LAMBDA software package

Python implementation, Version 1.0

Mathematical Geodesy and Positioning, Delft University of Technology

Contents

Chapter 1

Introduction

The key of precise GNSS applications is to resolve the double-differenced (DD) carrier phase ambiguities. One of the methods to do this is the LAMBDA method which was introduced by Teunissen (1993) and developed at the Delft University of Technology in the 1990s. LAMBDA stands for "Least-squares AMBiguity Decorrelation Adjustment". With the LAMBDA method, the integer least squares (ILS) ambiguity estimates are computed in two steps. First the ambiguities are decorrelated, by means of the Z-transformation. The decorrelation is essential to allow for an efficient search for the integer candidate which solves the integer minimization problem. This search is the second step in the LAMBDA method.

1.1 History of LAMBDA software development

LAMBDA was first implemented in Fortran-77, which is *Version 1.0* of the LAMBDA software. The implementation aspects are extensively described in De Jonge and Tiberius (1996a). The code was first translated into Matlab by Borre, see (Strang and Borre 1997), and further improved to *Version 2.1* by Joosten (2001).

In all versions the search was executed by enumerating all integer candidates inside the search space with a fixed size χ^2.

The main features and improvements of *Version 2.1* with respect to the earlier versions are:

- Good readability of the software.
- A modular approach with a limited set of subroutines, enabling one to perform for example only the decorrelation-step, only the search-step, or both steps.
- Possibility to find a number of candidates larger than 2, which can be useful for research purposes.
- The size of the search ellipsoid is based on bootstrapping instead of rounding (if requested number of candidate vectors is smaller than $n + 1$, with n the dimension of the float ambiguity vector).

1.2 Versatile LAMBDA software package in Matlab: Version 3.0

The Matlab *Version 3.0* includes more options:

- An alternative search strategy is implemented, based on searching in an alternating way around the conditional estimates and shrinking the search ellipsoid. These concepts were proposed in Teunissen (1993), see also De Jonge and Tiberius (1996a).
  Chang et al. (2005) implemented the technique in their MLAMBDA package.

- The bootstrapping and rounding estimators can be applied in case one does not want to apply ILS.

- It is possible to output the bootstrapping success rate with decorrelated ambiguities. This success rate is known to be a tight lower bound of the ILS success rate (Teunissen 1999).

- Partial ambiguity resolution can be applied, based on fixing a subset of the decorrelated ambiguities such that the success rate will be larger or equal to a minimum required success rate.

- The Ratio Test can be applied to decide on acceptance of the fixed solution. The model-driven Ratio Test with Fixed Failure rate can be applied, or the Ratio Test with a fixed (user-defined) threshold value.

1.3 LAMBDA software package in Python: Version 1.0

In order to support users developing their own GNSS processing software with integer ambiguity resolution in Python, the Matlab *Version 3.0* has been translated to Python. The only difference between the two is that the *search with enumeration* technique of the integer least-squares estimation is not provided in the Python implementation due to the lower speed of the method, compared to the *search and shrink* technique.

1.4 Disclaimer

This Python implementation (and also the older versions) are intended for educational / research purposes. Readability of the code has therefore been considered more important than computational speed. Hence, even though the LAMBDA method itself is optimized in terms of computational efficiency, the code could still be optimized to reduce the computation times. This should be taken into account if a user wants to make a comparison with other methods in terms of computation times. The most extensive source of information concerning the implementation aspects is De Jonge and Tiberius (1996a), to which we will refer often in this document.

1.5 Outline

In the next chapter, a brief review of the integer ambiguity resolution theory will be given. The three integer estimators will all be described in the order of complexity, as well as “optimality” in terms of probability of correct fixing (success rate). This means first the rounding estimator is described, followed by integer bootstrapping and integer least squares (ILS). The implemented ILS search strategy will be described as well. Next, the Partial Ambiguity Resolution (PAR) method will be introduced. Finally, ILS with the Ratio Test will be presented.

The third chapter gives an overview of the software package, with a description of all routines and how to use them. The final chapter is about the software availability and liability. A good overview of publications on the LAMBDA method and integer estimation in general is given at the end of this report.

Chapter 2

Integer ambiguity resolution theory

2.1 Parameter estimation

GNSS ambiguity resolution is the process of resolving the unknown cycle ambiguities of the DD carrier phase data as integers. The GNSS models on which ambiguity resolution is based, can all be cast in the following conceptual frame of linearized observation equations:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{a} + \boldsymbol{B}\boldsymbol{b} + \boldsymbol{\epsilon} \tag{2.1}$$

where $\boldsymbol{a} \in \mathbb{Z}^n$ is the integer parameter vector with DD integer ambiguities. $\boldsymbol{b} \in \mathbb{R}^p$ is the real-valued parameter vector, including baseline components and possibly tropospheric and ionospheric delay parameters, etcetera. The coefficient matrices are $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{m \times p}$, with $[\boldsymbol{A}\,\boldsymbol{B}]$ of full column rank. The observation vector $\boldsymbol{y} \in \mathbb{R}^m$ contains the observed-minus-computed pseudorange and carrier-phase observables, which is contaminated by the random noise vector $\boldsymbol{\epsilon}$. Generally, $\boldsymbol{\epsilon}$ is assumed to be normally distributed with mean zero and variance-covariance matrix $\boldsymbol{Q_{yy}}$.

In general, a four-step procedure is employed to solve model (1) based on the least squares criterion.

Step 1: Float solution

In the first step, the integer property of the ambiguities $\boldsymbol{a}$ is disregarded and the so-called float LS estimates together with their variance-covariance matrix are computed

$$\begin{bmatrix} \hat{\boldsymbol{a}} \\ \hat{\boldsymbol{b}} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{Q}_{\hat{a}\hat{a}} & \boldsymbol{Q}_{\hat{a}\hat{b}} \\ \boldsymbol{Q}_{\hat{b}\hat{a}} & \boldsymbol{Q}_{\hat{b}\hat{b}} \end{bmatrix} \tag{2.2}$$

Step 2: Integer estimation

In the second step, the float ambiguity estimate $\hat{\boldsymbol{a}}$ is used to compute the corresponding integer ambiguity estimate, denoted as

$$\check{\boldsymbol{a}} = S(\hat{\boldsymbol{a}}) \tag{2.3}$$

with $S : \mathbb{R}^n \longmapsto \mathbb{Z}^n$ the integer mapping from the n-dimensional space of reals to the n-dimensional space of integers. In this step, there are different choices of mapping function S possible, which correspond to the different integer estimation methods. Popular choices are integer least-squares (ILS), integer bootstrapping (IB) and integer rounding (IR). ILS is optimal, as it can be shown to have the largest success rate of all integer estimators, (Teunissen 1999). IR and IB, however, can also perform quite well, in particular after the LAMBDA decorrelation has been applied. Their advantage over ILS is that no integer search is required. Each of the methods will be discussed in more detail in the following subsections.

Step 3: Acceptance test

The third step is optional. It consists of deciding whether or not to accept the integer solution once integer estimates of the ambiguities have been computed. Several such tests have been proposed in the literature, cf. (Abidin 1993; Chen 1997; Euler and Schaffrin 1991; Han and Rizos 1996; Han 1997; Landau and Euler 1992; Tiberius and De Jonge 1995; Wang et al. 1998). Examples currently used in practice include the Ratio Test, the F-Ratio Test, the Difference Test and the Projector Test. These and other tests can be cast in the framework of Integer Aperture estimation, (Verhagen and Teunissen 2006; Teunissen and Verhagen 2011), which unifies steps 2 and 3 as described here.

Step 4: Fixed solution

In the fourth step, the float solutions of the remaining real-valued parameters solved in the first step are updated using the fixed integer parameters,

$$\check{\boldsymbol{b}} = \hat{\boldsymbol{b}} - \boldsymbol{Q}_{\hat{\boldsymbol{b}}\hat{\boldsymbol{a}}} \boldsymbol{Q}_{\hat{\boldsymbol{a}}\hat{\boldsymbol{a}}}^{-1} (\hat{\boldsymbol{a}} - \check{\boldsymbol{a}}) \tag{2.4}$$

If the integer ambiguity solution can be assumed to be deterministic (if success rate is very close to 1), the corresponding variance-covariance matrix of the fixed baseline solution is obtained as:

$$\boldsymbol{Q}_{\check{\boldsymbol{b}}\check{\boldsymbol{b}}} = \boldsymbol{Q}_{\hat{\boldsymbol{b}}\hat{\boldsymbol{b}}} - \boldsymbol{Q}_{\hat{\boldsymbol{b}}\hat{\boldsymbol{a}}} \boldsymbol{Q}_{\hat{\boldsymbol{a}}\hat{\boldsymbol{a}}}^{-1} \boldsymbol{Q}_{\hat{\boldsymbol{a}}\hat{\boldsymbol{b}}} \tag{2.5}$$

Note that the LAMBDA software package only deals with step 2 and optionally step 3. As such, the input consists of the float ambiguities $\hat{\boldsymbol{a}}$ and variance-covariance matrix $\boldsymbol{Q}_{\hat{\boldsymbol{a}}\hat{\boldsymbol{a}}}$. A user may use any GNSS data processing algorithm to obtain the float solution.

2.2 Decorrelation technique

In theory, one can perform the mapping (2.3) on the original DD ambiguities. However, due to the high correlation between the elements of the ambiguity vector as well as the poor precision of these elements, the integer solution obtained from the original DD ambiguities could be unreliable (in case of integer rounding or bootstrapping) or the computation very time-consuming (in case of integer least squares, see Section 2.5). By reparameterizing the ambiguities, the precision of the elements of the ambiguity vector can be improved while at the same time the correlation between the ambiguities is largely reduced. This reparameterization is referred to as the $\boldsymbol{Z}$-transformation, and transforms the original DD ambiguities into a new set of ambiguities as

$$\hat{\boldsymbol{z}} = \boldsymbol{Z}^T \hat{\boldsymbol{a}}$$

The corresponding (variance-)covariance matrices are transformed accordingly

$$\boldsymbol{Q}_{\hat{\boldsymbol{z}}\hat{\boldsymbol{z}}} = \boldsymbol{Z}^T \boldsymbol{Q}_{\hat{\boldsymbol{a}}\hat{\boldsymbol{a}}} \boldsymbol{Z} \quad \text{and} \quad \boldsymbol{Q}_{\hat{\boldsymbol{b}}\hat{\boldsymbol{z}}} = \boldsymbol{Q}_{\hat{\boldsymbol{b}}\hat{\boldsymbol{a}}} \boldsymbol{Z}$$

After transformation, the float solutions become

$$\begin{bmatrix} \hat{\boldsymbol{z}} \\ \hat{\boldsymbol{b}} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{Q}_{\hat{\boldsymbol{z}}\hat{\boldsymbol{z}}} & \boldsymbol{Q}_{\hat{\boldsymbol{z}}\hat{\boldsymbol{b}}} \\ \boldsymbol{Q}_{\hat{\boldsymbol{b}}\hat{\boldsymbol{z}}} & \boldsymbol{Q}_{\hat{\boldsymbol{b}}\hat{\boldsymbol{b}}} \end{bmatrix} \tag{2.6}$$

Now, the mapping function (2.3) corresponding to the integer method of choice is used to map $\hat{\boldsymbol{z}}$ to their integers, $\check{\boldsymbol{z}}$. The back-transformation provides the integer solution in terms of the original DD ambiguities

$$\check{\boldsymbol{a}} = \boldsymbol{Z}^{-T} \check{\boldsymbol{z}}$$

Note that the back-transformation is not required for the calculation of the corresponding fixed baseline solution, as this solution can be obtained directly with:

$$\check{\boldsymbol{b}} = \hat{\boldsymbol{b}} - \boldsymbol{Q}_{\hat{\boldsymbol{b}}\hat{\boldsymbol{z}}} \boldsymbol{Q}_{\hat{\boldsymbol{z}}\hat{\boldsymbol{z}}}^{-1} (\hat{\boldsymbol{z}} - \check{\boldsymbol{z}}) \tag{2.7}$$

2.3 Integer rounding

The simplest way to obtain an integer vector from the real-valued float solution is to round each of the entries of $\hat{\boldsymbol{z}}$ to its nearest integer. The corresponding integer estimator reads

$$\check{\boldsymbol{z}}_R = ([\hat{z}_1], \cdots, [\hat{z}_n])^T \tag{2.8}$$

where $[\cdot]$ stands for rounding to the nearest integer.

Note that if rounding is applied directly to the original ambiguities $\hat{\boldsymbol{a}}$ the result may be different from $\boldsymbol{Z}^{-T}\check{\boldsymbol{z}}$, with the latter resulting in a higher probability of correct fixing.

Figure 2.1 shows the 2-dimensional pull-in regions for rounding: all float solutions residing in a specific pull-in region will be fixed to the corresponding integer grid point in the centre of the pull-in region, i.e. all are *pulled* to the same integer vector. As an example, one float solution is shown with the red dot, and the corresponding integer solution is depicted with the blue circle.

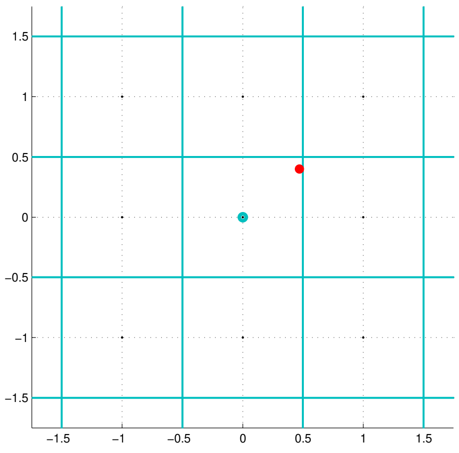

Figure 2.1: 2D Pull-in regions for integer rounding: unit squares.

2.4 Integer bootstrapping

The bootstrapped estimator still makes use of integer rounding, but it takes some of the correlation between the ambiguities into account. The bootstrapped estimator follows from a sequential least squares adjustment and it is computed as follows. If n ambiguities are available, we start with the most precise ambiguity, here assumed to be the last ambiguity $\hat{z}_n$, and round its value to the nearest integer. The remaining float ambiguities are corrected by virtue of their correlation with the last ambiguity. Then the one-but-last, but now corrected, real-valued ambiguity estimate is rounded to its nearest integer and all remaining $(n-2)$ ambiguities are then again corrected, but now by virtue of their correlation with this ambiguity. This process is continued until all ambiguities are considered. The components of the bootstrapped estimator $\check{\boldsymbol{z}}_{\text{B}}$ are given as

$$\begin{aligned} \check{z}_{n,\text{B}} &= [\hat{z}_n] \\ \check{z}_{n-1,\text{B}} &= [\hat{z}_{n-1|n}] = [\hat{z}_{n-1} - \sigma_{\hat{z}_{n-1}\hat{z}_n}\sigma_{\hat{z}_n}^{-2}(\hat{z}_n - \check{z}_{n,\text{B}})] \\ &\vdots \\ \check{z}_{1,\text{B}} &= [\hat{z}_{1|N}] = \left[\hat{z}_1 - \sum_{i=2}^{n}\sigma_{\hat{z}_1\hat{z}_{i|I}}\sigma_{\hat{z}_{i|I}}^{-2}(\hat{z}_{i|I} - \check{z}_{i,\text{B}})\right] \end{aligned} \tag{2.9}$$

where the short-hand notation $\hat{z}_{i|I}$ stands for the ith ambiguity obtained through a conditioning on the previous $I = \{i+1,\ldots,n\}$ sequentially rounded ambiguities. One should start with the most precise float ambiguity, which in this case is assumed to be $\hat{z}_n$. The real-valued sequential least squares solution can be obtained by means of the triangular decomposition of the variance-covariance matrix of the ambiguities: $\boldsymbol{Q}_{\hat{\boldsymbol{z}}\hat{\boldsymbol{z}}} = \boldsymbol{L}^T\boldsymbol{D}\boldsymbol{L}$, where $\boldsymbol{L}$ denotes a unit lower triangular matrix with entries

$$l_{i,j} = \sigma_{\hat{z}_j\hat{z}_{i|I}}\sigma_{\hat{z}_{i|I}}^{-2} \tag{2.10}$$

and $\boldsymbol{D}$ a diagonal matrix with the conditional variances $\sigma_{\hat{z}_{i|I}}^2$ as its diagonal elements.

Hence, Eq.(2.9) can be expressed as:

$$\begin{aligned}\check{z}_{j,\mathsf{B}} = [\hat{z}_{j|J}] &= \left[\hat{z}_j - \sum_{i=j+1}^{n} \sigma_{\hat{z}_j \hat{z}_{i|I}} \sigma^{-2}_{\hat{z}_{i|I}} (\hat{z}_{i|I} - \check{z}_{i,\mathsf{B}})\right] \\ &= \left[\hat{z}_j - \sum_{i=j+1}^{n} l_{i,j} (\hat{z}_{i|I} - \check{z}_{i,\mathsf{B}})\right] \end{aligned} \tag{2.11}$$

with $l_{i,j}$ from Eq.(2.10). Note that if one would like to start with $\hat{z}_1$ (being the most precise ambiguity), one should work with the $\boldsymbol{Q}_{\hat{\boldsymbol{z}}\hat{\boldsymbol{z}}} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\boldsymbol{T}}$ decomposition instead, see (De Jonge and Tiberius 1996a).

The success rate of integer bootstrapping can be evaluated exactly (Teunissen 1998e):

$$P_{s,\mathsf{B}} = P(\check{\boldsymbol{z}}_{\mathsf{B}} = \boldsymbol{z}) = \prod_{i=1}^{n} \left(2\Phi(\frac{1}{2\sigma_{\hat{z}_{i|I}}}) - 1 \right) \tag{2.12}$$

with z the true ambiguity vector and $\Phi(x)$ the cumulative normal distribution:

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\{-\frac{1}{2}t^2\} dt$$

It was already mentioned that the bootstrapping procedure should start with rounding the most precise float ambiguity. Moreover, from Eq.(2.9) it is clear that bootstrapping will generally result in different outcomes if applied to reparameterized ambiguities. It is known that bootstrapping performs close to optimal if applied to the decorrelated ambiguities $\hat{\boldsymbol{z}} = \boldsymbol{Z}^{\boldsymbol{T}}\hat{\boldsymbol{a}}$. This is because the sequential conditional variances are largely reduced by the decorrelation.

The decorrelating $\boldsymbol{Z}$-transformation is implemented such that the n-th ambiguity is the most precise one. Moreover, the decorrelation algorithm is implemented such that after each decorrelation step a reordering is performed, which guarantees that

$$\sigma_{\hat{z}_{i|I}} \leq \sigma_{\hat{z}_{j|I}} \quad \text{for } j < i \tag{2.13}$$

It should be noted that for the actual decorrelation this property is not a prerequisite. It is, however, important in case of bootstrapping, since it means that the ith transformed ambiguity has the smallest possible conditional variance, where the conditioning is on the previous $I = i+1, \ldots, n$ transformed ambiguities.

Figure 2.2 shows an example of the bootstrapping principle in the 2-dimensional case. The float solution is depicted with a red point. The grey line is the line:

$$z_1 = \hat{z}_1 - \sigma_{\hat{z}_2 \hat{z}_1} \sigma^{-2}_{\hat{z}_2} (\hat{z}_2 - z_2) \tag{2.14}$$

It shows how the value of the first ambiguity changes if the value of the second ambiguity is changed. Hence, if the second ambiguity is rounded to its nearest integer, $\check{z}_2 = [\hat{z}_2] = 0$ in this case, the corresponding conditional estimate $\hat{z}_{1|2}$ is obtained as the intersection of the grey line with the grid

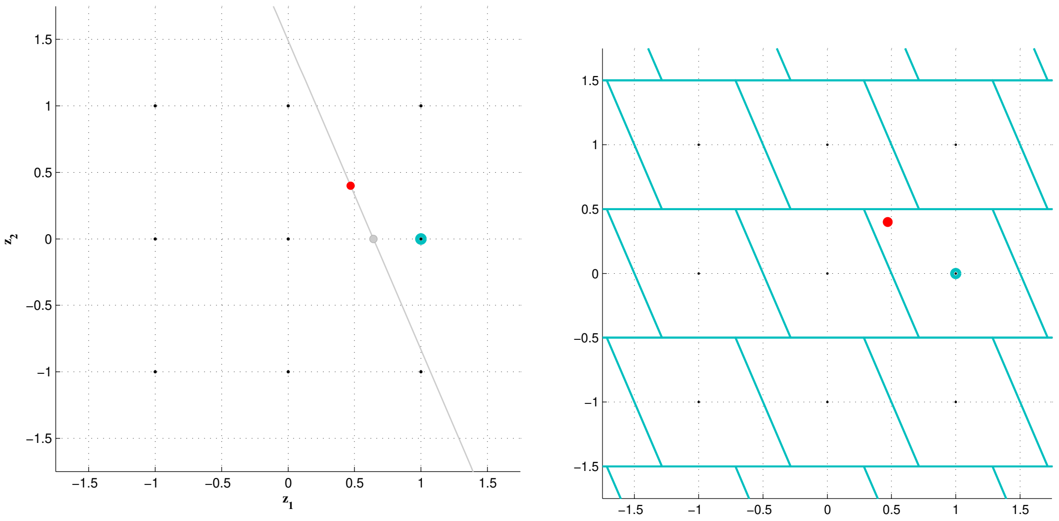

Figure 2.2: Principle and 2D pull-in regions for integer bootstrapping: parallelograms.

line $z_2 = \check{z}_2$ (grey point). Finally, $\hat{z}_{1|2}$ is rounded to its nearest integer to obtain here $\check{z}_1 = 1$. The bootstrapped solution is shown with the blue circle. Note that it is different from the integer rounding solution in Fig.2.1.

The right panel of Figure 2.2 shows the corresponding pull-in regions for integer bootstrapping, which in 2D are parallelograms.

2.5 Integer least squares

When solving the GNSS model of Eq.(2.1) in a least squares sense, but now with the additional constraint that the ambiguity parameters should be integer-valued, the integer estimator of the second step in the procedure becomes:

$$\check{\boldsymbol{a}} = \min_{\boldsymbol{z}\in\mathbb{Z}^n} (\hat{\boldsymbol{a}} - \boldsymbol{z})^T \boldsymbol{Q}_{\hat{a}\hat{a}}^{-1} (\hat{\boldsymbol{a}} - \boldsymbol{z}) \tag{2.15}$$

This estimator is known to be optimal, cf. (Teunissen 1999), which means that the probability of correct integer estimation is maximized.[1]

Figure 2.3 shows an example of the 2-dimensional pull-in regions for integer least squares. As an example, one float solution is shown with the red dot, and the corresponding integer solution is depicted with the blue circle. For this particular float solution, the same solution as with bootstrapping, see Fig.2.2 is obtained. In contrast to integer rounding and integer bootstrapping, an integer search is needed to determine $\check{\boldsymbol{a}}$.

This ILS procedure is efficiently mechanized in the LAMBDA method. Note that the success rate of ILS estimation is independent of the parameterization of the float ambiguities. The decorrelating $\boldsymbol{Z}$-transformation is only required in order to largely reduce the search time, such that the LAMBDA

[1]The ILS success rate cannot be evaluated exactly, but the bootstrapped success rate is known to be a tight lower-bound.

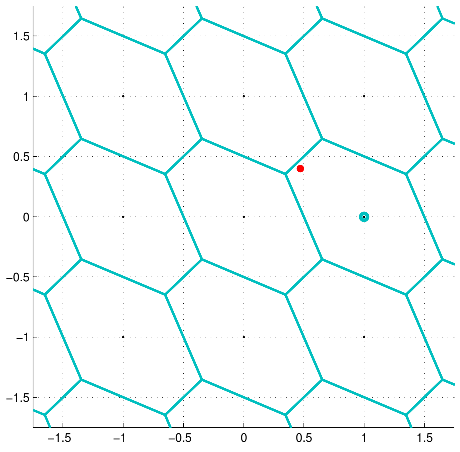

Figure 2.3: 2D Pull-in regions for integer least squares: hexagons.

method is highly efficient. The integer minimizer of Eq.(2.15) is obtained through a search over the integer grid points of an n-dimensional hyper-ellipsoid defined by the variance-covariance matrix $\boldsymbol{Q}_{\hat{\boldsymbol{z}}\hat{\boldsymbol{z}}}$ and with center $\hat{\boldsymbol{z}}$:

$$F(\boldsymbol{z}) = (\hat{\boldsymbol{z}} - \boldsymbol{z})^T \boldsymbol{Q}_{\hat{\boldsymbol{z}}\hat{\boldsymbol{z}}}^{-1} (\hat{\boldsymbol{z}} - \boldsymbol{z}) \leq \chi^2, \text{ with } \boldsymbol{z} \in \mathbb{Z}^n \tag{2.16}$$

The positive constant χ^2 determines the size of the search ellipsoid. The integer grid point $\boldsymbol{z}$ inside the hyper-ellipsoid which gives the minimum value of function $F(\boldsymbol{z})$ is the optimal ILS solution $\check{\boldsymbol{z}}$.

Figure 2.4 shows the effect of the decorrelation on the search space. The panel on the left shows the search space before decorrelation; it is very elongated and the search will be inefficient as many integer candidates will have to be evaluated to obtain the three integer vectors $\boldsymbol{a}^{(i)}$ inside the ellipse. The panel on the right shows the search space after decorrelation, with the three candidates $\boldsymbol{z}^{(i)} = \boldsymbol{Z}^T \boldsymbol{a}^{(i)}$.

Eq.(2.16) can be rewritten using the decomposition $\boldsymbol{Q}_{\hat{\boldsymbol{z}}\hat{\boldsymbol{z}}} = \boldsymbol{L}^T \boldsymbol{D} \boldsymbol{L}$:

$$(\hat{\boldsymbol{z}} - \boldsymbol{z})^T \boldsymbol{L}^{-1} \boldsymbol{D}^{-1} \boldsymbol{L}^{-T} (\hat{\boldsymbol{z}} - \boldsymbol{z}) \leq \chi^2 \tag{2.17}$$

Recall that the diagonal elements of $\boldsymbol{D}$ are the conditional variances of the transformed float ambiguities $\hat{z}_i$. The following notation will be used: $d_i = \sigma^2_{\hat{z}_{i|I}}$. Defining $\tilde{\boldsymbol{z}} = \boldsymbol{z} - \boldsymbol{L}^{-T}(\hat{\boldsymbol{z}} - \boldsymbol{z})$, we have

$$\boldsymbol{L}^T (\tilde{\boldsymbol{z}} - \boldsymbol{z}) = \hat{\boldsymbol{z}} - \boldsymbol{z} \tag{2.18}$$

where

$$\tilde{z}_i = \hat{z}_{i|I} = \hat{z}_i - \sum_{j=i+1}^{n} (\hat{z}_{j|J} - z_j) l_{j,i}, \; 1 \leq i \leq n \tag{2.19}$$

is the conditional estimate as in Eq.(2.11). Inserting Eq.(2.18) into Eq(2.17), the hyper-ellipsoid becomes

$$(\tilde{\boldsymbol{z}} - \boldsymbol{z})^T \boldsymbol{D}^{-1} (\tilde{\boldsymbol{z}} - \boldsymbol{z}) \leq \chi^2 \tag{2.20}$$

or equivalently

$$\frac{(\tilde{z}_1 - z_1)^2}{d_1} + \cdots + \frac{(\tilde{z}_i - z_i)^2}{d_i} + \cdots + \frac{(\tilde{z}_n - z_n)^2}{d_n} \leq \chi^2 \tag{2.21}$$

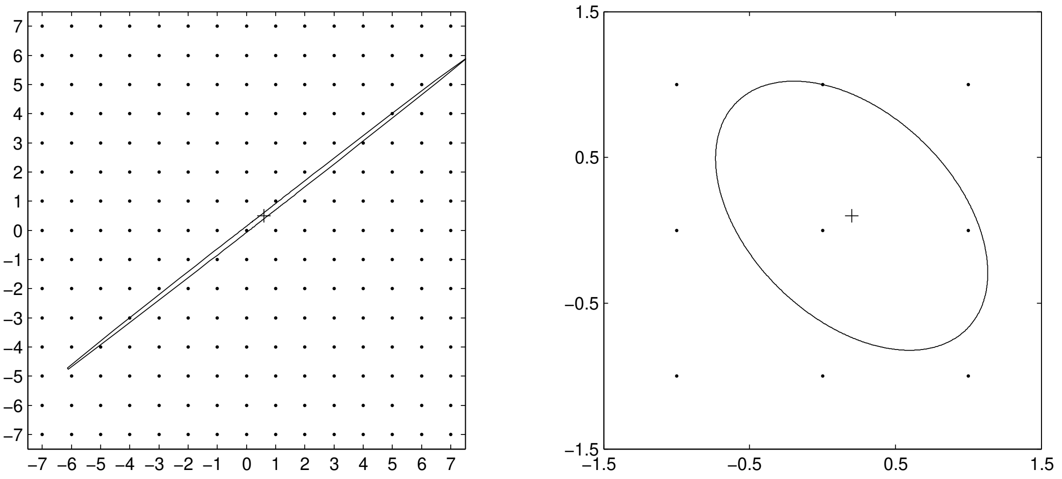
Figure 2.4: Search space before (left) and after (right) decorrelation. The float solution is indicated with the $+$.

Obviously for any $\boldsymbol{z}$ satisfying the bound Eq(2.21), also the following individual bounds are satisfied:

$$\tilde{z}_n - \sigma_{\hat{z}_n}\chi \leq z_n \leq \tilde{z}_n + \sigma_{\hat{z}_n}\chi \tag{2.22a}$$

$$\vdots$$

$$\tilde{z}_i - \sqrt{d_i\left(\chi^2 - \sum_{j=i+1}^{n}\frac{(z_j-\tilde{z}_j)^2}{d_j}\right)} \leq z_i \leq \tilde{z}_i + \sqrt{d_i\left(\chi^2 - \sum_{j=i+1}^{n}\frac{(z_j-\tilde{z}_j)^2}{d_j}\right)} \tag{2.22b}$$

$$\vdots$$

$$\tilde{z}_1 - \sqrt{d_1\left(\chi^2 - \sum_{j=2}^{n}\frac{(z_j-\tilde{z}_j)^2}{d_j}\right)} \leq z_1 \leq \tilde{z}_1 + \sqrt{d_1\left(\chi^2 - \sum_{j=2}^{n}\frac{(z_j-\tilde{z}_j)^2}{d_j}\right)} \tag{2.22c}$$

In the following, the two search techniques as implemented in the current LAMBDA software package are described.

2.5.1 Search with shrinking technique

The alternative search strategy based on shrinking the search ellipsoid during the search was already proposed in Teunissen (1993), and De Jonge and Tiberius (1996a). The implementation is based on (Chang et al. 2005).

The value of χ^2 is first set equal to infinity. If p integer candidates are requested, in the first step χ^2 is taken as the maximum of the squared norms $F(\boldsymbol{z}^{(1)}),\ F(\boldsymbol{z}^{(2)}),\ \cdots,\ F(\boldsymbol{z}^{(p)})$ for p integer candidates, $\boldsymbol{z}^{(1)},\ \boldsymbol{z}^{(2)},\ \cdots,\ \boldsymbol{z}^{(p)}$, respectively. The first candidate $\boldsymbol{z}^{(1)}$ will be equal to the bootstrapped solution according to Eq.(2.21). The remaining $p-1$ candidate vectors are selected by keeping the values of the last $2, \ldots, n$ ambiguities fixed to the bootstrapped solution, and setting the value of z_1 equal to the second-nearest integer, third-nearest integer, etcetera, with respect to the conditional estimate $\hat{z}_{1|2\ldots n}$ as in Eq.(2.11).

In this way we have $F(\boldsymbol{z}^{(1)}) \leq F(\boldsymbol{z}^{(2)}) \leq \cdots \leq F(\boldsymbol{z}^{(p)})$. Now the ellipsoidal region is shrunk by setting $\chi^2 = F(\boldsymbol{z}^{(p)})$. As a result, one can guarantee that at least p candidates are inside the ellipsoid.

Next, the search is continued at ambiguity level 2: the second-nearest integer of $\hat{z}_{2|3\ldots n}$ is taken. If this integer is inside the bounds as defined by Eq.(2.22), the corresponding conditional estimate $\hat{z}_{1|2\ldots n}$ is determined and rounded to the nearest integer, all other ambiguities (levels 3 to n) remain unchanged. If the resulting integer candidate vector $\boldsymbol{z}^{(p+1)}$ satisfies $F(\boldsymbol{z}^{(p+1)}) < \chi^2$, then χ^2 is replaced by $F(\boldsymbol{z}^{(p+1)})$ and at the same time its corresponding integer candidate vector is replaced by $\boldsymbol{z}^{(p+1)}$. As a result the ellipsoidal region is shrunk. One continues the search in this shrunken ellipsoid at ambiguity level 1 until no new candidates can be found there. In that case, the search is continued in the same way as before again in the second level, until no more new candidates can be found there either. In this way, one continues the search in all levels until no more candidates (besides the current p candidates) can be found whose squared norm is smaller than the current χ^2. The found p integer vectors are then the sought-for candidates.

2.5.2 Search strategy: example

Figure 2.5 show how the search procedures work for a 2-dimensional example.

The float solution is depicted with a blue asterisk. The grey line shows the line as defined in Eq.2.14. It shows that if $\hat{z}_2$ is rounded to its nearest integer 0, the conditional estimate $\hat{z}_{1|2}$ will be the intersection of the grey line with the grid line at $z_2 = 0$. With bootstrapping, this conditional estimate is then rounded to its nearest integer, in this case 1.

With the search-and-shrink strategy, the procedure is described stepwise and every step is shown as a separate panel in Figure 2.5. For each step the new candidate is shown in blue in the corresponding panel. Candidates from previous steps are shown in black. If at a certain stage a candidate is removed, as it is outside the shrunken ellipse, it is shown in red. For every step the old search ellipse (red) and the new shrunken ellipse (blue) are shown if shrinking is possible.

1. The bootstrapped solution of the decorrelated ambiguities is $[0\ 1]^T$ and this is our first candidate. Then the next 5 candidates are chosen by rounding the conditional ambiguity $\hat{z}_{1|2}$ to the 2nd, 3rd, 4th, and 5th nearest integer. We now have 6 candidates $\boldsymbol{z}^{(i)}$ (blue points), and the new size of the search ellipsoid is set to the maximum $F(\boldsymbol{z}^{(i)})$. The ellipse is shown in blue.

2. Round $\hat{z}_2$ to the second nearest integer, and round the corresponding new conditional estimate of the first ambiguity to the nearest integer. This will give you a new candidate (blue point in corresponding panel). It resides in the search ellipse. Therefore, the integer candidate with the largest $F(\boldsymbol{z}^{(i)})$ is removed, and χ^2 is now set to the largest value of the remaining candidates.

3. Next, round the conditional estimate of the first ambiguity from Step 2 to the second nearest-integer. Again it resides in the shrunken search ellipsoid. Therefore, the integer candidate with

the largest $F(\boldsymbol{z}^{(i)})$ is removed, and χ^2 is now set to the largest value of the remaining candidates.

4. Next, round the conditional estimate of the first ambiguity from Step 2 to the third nearest-integer. Again it resides in the shrunken search ellipsoid. Therefore, the integer candidate with the largest $F(\boldsymbol{z}^{(i)})$ is removed, and χ^2 is now set to the largest value of the remaining candidates.

5. Next, round the conditional estimate of the first ambiguity from step 2 to the fourth nearest-integer. This candidate is outside the search ellipsoid and is disregarded. No shrinking in this step.

6. A new candidate for the second ambiguity is obtained by rounding $\hat{z}_2$ to the third nearest integer. Rounding the corresponding new conditional estimate of the first ambiguity to the nearest integer results in a new candidate, which resides in the search ellipse. Therefore, the integer candidate with the largest $F(\boldsymbol{z}^{(i)})$ is removed, and χ^2 is now set to the largest value of the remaining candidates.

It can be seen that after Step 6, we have obtained 6 candidates and no other integer grid points can be found in the shrunken ellipse. Hence, these are the 6 best candidates in the ILS sense, which are then ordered with increasing $F(\boldsymbol{z}^{(i)})$. The one with the smallest $F(\boldsymbol{z}^{(i)})$ is the actual ILS solution.

2.6 Partial ambiguity resolution (PAR)

Even though all ambiguities sometimes cannot be reliably fixed, one could have sufficient confidence to fix a subset of the ambiguities. This is referred to as partial ambiguity resolution. In the current version of the LAMBDA software package, PAR is implemented based on decorrelation and bootstrapping, (Teunissen et al. 1999; Teunissen 2001a). A subset of the decorrelated ambiguities is fixed, with a corresponding bootstrapped success rate larger or equal to a minimum required value P_0. Hence, the goal is to select the largest possible subset such that:

$$\prod_{i=k}^{n}\left(2\Phi(\frac{1}{2\sigma_{\hat{z}_{i|I}}})-1\right) \geq P_0 \tag{2.23}$$

where only the last $n-k+1$ ambiguities are sequentially rounded. One starts with the nth decorrelated ambiguity, and check if the success rate of rounding this ambiguity is at least equal to P_0. If that is the case, one can continue with rounding the conditional ambiguity $\hat{z}_{n-1|n}$, since due to the property in Eq.(2.13) this ambiguity has the smallest conditional variance, where the conditioning is only on the nth ambiguity. This procedure continues until the success rate becomes too small. The remaining $k-1$ ambiguities are then conditioned on the $K=k,\ldots,n$ conditionally rounded ambiguities, but are

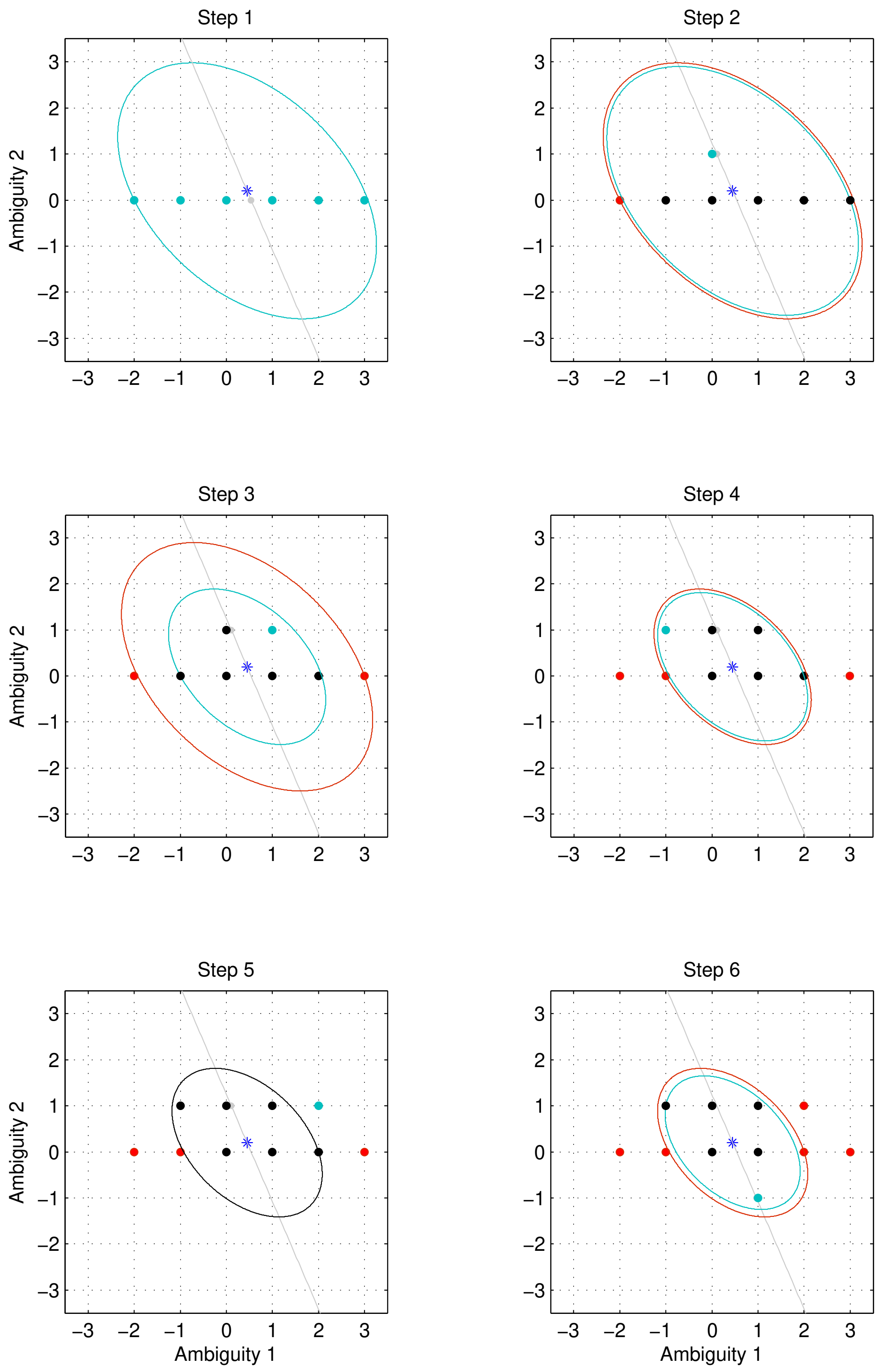

Figure 2.5: Search-and-shrink procedure for 2-dimensional example. Requested number of integer vectors is 6.

not sequentially rounded themselves. The complete 'fixed' ambiguity vector becomes thus:

$$\check{\boldsymbol{z}}_{\text{PAR}} = \begin{pmatrix} \hat{z}_{1|K} \\ \vdots \\ \hat{z}_{k-1|K} \\ [\hat{z}_{k|K}] \\ \vdots \\ [\hat{z}_{n-1|n}] \\ [\hat{z}_n] \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{z}}_{1|2} \\ \check{\boldsymbol{z}}_2 \end{pmatrix} \tag{2.24}$$

where $\check{\boldsymbol{z}}_2$ is the subset of ambiguities fixed to integers, and $\hat{\boldsymbol{z}}_{1|2}$ the subset which is not fixed to integers but conditioned on $\check{\boldsymbol{z}}_2$.

Of course it is possible to evaluate (2.23) before applying the sequential conditional rounding, so that it is known beforehand which $n-k+1$ ambiguities can be fixed to integers. This also implies that rather than applying bootstrapping, one could apply ILS to those $n-k+1$ decorrelated ambiguities, and then calculate the corresponding conditional estimates for the remaining $k-1$ ambiguities. This will result in a success rate which is equal to or higher than the bootstrapped success rate as calculated with Eq.(2.23), since it is known that ILS is optimal. This is how PAR is implemented in the LAMBDA software.

Note that the 'fixed' solution in terms of the original ambiguities can be obtained after applying the back-tranformation:

$$\check{\boldsymbol{a}}_{\text{PAR}} = \boldsymbol{Z}^{-T}\check{\boldsymbol{z}}_{\text{PAR}} \tag{2.25}$$

If not the complete set of ambiguities is fixed, $\check{\boldsymbol{a}}_{\text{PAR}}$ will generally not contain integer elements, since all elements are a linear function of all decorrelated ambiguities $\check{\boldsymbol{z}}_{\text{PAR}}$, which are not all integer-valued. In practice, the back-transformation is not required for the calculation of the corresponding fixed baseline solution, as this solution can be obtained directly with:

$$\begin{aligned} \check{\boldsymbol{b}} &= \hat{\boldsymbol{b}} - \boldsymbol{Q}_{\hat{b}\hat{a}}\boldsymbol{Q}_{\hat{a}\hat{a}}^{-1}(\hat{\boldsymbol{a}} - \check{\boldsymbol{a}}_{\text{PAR}}) \\ &= \hat{\boldsymbol{b}} - \boldsymbol{Q}_{\hat{b}\hat{z}_2}\boldsymbol{Q}_{\hat{z}_2\hat{z}_2}^{-1}(\hat{\boldsymbol{z}}_2 - \check{\boldsymbol{z}}_2) \end{aligned} \tag{2.26}$$

where $\boldsymbol{Q}_{\hat{b}\hat{z}_2}$ and $\boldsymbol{Q}_{\hat{z}_2\hat{z}_2}$ are the submatrices of $\boldsymbol{Q}_{\hat{b}\hat{z}} = \boldsymbol{Q}_{\hat{b}\hat{a}}\boldsymbol{Z}$ and $\boldsymbol{Q}_{\hat{z}\hat{z}}$, respectively, relating to $\hat{\boldsymbol{z}}_2$.

2.7 ILS with Ratio Test

Integer ambiguity resolution does not only concern integer estimation, but generally also involves applying an acceptance test on the integer solution, i.e. step 3 in Section 2.1. In the framework of Integer Aperture (IA) estimation steps 2 and 3 are unified: the float solution $\hat{\boldsymbol{a}}$ is taken as input and it is mapped to either an integer solution or to itself based on a certain acceptance criterion.

The Ratio Test is a member of the class of Integer Aperture estimators and probably one of the most popular acceptance tests, (Teunissen 2003a; Teunissen and Verhagen 2009; Teunissen and Verhagen 2011).

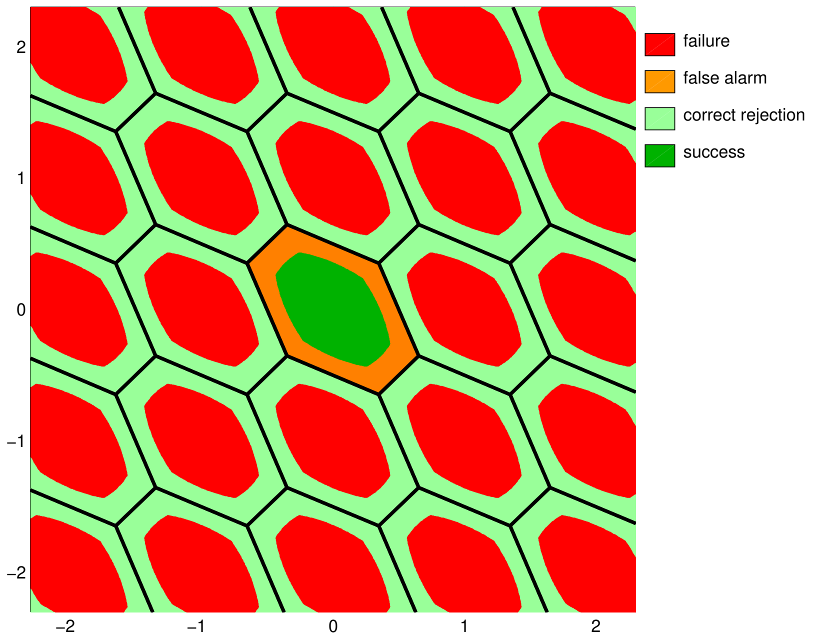

Figure 2.6: Acceptance regions with Ratio Test (bright green and red).

For the moment, only this test is included in the LAMBDA software package.

The Ratio Test is in fact a discrimination test: it tests the closeness of the float solution to the optimal integer solution compated to other integer candidates. The test is defined as:

$$\text{Accept } \check{\boldsymbol{a}} \text{ iff: } \quad \frac{F(\check{\boldsymbol{a}})}{F(\check{\boldsymbol{a}}')} \leq \mu \tag{2.27}$$

where μ is the threshold value, for which holds $0 \leq \mu \leq 1$ (since the optimal solution by definition has the smallest squared norm), and $\check{\boldsymbol{a}}'$ is the integer vector that returns the second smallest quadratic form $F(\boldsymbol{z})$, see Eq.(2.16).

The principle of the Ratio Test is illustrated in Figure 2.6: the acceptance regions are the bright green and red regions centered at all integer grid points. The value of μ determines the size of the acceptance regions. Four cases can be distinguished: correct acceptance (success), wrong acceptance (failure), unnecessary rejection (false alarm), and correct rejection. Obviously, a larger value for μ will result in a larger acceptance region, and higher probabilities of failure and success.

Note that in literature often the reciprocal of the above ratio is applied, with corresponding critical value $c = 1/\mu$. Often a fixed value is then chosen for c, e.g. 2 or 3. Teunissen and Verhagen (2009) introduced the model-driven Ratio Test, where μ is determined such that the failure rate *after* application of the Ratio Test will not exceed a user-defined value. Hence, the failure rate in this case is the probability of *wrong acceptance*. This approach is called the Fixed Failure rate Ratio Test (FF-RT).

In the LAMBDA toolbox, the user can choose to apply the Fixed Failure rate approach or to specify μ directly.

2.8 Quality of integer ambiguity resolution

The success rate can be used to evaluate the performance of an integer estimator. It is the probability of correct integer estimation and as such defined as:

$$P_s = P(\check{\boldsymbol{a}} = \boldsymbol{a}) = P(\hat{\boldsymbol{a}} \in S_{\boldsymbol{a}}) = \int\limits_{S_{\boldsymbol{a}}} f_{\hat{\mathbf{a}}}(\boldsymbol{x}|\boldsymbol{a}) d\boldsymbol{x} \tag{2.28}$$

with $f_{\hat{\mathbf{a}}}(\boldsymbol{x}|\boldsymbol{a})$ the probability density function (PDF) of the float ambiguities with mean $\boldsymbol{a}$, assumed to be the normal PDF and hence known:

$$f_{\hat{\mathbf{a}}}(\boldsymbol{x}|\boldsymbol{a}) = \frac{1}{\sqrt{\det(2\pi Q_{\hat{\boldsymbol{a}}\hat{\boldsymbol{a}}})}} \exp\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{a})^T Q_{\hat{\boldsymbol{a}}\hat{\boldsymbol{a}}}^{-1}(\boldsymbol{x}-\boldsymbol{a})\} \tag{2.29}$$

$S_{\boldsymbol{a}}$ is the pull-in region centered at the true but unknown integer $\boldsymbol{a}$. As the pull-in regions of the integer estimators are integer-translation invariant, the success rate can also be evaluated as:

$$P_s = \int\limits_{S_{\mathbf{0}}} f_{\hat{\mathbf{a}}}(\boldsymbol{x}|\mathbf{0}) d\boldsymbol{x} \tag{2.30}$$

The integration over the pull-in region is very complex in case of ILS and integer rounding, and hence it is difficult to evaluate (2.30) exactly. However, in case of bootstrapping the success rate can be evaluated using Eq.(2.12). The bootstrapping success rate is known to be a tight lower-bound for ILS and it is a upper bound for integer rounding, (Teunissen 1999). Therefore, the bootstrapping success rate will be provided as output from LAMBDA.

Alternatively, the performance of integer estimation could also be assessed based on the failure rate P_f. Since *success* and *failure* are the only possible outcomes, we have:

$$P_f = 1 - P_s \tag{2.31}$$

In case of Integer Aperture (IA) estimation, three instead of two outcomes need to be distinguished: *success* if the ambiguities are fixed correctly, *failure* if the ambiguities are fixed incorrectly, and *undecided* if the ambiguities are not fixed (i.e. rejected). The performance must thus be assessed based on the corresponding probabilities of success (s), failure (f) and undecided (u):

$$\begin{aligned} P_s &= P(\check{\boldsymbol{a}}_{\mathsf{IA}} = \boldsymbol{a}) \\ P_f &= P(\check{\boldsymbol{a}}_{\mathsf{IA}} = \boldsymbol{z} \neq \boldsymbol{a}) \\ P_u &= P(\check{\boldsymbol{a}}_{\mathsf{IA}} = \hat{\boldsymbol{a}}) \end{aligned} \tag{2.32}$$

We now have that the failure rate is equal to (compare to Eq.(2.31)):

$$P_f = 1 - P_s - P_u \tag{2.33}$$

The Fixed Failure rate approach provides a mechanism to control the failure rate. Another useful performance indicator is the so-called *successfix* rate, P_{sf}. It is the probability of successful fixing, i.e. the probability that the ambiguities are correct **if** fixed. It follows as the ratio:

$$P_{sf} = \frac{P_s}{P_s + P_f} \tag{2.34}$$

since the fix probability is equal to $P_s + P_f$. To have confidence in the integer outcomes of IA estimation, a user would like to have P_{sf} close to 1. This can be achieved by setting the failure rate P_f at a small-enough level.

Chapter 3

Routines and their usage

3.1 How to use the LAMBDA package

To use the routines of Python *Version 1.0*, you need to import the `LAMBDA` module into your main script:

```
import LAMBDA
```

3.2 The main routine

The main routine is

```
afixed,sqnorm,Ps,Qzhat,Z,nfixed,mu=LAMBDA.main(ahat,Qahat,method,ncands,P0,mu)
```

The vector of float ambiguities needs to be a 1D NumPy array of n elements. The corresponding variance-covariance matrix (2D NumPy array) should be square ($n \times n$), symmetric and positive-definite. The input and output arguments will be described first, followed by some examples of how the routine can be used if the user is interested in the different methods. In the Python implementation and the current document, the alias `np` is used for the `NumPy` package.

3.2.1 Input arguments

The following input arguments are needed, depending on the method of choice:

`ahat`	Float ambiguities ($\hat{\boldsymbol{a}}$, must be a 1D NumPy array!)
`Qahat`	Variance-covariance matrix of ambiguities ($\boldsymbol{Q}_{\hat{a}\hat{a}}$, must be a 2D NumPy array!)
`method`	1: ILS method based on shrink-and-search technique (uses `ssearch`) [DEFAULT] 2: Integer rounding method (uses `NumPy` function `round`) 3: Integer bootstrapping method (uses `bootstrap`) 4: Partial ambiguity resolution (PAR) (uses `parsearch`) 5: ILS method with Ratio Test (uses `ssearch`)
`ncands`	Number of requested integer vectors (only used with ILS/PAR) [DEFAULT=2]
`P0`	- with method 4 (PAR): Minimum required success rate [DEFAULT=0.995] - with method 5 (ILS + Ratio test): fixed failure rate for Ratio Test (available options: 0.01 and 0.001) [DEFAULT=0.001]
`mu`	Fixed threshold value for Ratio Test (value must be between 0 and 1)

The following output arguments are selected:

`afixed`	2D NumPy array of size $(n \times ncands)$ with the estimated integer candidates, sorted according to the corresponding squared norms, best candidate first. For integer rounding, bootstrapping and PAR $ncands = 1$
`sqnorm`	1D NumPy array of size $ncands$ with the distances between the integer candidates and float ambiguity vector in the metric of the variance-covariance matrix. Only available for ILS
`Ps`	Bootstrapped success rate. If ILS method is used, `Ps` is its lower bound; if rounding is used, `Ps` is its upper bound. If bootstrap method is used, `Ps` is the exact success rate
`Qzhat`	Variance-covariance matrix of decorrelated float ambiguities (corresponding to fixed subset in case of PAR)
`Z`	Transformation matrix with dimension - $(n \times n)$ for methods 1-3, 5 - $(n \times nfixed)$ for method 4 (PAR).
`nfixed`	Number of fixed ambiguities. - with methods 1 to 3: will always be equal to n - with method 4 (PAR): will be equal to the number of fixed decorrelated ambiguities - with method 5 (ILS + Ratio Test): will be equal to n if fixed solution is accepted, and 0 otherwise
`mu`	Threshold value used for Ratio Test

3.2.2 Integer remove-restore

In theory it is possible that an integer overflow occurs: the numeric value after an arithmetic operation becomes too large to be stored. With the current processors (32 bits or more) this is very unlikely to occur, but still the following integer remove-restore technique is applied, which guarantees that integer overflows cannot occur. Due to the integer-translation invariant property of integer estimators, it is namely possible to shift the float ambiguities over an integer number so that their values are all between -1 and 1. In Python this can be done with:

```
ahat,incr = np.modf(ahat)
```

After integer estimation, the removed integers can simply be restored with

```
afixed += numpy.matlib.repmat(incr.reshape(n,1),1,ncands)
```

This will always give the same fixed solution as when integer estimation is directly applied to $ahat$ without the integer remove-restore procedure.

3.2.3 Usage

The default method with the `LAMBDA` routine is ILS with the search-and-shrink procedure (`method` 1), with two integer candidates as output. Hence, the same output is obtained with the following two calls:

```
afixed,sqnorm,Ps,Qzhat,Z=LAMBDA.main(ahat,Qahat)

afixed,sqnorm,Ps,Qzhat,Z=LAMBDA.main(ahat,Qahat,1)
```

If a different number of integer candidates (default is 2) is required, this number can be specified by the optional input argument (`ncands`). For example, to obtain the 4 best integer vectors (ordered best, second-best, etcetera), use:

```
afixed,sqnorm,Ps,Qzhat,Z=LAMBDA.main(ahat,Qahat,1,4)
```

For PAR (`method` 4), the minimum required success rate can be specified by the optional input argument (`P0`) (should be a value between 0 and 1). If no value is given, the default value of 0.995 will be used. If for example the minimum required success rate should be 0.999, `LAMBDA` should be called with:

```
afixed,sqnorm,Ps,Qzhat,Z,nfixed=LAMBDA.main(ahat,Qahat,4,0.999)
```

`nfixed` the size of this subset.

For ILS with the Ratio Test, by default a fixed failure rate of 0.001 is used. If a user wants to use the value of 0.01 instead, this can be done with:

```
afixed,sqnorm,Ps,Qzhat,Z=LAMBDA.main(ahat,Qahat,5,0.01)
```

Alternatively, if the user wants to use a fixed threshold value mu equal to 0.7, the following command should be used:

```
afixed,sqnorm,Ps,Qzhat,Z=LAMBDA.main(ahat,Qahat,5,0.995,0.7)
```

The `LAMBDA` toolbox includes a demonstration routine `LAMBDAdemo` with more examples.

3.3 Routines used by `main`

3.3.1 `decorrel`: decorrelating Z-transformation

The main routine `LAMBDA` always applies integer estimation to the decorrelated ambiguities. With integer rounding and bootstrapping this is a prerequisite since it results in much higher success rates. For ILS it is necessary in order to make the search efficient in terms of computational speed.

The routine `decorrel` is used for the decorrelation. Starting from a variance-covariance matrix it will compute the $\boldsymbol{Z}$-transformation. Optionally it is possible to supply original float ambiguities, in which case the routine will return the decorrelated float ambiguities. The routine can be used as follows:

```
Qzhat,Z,L,D,zhat,iZt = decorrel(Qahat,ahat)
```

Inputs		
	`Qahat`	Variance-covariance matrix of the original ambiguities (2D array)
	`ahat`	Original ambiguities (1D array; optional)
Outputs		
	`Qzhat`	Variance-covariance matrix of decorrelated ambiguities
	`Z`	Transformation matrix
	`L, D`	L^TDL-decomposition of the decorrelated variance-covariance matrix of the ambiguities
	`zhat`	Decorrelated ambiguities (only if input argument ahat given)
	`iZt`	Inverse of $\boldsymbol{Z}^T$ matrix (needed for back-transformation)

This routine firstly executes the decomposition $\boldsymbol{Q}_{\hat{a}\hat{a}} = \boldsymbol{L^T D L}$ by running the subroutine

```
L,D = ldldecom(Qahat)
```

3.3.2 `ssearch`: ILS with search-and-shrink technique

The `ssearch` routine performs the search in the search ellipsoid with the search-and-shrink technique as described in Section 2.5.1.

```
afixed,sqnorm=ssearch(ahat,L,D,ncands)
```

Inputs

`ahat`	The decorrelated float ambiguities (1D array)
`L, D`	L^TDL–decomposition of the variance-covariance matrix of the decorrelated ambiguities
`ncands`	Number of candidates to be returned [DEFAULT=2]

Outputs

`afixed`	2D array ($n \times ncands$) with estimated integer candidates, sorted according to the corresponding squared norms, best candidate first
`sqnorm`	1D array with corresponding squared norms

The complete procedure to apply ILS with the search-and-shrink technique as implemented in `LAMBDA` is:

```
Qzhat,Z,L,D,zhat,iZt = decorrel(Qahat,ahat)
zfixed,sqnorm = ssearch(zhat,L,D,ncands)
afixed = iZt.dot(zfixed)
```

3.3.3 Integer rounding

For rounding the numpy function `np.round` is used. To improve the success rate of integer rounding, it is better to conduct the rounding on the decorrelated float ambiguities. The complete procedure to apply rounding as implemented in `LAMBDA` is:

```
Qzhat,Z,L,D,zhat,iZt = decorrel(Qahat,ahat)
zfixed = np.round(zhat)
afixed = iZt.dot(zfixed)
```

3.3.4 `bootstrap`: integer bootstrapping

With integer bootstrapping a sequential conditional rounding is applied to the float ambiguities, see Section 2.4. The routine `bootstrap` is executed with:

```
afixed=bootstrap(ahat,L)
```

Inputs

`ahat`	1D array with the float ambiguities
`L`	from $L^T DL$–decomposition of the variance-covariance matrix of the float ambiguities

Outputs

`afixed`	1D array of size n with the integer ambiguities

It is important to notice that the success rate of integer bootstrapping depends on the parameterization of the float ambiguities. It should be applied to the decorrelated ambiguities in order to obtain close to optimal performance.

The complete procedure to apply integer bootstrapping as implemented in `LAMBDA` is:

```
Qzhat,Z,L,D,zhat,iZt = decorrel(Qahat,ahat)
zfixed = bootstrap(zhat,L)
afixed = iZt.dot(zfixed)
```

3.3.5 `parsearch`: Partial ambiguity resolution

The PAR method as described in 2.6 is implemented in the routine `parsearch`, which can be used as follows:

```
zpar,sqnorm,Qzpar,Zpar,Ps,nfixed,zfixed = parsearch(zhat,Qzhat,L,D,P0,ncands)
```

Inputs

`zhat`	The decorrelated float ambiguities
`Qzhat`	The variance-covariance matrix of the decorrelated float ambiguities
`L, D`	$L^T DL$–decomposition of the variance-covariance matrix `Qzhat`
`P0`	User-defined success rate [DEFAULT=0.995]
`ncands`	Number of requested integer candidate vectors [DEFAULT=2]

Outputs

`zpar`	Subset of fixed ambiguities ($nfixed \times ncands$)
`sqnorm`	Squared norms corresponding to fixed subsets
`Qzpar`	Variance-covariance matrix of float ambiguities for the fixed subset
`Zpar`	$\boldsymbol{Z}$-matrix corresponding to fixed subset
`Ps`	Bootstrapped sucess rate of partial ambiguity resolution
`nfixed`	The number of fixed ambiguities
`zfixed`	[optional] Complete 'fixed' ambiguity vector where the remaining (non-fixed) ambiguities are adjusted according to their correlation with the fixed subset

The PAR algorithm should only be applied to decorrelated ambiguities. The complete procedure to apply PAR as implemented in LAMBDA is:

```
Qzhat,Z,L,D,zhat,iZt = decorrel(Qahat,ahat)
zpar,sqnorm,Qzpar,Zpar,Ps,nfixed,zfixed = parsearch(zhat,Qzhat,L,D,P0,1)
afixed = iZt.dot(zfixed)
```

Note that in Eq.(2.24): zpar= $\check{z}_2$ and zfixed= $\check{z}_{\mathrm{PAR}}$.
Furthermore in Eq.(2.26): Qzpar= $\boldsymbol{Q}_{\hat{\boldsymbol{z}}_2\hat{\boldsymbol{z}}_2}$ and $\boldsymbol{Q}_{\hat{\boldsymbol{b}}\hat{\boldsymbol{z}}_2} = \boldsymbol{Q}_{\hat{\boldsymbol{b}}\hat{\boldsymbol{a}}}$ Zpar.

3.3.6 ILS with Ratio Test

It is possible to apply ILS with the Ratio Test, see Section 2.7, where the threshold value is either determined using the Fixed Failure rate approach, or by using a fixed threshold value as specified by the user. In the first case, the routine ratioinv is used to determine the threshold value:

```
mu = ratioinv(Pf_FIX,Pf_ILS,n)
```

Inputs

Pf_FIX	Fixed failure rate (may be equal to 0.01 or 0.001)
Pf_ILS	ILS failure rate (or upper bound thereof)
n	Number of float ambiguities

Uses table stored in table1.txt and table10.txt.

The complete procedure to apply ILS with the Ratio Test as implemented in `LAMBDA` is:

```
Qzhat,Z,L,D,zhat,iZt = decorrel(Qahat,ahat)
zfixed,sqnorm = ssearch(zhat,L,D,2)
Ps_ILS = np.prod(2*norm.cdf(0.5/np.sqrt(D))-1)
if 1-Ps_ILS > Pf_FIX
   mu = ratioinv(Pf_FIX,1-Ps_ILS,n)
   if sqnorm[0]/sqnorm[1] > mu
      zfixed = zhat
   end
end
afixed = iZt.dot(zfixed)
```

Note that if the ILS failure rate (=`1-Ps_ILS`) is smaller than the required Fixed Failure rate, the fixed solution will always be accepted according to the FF-RT. This is the reason for including the `if`-statement which checks whethere (`1-Ps_ILS > Pf_FIX`) in the above procedure.

In case the user wants to specify the threshold value `mu`, it is not necessary to call `ratioinv` in the above procedure, and the aforementioned `if`-statement should be discarded as well.

There is no option implemented in the main `LAMBDA` routine to combine PAR with the Ratio Test. The procedure for that would be:

```
afixed,sqnorm,Ps_PAR,Qz,Z,nfixed = LAMBDA.main(ahat,Qahat,5)
if 1-Ps_PAR > Pf_FIX
   mu = LAMBDA.ratioinv(Pf_FIX,1-Ps_PAR,nfixed)
   if sqnorm[0]/sqnorm[1] > mu
      afixed = ahat
   end
end
```

Note that `ratioinv` should be applied with the PAR success rate (which is the output from `LAMBDA`), and with the number of fixed ambiguities `nfixed`.

Chapter 4

Getting started

It is most convenient to include the LAMBDA routine files in your working directory. The folder contains a demonstration routine `LAMBDAdemo` with examples of how the program can be used. Open the routine in your editor (e.g. Spyder); the comments will guide you through the different options. Some data examples (float ambiguity vector plus variance matrix) are included as well.

Chapter 5

Availability, Liability and Updates

5.1 Availability

The *Version 1.0* Python implementation of the LAMBDA software, as well as the Version 3.0 Matlab and Version 1.0 Fortran routines are available on request. A selection of papers on integer ambiguity resolution is included with the LAMBDA software package in a single PDF file.

5.2 Liability

Use of the accompanying LAMBDA software is allowed, but no liability for the use of the software will be accepted by the authors or their employer. Giving proper credits to the authors is the only condition posed upon the use of the LAMBDA software. We ask you to refrain from passing the software to third parties. Instead you are asked to pass our (e-mail) address to them, so we can send the software upon their request. The reason is, that in this way we have a complete overview of the users of the software, enabling us to keep everyone informed of further developments.

5.3 Updates

We welcome any suggestion for improvement of the code, the in-source documentation and the description in the report. We also would like to encourage you to communicate to us about results obtained with the LAMBDA method, and comparisons made with other methods. We would also be much obliged if you inform us in case you decide to use the method commercially. As said before, there are no restrictions on that, other than properly acknowledging the designers of the method and their employer. If you are planning to make a version in another language and would like to make it public, we would like you to contact us, in order to coordinate the efforts.

Bibliography

Abidin HA (1993). *Computational and geometrical aspects of on-the-fly ambiguity resolution*. Ph.D. Thesis, Dept. of Surveying Engineering, Techn. Report no.104, University of New Brunswick, Canada, 314 pp.

Boon F, Ambrosius B (1997). Results of real-time applications of the LAMBDA method in GPS based aircraft landings. In *Proc. of the International Symposium on Kinematic Systems in Geodesy, Geomatics and Navigation, Banff, Canada*, pp. 339–345.

Boon F, De Jonge PJ, Tiberius CCJM (1997). Precise aircraft positioning by fast ambiguity resolution using improved troposphere modelling. In *Proc. of ION GPS-1997, Kansas City MO*, pp. 1877–1884.

Chang X, Yang X, Zhou T (2005). MLAMBDA: a modified LAMBDA method for integer least-squares estimation. *Journal of Geodesy*, 79: 552–565.

Chen Y (1997). An approach to validate the resolved ambiguities in GPS rapid positioning. *Proc. of the International Symposium on Kinematic Systems in Geodesy, GeomaticsandNavigation,Banff, Canada*: 301–304.

De Jonge PJ, Tiberius C (1996a). The LAMBDA method for integer ambiguity estimation: implementation aspects, LGR-Series, No 12. Technical report, Delft University of Technology.

De Jonge PJ, Tiberius CCJM (1996b). Integer ambiguity estimation with the LAMBDA method. In *Proc. of IAG Symposium No. 115, GPS trends in terrestrial, airborne andspaceborneapplications,G. Beutler et al. (eds), Springer Verlag*, pp. 280–284.

De Jonge PJ, Tiberius CCJM, Teunissen PJG (1996). Computational aspects of the LAMBDA method for GPS ambiguity resolution. In *Proc. of ION GPS-1996, Kansas City MO*, pp. 935–944.

Euler HJ, Schaffrin B (1991). On a measure for the discernibility between different ambiguity solutions in the static-kinematic GPS-mode. In *Proceedings of* Kinematic Systems in Geodesy, Surveying, and Remote Sensing, *International Association of Geodesy Series*, Volume 107, pp. 285–295. Springer-Verlag, New York.

Han S (1997). Quality control issues relating to instantaneous ambiguity resolution forreal-timeGPSkinematic positioning. *Journal of Geodesy*, 71(6): 351–361.

Han S, Rizos C (1996). Validation and rejection criteria for integer least-squares estimation. *Survey Review*, 33(260): 375–382.

Jonkman NF (1998). *Integer GPS ambiguity estimation without the receiver-satellite geometry*. Delft Geodetic Computing Centre, LGR series No.18, Delft University of Technology,95pp.

Joosten P (2001). The LAMBDA-Method: Matlab^{TM} Implementation. Technical report, Mathematical Geodesy and Positioning, Delft University of Technology.

Joosten P, Tiberius CCJM (2000). Fixing the ambiguities: are you sure they're right. *GPS World*, 11(5): 46–51.

Joosten P, Tiberius CCJM (2002). LAMBDA: FAQs. *GPS Solutions*, 6(1-2): 109 – 114.

Landau H, Euler HJ (1992). On-the-fly ambiguity resolution for precise differential positioning. *Proc. of ION GPS-1992, Albuquerque NM*: 607–613.

Strang G, Borre K (1997). *Linear Algebra, Geodesy, and GPS*. Wellesley-Cambridge Press, Wellesley MA.

Teunissen PJG (1993). Least-squares estimation of the integer GPS ambiguities. invited lecture. In *Section IV Theory and Methodology, IAG General Meeting, August*, Beijing, China.

Teunissen PJG (1998a). A class of unbiased integer GPS ambiguity estimators. *Artificial Satellites*, 33(1): 4–10.

Teunissen PJG (1998b). *GPS carrier phase ambiguity fixing concepts*. In: PJG Teunissen and Kleusberg A, *GPS for Geodesy*, Springer-Verlag,Berlin.

Teunissen PJG (1998c). On the integer normal distribution of the GPS ambiguities. *Artificial Satellites*, 33(2): 49–64.

Teunissen PJG (1998d). Some remarks on GPS ambiguity resolution. *Artificial Satellites*, 32(3): 119–130.

Teunissen PJG (1998e). Success probability of integer GPS ambiguity rounding and bootstrapping. *Journal of Geodesy*, 72: 606–612.

Teunissen PJG (1999). An optimality property of the integer least-squares estimator. *Journal of Geodesy*, 73(11): 587–593.

Teunissen PJG (2000). ADOP based upperbounds for the bootstrapped and the least-squares ambiguitysuccessrates. *Artificial Satellites*, 35(4): 171–179.

Teunissen PJG (2001a). GNSS ambiguity bootstrapping: Theory and applications. In *Proc. KIS2001, International Symposium on Kinematic Systems in Geodesy, Geomatics and Navigation, June 5-8, Banff, Canada*, pp. 246–254.

Teunissen PJG (2001b). Integer estimation in the presence of biases. *Journal of Geodesy*, 75: 399–407.

Teunissen PJG (2001c). Statistical GNSS carrier phase ambiguity resolution: a review. In *Proc. of 2001 IEEE Workshop on Statistical Signal Processing, August 6-8, Singapore*, pp. 4–12.

Teunissen PJG (2002). The parameter distributions of the integer GPS model. *Journal of Geodesy*, 76(1): 41–48.

Teunissen PJG (2003a). Integer aperture GNSS ambiguity resolution. *Artificial Satellites*, 38(3): 79–88.

Teunissen PJG (2003b). Theory of carrier phase ambiguity resolution. *Wuhan University Journal of Natural Sciences*, 8: 471–484. 10.1007/BF02899809.

Teunissen PJG (2010). Mixed integer estimation and validation for next generation GNSS. In W. Freeden, M. Nashed, and T. Sonar (Eds.), *Handbook of Geomathematics*, pp. 1101–1127. Springer Berlin Heidelberg.

Teunissen PJG, De Jonge PJ, Tiberius CCJM (1996). The volume of the GPS ambiguity ambiguity search space and its relevance for integer ambiguity resolution. In *Proc. of ION GPS-1996, Kansas City MO*, pp. 889–898.

Teunissen PJG, De Jonge PJ, Tiberius CCJM (1998). Performance of the LAMBDA method for fast GPS ambiguity resolution. *Navigation*, 44(3): 373–383.

Teunissen PJG, Joosten P, Tiberius CCJM (1999). Geometry-free ambiguity success rates in case of partial fixing. In *Proc. of ION National Technical Meeting 1999 & 19th Biennal Guidance Test Symposium, San Diego CA*, pp. 201–207.

Teunissen PJG, Joosten P, Tiberius CCJM (2000). Bias robustness of GPS ambiguity resolution. In *Proc. of ION GPS-2000, Salt Lake City UT*, pp. 104–112.

Teunissen PJG, Odijk D (1997). Ambiguity Dilution of Precision: definition, properties and application. In *Proc. of ION GPS-1997, Kansas City MO*, pp. 891–899.

Teunissen PJG, Verhagen S (2008). GNSS Carrier Phase Ambiguity Resolution: Challenges and Open Problems. In *M Sideris (ed)* Observing our changing Earth*, International Association of Geodesy*, Volume 133, pp. 785–792. Springer Verlag, Berlin.

Teunissen PJG, Verhagen S (2009). The GNSS Ratio-Test Revisited - A better way of using it. *Survey Review*, 41(312): 138–151.

Teunissen PJG, Verhagen S (2011, March/April). Integer Aperture Estimation - A Framework for GNSS Ambiguity Acceptance Testing. *Inside GNSS*, 2011: 66–73.

Tiberius CCJM, De Jonge PJ (1995). Fast positioning using the LAMBDA method. In *Proc. of DSNS'95, Bergen, Norway, paper no.30*. The Nordic Institute of Navigation, Oslo.

Verhagen S (2005). On the reliability of integer ambiguity resolution. *Navigation*, 52(2): 99–110.

Verhagen S, Joosten P (2004). Analysis of integer ambiguity resolution algorithms. *European Journal of Navigation*, 2(4): 38–50.

Verhagen S, Teunissen PJG (2006). New global navigation satellite system ambiguity resolution method compared to existing approaches. *Journal of Guidance, Control, and Dynamics*, 29(4): 981–991.

Wang J, Stewart MP, Tsakiri M (1998). A discrimination test procedure for ambiguity resolution on-the-fly. *Journal of Geodesy*, 72(11): 644–653.