k]}$ be denoted by

$$\boldsymbol{x}_{[k|k]}^{(0)} \equiv [(\boldsymbol{z}_{1|k}^{(0)})^T, \ldots, (\boldsymbol{z}_{k-1|k}^{(0)})^T, (\boldsymbol{z}_{k|k}^{(0)})^T, (\boldsymbol{a}_k^{(0)})^T]^T.$$

Naturally we take

$$\boldsymbol{z}_{j|k}^{(0)} = \boldsymbol{z}_{j|k-1}, \quad j = 1, \ldots, k-1, \quad \boldsymbol{a}_k^{(0)} = \boldsymbol{a}_{k-1}.$$

For the initial estimate of $\boldsymbol{z}_k$, noting that the measurement equations at epoch k are $\boldsymbol{y}_k = \boldsymbol{B}_k \boldsymbol{z}_k + \boldsymbol{C}\boldsymbol{a} + \boldsymbol{v}_k$, we take

$$\boldsymbol{z}_{k|k}^{(0)} = \arg\min_{\boldsymbol{z}_k} \|(\boldsymbol{y}_k - \boldsymbol{C}\boldsymbol{a}_{k-1}) - \boldsymbol{B}_k \boldsymbol{z}_k\|_2. \tag{28}$$

Using the QR factorization (25), we can easily find $\boldsymbol{z}_{k|k}^{(0)}$ in (28) by solving the upper triangular system

$$\boldsymbol{R}_k \boldsymbol{z}_{k|k}^{(0)} = \boldsymbol{Q}_k^T (\boldsymbol{y}_k - \boldsymbol{C}\boldsymbol{a}_{k-1}).$$

After finding $\boldsymbol{z}_{k|k}^{(0)}$, we can obtain the modified active matrix of $\boldsymbol{A}_{[k]}$ at $\boldsymbol{x}_{[k|k]}^{(0)}$:

$$\boldsymbol{A}_{[k|k]}^{(0)} \equiv \begin{bmatrix} \boldsymbol{B}_{1|1} & & & & \boldsymbol{C}_1 \\ & \ddots & & & \vdots \\ & & \boldsymbol{B}_{k-1|k-1} & & \boldsymbol{C}_{k-1} \\ & & & \boldsymbol{B}_{k|k}^{(0)} & \boldsymbol{C}_k^{(0)} \end{bmatrix}, \tag{29}$$

where $[\boldsymbol{B}_{k|k}^{(0)}, \boldsymbol{C}_k^{(0)}]$ is the active matrix of $[\boldsymbol{B}_k, \boldsymbol{C}_k]$ at $\boldsymbol{x}_{[k|k]}^{(0)}$. If $[\boldsymbol{B}_{k|k}^{(0)}, \boldsymbol{C}_k^{(0)}]$ corresponds to measurements from less than 4 different satellites, then from the structure of $\boldsymbol{B}_k$ (note $\boldsymbol{B}_k = \left[\begin{smallmatrix} \boldsymbol{e} & \boldsymbol{E}_k \\ \sigma\boldsymbol{e} & \sigma\boldsymbol{E}_k \end{smallmatrix}\right]$) we can conclude that $\boldsymbol{B}_{k|k}^{(0)}$ does not have full column rank. So $\boldsymbol{A}_{[k|k]}^{(0)}$ does not have full column rank. In order to handle this problem, we use the strategy proposed in Section 2. Specifically, we add to the active matrix $[\boldsymbol{B}_{k|k}^{(0)}, \boldsymbol{C}_k^{(0)}]$ the row vectors from the rest of $[\boldsymbol{B}_k, \boldsymbol{C}]$ corresponding to other satellites and smallest residuals so that the row vectors of the updated $[\boldsymbol{B}_{k|k}^{(0)}, \boldsymbol{C}_k^{(0)}]$ corresponds to 4 different satellites. Then we can easily show that the updated $\boldsymbol{B}_{k|k}^{(0)}$ has full column rank under the earlier assumption that any 4 rows of $[\boldsymbol{e}, \boldsymbol{E}_j]$ for any j is nonsingular. In our later iterations at epoch k, we always use this strategy to ensure that the active matrices of $\boldsymbol{B}_k$ have full column rank. We then use $\boldsymbol{A}_{[k|k]}^{(0)}$ in lieu of the true active matrix of $\boldsymbol{A}_{[k]}$ at $\boldsymbol{x}_{[k|k]}^{(0)}$ for computing the search direction $\boldsymbol{h}_{[k]}^{(0)}$ at $\boldsymbol{x}_{[k|k]}^{(0)}$, i.e.,

we would like to solve the following linear system for $\boldsymbol{h}_{[k]}^{(0)}$ (cf. (7)):

$$\begin{aligned}&\big(\boldsymbol{A}_{[k|k]}^{(0)}\big)^T\boldsymbol{A}_{[k|k]}^{(0)}\boldsymbol{h}_{[k]}^{(0)}\\ =&\big(\boldsymbol{A}_{[k|k]}^{(0)}\big)^T\big[\boldsymbol{W}(\boldsymbol{x}_{[k|k]}^{(0)})\cdot\boldsymbol{r}\big(x_{[k|k]}^{(0)}\big)+\gamma\boldsymbol{s}\big(x_{[k|k]}^{(0)}\big)\big].\end{aligned}\tag{30}$$

In order to solve (30), we seek the QR factorization of $\boldsymbol{A}_{[k|k]}^{(0)}$. Applying the QR downdating technique to (25), we can obtain the QR factorization of $[\boldsymbol{B}_{k|k}^{(0)},\boldsymbol{C}_k^{(0)}]$:

$$\left[\begin{array}{c}(\boldsymbol{Q}_{k|k}^{(0)})^T\\ \hline (\bar{\boldsymbol{Q}}_{k|k}^{(0)})^T\end{array}\right]\left[\boldsymbol{B}_{k|k}^{(0)}\;\middle|\;\boldsymbol{C}_k^{(0)}\right]=\left[\begin{array}{c|c}\boldsymbol{R}_{k|k}^{(0)} & \hat{\boldsymbol{R}}_{k|k}^{(0)}\\ \hline \boldsymbol{0} & \bar{\boldsymbol{R}}_{k|k}^{(0)}\end{array}\right].\tag{31}$$

Then using the QR factorizations (27) and (31), we obtain from (29) that

$$\begin{aligned}&\left[\begin{array}{cc}\boldsymbol{Q}_{[k-1|k-1]}^T & \\ & (\boldsymbol{Q}_{k|k}^{(0)})^T\\ \hline \bar{\boldsymbol{Q}}_{[k-1|k-1]}^T & \\ & (\bar{\boldsymbol{Q}}_{k|k}^{(0)})^T\end{array}\right]\boldsymbol{A}_{[k|k]}^{(0)}\\ &=\left[\begin{array}{ccc}\boldsymbol{R}_{1|1} & & \hat{\boldsymbol{R}}_{1|1}\\ \ddots & & \vdots\\ \boldsymbol{R}_{k-1|k-1} & & \hat{\boldsymbol{R}}_{k-1|k-1}\\ & \boldsymbol{R}_{k|k}^{(0)} & \hat{\boldsymbol{R}}_{k|k}^{(0)}\\ \hline & & \tilde{\boldsymbol{R}}_{[k-1]}\\ & & \boldsymbol{0}\\ & & \bar{\boldsymbol{R}}_{k|k}^{(0)}\end{array}\right].\end{aligned}\tag{32}$$

The next step is to compute the following QR factorization by Householder transformations:

$$(\tilde{\boldsymbol{Q}}_{[k]}^{(0)})^T\left[\begin{array}{c}\tilde{\boldsymbol{R}}_{[k-1]}\\ \bar{\boldsymbol{R}}_{k|k}^{(0)}\end{array}\right]=\left[\begin{array}{c}\tilde{\boldsymbol{R}}_{[k]}^{(0)}\\ \boldsymbol{0}\end{array}\right].\tag{33}$$

In the implementation, the special structures of $\tilde{\boldsymbol{R}}_{[k-1]}$ and $\bar{\boldsymbol{R}}_{k|k}^{(0)}$ are used for computational efficiency. Here again the Q-factor $\tilde{\boldsymbol{Q}}_{[k]}$ does not need to be formed and stored. From (32) and (33), we see that there exists an orthogonal matrix $[\boldsymbol{Q}_{[k|k]}^{(0)},\bar{\boldsymbol{Q}}_{[k|k]}^{(0)}]$ such that

$$\left[\begin{array}{c}(\boldsymbol{Q}_{[k|k]}^{(0)})^T\\ \hline (\bar{\boldsymbol{Q}}_{[k|k]}^{(0)})^T\end{array}\right]\boldsymbol{A}_{[k|k]}^{(0)}=\left[\begin{array}{ccc}\boldsymbol{R}_{1|1} & & \hat{\boldsymbol{R}}_{1|1}\\ \ddots & & \vdots\\ \boldsymbol{R}_{k-1|k-1} & & \hat{\boldsymbol{R}}_{k-1|k-1}\\ & \boldsymbol{R}_{k|k}^{(0)} & \hat{\boldsymbol{R}}_{k|k}^{(0)}\\ \hline & & \tilde{\boldsymbol{R}}_{[k]}^{(0)}\\ & & \boldsymbol{0}\end{array}\right].\tag{34}$$

This is the QR factorization of $\boldsymbol{A}_{[k|k]}^{(0)}$. Here we do not form or store the Q-factor. Solving (30) by using the QR factorization (34), we obtain the search direction $\boldsymbol{h}_{[k]}^{(0)}$. Then applying the line search technique, we get the next iterate $\boldsymbol{x}_{[k|k]}^{(1)}$.

Now we can continue the iteration process. After obtaining $\boldsymbol{x}_{[k|k]}^{(1)}$, we can determine the active matrix $[\boldsymbol{B}_{k|k}^{(1)},\boldsymbol{C}_k^{(1)}]$ of $[\boldsymbol{B}_k,\boldsymbol{C}]$ at $\boldsymbol{x}_{[k|k]}^{(1)}$. The modified active matrix $\boldsymbol{A}_{[k|k]}^{(1)}$ of $\boldsymbol{A}_{[k|k]}$ at $\boldsymbol{x}_{[k|k]}^{(1)}$ is then obtained by replacing $[\boldsymbol{B}_{k|k}^{(1)},\boldsymbol{C}_k^{(1)}]$ at the bottom of $\boldsymbol{A}_{[k|k]}^{(0)}$ in (29) with $[\boldsymbol{B}_{k|k}^{(1)},\boldsymbol{C}_k^{(1)}]$. Then we just repeat the process in the previous paragraph to get a new iterate $\boldsymbol{x}_{[k|k]}^{(2)}$. When the iteration converges, we finally obtain the Huber's M-estimate at epoch k: $\boldsymbol{x}_{[k|k]}=[\boldsymbol{z}_{1|k}^T,\ldots,\boldsymbol{z}_{k|k}^T,\boldsymbol{a}_k^T]^T$. Then we start the computation for epoch $k+1$.

Here we give a remark. In (21) all measurement equations from epoch 1 to epoch k are used for estimation at epoch k. We obtain not only the estimate for the position at epoch k, but also the estimates for positions at all previous epochs—this is called smoothing. The computation cost at each epoch increases when k increases. So does the storage requirement. This may cause problems, especially for real time applications, when k becomes too large. In practice, after certain epochs, we may use only a fixed number of latest measurements for positioning. Our method with some modification can still be applied.

5.2 Simulation results

To demonstrate the performance of our method, we give some simulation results here. The simulation scenario is the same as that described in Section 4. The code measurements were constructed as (18) and carrier phase measurements were constructed as follows:

$$\boldsymbol{y}_k^\phi=\bar{\boldsymbol{y}}_k+\boldsymbol{e}\beta_k+\boldsymbol{a}+\boldsymbol{v}_k^\phi+\boldsymbol{b}^\phi,$$

where $\bar{\boldsymbol{y}}_k$ and β_k are the same as given in (18), the i-th element of $\boldsymbol{a}$ is λN^i with N^i being the difference of the number of integer cycles between satellite i and the roving receiver and that between satellite i and the stationary receiver at epoch 1, $\boldsymbol{v}_k^\phi\sim\mathcal{N}(\boldsymbol{0},\sigma_\phi^2\boldsymbol{I})$ with $\sigma_\phi=0.01m$, and $\boldsymbol{b}^\phi$ is the outlier vector.

We used the modified Newton method to compute Huber's M-estimates of the positions. In each epoch, the iteration process stops when the difference between the position estimates at two consecutive iteration steps is less than $0.001\,m$. The tuning constant γ was still set to be 1.5. For comparison, we also computed the LS

estimates of the positions by a recursive LS method proposed in [4].

We used 8 visible satellites and took the outlier vectors $\boldsymbol{b}^{\rho} = [0, 10, 0, 0, 0, 0, 0, 0]^T$ and $\boldsymbol{b}^{\phi} = [0, 0.1, 0, 0, 0, 0, 0, 0]^T$ for each epoch. One exception was that $\boldsymbol{b}^{\phi}$ was not added to the single differenced carrier phase measurements in the first epoch, since otherwise the Huber's M-estimation could not reduce the influence of the outliers in the carrier phase measurements for the reason we mentioned in Section 5.1. In order to reduce random effect on the results, we performed 100 simulation runs for the same satellite geometry, and each run had 500 epochs.

Figure 3 shows the *average* position errors in Huber's M-estimates and in LS estimates at each epoch. As for the code based relative positioning, we observe that Huber's M-estimation can reduce the effect of outliers and give better position estimates than the LS estimation. Both errors tends to decrease with epoch increasing. Figure 3 also gives the *average* smoothed position errors, where the smoothed position estimate at epoch k ($k < 500$) is the estimate of the position obtained at epoch 500. As it is expected, the smoothed position estimates are better than the corresponding regular position estimates, since more information was used by the former. We see that the smoothed position estimates at different epochs have almost the same accuracy. This is because the same estimate of the ambiguity vector was involved in the estimation of these positions.

Figure 4 displays the *average* number of iterations for the modified Newton method. For the initial a few epochs, the number of iterations is relatively large. But it drops dramatically later. For most epochs, it takes only 2 to 3 iterations.

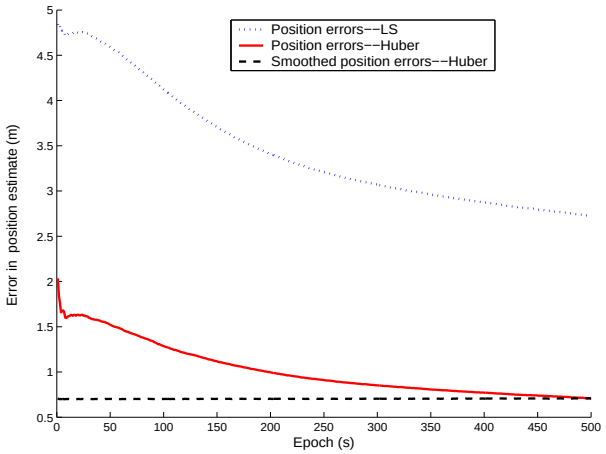

Figure 3: Average errors in the position estimates

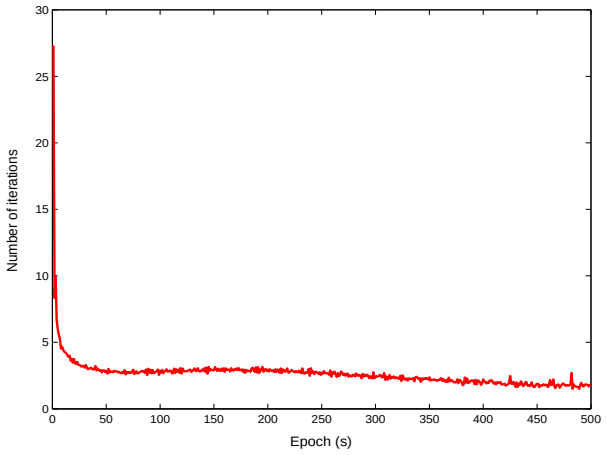

Figure 4: Average number of iterations for the modified Newton method

6 SUMMARY

The IRLS method is often used in Geodesy for robust M-estimation. For Huber's M-estimation, we proposed to use Newton's method instead, since updating/downdating techniques can be used to solve linear systems quickly during the iterations and also it usually converges faster than the IRLS method. We showed by simulations that for code based relative positioning, Newton's method indeed converges faster than the IRLS method. For code and carrier phase based relative positioning, we showed how to use the structures of the model to design an efficient modified Newton method for recursively computing Huber's M-estimates. When we designed the numerical method, we also took numerical stability and storage requirement into account. Simulation results indicated the proposed numerical methods are effective. In the future, we will consider applying the proposed methods to real data and test the effectiveness of the methods to real applications.

ACKNOWLEDGMENTS

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

References

[1] J. Antoch and H. Ekblom, *Recursive robust regression: computational aspect and comparison*, Com-

putational Statistics & Data Analysis, 19 (1995), pp. 115–234.

[2] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[3] X.-W. Chang , *Computation of the Huber M-estimator for a sparse structured problem*, Technical Report SOCS-3.7, School of Computer Science, McGill University, 2003, 13 pages.

[4] X.-W. Chang and C. C. Paige, *An algorithm for combined code and carrier phase based GPS positioning*, to appear in BIT Numerical Mathematics, 13 pages.

[5] P. J. Huber, *Robust Statistics*, Wiley, New York, 1981.

[6] The Institute of Navigation, Global Positioning System, Vol. 5, Papers published in Navigation, Alexandria, VA, 1998.

[7] K. R. Koch, *Parameter Estimation and Hypothesis Testing in Linear Models*, 2nd ed., Springer, Berlin, 1999.

[8] K. R. Koch and Y. Yang, *Robust Kalman filter for rank deficient observation models*, Journal of Geod., Vol. 72, pp. 436-441, 1998.

[9] K. Madsen and H. B. Nielsen, *Finite algorithms for robust linear regression*, BIT, 30 (1990), pp. 682–699.

[10] D. P. O'Leary, *Robust regression computation using iteratively reweighted least squares*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 466–488.

[11] M. R. Osborne, *Finite Algorithms in Optimization and Data Analysis*, John Wiley & Sons, Toronto, 1985.

[12] B. W. Parkinson, J. J. Spilker, P. Axelrad, P. Enge (Ed.), *Global Positioning System: Theory and Applications*, Vol. 1 and 2, American Institute of Aeronautics and Astronautics, Inc, 1996.

[13] J. G. Teunissen, *Quality control in integrated navigation systems*, IEEE Trans. Aerosp. Electr. System, Vol. 5, pp. 35-41, 1990.

[14] A. Wieser and F.K. Brunner, *Short static GPS sessions: robust estimation results*, GPS Solutions, 5 (2002), pp. 70–79.

[15] Y. Yang, *Robust estimation for dependent observations*, Manuscr. Geod., Vol. 19, pp. 10-17, 1994.

[16] Y. Yang, H. He and G. Xu, *Adaptively robust filtering for kinematic geodetic positioning*, Journal of Geodesy, Vol. 75, pp. 109-116, 2001.