Code and Carrier Phase Based Short Baseline GPS Positioning: Computational Aspects*

Xiao-Wen Chang, Christopher C. Paige, Lan Yin
School of Computer Science, McGill University, Montreal, Quebec, Canada H3A 2A7

Abstract

A recursive least squares algorithm is presented for short baseline GPS positioning using both carrier phase and code measurements. We take advantage of the structure of the problem to make the algorithm computationally efficient and use orthogonal transformations to ensure that the algorithm is numerically reliable. Details are given for computing position estimates and error covariance matrices with possible satellite rising and setting. Real data test results suggest our algorithm is effective.

1 Introduction

This paper is concerned with *kinematic* relative GPS positioning using both carrier phase and code measurements. Using these two types of measurement for positioning is not new, and much research has been conducted in this area. For example, Hatch [6], Ashjaee [1] and Goad [4] studied how to generate the so-called carrier-smoothed code measurements by using carrier phase and code measurements. Kleusberg [10], Hwang and Brown [8], Teunissen [13], Jin [9] and Tiberius [15] investigated how to use carrier phase and code measurements in GPS kinematic positioning. In [15] different combinations of these two types of measurement with different frequencies were considered. Most methods assume that a dynamic model for the roving receiver is available, and then use the Kalman filter. The main differences among these methods are that different information is assumed to be sent from the stationary receiver to the roving receiver, and different dynamic models are used.

Many methods given in the GPS literature do not address the computer implementation issue, which is crucial for software design. Although some do consider it, they usually consider only some of its aspects. Basically it has three aspects: numerical reliability, computational efficiency, and storage efficiency. In [3] we presented a recursive least squares (LS) approach for carrier phase based positioning. The approach is computationally efficient because it makes full use of the structure of the mathematical model and is numerically reliable because orthogonal transformations are used. Storage efficiency is also taken into account in this approach. The goal of this paper is to extend this approach to the combined case where both carrier phase and code measurements are used and refined models for the measurement equations are adopted. In this paper we assume the carrier phase and code measurements of the stationary receiver are available at the roving receiver. We do not use Kalman filtering, because

*This research was supported by NSERC of Canada Grant RGPIN217191-99, FCAR of Quebec Grant 2001-NC-66487, and NSERC-GEOIDE Network Project ENV#14 for Xiao-Wen Chang, and by NSERC of Canada Grant RGPIN9236-01 for Christopher C. Paige.

for many practical applications it is hard to model the dynamic behavior of the roving receiver. We believe that artificially including a dynamic model may degrade the estimation performance, while the additional computation in each epoch is unnecessarily expensive. However when a sufficiently accurate dynamic model *is* known, the approach given here can be extended to handle it. We consider only the L1 carrier since many receivers can only receive the L1 signal. But it is easy to extend our approach to the dual frequency case.

This paper is organized as follows. In Section 2 we give the mathematical model we use for position estimation. In Section 3 we show how to effectively use orthogonal transformations to make full use of the structure of the model to recursively compute the LS estimates of the positions and the corresponding error covariance matrices, and how to handle satellites rising and setting. In Section 4 we give the results of tests with real data. Finally a summary is given in Section 5.

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 $\boldsymbol{I}$ and its i-th column by $\boldsymbol{e}_i$, while $\boldsymbol{e} \equiv (1, 1, \ldots, 1)^T$ (we use $\equiv$ to mean 'is defined to be'). $\boldsymbol{I}_n$ will denote the $n \times n$ unit matrix. We use the 2-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 The mathematical model

Here we give the mathematical model for position estimation. Suppose there are two receivers s and r. The receiver s is *stationary* and its position is known, while the receiver r is *roving* and its position is to be determined. We want to find the baseline vector $\boldsymbol{x}$, i.e., the vector pointing from receiver s to receiver r. If the baseline is known, the position of the roving receiver will be known. In this paper, we assume the baseline is short (say, shorter than 10 kilometers).

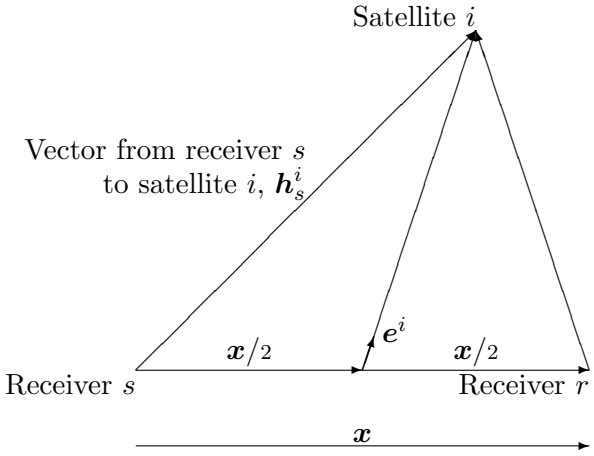


Figure 1: Geometry for two Receivers and one Satellite.

We need the following quantities to describe the geometry, that is, the relative positions of satellites

and receivers (see Figure 1):

$\boldsymbol{h}_s^i$	the vector from receiver s to satellite i,
$\boldsymbol{e}^i$	the *unit* vector from the midpoint of the baseline to satellite i,
ρ_s^i	the range in *wavelengths* from receiver s to satellite i,
λ	the wavelength (for the L1 carrier signal used here, $\lambda \approx 19cm$).

Note that superscripts indicate satellites, subscripts receivers. Since

$$\boldsymbol{x} = \boldsymbol{h}_s^i - \boldsymbol{h}_r^i, \qquad \boldsymbol{e}^i = \frac{\boldsymbol{h}_s^i - \boldsymbol{x}/2}{\|\boldsymbol{h}_s^i - \boldsymbol{x}/2\|} = \frac{\boldsymbol{h}_s^i + \boldsymbol{h}_r^i}{\|2\boldsymbol{h}_s^i - \boldsymbol{x}\|},$$

we have

$$(\|2\boldsymbol{h}_s^i - \boldsymbol{x}\|\boldsymbol{e}^i)^T\boldsymbol{x} = \|\boldsymbol{h}_s^i\|^2 - \|\boldsymbol{h}_r^i\|^2 = (\|\boldsymbol{h}_s^i\| - \|\boldsymbol{h}_r^i\|)(\|\boldsymbol{h}_s^i\| + \|\boldsymbol{h}_r^i\|) = \lambda(\rho_s^i - \rho_r^i)(\|\boldsymbol{h}_s^i\| + \|\boldsymbol{h}_s^i - \boldsymbol{x}\|),$$

giving

$$(\omega^i\boldsymbol{e}^i)^T\boldsymbol{x} = \lambda(\rho_s^i - \rho_r^i), \tag{1}$$

$$\omega^i \equiv \frac{\|2\boldsymbol{h}_s^i - \boldsymbol{x}\|}{\|\boldsymbol{h}_s^i\| + \|\boldsymbol{h}_s^i - \boldsymbol{x}\|}, \qquad \omega^i\boldsymbol{e}^i = \frac{2\boldsymbol{h}_s^i - \boldsymbol{x}}{\|\boldsymbol{h}_s^i\| + \|\boldsymbol{h}_s^i - \boldsymbol{x}\|}. \tag{2}$$

Notice that ω^i is close to 1, and we could replace it by 1 for many applications. But for some high precision GPS applications we may still want to keep ω^i. Furthermore, the cost of computing ω^i is negligible. The true baseline vector $\boldsymbol{x}$ will not be known, but we will see that at each step we will have an estimate of it, and this will be used to evaluate $\omega^i\boldsymbol{e}^i$ above.

Suppose the signal from satellite i arrives at receiver s at time t_k and its travel time is τ_s^i. At t_k the carrier phase measurement $\phi_s^i(t_k)$ (in wavelengths) and the code measurement $\tilde{\rho}_s^i(t_k)$ (in wavelengths) at receiver s for satellite i are (cf. [14, Sec. 5.2.2] or [15, Sec. 3.4])

$$\begin{aligned}\phi_s^i(t_k) = {} & \rho_s^i(t_k) - I_s^i(t_k) + T_s^i(t_k) + N_s^i + f[\delta t_s(t_k) - \delta t^i(t_k - \tau_s^i)] + D_s(t_k) - D^i(t_k - \tau_s^i) \\ & + \phi_s(t_0) - \phi^i(t_0) + \mu_s^i(t_k), \end{aligned} \tag{3}$$

$$\tilde{\rho}_s^i(t_k) = \rho_s^i(t_k) + I_s^i(t_k) + T_s^i(t_k) + f[\delta t_s(t_k) - \delta t^i(t_k - \tau_s^i)] + d_s(t_k) - d^i(t_k - \tau_s^i) + \nu_s^i(t_k), \tag{4}$$

where the "units" of each of the terms in the above equations is "number of wavelengths",

- $\phi_s^i(t_k)$: the carrier phase *measurement* at time t_k.
- $\tilde{\rho}_s^i(t_k)$: the code measurement at time t_k.
- $\rho_s^i(t_k)$: the range between receiver s at time t_k and satellite i at time $t_k - \tau_s^i$.
- $I_s^i(t_k)$: the ionospheric range error at time t_k.
- $T_s^i(t_k)$: the tropospheric range error at time t_k.
- N_s^i: the integer ambiguity.
- f: the frequency of the L1 carrier.
- $\delta t_s(t_k)$: the receiver clock error at time t_k.
- $\delta t^i(t_k - \tau_s^i)$: the satellite clock error at time $t_k - \tau_s^i$.

- $D_s(t_k)$: the receiver hardware delay for the carrier phase measurement at time t_k.
- $D^i(t_k - \tau_s^i)$: the satellite hardware delay for the carrier phase measurement at time $t_k - \tau_s^i$.
- $d_s(t_k)$: the receiver hardware delay for the code measurement at time t_k.
- $d^i(t_k - \tau_s^i)$: the satellite hardware delay for the code measurement at time $t_k - \tau_s^i$.
- $\phi_s(t_0)$: the initial phase of the receiver generated carrier signal at the initial time t_0.
- $\phi^i(t_0)$: the initial phase of the satellite generated carrier signal at the initial time t_0.
- $\mu_s^i(t_k)$: the carrier phase measurement noise, including multipath error, at time t_k.
- $\nu_s^i(t_k)$: the code measurement noise, including multipath error, at time t_k.

The ionospheric range error I_s^i, the tropospheric range error T_s^i and the satellite clock error δt^i can be modeled, see e.g. [11, Secs 4.4 & 5.3]. If they have been, we simply assume that (3)–(4) is the model after the corrections have been applied, and that the error terms in (3)–(4) are now the corresponding modeling errors. For simplified carrier phase and code measurement equations which do not consider the initial phases and hardware delays, see e.g. [11, Sec. 4.1].

Subtracting the carrier phase measurement equation corresponding to receiver r from (3) and noticing that

$$-[I_s^i(t_k)-I_r^i(t_k)]+[T_s^i(t_k)-T_r^i(t_k)]-f[\delta t^i(t_k-\tau_s^i)-\delta t^i(t_k-\tau_r^i)]-[D^i(t_k-\tau_s^i)-D^i(t_k-\tau_r^i)]-[\phi^i(t_0)-\phi^i(t_0)]$$

will be negligible since the baseline is short, we obtain the single differenced *carrier phase* measurement equation

$$\begin{aligned}\phi_s^i(t_k) - \phi_r^i(t_k) = \rho_s^i(t_k) - \rho_r^i(t_k) + N_s^i - N_r^i + f\delta t_s(t_k) - f\delta t_r(t_k) + D_s(t_k) - D_r(t_k) \\ + \phi_s(t_0) - \phi_r(t_0) + \mu_s^i(t_k) - \mu_r^i(t_k).\end{aligned} \tag{5}$$

In order to eliminate the number of dependent unknowns, we can simply combine the receiver clock errors, the receiver hardware delays, and the initial phases of the receiver generated carrier signals together. Notice that receivers s and r occur in every equation, so we can drop these indices and then indicate the time epoch k by subscript k. Thus by defining

$$\begin{gathered}\phi_k^i \equiv \phi_s^i(t_k) - \phi_r^i(t_k), \qquad N^i \equiv N_s^i - N_r^i, \qquad \mu_k^i \equiv \mu_s^i(t_k) - \mu_r^i(t_k), \\ \beta_k^\phi \equiv f\delta t_s(t_k) - f\delta t_r(t_k) + D_s(t_k) - D_r(t_k) + \phi_s(t_0) - \phi_r(t_0),\end{gathered}$$

we have from (5) and (1) that

$$\phi_k^i = \lambda^{-1}(\omega_k^i \boldsymbol{e}_k^i)^T \boldsymbol{x}_k + N^i + \beta_k^\phi + \mu_k^i. \tag{6}$$

Similarly by defining

$$\tilde{\rho}_k^i \equiv \tilde{\rho}_s^i(t_k) - \tilde{\rho}_r^i(t_k), \qquad \beta_k^\rho \equiv f\delta t_s(t_k) - f\delta t_r(t_k) + d_s(t_k) - d_r(t_k), \qquad \nu_k^i \equiv \nu_s^i(t_k) - \nu_r^i(t_k),$$

we obtain from (4) the single differenced *code* measurement equation

$$\tilde{\rho}_k^i = \lambda^{-1}(\omega_k^i \boldsymbol{e}_k^i)^T \boldsymbol{x}_k + \beta_k^\rho + \nu_k^i. \tag{7}$$

Following the literature (see e.g. [15, Sec. 3.4]), we assume that all μ_k^i in the carrier phase measurement equations (and all ν_l^j in the code measurement equations) for different satellites and different epochs

are unbiased independently distributed noises with the same normal distribution, and assume that μ_k^i and ν_l^j are independent. Suppose there are m visible satellites at epoch k. Writing

$$\boldsymbol{y}_k^\phi \equiv \begin{bmatrix} \phi_k^1 \\ \cdot \\ \phi_k^m \end{bmatrix}, \quad \boldsymbol{y}_k^\rho \equiv \begin{bmatrix} \tilde{\rho}_k^1 \\ \cdot \\ \tilde{\rho}_k^m \end{bmatrix}, \quad \boldsymbol{E}_k \equiv \lambda^{-1} \begin{bmatrix} (\omega_k^1 \boldsymbol{e}_k^1)^T \\ \cdot \\ (\omega_k^m \boldsymbol{e}_k^m)^T \end{bmatrix}, \quad \boldsymbol{a} \equiv \begin{bmatrix} N^1 \\ \cdot \\ N^m \end{bmatrix}, \quad \boldsymbol{v}_k^\phi \equiv \begin{bmatrix} \mu_k^1 \\ \cdot \\ \mu_k^m \end{bmatrix}, \quad \boldsymbol{v}_k^\rho \equiv \begin{bmatrix} \nu_k^1 \\ \cdot \\ \nu_k^m \end{bmatrix}, \tag{8}$$

we have from (6) and (7) that (remembering that $\boldsymbol{e} \equiv (1, 1, \ldots, 1)^T$)

$$\boldsymbol{y}_k^\phi = \boldsymbol{E}_k \boldsymbol{x}_k + \boldsymbol{a} + \boldsymbol{e}\beta_k^\phi + \boldsymbol{v}_k^\phi, \qquad \boldsymbol{v}_k^\phi \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2 \boldsymbol{I}_m), \tag{9}$$

$$\boldsymbol{y}_k^\rho = \boldsymbol{E}_k \boldsymbol{x}_k + \boldsymbol{e}\beta_k^\rho + \boldsymbol{v}_k^\rho, \qquad \boldsymbol{v}_k^\rho \sim \mathcal{N}(\boldsymbol{0}, \sigma_\rho^2 \boldsymbol{I}_m), \tag{10}$$

where we assume the standard deviations σ_ϕ and σ_ρ are known. These are the desired single differenced measurement equations at epoch k.

Notice that in $\boldsymbol{E}_k$, $\omega_k^i \boldsymbol{e}_k^i$ depends on the baseline $\boldsymbol{x}_k$, see (8) and (2). So we may write

$$\boldsymbol{E}_k \equiv \boldsymbol{E}(\boldsymbol{x}_k).$$

This $\boldsymbol{E}_k$ is known once $\boldsymbol{x}_k$ is known. Given an approximation to $\boldsymbol{x}_k$ (because $\boldsymbol{E}(\boldsymbol{x})$ is not very sensitive to changes in $\boldsymbol{x}$, our estimate of $\boldsymbol{x}_{k-1}$ is usually sufficient), we can compute our approximation to $\boldsymbol{E}_k$. Then given the measurements $\boldsymbol{y}_k^\phi$ and $\boldsymbol{y}_k^\rho$, we can obtain a better estimate of $\boldsymbol{x}_k$, and of $\boldsymbol{E}_k$ if necessary.

3 An efficient and numerically reliable approach

For solving a general linear LS problem, there are two typical methods: the normal equations method and the QR factorization method (which uses orthogonal transformations). If the coefficient matrix is ill-conditioned (this can happen in GPS due to poor geometry), the former may unnecessarily lose accuracy, while the latter does not, because it is numerically stable, see e.g. [5, Sec. 5.3]. Following [3], we will present a recursive least squares algorithm by orthogonal transformations to estimate positions based on the single differenced carrier phase and code measurement equations (9) and (10). We will make full use of the structure of the problem (for example using the fact that the integer ambiguity vector is *identical* in each step when there are no cycle slips or satellites setting and rising) to make the algorithm efficient. Note that Tiberius [15] also used a recursive algorithm. But he used the standard square root information filter (SRIF) approach based on double differenced carrier phase and/or code measurement equations. It is hard to see how this approach could take full advantage of this particular structure.

3.1 Position estimation

For the time being we assume that the $\boldsymbol{E}_k$ in (9) and (10) are known, and that the number of visible satellites does not change during the observation period.

First we eliminate β_k^ϕ from the carrier phase measurement equation (9). Let $\boldsymbol{P} \in \mathcal{R}^{m \times m}$ be a Householder transformation (see e.g. [5, p. 209]) such that

$$\boldsymbol{P}\boldsymbol{e} = \sqrt{m}\, \boldsymbol{e}_1, \qquad \boldsymbol{P} \equiv \boldsymbol{I} - \frac{2\boldsymbol{u}\boldsymbol{u}^T}{\boldsymbol{u}^T\boldsymbol{u}}, \qquad \boldsymbol{u} \equiv \boldsymbol{e}_1 - \frac{1}{\sqrt{m}}\, \boldsymbol{e}. \tag{11}$$

Note that $\boldsymbol{P}$ is symmetric and orthogonal. Partition $\boldsymbol{P}$ as $\boldsymbol{P} \equiv \begin{bmatrix} \boldsymbol{p}^T \\ \bar{\boldsymbol{P}} \end{bmatrix}$, then

$$\bar{\boldsymbol{P}} = \left[\frac{\boldsymbol{e}}{\sqrt{m}}, \quad \boldsymbol{I}_{m-1} - \frac{\boldsymbol{e}\boldsymbol{e}^T}{m-\sqrt{m}} \right]. \tag{12}$$

Multiplying (9) by $\boldsymbol{P}$ from the left, we obtain

$$\begin{bmatrix} \boldsymbol{p}^T \boldsymbol{y}_k^\phi \\ \bar{\boldsymbol{P}} \boldsymbol{y}_k^\phi \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}^T \boldsymbol{E}_k \\ \bar{\boldsymbol{P}} \boldsymbol{E}_k \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} \boldsymbol{p}^T \\ \bar{\boldsymbol{P}} \end{bmatrix} \boldsymbol{a} + \begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} \sqrt{m} \beta_k^\phi + \begin{bmatrix} \boldsymbol{p}^T \boldsymbol{v}_k^\phi \\ \bar{\boldsymbol{P}} \boldsymbol{v}_k^\phi \end{bmatrix}, \qquad \begin{bmatrix} \boldsymbol{p}^T \boldsymbol{v}_k^\phi \\ \bar{\boldsymbol{P}} \boldsymbol{v}_k^\phi \end{bmatrix} \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2 \boldsymbol{I}_m). \tag{13}$$

Since only the first equation in (13) involves the unknown (and here unwanted) term β_k^ϕ, it can be dropped for position estimation. So we have

$$\bar{\boldsymbol{P}} \boldsymbol{y}_k^\phi = \bar{\boldsymbol{P}} \boldsymbol{E}_k \boldsymbol{x}_k + \bar{\boldsymbol{P}} \boldsymbol{a} + \bar{\boldsymbol{P}} \boldsymbol{v}_k^\phi, \qquad \bar{\boldsymbol{P}} \boldsymbol{v}_k^\phi \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2 \boldsymbol{I}_{m-1}). \tag{14}$$

Notice that we have used the simple Householder transformation instead of the often used double differencing technique to eliminate the unknown variable β_k^ϕ. Its main advantage is that the transformed measurements are still uncorrelated. The Gram-Schmidt orthonormalization technique (see [12, pp. 124-132])), which can also eliminate β_k^ϕ and make the transformed measurements uncorrelated, is not as straightforward as the Householder transformation technique.

Since $\bar{\boldsymbol{P}}$ is $(m-1) \times m$, it does not have full column rank and we will not be able to get a unique estimate of $\boldsymbol{a}$. If we set $\bar{\boldsymbol{P}}\boldsymbol{a}$ as a new vector, we would lose the integer nature of $\boldsymbol{a}$. So we introduce the double difference integer ambiguity (DDIA) vector $\boldsymbol{z}$:

$$\boldsymbol{z} \equiv [N^2 - N^1, N^3 - N^1, \ldots, N^m - N^1]^T, \tag{15}$$

where without loss of generality we have chosen satellite 1 as the "reference" satellite. Notice $\boldsymbol{z}$ is still a vector of integers. Define

$$\boldsymbol{F} \equiv \boldsymbol{I}_{m-1} - \frac{\boldsymbol{e}\boldsymbol{e}^T}{m-\sqrt{m}}, \qquad \boldsymbol{J} \equiv [-\boldsymbol{e}, \boldsymbol{I}_{m-1}], \tag{16}$$

where $\boldsymbol{F}$ is nonsingular and $\boldsymbol{z} = \boldsymbol{J}\boldsymbol{a}$. It is easy to verify from (12) that

$$\bar{\boldsymbol{P}} = \boldsymbol{F}\boldsymbol{J}, \qquad \bar{\boldsymbol{P}}\boldsymbol{a} = \boldsymbol{F}\boldsymbol{J}\boldsymbol{a} = \boldsymbol{F}\boldsymbol{z}. \tag{17}$$

Thus (14) can be rewritten as

$$\bar{\boldsymbol{P}} \boldsymbol{y}_k^\phi = \bar{\boldsymbol{P}} \boldsymbol{E}_k \boldsymbol{x}_k + \boldsymbol{F}\boldsymbol{z} + \bar{\boldsymbol{P}} \boldsymbol{v}_k^\phi, \qquad \bar{\boldsymbol{P}} \boldsymbol{v}_k^\phi \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2 \boldsymbol{I}_{m-1}). \tag{18}$$

Similarly we can eliminate β_k^ρ from the code measurement equation (10) and obtain

$$\bar{\boldsymbol{P}} \boldsymbol{y}_k^\rho = \bar{\boldsymbol{P}} \boldsymbol{E}_k \boldsymbol{x}_k + \bar{\boldsymbol{P}} \boldsymbol{v}_k^\rho, \qquad \bar{\boldsymbol{P}} \boldsymbol{v}_k^\rho \sim \mathcal{N}(\boldsymbol{0}, \sigma_\rho^2 \boldsymbol{I}_{m-1}). \tag{19}$$

In order to make $\bar{\boldsymbol{P}} \boldsymbol{v}_k^\rho$ have the same covariance matrix as $\bar{\boldsymbol{P}} \boldsymbol{v}_k^\phi$, we define $\sigma \equiv \sigma_\phi / \sigma_\rho$ and multiply (19) by σ, then write the resulting equation and (18) together

$$\begin{bmatrix} \bar{\boldsymbol{P}} \boldsymbol{y}_k^\phi \\ \sigma \bar{\boldsymbol{P}} \boldsymbol{y}_k^\rho \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{P}} \boldsymbol{E}_k \\ \sigma \bar{\boldsymbol{P}} \boldsymbol{E}_k \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{z} + \begin{bmatrix} \bar{\boldsymbol{P}} \boldsymbol{v}_k^\phi \\ \sigma \bar{\boldsymbol{P}} \boldsymbol{v}_k^\rho \end{bmatrix}, \quad \begin{bmatrix} \bar{\boldsymbol{P}} \boldsymbol{v}_k^\phi \\ \sigma \bar{\boldsymbol{P}} \boldsymbol{v}_k^\rho \end{bmatrix} \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2 \boldsymbol{I}_{2(m-1)}). \tag{20}$$

For estimating $\boldsymbol{x}_k$ later, we would like to transform the coefficient matrix of $\boldsymbol{x}_k$ to upper triangular form. In order to do that, we first define the orthogonal matrix $\boldsymbol{G}$:

$$\boldsymbol{G} \equiv \begin{bmatrix} c\boldsymbol{I}_{m-1} & s\boldsymbol{I}_{m-1} \\ -s\boldsymbol{I}_{m-1} & c\boldsymbol{I}_{m-1} \end{bmatrix}, \quad \text{where } \gamma \equiv \sqrt{1+\sigma^2}, \quad c \equiv 1/\gamma, \quad s \equiv \sigma/\gamma, \quad \text{so } \boldsymbol{G}\begin{bmatrix} \boldsymbol{I}_{m-1} \\ \sigma\boldsymbol{I}_{m-1} \end{bmatrix} = \begin{bmatrix} \gamma\boldsymbol{I}_{m-1} \\ \boldsymbol{0} \end{bmatrix}.$$

Applying $\boldsymbol{G}$ to (20), we obtain

$$\begin{bmatrix} c\bar{\boldsymbol{P}}\boldsymbol{y}_k^\phi + s\sigma\bar{\boldsymbol{P}}\boldsymbol{y}_k^\rho \\ -s\bar{\boldsymbol{P}}\boldsymbol{y}_k^\phi + c\sigma\bar{\boldsymbol{P}}\boldsymbol{y}_k^\rho \end{bmatrix} = \begin{bmatrix} \gamma\bar{\boldsymbol{P}}\boldsymbol{E}_k \\ \boldsymbol{0} \end{bmatrix}\boldsymbol{x}_k + \begin{bmatrix} c\boldsymbol{F} \\ -s\boldsymbol{F} \end{bmatrix}\boldsymbol{z} + \begin{bmatrix} c\bar{\boldsymbol{P}}\boldsymbol{v}_k^\phi + s\sigma\bar{\boldsymbol{P}}\boldsymbol{v}_k^\rho \\ -s\bar{\boldsymbol{P}}\boldsymbol{v}_k^\phi + c\sigma\bar{\boldsymbol{P}}\boldsymbol{v}_k^\rho \end{bmatrix}. \tag{21}$$

We assume that in each epoch we have at least four visible satellites, i.e., $m \geq 4$. So $\bar{\boldsymbol{P}}\boldsymbol{E}_k$ almost always has full column rank, see e.g. [3, Sec 3.2]. Then we compute the QR factorization of $\gamma\bar{\boldsymbol{P}}\boldsymbol{E}_k$ by Householder transformations (see e.g. [5, Sec. 5.2.1]):

$$\boldsymbol{Q}_k^T(\gamma\bar{\boldsymbol{P}}\boldsymbol{E}_k) = \begin{bmatrix} \boldsymbol{R}_k \\ \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{Q}_k^T = \begin{bmatrix} \boldsymbol{U}_k \\ \boldsymbol{V}_k \end{bmatrix}, \quad 3 \times (m-1)\,\boldsymbol{U}_k, \quad (m-4) \times (m-1)\,\boldsymbol{V}_k, \tag{22}$$

where $\boldsymbol{Q}_k$ is an $(m-1) \times (m-1)$ orthogonal matrix and $\boldsymbol{R}_k$ is a 3×3 nonsingular upper triangular matrix. Multiplying (21) by $\mathrm{diag}(\boldsymbol{Q}_k^T, \boldsymbol{I}_{m-1})$ from the left gives

$$\begin{bmatrix} \boldsymbol{U}_k\bar{\boldsymbol{P}}(c\boldsymbol{y}_k^\phi + s\sigma\boldsymbol{y}_k^\rho) \\ \boldsymbol{V}_k\bar{\boldsymbol{P}}(c\boldsymbol{y}_k^\phi + s\sigma\boldsymbol{y}_k^\rho) \\ \bar{\boldsymbol{P}}(-s\boldsymbol{y}_k^\phi + c\sigma\boldsymbol{y}_k^\rho) \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_k \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}\boldsymbol{x}_k + \begin{bmatrix} c\boldsymbol{U}_k\boldsymbol{F} \\ c\boldsymbol{V}_k\boldsymbol{F} \\ -s\boldsymbol{F} \end{bmatrix}\boldsymbol{z} + \begin{bmatrix} \boldsymbol{U}_k\bar{\boldsymbol{P}}(c\boldsymbol{v}_k^\phi + s\sigma\boldsymbol{v}_k^\rho) \\ \boldsymbol{V}_k\bar{\boldsymbol{P}}(c\boldsymbol{v}_k^\phi + s\sigma\boldsymbol{v}_k^\rho) \\ \bar{\boldsymbol{P}}(-s\boldsymbol{v}_k^\phi + c\sigma\boldsymbol{v}_k^\rho) \end{bmatrix}. \tag{23}$$

Denote

$$\begin{aligned} \boldsymbol{y}_k &\equiv \boldsymbol{U}_k\bar{\boldsymbol{P}}(c\boldsymbol{y}_k^\phi + s\sigma\boldsymbol{y}_k^\rho), & \boldsymbol{v}_k &\equiv \boldsymbol{U}_k\bar{\boldsymbol{P}}(c\boldsymbol{v}_k^\phi + s\sigma\boldsymbol{v}_k^\rho), \\ \bar{\boldsymbol{y}}_k &\equiv \boldsymbol{V}_k\bar{\boldsymbol{P}}(c\boldsymbol{y}_k^\phi + s\sigma\boldsymbol{y}_k^\rho), & \bar{\boldsymbol{v}}_k &\equiv \boldsymbol{V}_k\bar{\boldsymbol{P}}(c\boldsymbol{v}_k^\phi + s\sigma\boldsymbol{v}_k^\rho), \\ \boldsymbol{g}_k &\equiv \bar{\boldsymbol{P}}(-s\boldsymbol{y}_k^\phi + c\sigma\boldsymbol{y}_k^\rho), & \boldsymbol{f}_k &\equiv \bar{\boldsymbol{P}}(-s\boldsymbol{v}_k^\phi + c\sigma\boldsymbol{v}_k^\rho). \end{aligned}$$

Then (23) can be rewritten as

$$\begin{bmatrix} \boldsymbol{y}_k \\ \bar{\boldsymbol{y}}_k \\ \boldsymbol{g}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_k \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}\boldsymbol{x}_k + \begin{bmatrix} c\boldsymbol{U}_k\boldsymbol{F} \\ c\boldsymbol{V}_k\boldsymbol{F} \\ -s\boldsymbol{F} \end{bmatrix}\boldsymbol{z} + \begin{bmatrix} \boldsymbol{v}_k \\ \bar{\boldsymbol{v}}_k \\ \boldsymbol{f}_k \end{bmatrix}. \tag{24}$$

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

$$\begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \underline{\boldsymbol{y}_k} \\ \bar{\boldsymbol{y}}_1 \\ \boldsymbol{g}_1 \\ \vdots \\ \bar{\boldsymbol{y}}_k \\ \boldsymbol{g}_k \end{bmatrix} = \left[\begin{array}{ccc|c} \boldsymbol{R}_1 & & & c\boldsymbol{U}_1\boldsymbol{F} \\ & \ddots & & \vdots \\ & & \boldsymbol{R}_k & c\boldsymbol{U}_k\boldsymbol{F} \\ \hline & & & c\boldsymbol{V}_1\boldsymbol{F} \\ & & & -s\boldsymbol{F} \\ & & & \vdots \\ & & & c\boldsymbol{V}_k\boldsymbol{F} \\ & & & -s\boldsymbol{F} \end{array}\right] \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \underline{\boldsymbol{x}_k} \\ \boldsymbol{z} \end{bmatrix} + \begin{bmatrix} \boldsymbol{v}_1 \\ \vdots \\ \underline{\boldsymbol{v}_k} \\ \bar{\boldsymbol{v}}_1 \\ \boldsymbol{f}_1 \\ \vdots \\ \bar{\boldsymbol{v}}_k \\ \boldsymbol{f}_k \end{bmatrix}. \tag{25}$$

Notice that forming $\boldsymbol{U}_j\boldsymbol{F}$ and $\boldsymbol{V}_j\boldsymbol{F}$ can be done efficiently by using the special form of $\boldsymbol{F}$ (see (16)). Since orthogonal transformations preserve the normal distribution, the transformed noise vector, the second term on the right side of (25), follows the distribution $\mathcal{N}(\boldsymbol{0}, \sigma_\phi^2 \boldsymbol{I}_{2k(m-1)})$.

Let $\boldsymbol{z}_{k|k}$ denote the LS estimate of $\boldsymbol{z}$ at epoch k. Then we see that $\boldsymbol{z}_{k|k}$ is the LS estimate for the sub-model formed by the last $k(2m-5)$ equations of (25):

$$\begin{bmatrix} \bar{\boldsymbol{y}}_1 \\ \boldsymbol{g}_1 \\ \vdots \\ \bar{\boldsymbol{y}}_k \\ \boldsymbol{g}_k \end{bmatrix} = \begin{bmatrix} c\boldsymbol{V}_1\boldsymbol{F} \\ -s\boldsymbol{F} \\ \vdots \\ c\boldsymbol{V}_k\boldsymbol{F} \\ -s\boldsymbol{F} \end{bmatrix} \boldsymbol{z} + \begin{bmatrix} \bar{\boldsymbol{v}}_1 \\ \boldsymbol{f}_1 \\ \vdots \\ \bar{\boldsymbol{v}}_k \\ \boldsymbol{f}_k \end{bmatrix}. \tag{26}$$

Once $\boldsymbol{z}_{k|k}$ has been computed, we observe from (25) that $\boldsymbol{x}_{1|k}, \boldsymbol{x}_{2|k}, \ldots, \boldsymbol{x}_{k|k}$, the LS estimates of $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ at epoch k, can be computed by solving the following upper triangular systems by back substitution:

$$\boldsymbol{R}_j \boldsymbol{x}_{j|k} = \boldsymbol{y}_j - c\boldsymbol{U}_j\boldsymbol{F}\boldsymbol{z}_{k|k}, \qquad j = 1, \ldots, k. \tag{27}$$

Notice that the $\boldsymbol{x}_{1|k}, \boldsymbol{x}_{2|k}, \ldots, \boldsymbol{x}_{k|k}$ can be computed in any order once $\boldsymbol{z}_{k|k}$ is available. So if $\boldsymbol{z}_{k|k}$ is updated, we could for example compute $\boldsymbol{x}_{k|k}$ without updating any of the earlier position estimates. Updating the earlier position estimates is called smoothing. If at a later time we want to smooth the position estimate at epoch j, then we see from (27) that $\boldsymbol{R}_j$ and $\boldsymbol{U}_j\boldsymbol{F}$ have to be stored. In practice, at any epoch we may just want to do smoothing for a fixed number of previous epochs rather than for all previous epochs, in order to avoid time and memory costs. For real-time applications, smoothing may not be necessary.

Now our task is to obtain the estimate $\boldsymbol{z}_{k|k}$ of $\boldsymbol{z}$ from (26). We use a recursive approach. Suppose at epoch $k-1$ we have computed the following orthogonal transformations:

$$\boldsymbol{T}_{k-1}^T \begin{bmatrix} c\boldsymbol{V}_1\boldsymbol{F} \\ -s\boldsymbol{F} \\ \vdots \\ c\boldsymbol{V}_{k-1}\boldsymbol{F} \\ -s\boldsymbol{F} \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}_{k-1} \\ \hline \boldsymbol{0} \end{bmatrix}, \qquad \boldsymbol{T}_{k-1}^T \begin{bmatrix} \bar{\boldsymbol{y}}_1 \\ \boldsymbol{g}_1 \\ \vdots \\ \bar{\boldsymbol{y}}_{k-1} \\ \boldsymbol{g}_{k-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_{k-1} \\ \hline \bar{\boldsymbol{b}}_{k-1} \end{bmatrix}, \tag{28}$$

where $\boldsymbol{T}_{k-1}$ is orthogonal, and $\boldsymbol{S}_{k-1}$ is nonsingular upper triangular with the same number of rows $m-1$ as $\boldsymbol{b}_{k-1}$. Then at epoch k after obtaining $\begin{bmatrix} c\boldsymbol{V}_k\boldsymbol{F} \\ -s\boldsymbol{F} \end{bmatrix}$ and $\begin{bmatrix} \bar{\boldsymbol{y}}_k \\ \boldsymbol{g}_k \end{bmatrix}$ (see the 2nd and 3rd equations in (24)), we perform the following orthogonal transformations by using Householder transformations:

$$\tilde{\boldsymbol{T}}_k^T \begin{bmatrix} \boldsymbol{S}_{k-1} \\ c\boldsymbol{V}_k\boldsymbol{F} \\ -s\boldsymbol{F} \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}_k \\ \hline \boldsymbol{0} \end{bmatrix}, \qquad \tilde{\boldsymbol{T}}_k^T \begin{bmatrix} \boldsymbol{b}_{k-1} \\ \bar{\boldsymbol{y}}_k \\ \boldsymbol{g}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_k \\ \hline \hat{\boldsymbol{b}}_k \end{bmatrix}, \qquad \bar{\boldsymbol{b}}_k \equiv \begin{bmatrix} \bar{\boldsymbol{b}}_{k-1} \\ \hat{\boldsymbol{b}}_k \end{bmatrix}, \tag{29}$$

where $\tilde{\boldsymbol{T}}_k$ is orthogonal, $\boldsymbol{S}_k$ is nonsingular upper triangular, and $\boldsymbol{S}_k$ and $\boldsymbol{b}_k$ each have $m-1$ rows. The Householder transformations can be implemented to take advantage of the upper triangular structure of $\boldsymbol{S}_{k-1}$. But the matrices $\tilde{\boldsymbol{T}}_k$ and $\boldsymbol{T}_{k-1}$ are neither formed nor stored. By using similar notation for the transformed noise vector, we get the transformed form of (26):

$$\begin{bmatrix} \boldsymbol{b}_k \\ \bar{\boldsymbol{b}}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}_k \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{z} + \begin{bmatrix} \boldsymbol{w}_k \\ \bar{\boldsymbol{w}}_k \end{bmatrix}, \qquad \begin{bmatrix} \boldsymbol{w}_k \\ \bar{\boldsymbol{w}}_k \end{bmatrix} \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2 \boldsymbol{I}_{k(2m-5)}). \tag{30}$$

Thus solving the upper triangular system

$$\boldsymbol{S}_k \boldsymbol{z}_{k|k} = \boldsymbol{b}_k \tag{31}$$

by back substitution, we obtain $\boldsymbol{z}_{k|k}$, the LS estimate of $\boldsymbol{z}$ at epoch k. After this we can solve (27) to obtain any $\boldsymbol{x}_{j|k}$, the estimate of $\boldsymbol{x}_j$ at epoch k.

Now let us deal with the nonlinearity problem. Since $\boldsymbol{E}_k = \boldsymbol{E}(\boldsymbol{x}_k)$ $(k = 1, 2, \ldots,)$, our problem of estimating the positions is actually nonlinear. We have to use approximations to $\boldsymbol{E}_k$ during the processing. Suppose we have obtained an estimate $\boldsymbol{x}_{k-1|k-1}$ of $\boldsymbol{x}_{k-1}$ at the end of epoch $k-1$. Then at epoch k, we use $\boldsymbol{E}(\boldsymbol{x}_{k-1|k-1})$ as an approximation to $\boldsymbol{E}(\boldsymbol{x}_k)$. This approximation is usually acceptable, since usually $\boldsymbol{x}_{k-1}$ and $\boldsymbol{x}_k$ are not far from each other, and $\boldsymbol{E}_{k-1}$ and $\boldsymbol{E}_k$ are very close. If necessary, after obtaining the estimate $\boldsymbol{x}_{k|k}$ of $\boldsymbol{x}_k$, we can use $\boldsymbol{E}(\boldsymbol{x}_{k|k})$ to approximate $\boldsymbol{E}_k$ and then re-compute an estimate of $\boldsymbol{x}_k$. We could even do some further iterations to get an improved estimate of $\boldsymbol{x}_k$. However neither of these last two produced any significant improvements. In fact $\boldsymbol{E}(\boldsymbol{x})$ varies so slowly as a function of $\boldsymbol{x}$ that it can save computations to only update $\boldsymbol{E}_k$ every five or so steps, depending on the application.

For the first epoch we can handle the nonlinearity of $\boldsymbol{E}_1 = \boldsymbol{E}(\boldsymbol{x}_1)$ in the following way. In many GPS applications we may know an approximate location of the roving receiver when we start to track the GPS signals. Then we can use this to construct an approximation to $\boldsymbol{E}_1$. If we do not have any information about the position of the roving receiver, we can take $\boldsymbol{x} = \boldsymbol{0}$ so that each $\boldsymbol{e}_1^i$ in $\boldsymbol{E}_1$ is the direction cosine from the *stationary* receiver to satellite i and $\omega_1^i = 1$, for $i = 1, \ldots, m$, see (8) and (2). Then after obtaining an initial estimate of $\boldsymbol{x}_1$ based on the carrier phase and code measurement equations at epoch $k = 1$, we can recompute $\boldsymbol{E}_1$ and then recompute the estimate of $\boldsymbol{x}_1$.

3.2 Computing the error covariance matrices

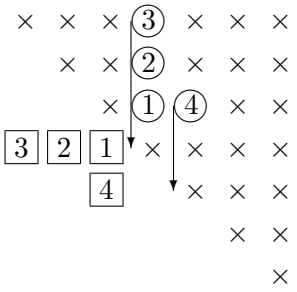
In order to have some idea of the errors in the estimates of positions, we would like to know the corresponding error covariance matrices $\mathrm{cov}\{\boldsymbol{x}_{j|k} - \boldsymbol{x}_j\}$, $j = 1, \ldots, k$. Since $\boldsymbol{E}_j$ depends on $\boldsymbol{x}_j$, the matrices $\boldsymbol{R}_j$ (see (22)) depend on the unknowns. Here we will assume that the $\boldsymbol{R}_j$ do not depend on the unknowns, and so we only *approximate* the true error covariance matrices. Combining the j-th equation of (25) and the top part of (30) gives

$$\begin{bmatrix} \boldsymbol{y}_j \\ \boldsymbol{b}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_j & c\boldsymbol{U}_j\boldsymbol{F} \\ \boldsymbol{0} & \boldsymbol{S}_k \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_j \\ \boldsymbol{z} \end{bmatrix} + \begin{bmatrix} \boldsymbol{v}_j \\ \boldsymbol{w}_k \end{bmatrix}, \qquad \begin{bmatrix} \boldsymbol{v}_j \\ \boldsymbol{w}_k \end{bmatrix} \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2 \boldsymbol{I}_{m+3}). \tag{32}$$

Note that (32) determines the estimate $\boldsymbol{x}_{j|k}$ of $\boldsymbol{x}_j$. In order to obtain $\mathrm{cov}\{\boldsymbol{x}_{j|k} - \boldsymbol{x}_j\}$, following [3] we decouple $\boldsymbol{x}_j$ and $\boldsymbol{z}$ in (32) by applying an orthogonal $\boldsymbol{Z}_{j|k}^T$ to the coefficient matrix from the left to zero the $c\boldsymbol{U}_j\boldsymbol{F}$ block:

$$\boldsymbol{Z}_{j|k}^T \begin{bmatrix} \boldsymbol{R}_j & c\boldsymbol{U}_j\boldsymbol{F} \\ \boldsymbol{0} & \boldsymbol{S}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{j|k} & \boldsymbol{0} \\ \bar{\boldsymbol{R}}_{j|k} & \boldsymbol{S}_{j|k} \end{bmatrix},$$

where Givens rotations (see for example [5, p. 215]) are used to take advantage of the triangular structure of $\boldsymbol{R}_j$ and $\boldsymbol{S}_k$ and produce upper triangular $\boldsymbol{R}_{j|k}$. Since $c\boldsymbol{U}_j\boldsymbol{F}$ is $3 \times (m-1)$, $3(m-1)$ rotations are needed. We zero $c\boldsymbol{U}_j\boldsymbol{F}$ column by column, and for each column we go from the bottom to the top. Only one element of $\boldsymbol{c}U_j\boldsymbol{F}$ and one corresponding diagonal element of $\boldsymbol{S}_k$ are used to construct one rotation. This process can be described schematically in the case $m = 5$ as follows, where the symbol $\textcircled{i}$ indicates the element is eliminated in the i-th rotation, while the symbol $\boxed{i}$ indicates this element is generated by the i-th rotation:



Then applying $\boldsymbol{Z}_{j|k}^T$ to (32) we obtain

$$\boldsymbol{Z}_{j|k}^T \begin{bmatrix} \boldsymbol{y}_j \\ \boldsymbol{b}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{j|k} & \boldsymbol{0} \\ \bar{\boldsymbol{R}}_{j|k} & \boldsymbol{S}_{j|k} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_j \\ \boldsymbol{z} \end{bmatrix} + \boldsymbol{Z}_{j|k}^T \begin{bmatrix} \boldsymbol{v}_j \\ \boldsymbol{w}_k \end{bmatrix}, \qquad \boldsymbol{Z}_{j|k}^T \begin{bmatrix} \boldsymbol{v}_j \\ \boldsymbol{w}_k \end{bmatrix} \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2 \boldsymbol{I}_{m+3}),$$

or equivalently, with obvious notation,

$$\begin{bmatrix} \boldsymbol{y}_{j|k} \\ \boldsymbol{b}_{j|k} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{j|k} & \boldsymbol{0} \\ \bar{\boldsymbol{R}}_{j|k} & \boldsymbol{S}_{j|k} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_j \\ \boldsymbol{z} \end{bmatrix} + \begin{bmatrix} \boldsymbol{v}_{j|k} \\ \boldsymbol{w}_{j|k} \end{bmatrix}, \qquad \begin{bmatrix} \boldsymbol{v}_{j|k} \\ \boldsymbol{w}_{j|k} \end{bmatrix} \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2 \boldsymbol{I}_{m+3}).$$

We see $\boldsymbol{y}_{j|k} = \boldsymbol{R}_{j|k}\boldsymbol{x}_{j|k}$ where $\boldsymbol{x}_{j|k}$ also satisfies (27), and so

$$\boldsymbol{R}_{j|k}(\boldsymbol{x}_{j|k} - \boldsymbol{x}_j) = \boldsymbol{v}_{j|k}, \qquad \mathrm{cov}\{\boldsymbol{x}_{j|k} - \boldsymbol{x}_j\} = \boldsymbol{R}_{j|k}^{-1}\mathrm{cov}\{\boldsymbol{v}_{j|k}\}\boldsymbol{R}_{j|k}^{-T} = \sigma_\phi^2(\boldsymbol{R}_{j|k}^T\boldsymbol{R}_{j|k})^{-1}.$$

3.3 Fixing integer ambiguities

In order to obtain high precision for position estimates in as few epochs as possible, the integer nature of the ambiguities should be used. Notice that in our algorithm in Section 3.1 we just regarded $\boldsymbol{z}$ as a general real vector, and did not fix it as a vector of integers. For some applications this will be sufficient, and it is not worth spending extra computation time to fix the integer ambiguities. Also if an integer ambiguity is not correctly fixed, it might cause a larger error in the position estimate. But if one wants, the results obtained here can be used to fix the integer ambiguities.

There are several approaches to fixing double difference integer ambiguities. One of the well-known approaches is called the LAMBDA method, see [14, Chap. 8]. This is based on $\mathrm{cov}\{\boldsymbol{z}_{k|k} - \boldsymbol{z}\}$, and one type of input for the LAMBDA method is the Cholesky factor of $[\mathrm{cov}\{\boldsymbol{z}_{k|k} - \boldsymbol{z}\}]^{-1}$, which actually has been obtained in Section 3.1. In fact, from the top part of (30) and (31),

$$\mathrm{cov}\{\boldsymbol{z}_{k|k} - \boldsymbol{z}\} = \mathrm{cov}\{\boldsymbol{S}_k^{-1}\boldsymbol{w}_k\} = \sigma_\phi^2\boldsymbol{S}_k^{-1}\boldsymbol{S}_k^{-T} = \sigma_\phi^2(\boldsymbol{S}_k^T\boldsymbol{S}_k)^{-1}. \tag{33}$$

Thus $\boldsymbol{S}_k/\sigma_\phi$ is the Cholesky factor of $[\mathrm{cov}\{\boldsymbol{z}_{k|k} - \boldsymbol{z}\}]^{-1}$.

If at epoch k, $\boldsymbol{z}_{k|k}$ is fixed as a vector of integers, then we can get the corresponding position estimates from (27). Starting from epoch $k+1$, we will not need to estimate $\boldsymbol{z}$ any more. We simply move $F\boldsymbol{z}$ to the left hand side of (18). So for any epoch l after k, we need only solve those upper triangular systems that we are interested in among (cf. (27))

$$\boldsymbol{R}_j\boldsymbol{x}_{j|l} = \boldsymbol{y}_j - c\boldsymbol{U}_j(\boldsymbol{F}\boldsymbol{z}_{k|k}), \qquad j = 1, \ldots, l, \quad l > k.$$

3.4 Handling the change of ambiguities

In Section 3.1 we assumed that we had the same satellites during the whole observation period. But if the observation span is long, there will be satellites setting and/or rising. Sometimes data for some satellites at some epochs may be missing. This can also be regarded as satellite setting and rising. As a result, the number of measurement equations may be different for different epochs, leading to different dimensions of the DDIA vector $\boldsymbol{z}$ for different epochs. The values of $\boldsymbol{z}$ may also change due to cycle slips. For methods for cycle slip detection and repair, see e.g. [7, Sec. 9.1.2]. But it is clear from this that correct repair of cycle slips may not be easy. So we assume that a method for cycle slip detection, but not repair, is incorporated in the positioning algorithm. When cycle slips are detected between two epochs, we just assume there were satellites setting and rising between these two epochs, although physically the setting and rising satellites are identical.

The major task is to update the estimate of the DDIA vector from epoch $k-1$ to epoch k, see (29)–(31). When this has been done, the position estimates at epoch k can easily be obtained, see (27). So the key to handling rising and setting satellites is to find the equivalents of (29)–(31) and (27). Since the DDIA vector may be different for different epochs, we will use notation such as $\boldsymbol{z}_k$ instead of $\boldsymbol{z}$. Because the number of visible satellites changes, we will write $\boldsymbol{F}_k$ instead of $\boldsymbol{F}$ in (16).

Remember from (15) that a DDIA element has the form $N^n - N^i$, where n is a non-reference satellite and i is the reference satellite. We will say that this element "corresponds to" the non-reference satellite n. It is important to be aware that we will use the *same* reference satellite for *every* element in *every* DDIA vector at a given epoch j. This reference satellite must be visible at epoch j, so if it sets between epochs $k-1$ and k, for some $k > j$, then (see Case 2 later) we will arrange to use a *different* reference satellite at epoch k. For $k = 1, 2, \ldots$, let $\tilde{\boldsymbol{z}}_k$ be the DDIA vector for some reference satellite which is visible at epoch k, where the elements of $\tilde{\boldsymbol{z}}_k$ correspond to *all* the other satellites we have encountered from epoch 1 to epoch k. If a satellite sets and rises again, it will be considered as a new satellite. Thus when a satellite rises, the dimension of $\tilde{\boldsymbol{z}}$ will increase by one, but a setting satellite leaves the dimension unchanged. In our constant satellite case $\tilde{\boldsymbol{z}}_k$ was just $\boldsymbol{z}$ in (15).

Assume at the end of epoch $k-1$ that we have obtained the equivalent of the top part of (30) for epoch $k-1$:

$$\tilde{\boldsymbol{b}}_{k-1} = \tilde{\boldsymbol{S}}_{k-1}\tilde{\boldsymbol{z}}_{k-1} + \tilde{\boldsymbol{w}}_{k-1}, \qquad \tilde{\boldsymbol{w}}_{k-1} \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2 \boldsymbol{I}), \tag{34}$$

where $\tilde{\boldsymbol{S}}_{k-1}$ is nonsingular upper triangular. We can partition $\tilde{\boldsymbol{z}}_{k-1}$ as

$$\tilde{\boldsymbol{z}}_{k-1} = \begin{bmatrix} \tilde{\boldsymbol{z}}_{k-1}^d \\ \boldsymbol{z}_{k-1} \end{bmatrix}, \tag{35}$$

where for $k = 1, 2, \ldots$, the elements of $\tilde{\boldsymbol{z}}_k^d$ correspond to the non-reference satellites which have gone *down* from epochs 1 until k, and the elements of $\boldsymbol{z}_k$ correspond to the non-reference satellites which are visible at epoch k. Here and later, when we talk about a DDIA vector at epoch k, a non-reference satellite means any satellite which is not the reference satellite at epoch k. Then with compatible partitioning, we can rewrite (34) as

$$\begin{bmatrix} \boldsymbol{b}_{k-1}^{(1)} \\ \boldsymbol{b}_{k-1} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{S}}_{k-1}^{(1)} & \tilde{\boldsymbol{S}}_{k-1}^{(2)} \\ \boldsymbol{0} & \boldsymbol{S}_{k-1} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{z}}_{k-1}^d \\ \boldsymbol{z}_{k-1} \end{bmatrix} + \begin{bmatrix} \boldsymbol{w}_{k-1}^{(1)} \\ \boldsymbol{w}_{k-1} \end{bmatrix}, \tag{36}$$

where both $\tilde{\boldsymbol{S}}_{k-1}^{(1)}$ and $\boldsymbol{S}_{k-1}$ are nonsingular upper triangular. Notice that if no satellites rise or set from epoch 1 to epoch $k-1$, then the top part of (36) will disappear and the bottom part is just the top part of (30) with k replaced by $k-1$. In the following we will combine (36) with the relevant equations at epoch k in order to obtain the position estimates. We consider two cases separately.

Case 1: The reference satellite at epoch $k-1$ remains at epoch k. We still use it as the reference satellite at epoch k. If a satellite sets between epochs $k-1$ and k, then we simply drop the corresponding two measurement equations from (20). If a satellite rises between epochs k–1 and k, then we just append the two equations at the obvious positions in (20), so the measurement equations obtained at epoch k involve only the ambiguities in the DDIA vector $\boldsymbol{z}_k$. The transformed measurement equations (24) at epoch k for the new situation can be written as

$$\boldsymbol{y}_k = \boldsymbol{R}_k\boldsymbol{x}_k + c\boldsymbol{U}_k\boldsymbol{F}_k\boldsymbol{z}_k + \boldsymbol{v}_k, \qquad \boldsymbol{v}_k \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2\boldsymbol{I}), \tag{37}$$

$$\begin{bmatrix}\bar{\boldsymbol{y}}_k \\ \boldsymbol{g}_k\end{bmatrix} = \begin{bmatrix}c\boldsymbol{V}_k\boldsymbol{F}_k \\ -s\boldsymbol{F}_k\end{bmatrix}\boldsymbol{z}_k + \begin{bmatrix}\bar{\boldsymbol{v}}_k \\ \boldsymbol{f}_k\end{bmatrix}, \qquad \begin{bmatrix}\bar{\boldsymbol{v}}_k \\ \boldsymbol{f}_k\end{bmatrix} \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2\boldsymbol{I}). \tag{38}$$

The main task is to combine (38) with (36) to obtain the equivalent of (34) for $\tilde{\boldsymbol{z}}_k$ at epoch k. The resulting LS estimate of $\tilde{\boldsymbol{z}}_k$ can then be used to give the position estimates based on (37) and the corresponding equations for epochs $j = 1, \ldots, k-1$,

$$\boldsymbol{y}_j = \boldsymbol{R}_j\boldsymbol{x}_j + c\boldsymbol{U}_j\boldsymbol{F}_j\boldsymbol{z}_j + \boldsymbol{v}_j, \qquad \boldsymbol{v}_j \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2\boldsymbol{I}). \tag{39}$$

The DDIA vector $\boldsymbol{z}_k$ can be partitioned as follows:

$$\boldsymbol{z}_k = \begin{bmatrix}\boldsymbol{z}_k^r \\ \boldsymbol{z}_k^u\end{bmatrix}, \tag{40}$$

where the elements of $\boldsymbol{z}_k^r$ correspond to the non-reference satellites which are visible at epoch $k-1$ and *remain* at epoch k, and those of $\boldsymbol{z}_k^u$ corresponds to the non-reference satellites which come *up* between epochs $k-1$ and k. Let $\boldsymbol{z}_k^d$ denote the DDIA vector whose elements correspond to the satellites which go *down* between epochs $k-1$ and k. Then for $k = 1, 2, \ldots$, we can summarize the DDIA vectors at epoch k as follows, where every element of every DDIA vector at epoch k involves the same *reference* satellite, which must be visible at epoch k:

- $\tilde{\boldsymbol{z}}_k$: (whose elements correspond to) all the non-reference satellites that were visible for at least one epoch from epoch 1 to epoch k;
- $\tilde{\boldsymbol{z}}_k^d$: all the non-reference satellites that go *down* between epochs 1 and k;
- $\boldsymbol{z}_k$: all the non-reference satellites that are visible at epoch k;
- $\boldsymbol{z}_k^r$: all the non-reference satellites that are visible at epoch $k-1$ and *remain* at epoch k;
- $\boldsymbol{z}_k^u$: all the non-reference satellites that come *up* between epochs $k-1$ and k;
- $\boldsymbol{z}_k^d$: all the non-reference satellites that go *down* between epochs $k-1$ and k.

Obviously $\boldsymbol{z}_{k-1}$ is a rearrangement of the combined elements of $\boldsymbol{z}_k^r$ and $\boldsymbol{z}_k^d$, so we can find a permutation matrix $\boldsymbol{\Pi}_k = [\boldsymbol{\Pi}_k^{(1)}, \boldsymbol{\Pi}_k^{(2)}]$ such that

$$\boldsymbol{\Pi}_k^T\boldsymbol{z}_{k-1} = \begin{bmatrix}(\boldsymbol{\Pi}_k^{(1)})^T\boldsymbol{z}_{k-1} \\ (\boldsymbol{\Pi}_k^{(2)})^T\boldsymbol{z}_{k-1}\end{bmatrix} = \begin{bmatrix}\boldsymbol{z}_k^d \\ \boldsymbol{z}_k^r\end{bmatrix}. \tag{41}$$

The following are the relationships between these DDIA vectors that we will use:

$$\tilde{\boldsymbol{z}}_k^d = \begin{bmatrix}\tilde{\boldsymbol{z}}_{k-1}^d \\ \boldsymbol{z}_k^d\end{bmatrix}, \qquad \tilde{\boldsymbol{z}}_k = \begin{bmatrix}\tilde{\boldsymbol{z}}_k^d \\ \boldsymbol{z}_k\end{bmatrix} = \begin{bmatrix}\tilde{\boldsymbol{z}}_{k-1}^d \\ \boldsymbol{z}_k^d \\ \boldsymbol{z}_k^r \\ \boldsymbol{z}_k^u\end{bmatrix}. \tag{42}$$

We are now ready to combine (36) and (38). Partition $\boldsymbol{F}_k = [\boldsymbol{F}_k^{(1)}, \boldsymbol{F}_k^{(2)}]$ compatibly with (40) so that in (38)

$$\begin{bmatrix} c\boldsymbol{V}_k\boldsymbol{F}_k \\ -s\boldsymbol{F}_k \end{bmatrix} \boldsymbol{z}_k = \begin{bmatrix} c\boldsymbol{V}_k\boldsymbol{F}_k^{(1)} \\ -s\boldsymbol{F}_k^{(1)} \end{bmatrix} \boldsymbol{z}_k^r + \begin{bmatrix} c\boldsymbol{V}_k\boldsymbol{F}_k^{(2)} \\ -s\boldsymbol{F}_k^{(2)} \end{bmatrix} \boldsymbol{z}_k^u.$$

In (36), use (41) to write

$$\boldsymbol{S}_{k-1}\boldsymbol{z}_{k-1} = \boldsymbol{S}_{k-1}\boldsymbol{\Pi}_k\,\boldsymbol{\Pi}_k^T\boldsymbol{z}_{k-1} = \left[\boldsymbol{S}_{k-1}\boldsymbol{\Pi}_k^{(1)} \;\middle|\; \boldsymbol{S}_{k-1}\boldsymbol{\Pi}_k^{(2)}\right] \begin{bmatrix} \boldsymbol{z}_k^d \\ \hline \boldsymbol{z}_k^r \end{bmatrix},$$

and similarly for $\tilde{\boldsymbol{S}}_{k-1}^{(2)}\boldsymbol{z}_{k-1}$. Then stacking (36) on top of (38) gives (see (42) and (40))

$$\left[\begin{array}{c} \boldsymbol{b}_{k-1}^{(1)} \\ \boldsymbol{b}_{k-1} \\ \bar{\boldsymbol{y}}_k \\ \boldsymbol{g}_k \end{array}\right] = \left[\begin{array}{cc|cc} \tilde{\boldsymbol{S}}_{k-1}^{(1)} & \tilde{\boldsymbol{S}}_{k-1}^{(2)}\boldsymbol{\Pi}_k^{(1)} & \tilde{\boldsymbol{S}}_{k-1}^{(2)}\boldsymbol{\Pi}_k^{(2)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}_{k-1}\boldsymbol{\Pi}_k^{(1)} & \boldsymbol{S}_{k-1}\boldsymbol{\Pi}_k^{(2)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & c\boldsymbol{V}_k\boldsymbol{F}_k^{(1)} & c\boldsymbol{V}_k\boldsymbol{F}_k^{(2)} \\ \boldsymbol{0} & \boldsymbol{0} & -s\boldsymbol{F}_k^{(1)} & -s\boldsymbol{F}_k^{(2)} \end{array}\right] \left[\begin{array}{c} \tilde{\boldsymbol{z}}_{k-1}^d \\ \boldsymbol{z}_k^d \\ \hline \boldsymbol{z}_k^r \\ \boldsymbol{z}_k^u \end{array}\right] + \left[\begin{array}{c} \boldsymbol{w}_{k-1}^{(1)} \\ \boldsymbol{w}_{k-1} \\ \bar{\boldsymbol{v}}_k \\ \boldsymbol{f}_k \end{array}\right]. \tag{43}$$

To solve the LS problem, we compute the following orthogonal transformations by the Householder transformations, which are the new versions of (29):

$$\tilde{\boldsymbol{T}}_k^T \left[\begin{array}{cc|cc} \tilde{\boldsymbol{S}}_{k-1}^{(1)} & \tilde{\boldsymbol{S}}_{k-1}^{(2)}\boldsymbol{\Pi}_k^{(1)} & \tilde{\boldsymbol{S}}_{k-1}^{(2)}\boldsymbol{\Pi}_k^{(2)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}_{k-1}\boldsymbol{\Pi}_k^{(1)} & \boldsymbol{S}_{k-1}\boldsymbol{\Pi}_k^{(2)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & c\boldsymbol{V}_k\boldsymbol{F}_k^{(1)} & c\boldsymbol{V}_k\boldsymbol{F}_k^{(2)} \\ \boldsymbol{0} & \boldsymbol{0} & -s\boldsymbol{F}_k^{(1)} & -s\boldsymbol{F}_k^{(2)} \end{array}\right] = \left[\begin{array}{c|c} \tilde{\boldsymbol{S}}_k^{(1)} & \tilde{\boldsymbol{S}}_k^{(2)} \\ \hline \boldsymbol{0} & \boldsymbol{S}_k \\ \hline\hline \boldsymbol{0} & \boldsymbol{0} \end{array}\right] \equiv \left[\begin{array}{c} \tilde{\boldsymbol{S}}_k \\ \hline\hline \boldsymbol{0} \end{array}\right],$$

$$\tilde{\boldsymbol{T}}_k^T \begin{bmatrix} \boldsymbol{b}_{k-1}^{(1)} \\ \boldsymbol{b}_{k-1} \\ \bar{\boldsymbol{y}}_k \\ \boldsymbol{g}_k \end{bmatrix} = \left[\begin{array}{c} \boldsymbol{b}_k^{(1)} \\ \hline \boldsymbol{b}_k \\ \hline\hline \hat{\boldsymbol{b}}_k \end{array}\right] \equiv \begin{bmatrix} \tilde{\boldsymbol{b}}_k \\ \hline\hline \hat{\boldsymbol{b}}_k \end{bmatrix}, \quad \tilde{\boldsymbol{T}}_k^T \begin{bmatrix} \boldsymbol{w}_{k-1}^{(1)} \\ \boldsymbol{w}_{k-1} \\ \bar{\boldsymbol{v}}_k \\ \boldsymbol{f}_k \end{bmatrix} = \left[\begin{array}{c} \boldsymbol{w}_k^{(1)} \\ \hline \boldsymbol{w}_k \\ \hline\hline \hat{\boldsymbol{w}}_k \end{array}\right] \equiv \begin{bmatrix} \tilde{\boldsymbol{w}}_k \\ \hline\hline \hat{\boldsymbol{w}}_k \end{bmatrix} \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2\boldsymbol{I}),$$

where both $\tilde{\boldsymbol{S}}_k^{(1)}$ and $\boldsymbol{S}_k$ are nonsingular upper triangular. This has completed the update and provided the equivalents of (34) and its expanded form (36) for epoch k (note the use of (42)):

$$\tilde{\boldsymbol{b}}_k = \tilde{\boldsymbol{S}}_k\tilde{\boldsymbol{z}}_k + \tilde{\boldsymbol{w}}_k, \qquad \tilde{\boldsymbol{w}}_k \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2\boldsymbol{I}), \tag{44}$$

$$\begin{bmatrix} \boldsymbol{b}_k^{(1)} \\ \boldsymbol{b}_k \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{S}}_k^{(1)} & \tilde{\boldsymbol{S}}_k^{(2)} \\ 0 & \boldsymbol{S}_k \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{z}}_k^d \\ \boldsymbol{z}_k \end{bmatrix} + \begin{bmatrix} \boldsymbol{w}_k^{(1)} \\ \boldsymbol{w}_k \end{bmatrix}. \tag{45}$$

We can now compute the LS estimates $\boldsymbol{z}_{k|k}$ of $\boldsymbol{z}_k$ and $\tilde{\boldsymbol{z}}_{k|k}^d$ of $\tilde{\boldsymbol{z}}_k^d$ by solving

$$\boldsymbol{S}_k\boldsymbol{z}_{k|k} = \boldsymbol{b}_k, \qquad \tilde{\boldsymbol{S}}_k^{(1)}\tilde{\boldsymbol{z}}_{k|k}^d = \boldsymbol{b}_k^{(1)} - \tilde{\boldsymbol{S}}_k^{(2)}\boldsymbol{z}_{k|k}, \qquad \tilde{\boldsymbol{z}}_{k|k} \equiv \begin{bmatrix} \tilde{\boldsymbol{z}}_{k|k}^d \\ \boldsymbol{z}_{k|k} \end{bmatrix}.$$

Notice that if no satellites rise or set between epochs $k-1$ and k, $\boldsymbol{z}_k^d$ and $\boldsymbol{z}_k^u$ will have no elements, so in (41) we take $\boldsymbol{\Pi}_k = \boldsymbol{\Pi}_k^{(2)} = \boldsymbol{I}$, giving $\boldsymbol{z}_{k-1} = \boldsymbol{z}_k^r = \boldsymbol{z}_k$, so in (43) the $2nd$ and $4th$ blocks of columns of the coefficient matrix disappear, $\boldsymbol{F}_k^{(1)} = \boldsymbol{F}_k$, and the orthogonal transformations revert to those of (29).

After this, we can use (37) and (39) to get $\boldsymbol{x}_{k|k}$ and $\boldsymbol{x}_{j|k}$ for $j = 1, \ldots, k-1$ by solving triangular systems

$$\boldsymbol{R}_k \boldsymbol{x}_{k|k} = \boldsymbol{y}_k - c\boldsymbol{U}_k \boldsymbol{F}_k \boldsymbol{z}_{k|k}, \qquad \boldsymbol{R}_j \boldsymbol{x}_{j|k} = \boldsymbol{y}_j - c\boldsymbol{U}_j \boldsymbol{F}_j \boldsymbol{z}_{j|k},$$

where $\boldsymbol{z}_{j|k}$ is the LS estimate of $\boldsymbol{z}_j$ at epoch k. This will be found by extracting the relevant elements from $\tilde{\boldsymbol{z}}_{k|k}$. Keeping track of this is part of the book-keeping we must do.

Now we would like to discuss how to estimate the error covariance matrices $\mathrm{cov}\{\boldsymbol{x}_{j|k} - \boldsymbol{x}_j\}$ for $j = 1, \ldots, k$. For $j = k$, combining (37) and the bottom part of (45), we obtain the same form as (32):

$$\begin{bmatrix} \boldsymbol{y}_k \\ \boldsymbol{b}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_k & c\boldsymbol{U}_k \boldsymbol{F}_k \\ \boldsymbol{0} & \boldsymbol{S}_k \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{z}_k \end{bmatrix} + \begin{bmatrix} \boldsymbol{v}_k \\ \boldsymbol{w}_k \end{bmatrix}, \qquad \begin{bmatrix} \boldsymbol{v}_k \\ \boldsymbol{w}_k \end{bmatrix} \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2 \boldsymbol{I}).$$

So we can use the approach given in Section 3.2 to estimate the error covariance matrix $\mathrm{cov}\{\boldsymbol{x}_{k|k} - \boldsymbol{x}_k\}$.

When $1 \le j \le k-1$, the situation is a little more complicated. Since the elements of $\boldsymbol{z}_j$ are part of $\tilde{\boldsymbol{z}}_k$, there exists a matrix $\boldsymbol{P}_{j|k}$, a permutation matrix with possible zero columns added, such that

$$\boldsymbol{z}_j = \boldsymbol{P}_{j|k} \tilde{\boldsymbol{z}}_k.$$

Then combining (39) and (44), we obtain the equivalent of (32) for this new situation:

$$\begin{bmatrix} \boldsymbol{y}_j \\ \tilde{\boldsymbol{b}}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_j & c\boldsymbol{U}_j \boldsymbol{F}_j \boldsymbol{P}_{j|k} \\ \boldsymbol{0} & \tilde{\boldsymbol{S}}_k \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_j \\ \tilde{\boldsymbol{z}}_k \end{bmatrix} + \begin{bmatrix} \boldsymbol{v}_j \\ \tilde{\boldsymbol{w}}_k \end{bmatrix}, \qquad \begin{bmatrix} \boldsymbol{v}_j \\ \tilde{\boldsymbol{w}}_k \end{bmatrix} \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2 \boldsymbol{I}).$$

Again the above form is the same as (32), so we can use the method given in Section 3.2 to estimate $\mathrm{cov}\{\boldsymbol{x}_{j|k} - \boldsymbol{x}_j\}$.

Case 2: The reference satellite (satellite 1, say) at epoch $k-1$ goes down between epochs $k-1$ and k. Without loss of generality we assume that satellite 2 is visible at epoch $k-1$, remains at epoch k, and is used as the reference satellite at epoch k. Suppose at epoch k we obtain (cf. (38))

$$\begin{bmatrix} \bar{\boldsymbol{y}}_k \\ \boldsymbol{g}_k \end{bmatrix} = \begin{bmatrix} c\boldsymbol{V}_k \boldsymbol{F}_k \\ -s\boldsymbol{F}_k \end{bmatrix} \boldsymbol{z}_k + \begin{bmatrix} \bar{\boldsymbol{v}}_k \\ \boldsymbol{f}_k \end{bmatrix}, \qquad \begin{bmatrix} \bar{\boldsymbol{v}}_k \\ \boldsymbol{f}_k \end{bmatrix} \sim \mathcal{N}(\boldsymbol{0}, \sigma_\phi^2 \boldsymbol{I}), \tag{46}$$

where $\boldsymbol{z}_k$ is the DDIA vector of visible satellites with satellite 2 as the reference satellite. At the end of epoch $k-1$ we have (34), where $\tilde{\boldsymbol{z}}_{k-1} \in \mathcal{R}^{m-1}$ say, uses satellite 1 as the reference satellite, and without loss of generality we assume

$$\tilde{\boldsymbol{z}}_{k-1} = [N^2 - N^1, N^3 - N^1, \ldots, N^m - N^1]^T.$$

Define the corresponding vector $\bar{\boldsymbol{z}}_{k-1}$ with satellite 2 as the reference satellite, along with the matrix $\boldsymbol{K}$,

$$\bar{\boldsymbol{z}}_{k-1} \equiv [N^1 - N^2, N^3 - N^2, \ldots, N^m - N^2]^T, \qquad \boldsymbol{K} \equiv \begin{bmatrix} -1 & \boldsymbol{0} \\ -\boldsymbol{e} & \boldsymbol{I}_{m-2} \end{bmatrix}.$$

Then it is easy to verify that

$$\boldsymbol{K}^2 = \boldsymbol{I}_{m-1}, \qquad \bar{\boldsymbol{z}}_{k-1} = \boldsymbol{K} \tilde{\boldsymbol{z}}_{k-1}. \tag{47}$$

This indicates that we can easily transform a DDIA vector with one satellite as the reference satellite to a DDIA vector with another satellite as the reference satellite. Define $\bar{\boldsymbol{S}}_{k-1} \equiv \tilde{\boldsymbol{S}}_{k-1} \boldsymbol{K}$. Then from (34) we have

$$\tilde{\boldsymbol{b}}_{k-1} = \tilde{\boldsymbol{S}}_{k-1} \tilde{\boldsymbol{z}}_{k-1} + \tilde{\boldsymbol{w}}_{k-1} = \tilde{\boldsymbol{S}}_{k-1} \boldsymbol{K} \boldsymbol{K} \tilde{\boldsymbol{z}}_{k-1} + \tilde{\boldsymbol{w}}_{k-1} = \bar{\boldsymbol{S}}_{k-1} \bar{\boldsymbol{z}}_{k-1} + \tilde{\boldsymbol{w}}_{k-1}.$$

We could apply an orthogonal transformation to the left of this to triangularize $\bar{\boldsymbol{S}}_{k-1}$, giving essentially the same situation as in Case 1, so we can now use that approach.

4 Real data tests

In order to demonstrate the effectiveness of our algorithm, we give real data test results here. All computations were performed in MATLAB 6.5 on an AMD Athlon running Windows XP. The two data sets were provided by VIASAT Geo-Technology Inc, a company in Montreal, Canada. The receivers which collected the data were made by Canadian Marconi Company. The sampling intervals for data set 1 and data set 2 were one second and two seconds, respectively. In data set 1, the user who held a receiver was walking in an open sky environment, and the baseline was about 200 m. In data set 2, the user who held a receiver was riding a small four wheel trail bike in an open sky environment, and the baseline was about 600 m. We used the position estimates obtained by VIASAT Geo-Technology software as the "true" positions. That software used a complex positioning algorithm which fixed the integer ambiguities, and it is believed that the errors in its estimates were about a few centimeters. In our test, we did not model the tropospheric and ionospheric errors, but we modeled the satellite clock error by using the standard quadratic polynomial, see e.g. [7, p. 181]. The differencing technique (see e.g. [7, Sec 9.1.2]) was used for cycle slip detection. In the test we took $\sigma = \sigma_\phi/\sigma_\rho = 0.001$, which is smaller than the typical value (see [11, p. 153]), but appeared to be a good choice for our data. This choice was recommended for this model of Marconi receiver by Jianjun Zhu [private communication], who carried out experiments on data set 1 using a different positioning algorithm. When we used a significantly different value for σ, the accuracy of our position estimates deteriorated.

The results of the errors in the position estimates (i.e., $\|\boldsymbol{x}_{k|k} - \boldsymbol{x}_k\|$, where $\boldsymbol{x}_k$ is the position estimate at epoch k obtained by VIASAT Geo-Technology software and $\boldsymbol{x}_{k|k}$ is the corresponding position estimate obtained by our algorithm) and the errors in the smoothed position estimates at epoch 1200 (i.e., $\|\boldsymbol{x}_{j|1200} - \boldsymbol{x}_j\|$, $j = 1, \ldots, 1199$) for the two data sets are shown in Figures 2 and 3, respectively. Here for simplicity, Figures 2 and 3 show only the 2-norm of the 3-dimensional error, rather than the individual components (i.e., North, East, Up) of the error. For data set 1, over the test period (1200 seconds) there were a total of 12 visible satellites; the maximum number of satellites appearing

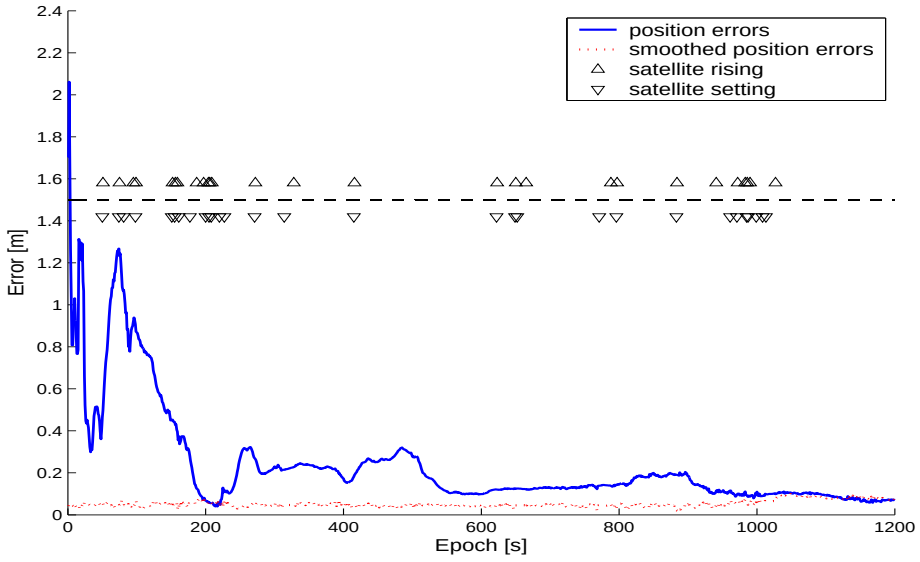


Figure 2: Estimated position errors and smoothed position errors for data set 1

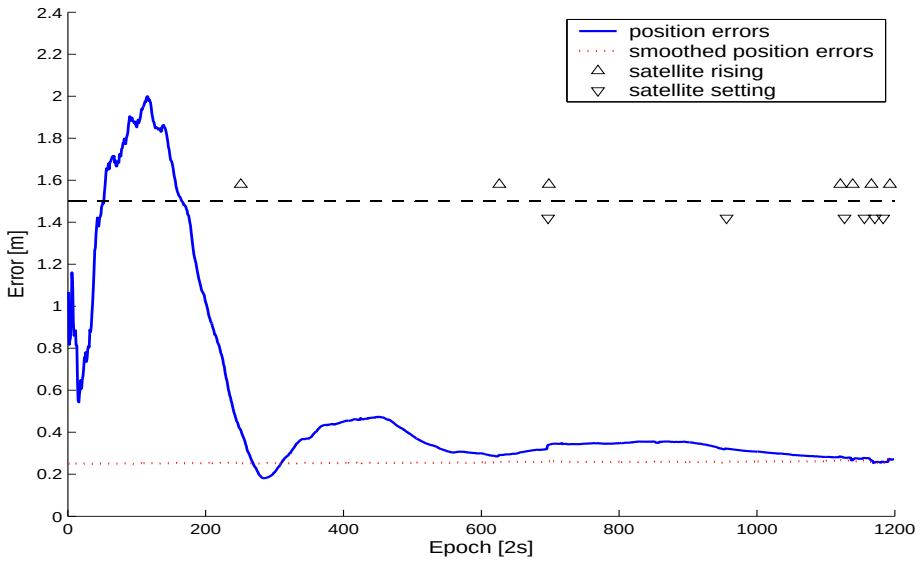


Figure 3: Estimated position errors and smoothed position errors for data set 2

at one epoch was 10; for most of the time in this period there were 6 satellites, and the reference satellite changed twice in the computation. For data set 2, over the test period (2400 seconds) there were a total of 11 visible satellites; the maximum number of satellites appearing at one epoch was 9; there were 7 satellites for a little more than half of this period, and the reference satellite changed once in the computation. From Figures 2 and 3, we observe that after 100-200 epochs, we can get submeter accuracy. The two figures indicate that the accuracy of the smoothed position estimates $\boldsymbol{x}_{j|1200}$ $(j = 1, \ldots, 1199)$ at epoch 1200 is similar to the accuracy of $\boldsymbol{x}_{1200|1200}$. We found in our test that this was also true for other epochs. From the two figures, we see that satellite setting and rising does not have any obvious effect on the precision of our position estimates.

It is not clear to us why the position estimates for data set 1 were eventually more accurate than for data set 2. The computations did not indicate that the geometry in data set 2 was poorer, both baselines are quite small, and we doubt that the different sampling rate would make that much difference to the final accuracy. We are unable to rule out possible faults in the information used (note that the position estimates obtained by VIASAT Geo-Technology software were regarded as the "true" positions). This highlights an apparent difficulty in this area — as far as we know there are no reliably certified test data sets on which to test and compare either models or algorithms.

5 Summary

A recursive LS approach was presented for combining carrier phase and code measurements for short baseline GPS positioning. Unlike many publications in GPS, we gave full computational details for computing position estimates (including smoothed position estimates) as well as the corresponding error covariance matrices, and included the computation for possible satellite setting and rising. Hopefully a reader can implement the algorithm without difficulty. Our algorithm is numerically reliable, since we use numerically stable orthogonal transformations. It is also efficient, since it takes full advantage of the structure of the problem. The computed results on the practical test data were positive.

Acknowledgments

We would like to thank Claude St-Pierre and Jianjun Zhu for providing the real data.

References

[1] J. Ashjaee, Ashtech XII GPS technology, *Proceedings of IEEE PLANTS'90 Position Location and Navigation Symposium*, Las Vegas, IEEE, New York, pp. 184–190.

[2] J. D. Bossler, Clyde C. Goad, Peter L. Bender, Using the Global Positioning System (GPS) for geodetic positioning, *Bulletin Géodésique*, Vol. 54, 1980, pp. 553–563.

[3] X.-W. Chang and C.C. Paige, An orthogonal transformation algorithm for GPS positioning, SIAM J. Sci. Comp., Vol. 24., pp. 1710-1732, 2003.

[4] C.C. Goad, Optimal filtering of pseudoranges and phases from single-frequency GPS receivers, *Navigation, Journal of The Institute of Navigation*, Vol. 37, Spring 1990, pp. 249–262.

[5] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University press, Baltimore, MD, 1996.

[6] R. Hatch, The synergism of GPS code and carrier measurements, *Proceedings of the Third International Symposium on Satellite Doppler Positioning*, New Mexico State University, New Mexico, February 8–12, 1982, Vol. 2, pp. 1213–1231.

[7] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, *GPS Theory and Practice*, 5th ed., Springer, New York, 2001.

[8] P.Y.C. Hwang and R.G. Brown, GPS navigation; combining pseudo-range with continuous carrier-phase using a Kalman filter, *Navigation, Journal of The Institute of Navigation*, Vol. 37, pp. 181–196, 1990.

[9] X.X. Jin, Algorithm for carrier-adjusted DGPS positioning and some numerical results, *Journal of Geodesy*, Vol. 71, pp. 411–422, 1997.

[10] A. Kleusberg, Kinematic relative positions using GPS code and carrier beat phase observations, *Manuscripta Geodaetica*, Vol. 10, 1986, pp. 257–274.

[11] P. Misra and P. Enge, *Global Positioning System, Signals, Measurements, and Performance*, Ganga-Jamuna Press, Lincoln, Massachusetts, 2001.

[12] B.W. Remondi, *Using the Global Positioning System Phase Observable for Relative Geodesy: Modeling, Processing, and Results*, Ph.D. thesis. The University of Texas at Austin, 1984.

[13] P.J. Teunissen, The GPS phase-adjusted pseudo-range, *Proceedings of the Second International Workshop on High Precision Navigation*, K. Linkwitz and U. Hangleiter (eds), Dümmler, Bonn, 1991, pp. 115-125.

[14] P. J. Teunissen and A. Kleusberg (ed). *GPS for Geodesy*, 2nd ed. Springer, Berlin, 1998.

[15] C.C.J.M. Tiberius, Recursive Data Processing for Kinematic GPS Surveying, PhD thesis, Department of Mathematical Geodesy and Positioning, Delft University of Technology, 1998.