The LAMBDA-Method: Matlab$^{\text{TM}}$ Implementation

Version 2.1, 12th March 2001

Peter Joosten
Mathematical Geodesy and Positioning
Civil Engineering and Geosciences
Delft University of Technology, The Netherlands
Email: `P.Joosten@geo.tudelft.nl`

12th March 2001

1 Introduction

One of the keys to precise positioning using GPS is to resolve the double differenced carrier phase ambiguities. One of the methods to do this is the LAMBDA-method, developed at the Delft University of Technology. LAMBDA stands for “Least-squares AMBiguity Decorrelation Adjustment”. When using the LAMBDA-method, the integer least squares ambiguity estimates are computed in two steps. First the ambiguities are decorrelated, by means of the Z-transformation. Then the integer minimization problem is solved by a discrete search over an ellipsoidal region, the ambiguity search ellipsoid.

The method has been introduced in [7] by P.J.G. Teunissen. The method was first implemented in Fortran-77, by P.J. de Jonge and C.C.J.M. Tiberius. A detailed description of this implementation is given in [3]. The code was translated into Matlab$^{\text{TM}}$ by K. Borre (see [6])

When a research on the stochastic properties of the estimated integer ambiguities was started in 1998, initially this Matlab$^{\text{TM}}$ implementation was used. However it was found that some improvements were necessary. One of the biggest drawbacks of the earlier version was the fact that it was not possible to find more than two candidates. As a result of it, it was decided to create a new version, based on both the description in [3] and [6].

Compared to the earlier version, this new version offers:

- Improved readability of the software.
- A more modular approach, enabling one to perform for example only the decorrelation-step, only the search-step, or both steps.
- It is now possible to find a number of candidates different from 2, which can be useful for research purposes.
- Sometimes intermediate output can be very helpful for research purposes, sometimes it can be rather annoying. It is now possible to switch on or off this output.
- In the original code, not much of Matlab$^{\text{TM}}$ ’s vectorized computation possibilities are used. Using this can improve both readability of the code and computational speed.

- The original code is a rather extensive set of subroutines. Readability was improved by reducing the number of subroutines.
- The dimension of the problem is derived from the input using the `size`-function. This decreases the number of arguments to functions, and therefore the probability of mistakes.

2 Ambiguity resolution in brief

2.1 Parameter estimation

GPS ambiguity resolution is the process of resolving the unknown cycle ambiguities of the double difference carrier phase data as integers. The GPS models on which ambiguity resolution is based, can all be cast in the following conceptual frame of linear(ized) observation equations:

$$E\{y\} = Aa + Bb + e \qquad (1)$$

Where y is the given GPS data vector, a and b are the unknown parameter vectors of order n and p respectively, and where e is the noise vector of order m. The matrices A and B are the corresponding design matrices of order $m \times n$ and $m \times p$ respectively. The data vector y will usually consist of the "observed minus computed" single- or dual-frequency DD phase and/or pseudo range (code) observations, accumulated over all observation epochs. The entries of vector a are then the DD carrier phase ambiguities, expressed in units of cycles rather than range. They are known to be integers. The entries of vector b will consist of the remaining unknown parameters, such as for instance baseline components (coordinates) and possibly atmospheric delay parameters (troposphere, ionosphere). The procedure which is usually followed for solving the above model can be divided into three steps (for more details the reader is referred to e.g. [7], [3] or to the textbooks [23], [5] and [6]).

- In the first step one simply disregards the integer constraints on the ambiguities and performs a standard adjustment. As a result one obtains the (real-valued) least-squares estimates of a and b, together with their variance-covariance matrix. This solution is often referred to as the "float" solution, and denoted by $\hat{a}$ and $\hat{b}$. The corresponding variance-covariance matrices are denoted by $Q_{\hat{a}}$ and $Q_{\hat{b}}$.
- In the second step the "float" ambiguity estimate $\hat{a}$ is used to compute the corresponding integer ambiguity estimate. This implies that a mapping $F : R^n \to Z^n$, from the n-dimensional space of reals to the n-dimensional space of integers is introduced.
- Once the integer ambiguities are computed, they are used in the third step to finally correct the "float" estimate of b, using the following equation:

$$\begin{array}{rcl} \check{b} & = & \hat{b} - Q_{\hat{b}\hat{a}} Q_{\hat{a}}^{-1} (\hat{a} - \check{a}) \\ Q_{\check{b}} & = & Q_{\hat{b}} - Q_{\hat{b}\hat{a}} Q_{\hat{a}}^{-1} Q_{\hat{a}\hat{b}} \end{array} \qquad (2)$$

  The resulting $\check{b}$ and corresponding $Q_{\check{b}}$ are referred to as the "fixed" solution. Note that the resulting variance-covariance matrix is based on the assumption that after the "fixing step" the ambiguities are *known* quantities. Whether or not this is an acceptable assumption depends on the actual situation, and is left to the user.

2.2 Integer least-squares estimation

In the previous paragraph a mapping from the n-dimensional space of reals to the n-dimensional space of integers was mentioned. Numerous possible mappings exist, ranging from simply rounding each element of the ambiguity vector to its nearest integer, to much more complicated mappings as the integer least squares estimator.

Recently it has been proven (see [18] or [17]) that the use of the integer least squares estimator is optimal in a sense that this estimator maximizes the probability that indeed the correct integers are found. Unfortunately no closed form equations exist which solves this problem. Instead the solution need to be found by means of a search procedure. This procedure is described in detail in chapter 4 of [3].

2.3 Decorrelation of ambiguities

In theory we could perform the search as mentioned in the previous section on the original double difference ambiguities. However due to the usually high correlation between the elements of the ambiguity vector, as well as the poor precision of these elements, this would be a cumbersome process. By reparameterizing the ambiguities, we can increase the precision of the elements of the ambiguity vector while at the same time decreasing the correlation between them. This reparameterization is referred to as the Z-transformation, and transforms the original double difference ambiguities into a new set of ambiguities $z = Z^* a$. The corresponding variance-covariance matrix is transformed accordingly: $Q_{\hat{z}} = Z^* Q_{\hat{a}} Z$. The Z-matrix, or transformation matrix depends on the variance-covariance matrix of the float ambiguities. Routine `decorrel` will compute the Z-matrix for each given variance-covariance matrix. The underlying algorithm is described in detail in chapter 3 of [3].

2.4 Summary

The LAMBDA-method is a method to solve the problem of mapping estimated ambiguities from the n-dimensional space of reals to the n-dimensional space of integers. It offers a way of finding these integer ambiguities according to the integer least squares criterion. It has been found that this problem can be solved in a much more efficient way, when the ambiguities are decorrelated first. Therefore the LAMBDA-method consists of two steps. First the ambiguities are decorrelated, by means of the Z-transformation. Then the integer minimization problem is solved by a discrete search over an ellipsoidal region, the ambiguity search ellipsoid. This also explains the name “LAMBDA”, which stands for “Least-squares AMBiguity Decorrelation Adjustment”.

Summarizing, the LAMBDA-method uses the least squares estimator to estimate integer ambiguities, using float ambiguities and the corresponding variance-covariance matrix as input.

3 Usage of the Matlab^{TM} routines

3.1 On the input

Input for the problem in general consists of a vector of float ambiguities, and the corresponding variance-covariance matrix. Assuming the dimension of the problem is n, the vector of float ambiguities ($\hat{a}$) needs to be a **column**-vector of n elements. The corresponding variance-covariance matrix ($Q_{\hat{a}}$) should be square ($n \times n$), symmetric and positive definite.

When using the main-routines, `lambda1.m` or `lambda2.m`, this is checked. These checks can be relatively time-consuming on slow systems. If this is the case in your situation, these tests can easily be switched off by out-commenting them in the source. Also, if you decide to use the separate routines by itself, you are responsible for the correctness of the input yourself.

3.2 Integer least squares estimation

In order to perform a complete integer least-squares estimation, two routines are provided. The first routine (`lambda1.m`) offers several extra possibilities, such as intermediate output and a user-selectable number of candidates to be returned. The second routine (`lambda2.m`) performs a straightforward estimation. It will return 2 candidates, the best, and the second-best, which can be used for validation purposes. It is suggested to use `lambda1.m` for research and analysis, and `lambda2.m` for production work[1].

The routines are used as follows:

```
[afixed,sqnorm,Qahat,Z] = lambda1 (afloat,Qahat,[ncands,fidlog]);
```

and

```
[afixed,sqnorm,Qahat,Z] = lambda2 (afloat,Qahat);
```

respectively. Note that according to Matlab^{TM} practice, output arguments in which you are not interested may be left out. Thus, if you are only interested in estimated integer ambiguities, and nothing else, the following call would be most appropriate:

```
[afixed] = lambda2 (afloat,Qahat);
```

3.3 Decorrelation

If you are just interested in the decorrelation process, and the corresponding Z-transformation, it is possible to use routine `decorrel.m`. Starting from a variance-covariance matrix it will compute the Z-transformation (Z-matrix). Optionally it is possible to supply original ambiguities, in which case the routine will return the decorrelated float ambiguities. The routines are called as follows:

```
[Q,Z,L,D] = decorrel (Q);
```

or

```
[Q,Z,L,D,z] = decorrel (Q,a);
```

3.4 Search

If you have ambiguities, which are already decorrelated, or for some reason you would like to perform the search on the original untransformed double difference ambiguities, you can use the routine `lsearch.m`. This routine requires as input L and D as will result from the L^tDL-decomposition of the variance-covariance matrix. `lsearch.m` also will need the size of the search-ellipsoid as input. Furthermore it is required that the float ambiguities have a value between -1 and 1. So if necessary, they should be shifted over an integer number of gridpoints in order to meet this requirement. After estimating the integers, they should be shifted backwards. An appropriate calling-sequence of routines would be:

[1]Although for production work, because of execution-speed using a compiled version of the original FORTRAN code might be more appropriate

```
incr                 = afloat - rem(afloat,1);
afloat               = rem(afloat,1);

[L,D]                = ldldecom (Qahat);
Chi2                 = chistart (D,L,afloat,ncands);
[afixed,sqnorm,ierr] = lsearch (afloat,L,D,Chi2,ncands);

afixed               = afixed + repmat(incr,1,ncands);
```

where `ncands` indicates the number of candidates to be computed. See the information on the separate routines for further details.

3.5 Demonstration routine

`ldemo.m` serves as a demonstration of the use of the LAMBDA-routines. The demonstration allows you to enter or modify float ambiguities and the corresponding covariance matrix. It will then estimate the integer ambiguities, using the LAMBDA-method.

Use of this demonstration requires a computer which is capable of displaying graphics, the routines were tested on a PC running Windows 95, and on a Unix-system using an X-terminal. The LAMBDA-routines itself do not require any graphics capabilities.

The demonstration has the following limitations:

1. Dimension of the problem is limited to 6
2. All numbers (floats and elements of the variance-covariance matrix) should fit to the format “%8.4f”, this in order to make the demo “readable”).
3. Although some checks on the validity of the input are performed, they are not complete. Erroneous input can cause the LAMBDA-routines itself to complain, in which case the messages will be written to the standard Matlab^{TM} screen, instead of the demo-screen.

After starting Matlab^{TM} , the demonstration can be started by the command:

```
ldemo
```

This will present you with the following screen:

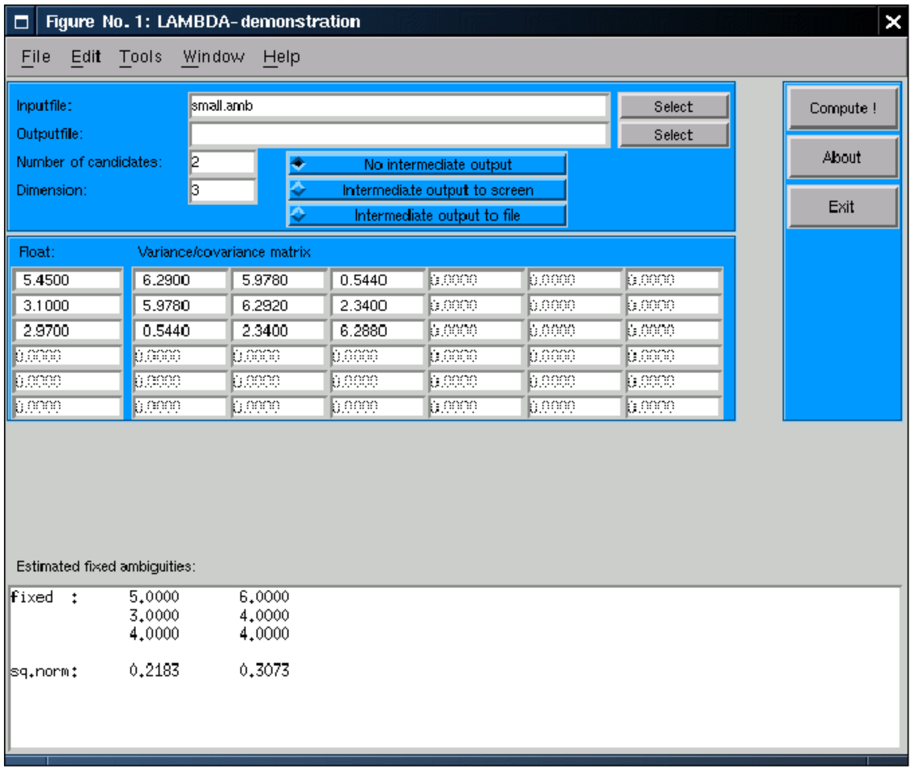
Figure No. 1: LAMBDA-demonstration
File Edit Tools Window Help
Select
Select
Compute !
About
Exit

Inputfile: small.amb
Outputfile:
Number of candidates: 2
Dimension: 3

[✓ No intermediate output]
Intermediate output to screen
Intermediate output to file

Float:	Variance/covariance matrix					
5.4500	6.2900	5.9780	0.5440	0.0000	0.0000	0.0000
3.1000	5.9780	6.2920	2.3400	0.0000	0.0000	0.0000
2.9700	0.5440	2.3400	6.2880	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Estimated fixed ambiguities:

```
fixed  :     5.0000       6.0000
             3.0000       4.0000
             4.0000       4.0000

sq.norm:     0.2183       0.3073
```

This demonstration is intended to be self-explanatory, however a few remarks may be helpful:

- After modifying the input, new corresponding integers are not estimated automatically. Clicking the "compute!" button carries out the estimation process.
- If you are editing the variance-covariance manually, the necessary symmetry is taken care of automatically. If however you are editing a file (like `small.amb` in the example above) which you want to use, you should make sure yourself that this matrix is symmetric.
- If you want intermediate output to file, be sure to select an outputfile. If this file already exists, it will be overwritten. If you select "Intermediate output to screen", this intermediate output is written to the standard Matlab^{TM} screen/window.

4 Routines

4.1 `decorrel.m`

Performs a decorrelation of the variance-covariance matrix $Q_{\hat{a}}$. This is done by a reparameterization, or as we call it, a Z-transformation. The routine computes the Z-transformation, and uses it to decorrelate the original ambiguities and the corresponding variance-covariance matrix as follows:

Usage: `[Q,Z,L,D,z] = decorrel (Q,a);`

with	`Q`	Input	Variance-covariance matrix of the original ambiguities
	`a`	Input	Original ambiguities
	`Q`	Output	Decorrelated variance-covariance matrix
	`Z`	Output	Transformation matrix
	`L`	Output	L as obtained from a $L^t DL$-decomposition of the decorrelated variance-covariance matrix of the ambiguities
	`D`	Output	D as obtained from this $L^t DL$-decomposition
	`z`	Output	Decorrelated ambiguities

4.2 `lambda1.m`

This is one of the main routines. Starting from original ambiguities and the corresponding variance-covariance matrix, both the decorrelation and the search process are carried out. The routine delivers the estimated integer ambiguities, the decorrelated variance-covariance matrix and the Z-transformation matrix. It is possible to write intermediate results to either the screen or file. The number of candidates to be returned can be chosen by the user.

Usage: `[afixed,sqnorm,Qzhat,Z] = lambda1 (afloat,Qahat[,ncands,fidlog]);`

with:	`afloat`	Input	Original float ambiguities
	`Qahat`	Input	Corresponding variance-covariance matrix
	`ncands`	Input	Number of candidates to be returned (optional, default = 2)
	`fidlog`	Input	File-identifier for intermediate output (optional, default = 0) 0: No intermediate output (default) 1: Intermediate output to screen n: Intermediate output to file (needs to be opened)
	`afixed`	Output	Array with estimated integer candidates, sorted according to the corresponding squared norms, best candidate first
	`sqnorm`	Output	Corresponding squared norms
	`Qzhat`	Output	Decorrelated variance-covariance matrix
	`Z`	Output	Z-transformation matrix

In case the number of requested candidates is equal to or less than the dimension of the problem plus one (`ncands` $\leq n+1$), the size of the search ellipsoid is computed using routine chistart (see below for a description). Otherwise an approximation (based on the volume of the search-ellipsoid) is used, which does not guarantee that the requested number of candidates will fall inside the ellipsoid. This approximation is chosen in such a way, that chances for success are high. If the requested number of candidates will not be found, an error message will be generated.

4.3 `lambda2.m`

This is the other one of the main routines. Compared to routine `lambda1.m`, this routine is simplified. It is not possible to generate intermediate output, and it is not possible to select the number of candidates to be computed (the routine will return 2 candidates).

Usage: `[afixed,sqnorm,Qahat,Z] = lambda2 (afloat,Qahat);`

with:	`afloat`	Input	Original float ambiguities
	`Qahat`	Input	Corresponding variance-covariance matrix
	`afixed`	Output	Array with estimated integer candidates, sorted according to the corresponding squared norms, best candidate first
	`sqnorm`	Output	Corresponding squared norms
	`Qzhat`	Output	Decorrelated variance-covariance matrix
	`Z`	Output	Z-transformation matrix

4.4 `chistart.m`

This routine computes a suitable size for the search ellipsoid, where “suitable” means that the search ellipsoid needs to be a small as possible, in order to assure a fast solution, but at the same time containing at least the correct integer candidate.

Usage: `Chi2 = chistart (D,L,a,ncands[,factor]);`

with:	`L`	Input	L as obtained from a L^tDL-decomposition of the decorrelated variance- covariance matrix of the ambiguities
	`D`	Input	D as obtained from this L^tDL-decomposition
	`a`	Input	Z-transformed float ambiguities
	`ncands`	Input	Number of candidates to be returned
	`factor`	Input	Multiplication factor for the volume (see below, optional, default = 1.5)
	`Chi2`	Output	Size of the search ellipsoid (squared norm)

If the requested number of candidates is no more than $n+1$, with n the dimension of the problem, the size of the ellipsoid can be computed in such a way that it is guaranteed that at least this number of candidates, will be inside the ellipsoidal region. If applied to the decorrelated ambiguities, it is likely that there will be not much more candidates inside this ellipsoidal region, which assures a faster search. If this method would be applied to the original ambiguities, an abundance of candidates may be the result.

The algorithm is based on the bootstrapping estimator, where in a first step the best determined ambiguity is rounded to its nearest integer. The remaining ambiguities are then corrected, using their correlation with the first ambiguity. Then the second-best element of the ambiguity vector is rounded to the nearest integer, and the remaining ones are corrected. This process is continued until all elements are estimated as integers. The components of the bootstrapped estimator are given as:

$$\begin{array}{rcccl} \check{a}_{B,1} & = & [\hat{a}_1] & & \\ \check{a}_{B,2} & = & [\hat{a}_{2|1}] & = & [\hat{a}_2 - \sigma_{\hat{a}_2\hat{a}_1}\sigma_{\hat{a}_1}^{-2}(\hat{a}_1 - \check{a}_{B,1})] \\ & \vdots & & & \\ \check{a}_{B,n} & = & [\hat{a}_{n|N}] & = & [\hat{a}_n - \sum_{i=1}^{n-1}\sigma_{\hat{a}_n\hat{a}_{i|I}}\sigma_{\hat{a}_{i|I}}^{-2}(\hat{a}_{i|I} - \check{a}_{B,i})] \end{array} \quad (3)$$

where the shorthand notation $\hat{a}_{i|I}$ stands for the ith float ambiguity obtained through a conditioning on the previous $I = \{1,\ldots,(i-1)\}$ sequentially rounded ambiguities. If more than 1 candidate is requested, components of the candidate vector are also choosen as the second nearest integer to the real value, to assure that the requested number of candidates will fall within the search space.

Note that this differs from the original implemenation as described in [3], where the size of the search ellipsoid is set based in candidates obtained by *ordinary rounding* instead of *conditional rounding*. It was found however that in some extreme cases this lead to excessively long search times.

If more than $n+1$ candidates are requested, using the Matlab implementation, it is currently not possible[2] to compute a size that guarantees that at least the requested number of candidates will be found. In this case the fact that the volume of the search-ellipsoid is a good indicator for the number of candidates inside this ellipsoid is utilized. Volume and squared norm of the search-ellipsoid are related as follows:

$$E_n = \chi^n \sqrt{\|Q_{\hat{a}}\|} V_n \quad ; \quad V_n = \frac{2}{\pi} \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2})}$$

This approximation usually works well enough if the volume (E_n) is taken as 1.5 times the number of candidates requested. It might happen that the ellipsoid does not contain enough candidates. In this case it would be necessary to increase `factor`.

4.5 `ldldecom.m`

This routine performs a L^tDL-decomposition. Note that the decomposition is performed "backwards", starting at the end of the input-matrix. As a consequence the result is such that the following equation is satisfied:

$$Q = L^t * D * L$$

This corresponds to the *third* option as described on page 8 of [3]. Use of this routine is straightforward:

Usage: `[L,D] = ldldecom(Q);`

with	`Q`	Input	Matrix to be decomposed
	`D,L`	Output	L^tDL-decomposition

4.6 `lsearch.m`

This routine performs the actual search for the integer ambiguities. The purpose is to find the candidate $\check{a}$, which minimizes following expression:

$$(\hat{a}-\check{a})^* Q_{\hat{a}}^{-1} (\hat{a}-\check{a})$$

In order to do this, a discrete search is performed over the following ellipsoid:

$$(\hat{a}-\check{a})^* Q_{\hat{a}}^{-1} (\hat{a}-\check{a}) \leq \chi^2$$

This will return the candidate $\check{a}$ which has the shortest distance to the float solution $\hat{a}$, in the metric of the variance-covariance matrix. When calling this routine, it is assumed that the ambiguities are already decorrelated. Using the decorrelated float ambiguities and the L^tDL-decomposition of the corresponding variance-covariance matrix, the requested number of candidates is searched for. The routine will return (if possible), the `ncands` best candidates, sorted according the corresponding squared norms.

[2]Note that in theory it is possible to choose the size of the search-ellipsoid in such a way that it is guarenteed that the requested number of candidates will be found, by performing bootstrapping but using the 3rd, 4th and so on, nearest integer. This is however not implemented in the Matlab^{TM} routines.

Usage: `[afixed,sqnorm,ierr] = lsearch (afloat,L,D,Chi2,ncands);`

with:	`afloat`	Input	Decorrelated float ambiguities
	`L`	Input	L as obtained from a L^tDL-decomposition of the decorrelated variance-covariance matrix of the ambiguities
	`D`	Input	D as obtained from this L^tDL-decomposition
	`Chi2`	Input	Squared size of the search ellipsoid
	`ncands`	Input	Number of candidates requested by the user
	`afixed`	Output	Array with estimated integer candidates, sorted according to the corresponding squared norms, best candidate first
	`sqnorm`	Output	Corresponding squared norms
	`ierr`	Output	Error-code: 0: No errors found 1: The number of candidates found is less than requested

4.7 `writemat.m`

Write a matrix to a file or screen

Usage: `writemat (fid,text,matrix[,format]);`

with:	`fid`	Input	File-identifier: 0: Do not write 1: Write to the screen (Matlab^{TM} default) n: Write to a file, which should be open prior to using this routine
	`text`	Input	Descriptive text
	`matrix`	Input	The matrix to be printed
	`format`	Input	Format to be used (optional, default = “%15.5f”)

5 Computational aspects

Since this Matlab^{TM} version of the LAMBDA-method is intended mainly or research purposes, more attention was paid to readability of the code than to computational speed. It might be interesting to have a look at some timing results, especially to find out which part of the process is the most time-consuming. Two sample datasets were used, a 3-dimensional and a 12-dimensional problem. For the computations, a Pentium PC, 120Mhz, running Windows 95 was used. The table below summarizes the results (using routine `lambda2.m`):

Case	Total time (sec)	Decorrelation	Chistart	search	Other
3D, V2.0	0.0437	56 %	10 %	29 %	5 %
3D, V1.0	0.0708	40 %	8 %	19 %	33 %
12D, V2.0	0.2603	94 %	1 %	5 %	0 %
12D, V1.0	0.4085	93 %	1 %	4 %	2 %

Clearly the improvement over the original Matlab^{TM} version (here referred to as V1.0) can be seen. This improvement is mainly caused by using a vectorized approach where possible. Another thing that can be seen here, is the fact that most of the time is spend on the decorrelation process, especially with the larger dimensions.

6 Installation/System requirements

6.1 System requirements

The routines were written and tested on a PC, running Windows 95, with Matlab^{TM} version 5.1. Later on, the routines were also tested using Matlab 5.2 on a PC, running Linux (RedHat 5.2, kernel 2.0.36) as well as on a HP9000, running the HPUX operating system (version B.10.20). Although not tested, it is believed that the routines will run on any system on which Matlab^{TM} is installed. As far as known, none of the newer additions to Matlab^{TM} were used, so it should work on older versions as well. The included demonstration requires graphical capabilities, so a windows-system is necessary.

6.2 Installation

The software is provided as either a “zip-file”, or a unix “tar-file”. Extract all files, using your favorite software, to any directory you wish. In order to use the routines and/or the demo, there are two possibilities. First your can go to the directory where you place the files, using the `cd`-command. Alternatively you can add this directory to the Matlab^{TM} path. This can be done using one of the following commands, depending on your operating system:

```
Unix:        path (path,'/yourhome/yourdirectory');
VMS:         path (path,'YOURDISK:[yourhome.yourdirectory]');
DOS/Windows: path (path,'yourhome/yourdirectory');
Mac:         path (path,'Yourdisk:yourdirectory:');
```

It might be a good idea to place such a statement in your startup.m file, to avoid the necessity of issuing this command every time you wish to use the routines. This file can be found in directory `matlab/toolbox/local`, where Matlab is the root-directory of your Matlab^{TM} installation. If this file does not exist, you can create it yourself, using your favorite editor. For further information, see your local installation of Matlab^{TM} , or ask your local guru.

Distributed files:

`amb18.mat`	Example, based on a kinematic survey
`chistart.m`	M-file to compute the size for the search ellipsoid
`Contents.m`	M-file, table of contents for Matlab help system
`decorrel.m`	M-file to decorrelate the ambiguities
`geofree.mat`	Example, 2-dimensional, can be used in demonstration
`lambda.ps`	Documentation, in PostScript format
`lambda.pdf`	Documentation in Adobe’s portable document format (PDF)
`lambda1.m`	M-file, main routine, complete version, with options
`lambda2.m`	M-file, main routine, straightforward solution, no options
`large.mat`	Example, 12-dimensional, can **not** be used in demonstration
`ldemo.m`	M-file, demonstration
`ldldecom.m`	M-file, L^tDL-decomposition
`lsearch.m`	M-file, ambiguity search
`readme.txt`	Readme file with installation information
`sixdim.mat`	Example, 6-dimensional, can be used in demonstration
`small.mat`	Example, 3-dimensional, can be used in demonstration
`writemat.m`	M-file, write matrices, only used in demonstration

7 Availability

These Matlab^{TM} routines, as well as the original Fortran-routines are available on request from:
P. Joosten, Email: `P.Joosten@geo.tudelft.nl`.

This will give you the complete set of Matlab^{TM} routines, as well as the demonstration program and some sample data. Also included is this manual, in PDF and PostScript format.

General information on the LAMBDA-method can be found at:
`http://www.geo.tudelft.nl/mgp/lambda/index.html`,
where you can find an extensive bibliography on the LAMBDA-method, as well as a short description.

8 Use and 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, the Delft University of Technology.

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.

9 Testing and 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.

At this moment, several implementations in other computer languages (C, C++) have been derived from the original Fortran version; none of them has been made public yet. If you are planning to make a version in an other language, and would like to make it public, we would like you to contact us, in order to coordinate the efforts. Updated information about the LAMBDA method will be available at http://www.geo.tudelft.nl/mgp/lambda/

References

[1] DE JONGE, P., AND TIBERIUS, C. A new GPS ambiguity estimation method based on integer least squares. In *Proceedings of the 3rd International Symposium on Differential Satellite Navigation Systems DSNS'94, London, UK, April 18-22, paper no. 73, 9pp.* (1994).

[2] DE JONGE, P., AND TIBERIUS, C. Integer estimation with the LAMBDA method. In *Proceedings IAG Symposium No. 115, 'GPS trends in terrestrial, airborne and spaceborne applications* (1996), G. B. et al, Ed., Springer Verlag, pp. 280–284.

[3] DE JONGE, P., AND TIBERIUS, C. The LAMBDA method for integer ambiguity estimation: implementation aspects. Tech. Rep. LGR Series, No. 12, Delft Geodetic Computing Centre, Delft University of Technology, The Netherlands, 1996.

[4] DE JONGE, P., TIBERIUS, C., AND TEUNISSEN, P. Computational aspects of the LAMBDA method for GPS ambiguity resolution. In *Proceedings ION GPS-96, 9th International Technical Meeting of the satellite Division of the Institute of Navigation, Kansas City, Missouri, Sept 17-20* (1996), pp. 935–944.

[5] HOFMANN-WELLENHOF, B., LICHTENEGGER, H., AND COLLINS, J. *Global Positioning System: Theory and Practice*, fourth ed. Springer Verlag, Berlin, 1997.

[6] STRANG, G., AND BORRE, K. *Linear Algebra, Geodesy, and, GPS*. Wellesley-Cambridge Press, 1997.

[7] TEUNISSEN, P. Least-squares estimation of the integer GPS ambiguities. In *IAG General Meeting, Invited Lecture, Section IV Theory and Methodology, Beijing, China* (August 1993).

[8] TEUNISSEN, P. A new method for fast carrier phase ambiguity estimation. In *Proceedings of the IEEE Position, Location and Navigation Sympoisum PLANS'94, Las Vegas, NV, April 11-15* (1994), pp. 562–573.

[9] TEUNISSEN, P. The invertible GPS ambiguity transformations. *Manuscriptae Geodeticae 20* (1995), 489–497.

[10] TEUNISSEN, P. The least-squares ambiguity decorrelation adjustment: a method for fast GPS integer ambiguity estimation. *Journal of Geodesy 70*, 1-2 (1995), 65–82.

[11] TEUNISSEN, P. An analytical study of ambiguity decorrelation using dual frequency code and carrier phase. *Journal of Geodesy 70*, 8 (1996), 515–528.

[12] TEUNISSEN, P. On the geometry of the ambiguity search space with and without ionosphere. *Zeitschrift für Vermessungswesen 121*, 7 (1996), 332–341.

[13] TEUNISSEN, P. On the GPS double difference ambiguities and their partial search space. In *III Hotine-Marussi symposium on Mathematical Geodesy, L'Aquila, Italy, May 29-Jnue 3, 1994, Published in the proceedings IAG Symposium No. 114, 'Geodetic Theory Today* (1996), Springer Verlag, pp. 39–48.

[14] TEUNISSEN, P. Rank defect integer least-squares with applications to GPS. *Bolletino di Geodesia e Scienze Affini 55*, 3 (1996), 225–238.

[15] TEUNISSEN, P. Size and shape of l1/l2 ambiguity search space. In *XXI General Assembly of IUGG, Boulder, CO, July 2-14, 1995, published in the proceedings IAG Symposium No. 115, 'GPS trends in terrestrial, airborne and spaceborne applications* (1996), G. B. et al, Ed., Springer Verlag, pp. 275–279.

[16] TEUNISSEN, P. The least-squares ambiguity decorrelation adjustment: its performance on short GPS baselines and short observations spans. *Journal of Geodesy 71*, 10 (1997), 589–602.

[17] TEUNISSEN, P. The probability distribution of the GPS baseline for a class of integer ambiguity estimators. *Journal of Geodesy 73* (1999), 275–284.

[18] TEUNISSEN, P. A theorem on maximizing the probability of correct integer estimation. *Artificial Satellites 34 (1)* (1999), 3–9.

[19] TEUNISSEN, P., DE JONGE, P., AND TIBERIUS, C. On the spectrum of the GPS DD-ambiguities. In *Proceedings ION GPS-94, 7th International Technical Meeting of the Satellite Division of the Institute of Navigation, Salt Lake City, UT, Sept 20-23* (1994), pp. 115–124.

[20] TEUNISSEN, P., DE JONGE, P., AND TIBERIUS, C. The lambda method for fast GPS surveying. In *Proceedings of International Symposium 'GPS technology applications',Bucharest, Romania, September 26-29* (1996), pp. 203–210.

[21] TEUNISSEN, P., DE JONGE, P., AND TIBERIUS, C. A new way to fix carrier-phase ambiguities. *GPS World 6*, 4 (1996), 58–61.

[22] TEUNISSEN, P., DE JONGE, P., AND TIBERIUS, C. The volume of the GPS ambiguity ambiguity search space and its relevance for integer ambiguity resolution. In *Proceedings ION GPS-96, 9th International Technical Meeting of the Satellite Division of the Institute of Navigation, Missouri, NV, Sept 17-20* (1996), pp. 889–898.

[23] TEUNISSEN, P., AND KLEUSBERG, A., Eds. *GPS for Geodesy*, second enlarged ed. Springer Verlag, Berlin, 1998.

[24] TEUNISSEN, P., AND TIBERIUS, C. Integer least-squares estimation of the GPS phase ambiguities. In *Proceedings of the International Symp. On Kinematic Systems in Geodesy, Aug, 30-Sept. 2, 1994, Banff, Canada, 11 pp* (1994).

[25] TIBERIUS, C., AND DE JONGE, P. Fast positioning using the LAMBDA method. In *Proc. 4th Int. Symposium on Differential Satellite Navigation Systems (DSNS), 24-28 April 1995, Paper no. 30, Bergen, Norway, pp. 1-8* (1995).

[26] TIBERIUS, C., AND DE JONGE, P. Introduction to GPS surveying (part 4). *Geomatics Info Magazine 10* (1995), 61–67.

Contents

1 Introduction **1**

2 Ambiguity resolution in brief **2**
2.1 Parameter estimation 2
2.2 Integer least-squares estimation 2
2.3 Decorrelation of ambiguities 3
2.4 Summary 3

3 Usage of the Matlab^{TM} routines **3**
3.1 On the input 3
3.2 Integer least squares estimation 4
3.3 Decorrelation 4
3.4 Search 4
3.5 Demonstration routine 5

4 Routines **6**
4.1 `decorrel.m` 6
4.2 `lambda1.m` 7
4.3 `lambda2.m` 7
4.4 `chistart.m` 8
4.5 `ldldecom.m` 9
4.6 `lsearch.m` 9
4.7 `writemat.m` 10

5 Computational aspects **10**

6 Installation/System requirements **11**
6.1 System requirements 11
6.2 Installation 11

7 Availability **12**

8 Use and Liability **12**

9 Testing and Updates **12**