Integer Parameter Estimation in Linear Models with Applications to GPS[1]

Arash Hassibi   Stephen Boyd

Internet: arash@isl.stanford.edu & boyd@isl.stanford.edu
Tel: (650) 723-9833 & (650) 723-0002
Fax: (650) 723-8473

Information Systems Laboratory
Electrical Engineering Department
Stanford University
Stanford, CA 94305-4055

To Appear in IEEE Trans. on Signal Processing, November 1998

Abstract

We consider parameter estimation in linear models when some of the parameters are known to be integers. Such problems arise, for example, in positioning using carrier phase measurements in the global positioning system (GPS), where the unknown integers enter the equations as the number of carrier signal cycles between the receiver and the satellites when the carrier signal is initially phase locked.

Given a linear model, we address two problems: (1) The problem of *estimating* the parameters, and, (2) the problem of *verifying* the parameter estimates. We show that with additive Gaussian measurement noise:

- The maximum likelihood estimates of the parameters are given by solving an integer least-squares problem. Theoretically, this problem is very difficult computationally (NP-hard).
- Verifying the parameter estimates (computing the probability of estimating the integer parameters correctly) requires computing the integral of a Gaussian probability density function over the Voronoi cell of a lattice. This problem is also very difficult computationally.

However, by using a polynomial-time algorithm due to Lenstra, Lenstra, and Lovász (LLL algorithm),

- The integer least-squares problem associated with estimating the parameters can be solved efficiently in practice.
- Sharp upper and lower bounds can be found on the probability of correct integer parameter estimation.

We conclude the paper with simulation results that are based on a synthetic GPS setup.

Key words— Integer parameter estimation, Linear model, Integer least-squares, GPS.

[1]Research supported in part by AFOSR (under F49620-95-1-0318 and AF F49620-97-1-0459), NSF (under ECS-9707111 and EEC-9420565), and MURI (under F49620-95-1-0525). The US Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

1 Problem Statement

Throughout this paper, we assume that the observation $y \in \mathbf{R}^N$ depends on the unknown vectors $x \in \mathbf{R}^p$ (*real*) and $z \in \mathbf{Z}^q$ (*integer*) through the linear relationship

$$y = Ax + Bz + v, \tag{1}$$

where $A \in \mathbf{R}^{N \times p}$ (full column rank) and $B \in \mathbf{R}^{N \times q}$ (full column rank) are known matrices. The measurement noise $v \in \mathbf{R}^N$ is assumed to be Gaussian with known mean and covariance matrix. Without any loss of generality, we consider the case where the mean is zero and the covariance matrix is identity, i.e., $v \sim \mathcal{N}(0, I)$ (we can always re-scale equation (1) by the square root of the covariance matrix and remove the mean). The more general setup, including *a priori* knowledge on the distribution of x, and with component-wise constraints on z, is treated in §6.1.

Given y, A, and B, our goal is to *find* and *verify* estimates of the unknown parameters x and z. Therefore, two problems are addressed in this paper. First the problem of *estimating* the parameters, and second, the problem of *verifying* the parameter estimates.

1.1 Estimation problem

In order to obtain reasonable estimates for x and z we need a suitable estimation criterion. Here we consider maximum likelihood (ML) estimation. By definition, the ML estimates x_{ML} and z_{ML} of x and z respectively are found by maximizing the probability of observing y, i.e.,

$$(x_{\mathrm{ML}}, z_{\mathrm{ML}}) = \underset{(x,z) \in \mathbf{R}^p \times \mathbf{Z}^q}{\operatorname{argmax}} \quad p_{Y|X,Z}(y|x,z). \tag{2}$$

1.2 Verification problem

Since z is an integer-valued vector, we have the chance of estimating (or detecting) it exactly. Thus, for verifying the estimate of z, a reasonable choice is to compute the probability of estimating z correctly, i.e.,

$$P_{\mathrm{c}} = \mathbf{Prob}(z_{\mathrm{ML}} = z). \tag{3}$$

Clearly, the larger this value the higher the reliability on the estimate z_{ML}. Another reliability measure, associated with the real parameter x, is given by

$$\mathbf{Prob}(\|x - x_{\mathrm{ML}}\| < \epsilon), \tag{4}$$

where $\|\cdot\|$ denotes the Euclidean norm. This gives the probability of x_{ML} lying in a Euclidean ball of radius ϵ of its actual value. A combined reliability measure is

$$\mathbf{Prob}(\|x - x_{\mathrm{ML}}\| < \epsilon \ \ \& \ \ z_{\mathrm{ML}} = z) = \mathbf{Prob}(\|x - x_{\mathrm{ML}}\| < \epsilon \,|\, z_{\mathrm{ML}} = z) \cdot \mathbf{Prob}(z_{\mathrm{ML}} = z). \tag{5}$$

In practical cases, x_{ML} is "close" to x only if $z_{\mathrm{ML}} = z$, and therefore for "small" ϵ, (4) and (5) are "almost" equal. We will focus on reliability measures (3) and (5) in this paper.

2 Brief Overview

Recently, there has been a growing research interest in integer parameter estimation in linear models due to its application in the global positioning system (GPS), which is a navigation system that enables quick, accurate, and inexpensive positioning anywhere on the globe at any time. Roughly speaking, a GPS user calculates its position through triangulation by measuring its range (or distance) to satellites orbiting the earth using electromagnetic signals [HWLC97, PS96, Hur89, Hur93]. A special feature of GPS, which initially was not generally understood, is the ability to create extremely precise ranging signals by reproducing and tracking the RF carriers of the satellites. Because the carriers have a wavelength of 19cm, tracking them to 1/100th of a wavelength provides a precision of 2mm. Although modern receivers can attain these tracking precisions, one must still determine which carrier cycle is being tracked (e.g., relative to the start of modulation). In other words, carrier cycle resolution is vital for centimeter level accuracy in positioning. Using this carrier tracking method, the range measurement, y, can be well approximated to be linear in the position of the user, x, and the ambiguous cycles of the carrier signals, z, as given in (1).

Computing the ML estimates of x and z by observing y can be reduced to solving a least-squares problem in x and z. If there were no integer parameters ($B = 0$), we could compute x_{ML} explicitly as $x_{\mathrm{ML}} = (A^T A)^{-1} A^T y$. However, because z is an integer vector, the least-squares becomes very difficult to solve, at least in theory. In terms of computational complexity, the problem is at least as hard as a wide variety of other problems, for which no polynomial-time algorithm is known, or thought to exist. One purpose of this paper is to show that although this problem is *theoretically* difficult, it is possible to solve it efficiently in practical cases such as GPS.

A good overview of different approaches to the estimation problem can be found in [HWLC97, KT96] and references therein. In our opinion, Teunissen [Teu93, Teu94, Teu95b, Teu95a] was the first to address the estimation problem rigorously. Teunissen's approach, as noted in [Teu95a], is based on ideas from H. W. Lenstra [Len81] and eventually resulted in the LAMBDA (Least-squares AMBiguity Decorrelation Adjustment) method (see, e.g., [JT96, SB97]). Lenstra's work in [Len81] was modified a year later and led to the discovery of the LLL algorithm [LLL82, Len83], which can be thought of as an algorithm to approximate a set of real numbers by rational numbers with a common small denominator (simultaneous Diophantine approximation). We will show that the LLL algorithm (or its variations) can be used as a tool for a practically efficient method for solving the estimation problem (see, also [HB95, HB96, SB97]). Two nice features of the LLL algorithm that enable such a method, are its practical efficiency and its guaranteed performance. It is not clear whether Teunissen's "ambiguity decorrelation adjustment" method of [Teu95a] has any guaranteed performance or polynomial complexity, although it is reported to work well in practice.

The problem of verifying the ML estimates is also made computationally difficult because of the presence of the integer parameters. It turns out that the verification problem can also be approached using the LLL algorithm to speed up the algorithms for computing and bounding P_c as a measure of reliability. As far as we know, prior to our earlier publications [HB95, HB96], there had not been a sound theoretical treatment for the verification problem, although the (heuristic) methods frequently used in GPS applications had been reported to work well in practice [Teu95a, HWLC97]. Some of these methods, for example those in [TJ96, Teu97a], use χ^2-tests and the notion of the shortest lattice vector as we do, but a precise mathematical justification is lacking and there is no apparent connection between the reliability measures in these verification methods and $P_{\rm c}$.

To summarize, in this paper we will show that by using the LLL algorithm it is possible to solve the estimation and verification problems efficiently in practical cases such as GPS, although both of these problems are theoretically difficult. In §3 we formulate the estimation and verification problems more explicitly. In §4 we introduce the LLL algorithm and it is shown how it can be used to help solve the verification problem. In §5 the solution to the estimation problem is addressed where again the LLL algorithm plays a major role. Extensions to model (1) are considered in §6, simulation results (based on GPS) are given in §7, and the paper is concluded in §8.

Finally, it should be noted that although most recent applications in integer parameter estimation have been in GPS, there are many other fields in which these results could have impact. These include, for example, radar imaging, MRI, and communications (especially for multi-input/multi-output channels).

3 Problem formulation

Since $v \sim \mathcal{N}(0, I)$, the probability distribution of y given x and z is $\mathcal{N}(Ax + Bz, I)$ or

$$p_{Y|X,Z}(y|x,z) = \frac{1}{(2\pi)^{N/2}} e^{-\|y-Ax-Bz\|^2/2},$$

and therefore from (2)

$$(x_{\rm ML}, z_{\rm ML}) = \underset{(x,z)\,\in\,\mathbf{R}^p\times\mathbf{Z}^q}{\operatorname{argmin}} \|y - Ax - Bz\|^2,$$

Using completion of squares arguments it can be shown that

$$(x_{\rm ML}, z_{\rm ML}) = \underset{(x,z)\,\in\,\mathbf{R}^p\times\mathbf{Z}^q}{\operatorname{argmin}} \left(\left(x - \hat{x}_{|z}\right)^T \Xi^{-1} \left(x - \hat{x}_{|z}\right) + (z-\hat{z})^T \Upsilon^{-1} (z - \hat{z}) + y^T \Delta y \right), \quad (6)$$

where

$$\begin{aligned} \Xi &= (A^T A)^{-1}, & \Upsilon &= \left(B^T \left(I - A\Xi A^T\right) B\right)^{-1}, \\ \hat{x}_{|z} &= \Xi A^T (y - Bz), & \hat{z} &= \Upsilon B^T \left(I - A \Xi A^T\right) y, \end{aligned} \quad (7)$$

and Δ is a positive definite matrix that only depends on A and B. Note that our underlying assumption is that the inverses in (7) exist.

In (6) the term $y^T \Delta y$ is constant and does not affect $x_{\rm ML}$ and $z_{\rm ML}$. Since Ξ is positive definite, the first term in (6) is always non-negative and therefore, its minimizer for a given z is $x = \hat{x}_{|z}$ with a minimum value of zero. The second term in (6) does not depend on x and its minimum can be found by minimizing over z alone. Therefore, z_{ML} is simply the minimizer of the second term and $x_{\mathrm{ML}} = \hat{x}_{|z_{\mathrm{ML}}}$. In other words,

$$z_{\mathrm{ML}} = \underset{z \in \mathbf{Z}^q}{\operatorname{argmin}} \ (z - \hat{z})^T \Upsilon^{-1} (z - \hat{z}), \quad x_{\mathrm{ML}} = \hat{x}_{|z_{\mathrm{ML}}}. \tag{8}$$

Clearly, z_{ML} (and therefore x_{ML}) is given by the solution to a least-squares problem involving integer variables. If $\hat{z}$ were in $\mathbf{Z}^q$, then of course we would have $z_{\mathrm{ML}} = \hat{z}$. A simple approach is to *round* (each component of) $\hat{z}$ to the nearest integer vector. This simple approach does yield z_{ML} when the matrix Υ is diagonal, or nearly diagonal. But in the general case (and in practice) rounding $\hat{z}$ will give an integer vector that is very far from optimal. The problem of computing z_{ML} is known to be NP-hard [GLS88, Boa81].

In order to calculate the measures of reliability P_{c} and $\mathbf{Prob}(\|x - \hat{x}\| < \epsilon \mid z = z_{\mathrm{ML}})$ we need to know the probability distributions of our estimates. Note that we can easily read the covariance matrices of $\hat{z}$ and $\hat{x}_{|z}$ from (6) as $\mathbf{cov}(\hat{z}) = \Upsilon$ and $\mathbf{cov}\left(\hat{x}_{|z}\right) = \Xi$. Since $\hat{z}$ and $\hat{x}_{|z}$ are linear functions of a Gaussian random variable $y \sim \mathcal{N}(Ax + Bz, I)$, they are Gaussian random variables themselves and we have

$$\hat{z} \sim \mathcal{N}(z, \Upsilon) \quad \text{and} \quad \hat{x}_{|z} \sim \mathcal{N}(x, \Xi). \tag{9}$$

The probability distribution of $\hat{z}$ suggests that $\hat{z} = z + u$ where $u \sim \mathcal{N}(0, \Upsilon)$. Multiplying both sides by $G \triangleq \Upsilon^{-1/2}$ (the whitening matrix of u) and by defining $\tilde{y} \triangleq G\hat{z}$ we get

$$\tilde{y} = Gz + \bar{u}, \quad \text{where} \quad \bar{u} \sim \mathcal{N}(0, I). \tag{10}$$

Also, (8) can be written in terms of G and $\tilde{y}$ in the following equivalent form

$$z_{\mathrm{ML}} = \underset{z \in \mathbf{Z}^q}{\operatorname{argmin}} \ \|\tilde{y} - Gz\|^2. \tag{11}$$

The set $\{Gz \mid z \in \mathbf{Z}^q\}$ is a lattice in $\mathbf{R}^q$. Equation (11) states that z_{ML} is found by computing the closest lattice point to the vector $\tilde{y}$. In addition, according to (10), $\tilde{y}$ is off from Gz by $\bar{u}$. Therefore, as long as $\bar{u}$ is small enough, such that $\tilde{y}$ remains closest to Gz than any other point in the lattice, the estimate of z is correct (i.e., $z_{\mathrm{ML}} = z$). This is equivalent to $Gz + \bar{u}$ remaining inside the *Voronoi cell* of the lattice point Gz, i.e., the set of all points closer to Gz than any other point in the lattice. Because of the periodic structure of the lattice, the Voronoi cell of Gz is the translation of the Voronoi cell of the origin V_0 by the vector Gz, and therefore we can write

$$P_{\mathrm{c}} = \mathbf{Prob}(\bar{u} \in V_0) \quad \text{where} \quad \bar{u} \sim \mathcal{N}(0, I). \tag{12}$$

So P_{c} is equal to the probability of a q-dimensional normal random variable falling inside the Voronoi cell V_0.

According to (9), $\hat{x}_{|z} = x + w$ where $w \sim \mathcal{N}(0, \Xi)$, and if $z = z_{\mathrm{ML}}$ we have

$$\mathbf{Prob}(\|x - x_{\mathrm{ML}}\| < \epsilon \,|\, z = z_{\mathrm{ML}}) = \mathbf{Prob}(\|w\| < \epsilon),$$

which is equal to the probability of a p-dimensional normal random variable falling inside the ellipsoid $\{\ x \mid \|\Xi^{1/2} x\| \leq \epsilon\ \}$. From (5) we finally get

$$\mathbf{Prob}(\|x - x_{\mathrm{ML}}\| < \epsilon \ \ \& \ \ z = z_{\mathrm{ML}}) = P_{\mathrm{c}} \cdot \mathbf{Prob}(\|w\| < \epsilon) \tag{13}$$

Formulas (8), (11), (12), and (13) give the estimates and their measures of reliability, and will be used in later sections to solve the estimation and verification problems.

4 Verification Problem

As noted in §1, the probability of detecting the integer parameter z correctly, P_{c} in (12), can be used to verify the ML estimates. The reliability measure given in (13) also requires the calculation of P_{c}. Therefore, an essential step in verifying x_{ML} and z_{ML} is the calculation of P_{c}.

As noted in the previous section, P_{c} is given by the integral of a normal probability distribution over the Voronoi cell of the origin in the lattice $\{Gz \mid z \in \mathbf{Z}^q\}$. It turns out that the problem of finding the Voronoi cell of a lattice and performing the integration is a very challenging one computationally [HB95]. Therefore, we are motivated to find fast ways to approximate P_{c}, that for example, would enable real-time reliability tests instead. In fact, easily computable upper and lower bounds on P_{c} exist which become tight as P_{c} approaches unity. This is a very fortunate property, since in practice, we are mostly interested in our estimates when the probability of correct integer parameter estimation is close to one (high reliability), when the bounds are good.

In §4.1 we will see that the choice of G for a given lattice is highly non-unique. As a matter of fact for $q > 1$, there exists infinitely many matrices G_1 and G_2 with $G_1 \neq G_2$ such that

$$\{G_1 z \mid z \in \mathbf{Z}^q\} = \{G_2 z \mid z \in \mathbf{Z}^q\}.$$

Although P_{c} only depends on the lattice and not on the specific choice of G, this is not necessarily true for the upper and lower bounds on P_{c}. Therefore, we can sharpen these bounds by optimizing over the family of admissible matrix representations for the lattice. In the next section, we introduce the LLL algorithm that is an extremely useful tool for sharpening the bounds on P_{c}, and as will become clear later, it is also useful for improving the efficiency of the algorithm for solving the integer least-squares problem for computing z_{ML}.

4.1 Lattices and basis reduction

Let us denote the lattice $L \subset \mathbf{R}^q$ generated by the matrix $G = [g_1\, g_2 \cdots g_q] \in \mathbf{R}^{q \times q}$ by

$$L = \mathbf{L}(G) = \{Gz \mid z \in \mathbf{Z}^q\}\,.$$

The set of vectors $\{g_1, g_2, \ldots, g_q\}$ is called *a* basis for L since all lattice points are integer linear combinations of these vectors. The same lattice can be generated by other bases as well. For example, let F be any *unimodular* matrix, i.e., a matrix such that the elements of F and its inverse F^{-1} are all integers (or equivalently, the elements of F are integers and $|\det F| = 1$). Then the mapping from z to Fz is a one-to-one and onto from the integer lattice into itself. It follows that G and GF generate the same lattice. Conversely, *all* bases of a lattice arise by transformation via unimodular matrices. Thus,

$$\mathbf{L}(G) = \mathbf{L}(\bar{G}) \Longleftrightarrow \bar{G} = GF \text{ for some unimodular matrix } F.$$

In other words, the set of unimodular matrices characterize the set of bases for a given lattice.

Of the many possible bases of a lattice, one would like to select one that is in some sense nice or simple, or *reduced*. There are many different notions of reduced bases in the literature; one of them consists of requiring that all its vectors be short in the sense that the product $\|g_1\| \, \|g_2\| \cdots \|g_q\|$ is minimum. This problem is known as the *minimum basis problem* and has been proved to be NP-hard [GLS88].

Although the minimum basis problem is NP-hard, there is a *polynomial time* algorithm that, given any lattice $L = \mathbf{L}(G)$ with G having rational elements, computes a new basis for L that is in some sense *reduced*. (In practice, G always has rational elements due to finite precision computation, but it should be noted that the algorithm still terminates in a finite number of steps if G has irrational elements.) This algorithm is due to A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász (LLL) and is very efficient in practice (cf. [GLS88]). Given a lattice generator matrix G, the LLL algorithm produces a new lattice generator matrix $\bar{G} = GF$ (with F unimodular) such that $\bar{G}$ has two nice properties. Roughly speaking these are: (1) The columns of $\bar{G}$ (the basis vectors) are "almost orthogonal", and, (2) the norm of the columns of $\bar{G}$ (the length of the basis vectors) are not "arbitrarily large".

More specifically, if the columns of $\bar{G}$ (the basis vectors) are $\bar{g}_1, \bar{g}_2, \ldots, \bar{g}_q$, then

$$\bar{g}_j = \sum_{i=1}^{j} \mu_{ji} \bar{g}_i^*$$

in which $\mu_{jj} = 1$, $|\mu_{ji}| \leq \frac{1}{2}$ for $i \neq j$, and $\{\bar{g}_i^*\}_{i=1}^q$ is an orthogonal basis found by performing the Gram-Schmidt orthogonalization procedure on $\{\bar{g}_i\}_{i=1}^q$. The fact that μ_{jj} is at least twice as large as $|\mu_{ji}|$ for $i \neq j$ shows that the basis vectors $\{\bar{g}_i\}_{i=1}^q$ are "almost orthogonal". Moreover we have

$$\|\bar{g}_1\| \|\bar{g}_2\| \cdots \|\bar{g}_q\| \leq 2^{q(q-1)/4} |\det G|, \tag{14}$$

and

$$\|\bar{g}_1\| \leq 2^{(q-1)/4} \sqrt[q]{|\det G|}, \tag{15}$$

$$\|\bar{g}_1\| \leq 2^{(q-1)/2} \min_{v \in L, \; v \neq 0} \|v\| \tag{16}$$

which show that the length of the basis vectors cannot be "arbitrarily large" (note that $|\det G|$ is constant for any representation of the lattice since all bases of a lattice arise by transformation via unimodular matrices which have a determinant of ± 1). Although the bounds obtained from (14), (15) and (16) appear to be very loose for moderately large values of q, the LLL algorithm performs substantially better in practice [OtR85].

We refer the more interested reader to Chapter 5 of [GLS88] and references therein for a more complete discussion of basis reduction in lattices. In fact, there are other polynomial time algorithms for basis reduction in lattices that give better worst-case bounds than those given here but have higher worst-case complexity [Kan83, Sch85]. The LLL algorithm is described in the Appendix.

4.2 Calculating P_{c}

Computing P_{c} as given in (12) requires the knowledge of V_0, the Voronoi cell of the origin in the lattice generated by G. The Voronoi cell of a point Gz_0 is the set of all points in $\mathbf{R}^q$ which are closer to Gz_0 than any other point in the lattice. When the Euclidean norm is our distance measure, the Voronoi cells of the lattice points become polytopic, i.e., an intersection of half-spaces in $\mathbf{R}^q$. By definition

$$V_0 = \left\{x \mid \|x-0\|^2 \le \|x-Gz\|^2,\ \forall z \in \mathbf{Z}^q\right\} = \left\{x \mid (Gz)^T x \le z^T G^T G z/2,\ \forall z \in \mathbf{Z}^q\right\}. \tag{17}$$

The shape of the Voronoi cells are similar and are translated versions of V_0 (Figure 1). Also note

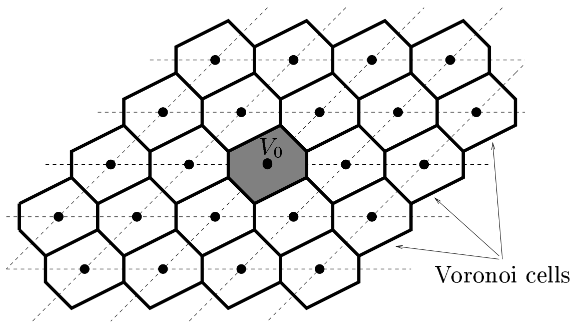

Figure 1: The Voronoi cells of the lattice points are translations of V_0.

that V_0 is *symmetric* with respect to the origin.

V_0 as defined in (17) is an intersection of infinitely many half-spaces. Such a description is not practical to perform a (numerical) integration to compute P_{c}, and therefore, we should eliminate the redundant half-spaces to obtain a finite description for V_0. Although this is theoretically possible and the use of the LLL algorithm speeds up the method (cf. [HB95]), it might not be possible in, for example, real-time applications. Adding to this the computational cost and deficiencies of the numerical algorithm to perform the integral of the Gaussian probability density function over V_0, we conclude that, although computing P_{c} is possible, it requires extensive computation that might be infeasible in practice or not worthwhile. Therefore, finding easily computable bounds on P_c is practically important.

4.3 Computing upper and lower bounds on P_c

4.3.1 Upper bound using $|\det G|$

The volume of the parallelepiped formed by the basis vectors of a non-degenerate lattice is given by $|\det G|$. Since the Voronoi cells of the lattice points partition the space with the same density (cells per unit volume) as the parallelepiped cells, it follows that the volume of the Voronoi cells are also equal to $|\det G|$. Therefore, $|\det G|$ is the volume of the region for integrating the noise probability density function, and is related to the probability of correct integer parameter estimation P_c: roughly speaking, the larger this value, the larger the probability of correct integer estimation. Of course, the shape of the Voronoi cell is also a major factor in P_c. We can however develop an upper bound on P_c that is based on $|\det G|$, the volume of a Voronoi cell. Among all regions with a given volume, the one that maximizes the probability (under a normal distribution) is a Euclidean ball (simply because the sublevel sets of a normal distribution are Euclidean balls). Therefore the probability of a Euclidean ball of volume equal to $|\det G|$ is larger than P_c.

A q-dimensional Euclidean ball of radius ρ has volume $\alpha_q \rho^q$ with $\alpha_q = \pi^{q/2}/\Gamma(q/2+1)$, where Γ is the Gamma function ($\Gamma(n) = (n-1)!$ for positive integer n). Therefore, the radius of a q-dimensional Euclidean ball of volume $|\det G|$ is $\rho = \sqrt[q]{|\det G|/\alpha_q}$ and an upper bound on P_c becomes

$$P_{\mathrm{c,up}} = \mathbf{Prob}\left(\|v\| < \sqrt[q]{|\det G|/\alpha_q}\right) \quad \text{with} \quad v \sim \mathcal{N}(0, I).$$

The sum of squares of q independent zero-mean, unit variance normally distributed random variables is a χ^2 distribution with q degrees of freedom (cf. [Koc88]). Thus, $\|v\|^2$ is χ^2 with q degrees of freedom and we get

$$P_{\mathrm{c,up}} = F_{\chi^2}\left((|\det G|/\alpha_q)^{2/q}; q\right), \tag{18}$$

where $F_{\chi^2}(\,\cdot\,; q)$ is the cumulative distribution function of a χ^2 random variable with q degrees of freedom. This bound is a very cheap one computationally since it only requires the determinant of G followed by a χ^2 cumulative distribution function table lookup. In practice, $P_{\mathrm{c,up}}$ turns out to be a very good approximation to P_c. Simulations in §7 illustrate this fact. Let us note that there are many other methods to compute upper bounds on P_c and we refer the interested reader to [HB95].

4.3.2 Lower bound based on $d_{\min}$

Given the lattice $L = \mathbf{L}(G)$ with $G \in \mathbf{R}^{q\times q}$, the *shortest lattice vector* or the *minimum distance* of the lattice $d_{\min}$ is defined by

$$d_{\min} = \min_{z \in \mathbf{Z}^q,\ z \neq 0} \|Gz\|. \tag{19}$$

$d_{\min}$ is actually the distance between the origin and its closest neighbor in the lattice. A ball of radius $d_{\min}/2$ is the largest ball centered at the origin that lies in V_0 (Figure 2).

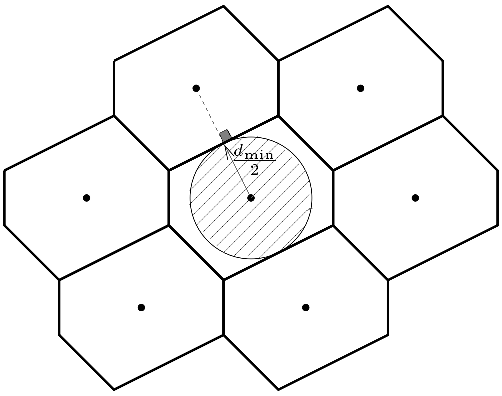

Figure 2: The ball centered at the origin and of radius $d_{\min}/2$ lies inside the Voronoi cell of the origin. The probability of noise falling in this ball gives a lower bound on $P_{\rm c}$.

Clearly, the probability of noise falling in a ball centered at the origin and of radius $d/2$ where $d \le d_{\min}$ gives us a lower bound on $P_{\rm c}$. This lower bound is given by

$$P_{\rm c,low} = \textbf{Prob}\left(\|u\| < \frac{d}{2}\right) \quad \text{where} \quad u \sim \mathcal{N}(0, I) \quad \text{and} \quad d \le d_{\min},$$

or in terms of the χ^2 cumulative distribution function

$$P_{\rm c,low} = F_{\chi^2}\left(\frac{d^2}{4}; q\right) \quad \text{where} \quad d \le d_{\min}. \tag{20}$$

Obviously, this lower bound is best when $d = d_{\min}$.

In practice, we usually only care for $P_{\rm c}$ being exact when $P_{\rm c}$ is close to unity. It can be shown that as $P_{\rm c}$ approaches unity, $P_{\rm c,low}$ with $d = d_{\min}$ becomes tight[2] and therefore we will not be too conservative in accepting $z_{\rm ML}$ when, say, $P_{\rm c,low} \cong 0.99$ [HB95].

4.3.3 Computing $d_{\min}$ and lower bounds on $d_{\min}$

So far we have not discussed how to compute $d_{\min}$. Unfortunately, computing $d_{\min}$ is conjectured to be NP-hard, and no polynomial-time algorithm has been found to compute $d_{\min}$. A generic algorithm for finding $d_{\min}$ is given in [HB95]. It is important to note that if the columns of G are a reduced basis in the sense of the LLL algorithm, the efficiency of this algorithm is much better. However, we have no guarantee that this algorithm will stop in polynomial time. Therefore, it is

[2]It can be shown that the lower bound error $(P_{\rm c} - P_{\rm c,low})$ is largest when V_0 is a slab. In this case, it is easy to verify that as $P_{\rm c} \to 1$, $(P_{\rm c} - P_{\rm c,low}) \to 0$ (cf. [HB95]).

desirable to find methods for computing *lower bounds* on $d_{\min}$ instead, that have very low computational complexity. These bounds can be used in (20) to get very fast lower bounds on P_c. One of the many methods (cf. [HB95]) for computing a lower bound on $d_{\min}$ is suggested in [GLS88] and is as follows.

Gram-Schmidt method lower bound on $d_{\min}$. Suppose $G = [g_1\, g_2 \cdots g_q]$ and $(g_1^*, g_2^*, \ldots, g_q^*)$ is the Gram-Schmidt orthogonalization of $(g_1, g_2, \ldots, g_q)$ so that $g_1^* = g_1$ and

$$g_j^* = g_j - \sum_{i=1}^{j-1} \frac{g_j^T g_i^*}{\|g_j^*\|^2} g_i^* \quad \text{for} \quad j = 2, 3, \ldots, q. \tag{21}$$

Then a lower bound d on $d_{\min}$ is

$$d = \min(\|g_1^*\|, \|g_2^*\|, \ldots, \|g_q^*\|). \tag{22}$$

If the columns of G are a reduced basis in the sense of the LLL algorithm, this lower bound is guaranteed not to be arbitrarily far from $d_{\min}$ (i.e., there is a bound in terms of q on how large $|d - d_{\min}|$ can be [GLS88]). Therefore, the LLL algorithm can be used to sharpen this lower bound, which in effect, sharpens the lower bound $P_{c,\mathrm{low}}$.

4.4 Verifying the real parameter estimate

Until now, we have mainly discussed the issue of calculating (or bounding) the probability of correct integer estimation P_c, i.e., the measure of reliability for the integer parameter estimate. Once the calculation of P_c or its bounds suggest that the integer parameters are resolved correctly, the covariance matrix $\mathbf{E}(x_{\mathrm{ML}} - x)(x_{\mathrm{ML}} - x)^T$ is known as given in §3, and one can verify the *real* parameter estimate x_{ML}. For example, since the distribution of $x_{\mathrm{ML}} - x$ is zero-mean Gaussian, $(x_{\mathrm{ML}} - x)^T \Xi^{-1} (x_{\mathrm{ML}} - x)$ has a χ^2 distribution with p degrees of freedom and therefore ellipsoidal confidence regions can be found on x_{ML} using χ^2 distribution table lookups (cf. [Koc88]).

5 Estimation Problem

5.1 Nearest lattice vector problem

The main computational task of the estimation step is the solution to an integer least-squares problem which gives the maximum likelihood estimate of the integer parameter. According to §3

$$z_{\mathrm{ML}} = \underset{z \in \mathbf{Z}^q}{\operatorname{argmin}} \; \|\tilde{y} - Gz\|^2, \tag{23}$$

or equivalently

$$z_{\mathrm{ML}} = \underset{z \in \mathbf{Z}^q}{\operatorname{argmin}} \; (z - \hat{z})^T \Upsilon^{-1} (z - \hat{z}), \tag{24}$$

where $\hat{z} = G^{-1}\tilde{y}$ and $\Upsilon = (G^TG)^{-1}$.

Equation (23) can be interpreted as finding the closest lattice point in $L = \mathbf{L}(G)$ to $\tilde{y}$. This problem is known as the *nearest lattice vector problem* and is NP-hard [GLS88, Boa81]. However, in practice, there are reasonably efficient ways to solve this problem which is the main topic of this section.

Note that if $\hat{z}$ in (24) is a vector with integer coordinates, z_{ML} can be found immediately as $z_{\mathrm{ML}} = \hat{z}$ which would give a minimum value of zero for the quadratic $(z-\hat{z})^T\Upsilon^{-1}(z-\hat{z})$. This only happens when the noise level is zero so the observed $\tilde{y}$ is one of the lattice points. When Υ is diagonal, z_{ML} can be simply found by rounding the components of $\hat{z}$ to the nearest integer since the integer variables are separable as

$$(z-\hat{z})^T\Upsilon^{-1}(z-\hat{z}) = \sum_{i=1}^{q}(z_i-\hat{z}_i)^2/\Upsilon_{ii}.$$

A diagonal Υ corresponds to a generator G with orthogonal columns. This observation suggests that if Υ is "almost" diagonal, or equivalently, the columns of G are "almost" orthogonal, rounding $\hat{z}$ in (24) would give a close approximation to z_{ML}.

Since we can replace G by GF, where F is any unimodular matrix, we can use the LLL algorithm to find an almost orthogonal basis for a given lattice $L = \mathbf{L}(G)$. Rounding can then be applied to get a hopefully good approximation for z_{ML}. This approximation can serve as a good initial guess for z_{ML} in any algorithm for solving the optimization problems (23) or (24).

5.2 Suboptimal polynomial-time algorithms

In this subsection, we describe a suboptimal polynomial-time algorithm for calculating the minimum of $\|\tilde{y} - Gz\|$ or $(z-\hat{z})^T\Upsilon^{-1}(z-\hat{z})$ over the integer lattice. Suboptimal algorithms of this kind are important for a few reasons. First, they can be performed efficiently with a guaranteed low worst-case complexity. Secondly, suboptimal algorithms provide a relatively good initial guess or relatively tight upper bound for any global optimization algorithm. Finally, these suboptimal algorithms might find the global optimum as they often do in practice. If $d_{\min}$ or any lower bound $d \le d_{\min}$ is known, a sufficient condition for the suboptimal minimizer z_{sub} to be the global minimizer z_{ML} is simply given by

$$\|\tilde{y} - Gz_{\mathrm{sub}}\| \le \frac{d}{2} \Longrightarrow z_{\mathrm{sub}} = z_{\mathrm{ML}}, \tag{25}$$

as there is only one lattice point in a ball centered at Gz_{sub} and with radius $d_{\min}/2$ (Figure 3).

Suboptimal algorithm for finding an approximately nearest lattice point based on rounding. Suppose that $G \in \mathbf{R}^{q\times q}$ and $\tilde{y} \in \mathbf{R}^q$ are given. A suboptimal solution $z_{\mathrm{sub}} \in \mathbf{Z}^q$ in the

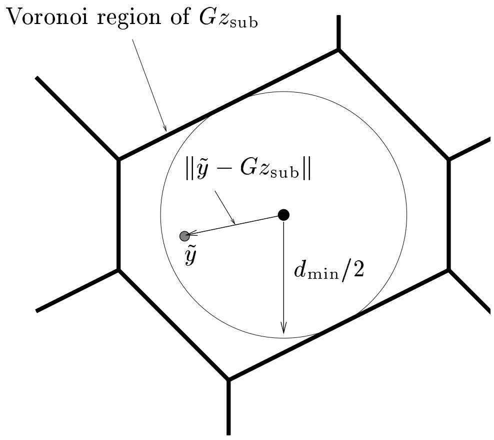

Figure 3: If $\|\tilde{y} - Gz_{\text{sub}}\| \leq d_{\min}/2$ then it is guaranteed that z_{sub} is the global minimizer z_{ML}. $d_{\min}$ or any lower bound on $d_{\min}$ is usually a byproduct of the verification step and can be used without any additional computation.

sense that

$$\|\tilde{y} - Gz_{\text{sub}}\| \leq \left(1 + 2q\,(4.5)^{q/2}\right) \min_{z \in \mathbf{Z}^q} \|\tilde{y} - Gz\|, \tag{26}$$

exists that can be found as follows [Bab86].

- Perform the LLL algorithm on the initially given basis vectors which are the columns of G. This results in a new lattice generator matrix $\bar{G}$ which is almost orthogonal and a unimodular matrix F such that $\bar{G} = GF$.
- $z_{\text{sub}} \leftarrow F\lceil \bar{G}^{-1}\tilde{y} \rfloor$ where $\lceil\cdot\rfloor$ is the component-wise rounding operation to the nearest integer.
- If $\|\tilde{y} - Gz_{\text{sub}}\| \leq d_{\min}/2$ where $d_{\min}$ is the minimum distance of L, z_{sub} is the *global minimizer* of $\|\tilde{y} - Gz\|$ for $z \in \mathbf{Z}^q$.

Clearly, the reason why component-wise rounding gives a good suboptimal solution is the fact that $\bar{G}$ is almost orthogonal because of the use of the LLL algorithm. Another heuristic to get a suboptimal solution is to do the component-wise rounding recursively, i.e., round only one of the components of $\bar{G}^{-1}\tilde{y}$ (e.g., the one closest to an integer) at a time, then fix that component in the least-squares problem and repeat.

Another suboptimal polynomial time algorithm for finding an "approximately" nearest lattice point is due to Babai (1986) [GLS88, Bab86]. In this method, z_{sub} is found by recursively computing the closest point in sub-lattices of L to $\tilde{y}$. The provable worst-case bound we get is better than (26) with the price of some additional computation. As reported in [OtR85] and from our own experience, it should be noted that these suboptimal algorithms work much better in practice than what the worst-case bounds suggest.

Using the worst-case performance bounds of these suboptimal algorithms, it is possible to find a lower bound on the probability that $z_{\mathrm{sub}} = z$ given $z_{\mathrm{ML}} = z$, i.e., the probability that the suboptimal estimate is correct given the optimal estimate is correct. Suppose that the known worst-case suboptimality factor of the suboptimal algorithm is $\gamma_q > 1$ (e.g., in (26), $\gamma_q = 1 + 2q(4.5)^{q/2}$) so that

$$\|\tilde{y} - Gz_{\mathrm{sub}}\| \leq \gamma_q \|\tilde{y} - Gz_{\mathrm{ML}}\|.$$

We have

$$\begin{aligned} \|G(z_{\mathrm{sub}} - z_{\mathrm{ML}})\| &= \|(\tilde{y} - Gz_{\mathrm{ML}}) - (\tilde{y} - Gz_{\mathrm{sub}})\| \\ &\leq \|\tilde{y} - Gz_{\mathrm{ML}}\| + \|\tilde{y} - Gz_{\mathrm{sub}}\| \\ &\leq (1+\gamma_q)\|\tilde{y} - Gz_{\mathrm{ML}}\|. \end{aligned}$$

If $z_{\mathrm{sub}} \neq z_{\mathrm{ML}}$ then $\|G(z_{\mathrm{sub}} - z_{\mathrm{ML}})\| \geq d_{\min}$ so that $\|\tilde{y} - Gz_{\mathrm{ML}}\| \geq d_{\min}/(1+\gamma_q)$. But $d_{\min}$ can be (lower) bounded by P_{c} as $d_{\min} \geq 2Q^{-1}\left((1-P_{\mathrm{c}})/2\right)$ where Q^{-1} is the inverse function of the Q function (the probability of the tail of the Gaussian PDF)[3]. Therefore if $z_{\mathrm{sub}} \neq z_{\mathrm{ML}}$ we have $\|\tilde{y} - Gz_{\mathrm{ML}}\| \geq 2Q^{-1}\left((1-P_{\mathrm{c}})/2\right)/(1+\gamma_q)$, or equivalently,

$$\|\tilde{y} - Gz_{\mathrm{ML}}\| \leq \frac{2Q^{-1}\left((1-P_{\mathrm{c}})/2\right)}{1+\gamma_q} \Longrightarrow z_{\mathrm{sub}} = z_{\mathrm{ML}}.$$

Now if $z_{\mathrm{ML}} = z$ then $\|\tilde{y} - Gz_{\mathrm{ML}}\|^2$ is χ^2 with q degrees of freedom and we finally get

$$\mathbf{Prob}\left(z_{\mathrm{sub}} = z \mid z_{\mathrm{ML}} = z\right) \geq F_{\chi^2}\left(\frac{4Q^{-1}\left((1-P_{\mathrm{c}})/2\right)^2}{(1+\gamma_q)^2}; q\right). \tag{27}$$

The interesting point about (27) is that as P_{c} gets larger, the bound on the probability gets larger, which means that the suboptimal algorithm is guaranteed to perform better. In other words, for large enough P_{c}, or when the reliability on the integer estimate is high, these suboptimal algorithms have a guarantee on their performance. Again, we must note that, in practice, the performance is much better than what the worst-case bound suggests.

5.3 Searching for integer points inside an ellipsoid

Once a suboptimal solution to the integer least-squares problem has been found, we need to check whether any better solution exists or not. This problem is equivalent to checking whether an ellipsoid contains any points with integer coordinates. We consider this problem in this subsection.

Suppose that the ellipsoid $\mathcal{E}$ is given by either of the two equivalent descriptions

$$\mathcal{E} = \{x \mid x \in \mathbf{R}^q,\ \|\tilde{y} - Gx\| < r\} \quad \text{or} \quad \mathcal{E} = \left\{x \mid x \in \mathbf{R}^q,\ (x-\hat{z})^T \Upsilon^{-1}(x-\hat{z}) < r^2\right\}, \tag{28}$$

[3] The reason is simply that a slab of width $d_{\min}$ covers the Voronoi cell, see [HB95].

where $G \in \mathbf{R}^{q \times q}$, $\Upsilon = (G^T G)^{-1}$ and $\hat{z} = G^{-1}\tilde{y}$. Our goal is to find at least one integer point in $\mathcal{E}$ or prove that no such point exists. In other words, we would like to find, if any, a $z \in \mathbf{Z}^q$ such that $\|\tilde{y} - Gz\| < r$ or equivalently $(z - \hat{z})^T \Upsilon^{-1} (z - \hat{z}) < r^2$.

One can easily bound the integer points $z = [z_1 \ z_2 \cdots \ z_q]^T$ satisfying $(z - \hat{z})^T \Upsilon^{-1} (z - \hat{z}) < r^2$ by bounding z in each dimension, i.e., to bound z_i for $i = 1, \ldots, q$. This is equivalent to finding the *minimum volume box* covering an ellipsoid (Figure 4). The result is

$$\lceil \hat{z}_i - r\sqrt{\Upsilon_{ii}} \rceil \leq z_i \leq \lfloor \hat{z}_i + r\sqrt{\Upsilon_{ii}} \rfloor, \tag{29}$$

for $i = 1, 2, \ldots, q$ where $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ are the rounding up and rounding below operations respectively. Clearly, (29) gives an *explicit* outer approximation for integer points that lie inside the ellipsoid. However, this outer approximation can be very bad (i.e., contain many more integer points than the ellipsoid) if the minimum volume box approximation is not good. Therefore, in such cases, checking whether the ellipsoid contains an integer point by searching over the integer point candidates (29) is very inefficient.

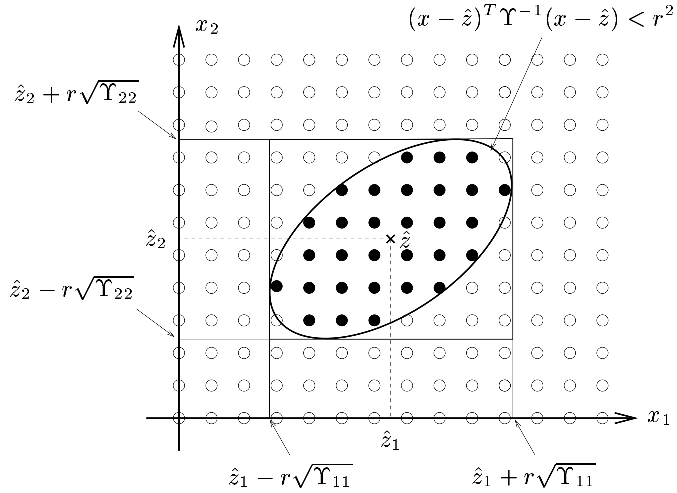

Figure 4: The minimum volume box covering the ellipsoid $(x - \hat{z})^T \Upsilon^{-1} (x - \hat{z}) < r^2$ is related to the diagonal entries of Υ and r.

The minimum volume box approximation is best when the axes of the ellipsoid are parallel to the coordinate axes, or equivalently, Υ is diagonal. Therefore, if we perform the LLL algorithm on the originally given basis vectors, so that we are guaranteed to get an "almost" orthogonal G or diagonal Υ, the minimum volume box covering the ellipsoid is hopefully a good approximation to the ellipsoid, and contains fewer integer points satisfying (29). Now by checking whether any of these integer points satisfy the inequality $(z - \hat{z})^T \Upsilon^{-1} (z - \hat{z}) < r^2$, we can determine whether there is an integer point inside the ellipsoid or not.

Even if Υ is diagonal, the volume of the minimum volume box is larger than the volume of the ellipsoid by a factor of $\alpha_q = \pi^{q/2} / \Gamma(q/2+1)$. This factor is still very large for even moderate values of q, and therefore, the set of integer vectors z satisfying (29) could be much larger than the set of integer points inside the ellipsoid. Hence, the method just explained for finding an integer point inside the ellipsoid could still be inefficient in practice.

A more efficient method for searching for integer points inside an ellipsoid is to bound z_i *recursively* for $i = 1, 2, \ldots, q$. This can be easily done when G is lower triangular as follows. Suppose that

$$G = \begin{bmatrix} g_{11} & 0 & \cdots & 0 \\ g_{21} & g_{22} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ g_{q1} & g_{q2} & \cdots & g_{qq} \end{bmatrix}$$

is lower triangular. Then $\|\tilde{y} - Gz\| < r^2$ gives

$$(\tilde{y}_1 - g_{11}z_1)^2 + (\tilde{y}_2 - g_{21}z_1 - g_{22}z_2)^2 + \ldots + (\tilde{y}_q - g_{q1}z_1 - g_{q2}z_2 - \ldots - g_{qq}z_q)^2 < r^2,$$

and therefore

$$\lceil \tfrac{\tilde{y}_1 \mp r}{g_{11}} \rceil \leq z_1 \leq \lfloor \tfrac{\tilde{y}_1 \pm r}{g_{11}} \rfloor,$$

$$\lceil \tfrac{\tilde{y}_2 - g_{21}z_1 \mp \sqrt{r^2 - (\tilde{y}_1 - g_{11}z_1)^2}}{g_{22}} \rceil \leq z_2 \leq \lfloor \tfrac{\tilde{y}_2 - g_{21}z_1 \pm \sqrt{r^2 - (\tilde{y}_1 - g_{11}z_1)^2}}{g_{22}} \rfloor,$$

$$\lceil \tfrac{\tilde{y}_3 - g_{31}z_1 - g_{32}z_2 \mp \sqrt{r^2 \mp (\tilde{y}_1 - g_{11}z_1)^2 - (\tilde{y}_2 - g_{21}z_1 - g_{22}z_2)^2}}{g_{33}} \rceil \leq z_3 \leq \lfloor \tfrac{\tilde{y}_3 - g_{31}z_1 - g_{32}z_2 \pm \sqrt{r^2 - (\tilde{y}_1 - g_{11}z_1)^2 - (\tilde{y}_2 - g_{21}z_1 - g_{22}z_2)^2}}{g_{33}} \rfloor,$$

and so on. Thus z_i, the ith element of z, satisfies the inequality

$$l_i \leq z_i \leq u_i \tag{30}$$

with

$$l_i = \lceil \frac{t_i - \text{sgn}(g_{ii}) r_i}{g_{ii}} \rceil \quad \& \quad u_i = \lfloor \frac{t_i + \text{sgn}(g_{ii}) r_i}{g_{ii}} \rfloor \tag{31}$$

in which

$$r_1 = r, \quad r_i = \sqrt{r_{i-1}^2 - \left(\tilde{y}_{i-1} - \sum_{k=1}^{i-1} g_{(i-1)k} z_k\right)^2} \quad \text{for} \quad i = 1, 2, \ldots, q-1 \tag{32}$$

and

$$t_1 = \tilde{y}_1, \quad t_i = \tilde{y}_i - \sum_{k=1}^{i-1} g_{ik} z_k \quad \text{for} \quad i = 2, 3, \ldots, q. \tag{33}$$

Note that the upper bound u_i might become less than the lower bound l_i as a result of lower and upper rounding. In this case, there is no integer point in the ellipsoid with the values chosen for the coordinates $z_1, z_2, \ldots, z_{i-1}$, and therefore, we need to choose other values for $z_1, z_2, \ldots, z_{i-1}$ and continue. Whenever for all possible values of z_1 satisfying $l_1 \leq z_1 \leq u_1$, we get an upper bound that is less than the lower bound for one of the coordinates, it can be concluded that no point with integer coordinates exists inside the ellipsoid. This suggests the following search method.

Searching for integer points inside an ellipsoid. Suppose $G \in \mathbf{R}^{q\times q}$, $\tilde{y} \in \mathbf{R}^q$, and $r \in \mathbf{R}$ are given. In order to find a point $\|\tilde{y} - Gx\| < r$ for $x \in \mathbf{R}^q$ or prove that no such point exists:

- Find unitary transformation Θ such that ΘG is lower triangular.
- $\tilde{y} \leftarrow \Theta\tilde{y}$; $G \leftarrow \Theta G$; $i \leftarrow 1$;
- **while** 1
  - Compute l_i and u_i from equations (31)-(33); $S_i \leftarrow \{l_i, l_i+1, \ldots, u_i\}$;
  - **while** $S_i = \emptyset$
    - **if** $i = 1$; no integer point exists in ellipsoid; **stop.**
    - $i \leftarrow i - 1$;
  - **end**
  - Pick z_i as the element in S_i which is closest to the average of the largest and smallest elements in S_i; $S_i \leftarrow S_i - \{z_i\}$;
  - **if** $i = q$ **then** $z^* = [z_1\ z_2 \cdots\ z_q]^T$ is an integer point inside the ellipsoid; **stop.**
  - $i \leftarrow i + 1$;
- **end**

Note that in the above search method S_i gives the set of possible values for z_i at each iteration. z_i is taken as the element in S_i that is closest to the average of its largest and smallest elements. For example, if $S_i = \{1, 2, 3\}$ then $(1+3)/2 = 2$ and $z_i = 2$ is chosen. For $S_i = \{5, 7, 10, 11\}$, the average of the largest and smallest elements is $(5+11)/2 = 8$ and therefore we should pick $z_i = 7$. The reason for choosing z_i like this is that, intuitively, the points closer to the midpoint of the interval of possible values for z_i correspond to the "fatter" region of the ellipsoid that is more likely to contain an integer point. In addition, these points correspond to the center of the ellipsoid, and will hopefully result in lower values of $\|\tilde{y} - Gz\|$. This is specially important in the algorithm for finding the global minimum of $\|\tilde{y} - Gz\|$.

5.4 Global optimization algorithm

Now we have all the tools to describe a global optimization method to solve the *integer least-squares* or *nearest lattice vector* problem to find z_{ML} in (23) or (24).

Global optimization algorithm for finding z_{ML}. Suppose $G \in \mathbf{R}^{q\times q}$ and $\tilde{y} \in \mathbf{R}^q$ are given. Furthermore, let d be the minimum distance or a lower bound on the minimum distance in the lattice $L = \mathbf{L}(G)$. z_{ML}, the minimizer of $\|\tilde{y} - Gz\|$ for $z \in \mathbf{Z}^q$, can be obtained as follows:

- Compute a suboptimal solution z_{sub} to the integer least-squares problem $\mathrm{argmin}_z \|\tilde{y} - Gz\|$ using the method of §5.2.
- **if** $\|\tilde{y} - Gz_{\mathrm{sub}}\| \le d/2$ **then** $z_{\mathrm{ML}} \leftarrow z_{\mathrm{sub}}$**; stop.**
- $r \leftarrow \|\tilde{y} - Gz_{\mathrm{sub}}\|$; $z^* \leftarrow z_{\mathrm{sub}}$;
- **while** 1
  - Search for an integer point inside the ellipsoid $\|\tilde{y} - Gz\| < r$ using the method in §5.3; **if** no such point exists **then** $z_{\mathrm{ML}} \leftarrow z^*$**; stop.**
  - Let z^* be the integer point found in the previous step; $r \leftarrow \|\tilde{y} - Gz^*\|$;
  - **if** $r \le d/2$ **then** $z_{\mathrm{ML}} \leftarrow z^*$**; stop.**
- **end**

Note that in the first step we perform a suboptimal polynomial time algorithm to find a good initial integer parameter estimate z_{sub}. If we are lucky, $z_{\mathrm{sub}} = z_{\mathrm{ML}}$, but even if this is not true, we will hopefully get a small initial r and therefore a small search space. The algorithm stops at the second step whenever $r \le d/2$ as it is guaranteed that no integer point lies inside the ellipsoid. The algorithm still works without any knowledge of d as we can simply take $d = 0$ in such cases. However, a d that is reasonably close to $d_{\min}$ may reduce the number of outer iterations by one.

In practice, simulations show that this global optimization method is very efficient for problems with a few ten integer variables, although the worst-case complexity is not polynomial. Such problems can be solved almost instantly on a typical personal computer[4].

5.5 Computing the real parameter estimate

Once z_{ML} is found by the global optimization algorithm in §5.4, the maximum likelihood estimate of x, the real parameter, is straightforward. x_{ML} is simply given as in §3 using equations (7) and (8).

6 Extensions

In this section we consider extensions to model (1). We first consider the more general case in which *a priori* knowledge on the statistics of x is assumed, and then a special case in which x has

[4]Contact the authors for an implementation of this global optimization algorithm in Matlab.

state-space structure. Finally, we show how to modify the estimation algorithms to deal with the case in which component-wise constraints are added on z.

6.1 Maximum a priori (MAP) estimates

When there is *a priori* knowledge on the distribution of the parameters, an alternative to maximum likelihood estimation is maximum a posteriori (MAP) estimation. In this case the estimates are found as to maximize the probability of x and z having observed y. In other words,

$$(x_{\text{MAP}}, z_{\text{MAP}}) = \underset{(x,z) \in \mathbf{R}^p \times \mathbf{Z}^q}{\operatorname{argmax}} \; p_{X,Z|Y}(x, z|y). \tag{34}$$

If we assume that the noise v is $\mathcal{N}(0, \Sigma)$ and x is known *a priori* to be $\mathcal{N}(0, \Pi_0)$, MAP estimates for x and z can be found similarly by solving a least-squares problem. Equation (6) still holds but with a different definition for Ξ, Υ, $\hat{x}_{|z}$ and $\hat{z}$. It can be shown that these definitions should be

$$\begin{aligned} \Xi &= (\Pi_0^{-1} + A^T \Sigma^{-1} A)^{-1}, & \Upsilon &= \left(B^T (\Sigma + A \Pi_0 A^T)^{-1} B\right)^{-1}, \\ \hat{x}_{|z} &= \Xi A^T \Sigma^{-1} (y - Bz), & \hat{z} &= \Upsilon B^T (\Sigma + A \Pi_0 A^T)^{-1} y. \end{aligned} \tag{35}$$

(Obviously, equations (7) can be recovered from (35) by setting $\Sigma = I$ and $\Pi_0^{-1} = 0$.) The inverses in (35) always exist when A, B are full column rank and Σ, Π_0 are nonsingular. The equations for $\tilde{y}$, G, P_{c}, etc., are as before but using these new values for Ξ, Υ, $\hat{x}_{|z}$ and $\hat{z}$. z_{MAP} and x_{MAP} are given by similar formulas to (8) with the ML subscript replaced by MAP.

6.2 State-space structure

Roughly speaking, in the case of N observations ($y \in \mathbf{R}^N$), calculating $\tilde{y} = \Upsilon^{-1/2} \hat{z}$ and $G = \Upsilon^{-1/2}$ using (7) or (35) requires $O(N^3)$ operations for computing the various matrix inversions involved that is a substantial amount of work for large N. Usually, $N \gg q$ and therefore calculating $\tilde{y}$ and G can be even more computationally intensive than solving the integer least-squares problem. Moreover, in many applications, the data comes in sequentially and it is desirable to find $\tilde{y}$ and G for sequentially increasing values of N without requiring any data storage. When x has finite-dimensional (state-space) structure it is possible to compute $\tilde{y}$ and G recursively, and with a (significantly) smaller number of floating point operations.

Assume a finite-dimensional (state-space) structure for x_k ($k = 0, 1, 2, \ldots, N$) as

$$x_{k+1} = F_k x_k + G_k u_k, \tag{36}$$

and suppose that the observation y_k depends on $x_k \in \mathbf{R}^p$ and $z \in \mathbf{Z}^q$ through the linear relationship

$$y_k = H_k x_k + J_k z + v_k \tag{37}$$

where[5] F_k, G_k, H_k, J_k are known matrices, and $\{x(0), u_k, v_k\}$ are zero-mean (jointly) Gaussian random variables such that

$$\mathbf{E}x(0)x(0)^T = \Pi, \quad \mathbf{E}u_k x(0)^T = 0, \quad \mathbf{E}v_k x(0)^T = 0, \quad \mathbf{E}\begin{bmatrix} u_k \\ v_l \end{bmatrix}\begin{bmatrix} u_k \\ v_l \end{bmatrix}^T = \begin{bmatrix} Q_k & S_k \\ S_k^T & R_k \end{bmatrix}\delta_{kl}.$$

The goal is to compute the MAP estimate of z (and x_k) given $y_0, \ldots, y_N$. Using the connection between the standard least-squares problem and the Kalman filter, it is possible to derive formulas for updating $\tilde{y}$ and G recursively, and therefore, to compute z_{MAP} from equation (11).

More specifically, it can be shown that (cf. [HB95]) the MAP estimate of $z \in \mathbf{Z}^q$ given $y_0, \ldots, y_k$ is given by

$$z_{\mathrm{MAP}}^{(k)} = \underset{z \in \mathbf{Z}^q}{\operatorname{argmin}} \|\tilde{y}^{(k)} - G^{(k)} z\|$$

where

$$G^{(k)} = (\Theta^{(k)})^{1/2}, \quad \tilde{y}^{(k)} = G^{(k)}\hat{z}^{(k)},$$

and Θ_k and $\hat{z}^{(k)}$ can be recursively computed through

$$\begin{aligned} \Theta^{(k)} &= \Theta^{(k-1)} + \Omega_k^T R_{e,k}^{-1} \Omega_k, \ \Theta^{(-1)} = 0, \\ \hat{z}^{(k)} &= (\Theta^{(k)})^{-1}\left(\Theta^{(k-1)}\hat{z}^{(k-1)} + \Omega_k^T R_{e,k}^{-1} e_k\right), \ \hat{z}^{(-1)} = 0, \end{aligned}$$

in which Ω_k and e_k are found by the *inverse* system with inputs J_k and y_k respectively, i.e.,

$$\check{x}_{k+1} = (F_k - K_{p,k}H_k)\check{x}_k + K_{p,k}J_k, \quad \Omega_k = -H_k\check{x}_k + J_k$$

and

$$\hat{x}_{k+1} = (F_k - K_{p,k}H_k)\hat{x}_k + K_{p,k}y_k, \quad e_k = -H_k\hat{x}_k + y_k$$

with $\check{x}_0 = 0$, $\hat{x}_0 = 0$ and

$$\begin{aligned} R_{e,k} &= R_k + H_k P_k H_k^T, & K_{p,k} &= (F_k P_k H_k^T + G_k S_k) R_{e,k}^{-1}, \\ F_{p,k} &= F_k - K_{p,k} H_k, & P_{k+1} &= F_k P_k F_k^T + G_k Q_k G_k^T - K_{p,k} R_{e,k} K_{p,k}^T, \ P_0 = \Pi \end{aligned} \tag{38}$$

for $k = 0, 1, 2, \ldots, N$. Moreover, the probability of correctly estimating z ($z_{\mathrm{ML}} = z$) at time k, is given by the integral of a q-dimensional normal random variable over the Voronoi cell of the origin in the lattice $\{G^{(k)}z \mid z \in \mathbf{Z}^q\}$.

Once $z_{\mathrm{ML}}^{(N)}$ is calculated, standard Kalman filter-based methods (e.g., the *two-pass smoothing* algorithm in [AM79]) can be used to estimate x_k given the observations $(y_k - J_k z_{\mathrm{ML}}^{(N)})$ for $k = 0, 1, \ldots, N$.

[5] For example, (36) might describe the dynamics of a structure and (37) be the linearized equations for the pseudo-ranges obtained from GPS sensors mounted on the structure. As another example, these equations might describe the GPS pseudo-ranges obtained in a surveying scenario [HB95].

As a final note, we should add that when x is constant (no dynamics involved), there exists an obvious state-space formulation for x, i.e.,

$$x_{k+1} = I x_k, \quad y_k = H_k x_k + J_k z + v_k.$$

Thus, $F_k = I,\ G_k = 0,\ Q_k = 0,\ S_k = 0\ \ \forall k$, and we can simply use the recursive formulas derived previously.

6.3 Component-wise constraints on the integer parameter

In previous sections, the integer parameter z in (1) or (37) was assumed to be *unconstrained*. However, in many practical cases (e.g., communications) z is bound to lie in a given *constellation* or subset of $\mathbf{Z}^q$, and therefore, the estimation algorithms of §5 need to be modified. Fortunately, this is very easy to do when, for example, there are component-wise constraints on z.

Specifically, suppose that $z_i \in T_i$ for $i = 1, \ldots, q$ in which $T_i \subseteq \mathbf{Z}$ is given. The ML (or MAP) estimate of z is given by an integer least-squares problem similar to (11) but with the additional constraints $z_i \in T_i$ where the matrices $\tilde{y}$ and G are defined as before. To deal with these additional constraints, in order to compute z_{ML}, the suboptimal algorithm of §5.2 can be modified by mapping the ith component of $F\lceil \bar{G}^{-1}\tilde{y}\rfloor$ to its closest element in T_i, and the algorithm of §5.3 can be modified by replacing the assignment $S_i \leftarrow \{l_i, l_i+1, \ldots, u_i\}$ by $S_i \leftarrow \{l_i, l_i+1, \ldots, u_i\} \cap T_i$. Once z_{ML} is found using the global optimization algorithm of §5.4 with the mentioned modifications, x_{ML} can be computed using the exact same formulas.

7 Simulations

7.1 Navigation in one dimension using sinusoidal beacons

In this section we illustrate an example that is the basis of most EDM (Electromagnetic Distance Measurement) equipment used by surveyors and geodesists [Bur74]. For simplicity we consider a one-dimensional case. The goal is to estimate an unknown position $x_0 \in \mathbf{R}$ using sinusoidal signals sent by two beacons (transmitters) located at known positions s_1 and s_2 (Figure 5).

Specifically, the (synchronized) receiver is located at x_0 and measures the phase of the signals emitted from s_1 and s_2. Due to the periodic nature of the signals, the receiver can only measure these phases modulo some integer multiple of the wavelengths. Let y_1 and y_2 be the measured phases. By simple geometry (assuming no measurement noise)

$$x_0 - \lambda_1 z_1 = y_1 + s_1, \quad x_0 - \lambda_2 z_2 = y_2 + s_2$$

or

$$\begin{bmatrix} y_1 + s_1 \\ y_2 + s_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} x_0 - \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}. \tag{39}$$

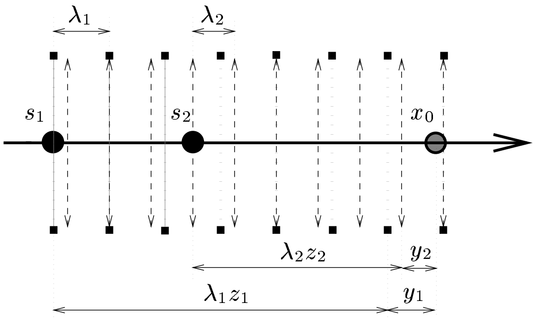

Figure 5: Beacons at points s_1 and s_2 transmit sinusoidal signals of wavelength λ_1 and λ_2 respectively. The (synchronized) receiver at x_0 locates itself by measuring the phase of the signals emitted from the beacons. Estimating x_0 can be cast as estimating integer variables in a linear model.

As can be clearly seen, this gives us two linear equations in three unknowns $x_0 \in \mathbf{R}$, $z_1 \in \mathbf{Z}$ and $z_2 \in \mathbf{Z}$.

By measuring y_1, it becomes clear that x_0 is located at distance y_1 plus an integer multiple of λ_1 from s_1. Therefore, the possible values for x_0 are points that are at distance λ_1 apart from each other. Similarly, by measuring y_2, another set of possible values for x_0 is achieved that consists of points λ_2 apart from each other. Hence, by combining both measurements y_1 and y_2, the possible values for x_0 become points that are at distance λ apart where $\lambda = m_1\lambda_1 = m_2\lambda_2$ with $m_1, m_2 \geq 1$ and $(m_1, m_2) = 1$ (i.e., m_1 and m_2 are relatively prime). Therefore, having two transmitters is equivalent to having a single transmitter with *larger* (or equal) wavelength. If λ is large and we have a rough idea of where x_0 is located (say with an uncertainty less than λ), x_0 can be found with high reliability. If there are more than just two transmitters with different wavelengths, the equivalent wavelength λ is increased, and therefore, the uncertainty in x_0 is reduced. For example, with three transmitters of wavelengths λ_1, λ_2 and λ_3, the equivalent λ becomes $\lambda = m_1\lambda_1 = m_2\lambda_2 = m_3\lambda_3$ where $(m_1, m_2, m_3) = 1$.

If λ_1/λ_2 is irrational, there are no integer numbers m_1 and m_2 such that $m_1\lambda_1 = m_2\lambda_2$. In this case, it can be simply verified that the choice of x_0 satisfying (39) is unique, and hence x_0 can be determined *exactly.*

Now suppose that there *is* measurement noise, so that

$$\begin{bmatrix} y_1 + s_1 \\ y_2 + s_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} x_0 - \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

where $v_1, v_2 \sim \mathcal{N}(0, \sigma^2)$ and, v_1 and v_2 are uncorrelated. Also, we assume that $x_0 \sim \mathcal{N}(0, \sigma_x^2)$. Using formulas (35) we have

$$G = \frac{1}{\sqrt{2}} \begin{bmatrix} \frac{\lambda_1}{\sigma} & -\frac{\lambda_2}{\sigma} \\ \frac{\lambda_1}{\sqrt{\sigma^2+2\sigma_x^2}} & \frac{\lambda_2}{\sqrt{\sigma^2+2\sigma_x^2}} \end{bmatrix}.$$

Assuming the values $\lambda_1 = 0.5$, $\lambda_2 = 4$, $\sigma = 0.01$, $\sigma_x = 10$ we get $P_c = 0.158$. Now assuming that $\lambda_1 = \sqrt[3]{2}$, $\lambda_2 = \sqrt[3]{4}$, $\sigma = 0.01$, $\sigma_x = 10$ we get $P_c = 0.886$. (The values for P_c were computed using Monte Carlo with 10^5 random variates.) Note that although $\lambda_1\lambda_2 = 2$ is equal in both cases (and therefore so are $\det G$ and $P_{c,\text{up}}$), P_c is much higher in the second case. The reason is that in the first case $\lambda_1/\lambda_2 = 1/8$ is rational while in the second case $\lambda_1/\lambda_2 = 1/\sqrt[3]{2}$ is irrational. This is intuitively reasonable by our discussion for the noiseless case. Without measurement noise, x_0 can be estimated exactly if λ_1/λ_2 is irrational. With measurement noise, x_0 can be estimated "better" (or P_c is relatively higher) when λ_1/λ_2 is irrational.

This discussion suggests that P_c is a very complicated function of λ_1/λ_2 (for constant $\lambda_1\lambda_2$). Roughly speaking, a rational λ_1/λ_2 corresponds to a relatively smaller P_c, while an irrational λ_1/λ_2 corresponds to a relatively larger P_c. Figure 6 shows a plot of P_c as a function of λ_1/λ_2 for 200 values, with $\lambda_1\lambda_2 = 2$, $\sigma = 0.01$ and $\sigma_x = 10$. Even this sampling shows the erratic behavior suggested by the discussion above. ($P_{c,\text{up}} = 0.8947$ is constant for $\lambda_1\lambda_2 = 2$).

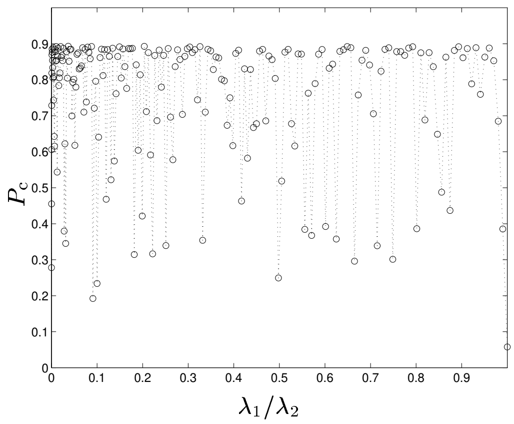

Figure 6: P_c as a function of λ_1/λ_2 for $\lambda_1\lambda_2 = 2$, $\sigma = 0.01$ and $\sigma_x = 10$. Computed points are shown with a 'o'.

Note that these ideas hold in higher dimensions, and in general, P_c is a very complicated function of the wave vectors of the signals emitted from the transmitters. As will become clear in the next subsection, the transmitters (or receiver) need not to be stationary to get a linear model involving integer parameters. In fact, moving transmitters are better as far as estimation is concerned, because the change of wave vectors of such transmitters has the same effect as an increased number of *stationary* transmitters. In other words, we get better estimates when the relative receiver-transmitter geometry changes with time. In GPS using carrier phase measurements this geometry change is crucial, and is caused by *orbiting* satellites transmitting RF signals (see next simulation).

7.2 GPS setup

In general, navigation using phase measurements from sinusoidal signals transmitted from far away locations in space with known coordinates can be cast as an integer parameter estimation problem in a linear model. The global positioning system (GPS) is one such case, in which the sinusoidal signals are sent by satellites orbiting the earth. The phase measurement noise in GPS, however, is not completely Gaussian, and depends on non-random effects such as receiver/satellite clock bias, residual atmospheric errors, multipath, etc. In practice, measurements from different satellites and receivers are combined (e.g., subtracted) to partially remove these non-random effects. Refer to [HWLC97, Hur93, Lei95, PS96] for a thorough discussion of such techniques.

The simulations in this section are based on a synthetic two-dimensional setup similar to GPS (see Figure 7). We assume that the position of the (GPS) receiver x, that is to be determined, can be modeled as a zero-mean Gaussian random variable with variance σ_x^2 in each dimension. The coordinate axes are chosen as in Figure 7 such that the origin is a point on the surface of the earth (a point on the periphery of a circle of radius equal to that of the earth $R_{\mathrm{e}} = 6357\mathrm{km}$). We suppose that there are three visible satellites orbiting the earth with an altitude of 20200km and with a period of 12 hours (angular velocity of $\omega = 1/120°\mathrm{sec}^{-1}$). The satellites are transmitting a carrier signal of wavelength $\lambda = 19\mathrm{cm}$ each, and their coordinates are known to the receiver. The receiver, which is assumed to be completely synchronized with the satellites (meaning that it can generate the transmitted carrier signals), measures the phase of the *received* carrier signals every $T = 2$ seconds and unwraps them as times goes by. By multiplying these (unwrapped) phase measurements by the wavelength λ divided by 2π, the receiver can measure its distance (or range) to each satellite up to some additive noise which is assumed to be $\mathcal{N}(0, \sigma^2)$, and, of course, an *integer* multiple of the wavelength. (This integer multiple can be thought as the number of carrier signal cycles between the receiver and the satellite when the carrier signal is initially phase locked.) By linearizing the range equations, the problem becomes one of estimating a real parameter (the coordinates of the receiver x) and an integer parameter (the integer multiples of the wavelengths) in a linear model as in (1).

In the simulation that follows, the actual location of the receiver is $x^T = [-50\ 100]$ that will be estimated using the carrier phase measurements. We assume that the standard deviation of x is $\sigma_x = 100\mathrm{m}$ along each coordinate axis. The satellites make angles of 90°, 120° and 45° with the x_1 axis initially and the direction of rotation for all of them is clockwise. The variance of phase measurement noise in units of length is taken as $\sigma = 0.01\mathrm{m}$. Using the carrier phase measurements taken over a period of 200 seconds, the receiver tries to find its position $x \in \mathbf{R}^2$ (as well as the ambiguous integer multiples of the wavelengths $z \in \mathbf{Z}^3$) as a function of time by solving for the MAP estimates using the method described in this paper. The final angles of the satellites (after 200 seconds) with respect to the x_1 axis become 88.3°, 118.3° and 43.3°.

Figure 8 shows the *exact* P_{c} (computed by Monte Carlo using 1500 random variates), and $P_{\mathrm{c,up}}$,

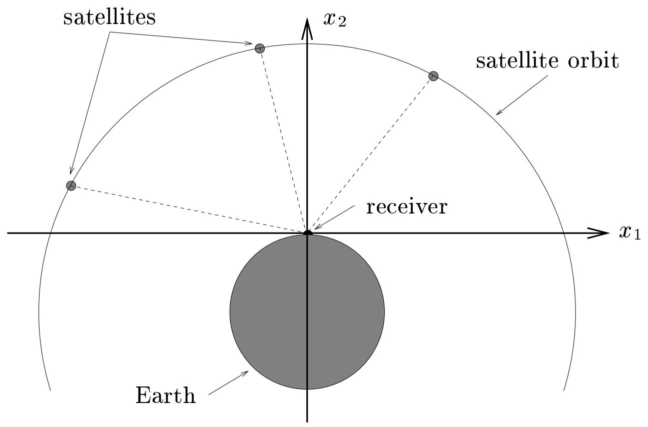

Figure 7: Simulations in this section are based on a simplified GPS setup in two dimensions. The radius of the the shaded circle (earth) is 6357km. Three satellites are assumed that orbit the earth at an altitude of 20200km and period of 12 hours.

the upper bound using $|\det G|$, as a function of time $t \leq 200$. As time t is increased, P_{c} and $P_{\mathrm{c,up}}$ increase and approach unity. Intuitively, this makes sense since the geometry of the problem is changing as a function of time (because of the rotation of the satellites), and therefore, we get more and more information to estimate z. As can be seen, $P_{\mathrm{c,up}}$ is a very sharp bound on P_{c}. Note that computing $P_{\mathrm{c,up}}$ is very easy and only requires computing $\det G$ (for which analytical closed form expressions are also available for a variety of GPS models [Teu97b]) followed by a χ^2 table lookup.

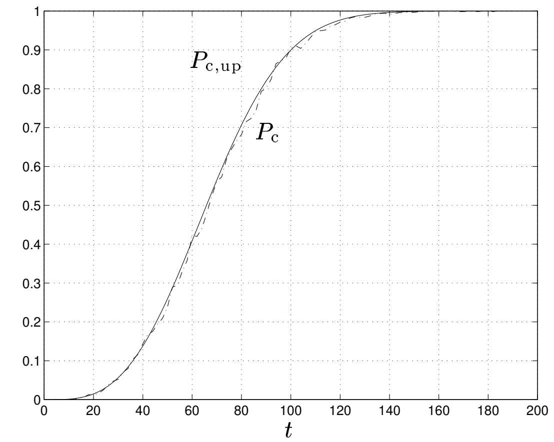

Figure 8: Exact P_{c} (*dash-dotted curve*) and $P_{\mathrm{c,up}}$ (upper bound using $|\det G|$; *solid curve*) as a function of time. $P_{\mathrm{c,up}}$ is a very good approximation to P_{c}. Note that since the Monte Carlo method for finding P_{c} is inexact, this error $P_{\mathrm{c,up}} - P_{\mathrm{c}}$ occasionally becomes negative. The error is relatively small, specially for low and high P_{c}s. The maximum error is roughly 0.03.

In practice, it might not be computationally feasible to compute P_{c} (e.g., in real-time applications), and an easily computable lower bound on P_c is required to complement the information obtained from the upper bound $P_\text{c,up}$. The lower bound $P_\text{c,low}$ in (20) can be used for this purpose in which d is the lower bound on d_min using the Gram-Schmidt method. Figure 9 shows $P_\text{c,up}$ and $P_\text{c,low}$ vs. time t. Note that although P_c is a monotonic (and continuous) function of t, this is not necessarily true for its bounds. For example at $t \approx 70$ there is a jump in the value of $P_\text{c,low}$. As t is increased, both $P_\text{c,up}$ and $P_\text{c,low}$ approach unity, while their gap approaches zero (cf. §4.3.2). This is a very good feature as we are not too conservative in bounding P_c when there is high reliability in our estimates (which could be the only case when we are really interested in our estimates). If we assume a 99% reliability, we can say that our estimate on z is unreliable for $t < t_1$ and is reliable for $t > t_2$.

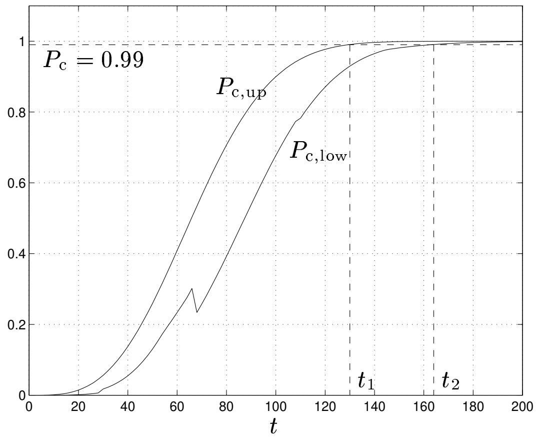

Figure 9: $P_\text{c,up}$ and $P_\text{c,low}$ as a function of time. The gap becomes smaller as P_c increases.

The next two plots (Figures 10 and 11) show the results of the estimation process. Figure 10 shows the times at which the integer parameters are estimated correctly, as well as the norm of the error in the estimated real parameter $\|x - x_\text{ML}\|$ as a function of time. This figure shows that when P_c becomes relatively high, the integer parameters are resolved as we would expect. Also, once the integer parameters are resolved, the error in the *real* parameter estimate becomes insignificant. This justifies the use of the probability of correct integer estimation P_c as a measure of reliability for the real parameter as well.

Figure 11 compares the times at which z is resolved correctly using global optimization and when using simple rounding (i.e., $z_\text{est} = \lceil G^{-1}\tilde{y} \rfloor$, without using the LLL algorithm). It takes a much longer time in the simple rounding method for the integer parameter estimates to settle (150 as compared to 85 seconds.)

We have plot the number of inner iterations in the global optimization algorithm vs. time in Figure 12. A number of iterations equal to *zero* means that the initial value for z found in the algorithm (by performing the LLL algorithm and rounding off) is guaranteed to be the global minimizer since the error norm is less than $d/2$ (condition (25) where d is the lower bound on

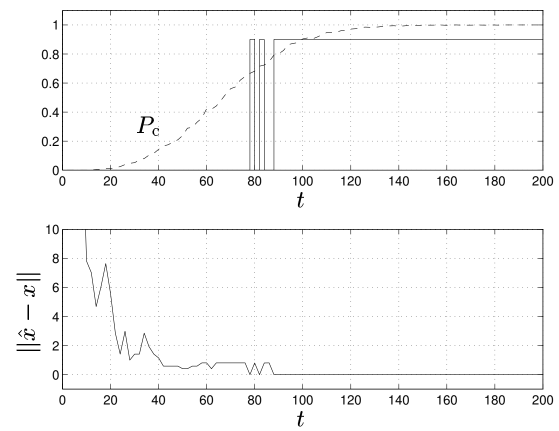

Figure 10: (*Top:*) P_c as a function of time (*dash-dotted curve*) and times at which the integer parameters are resolved correctly using global optimization (*solid curve*, a *high* means that the integer parameters are resolved correctly and a *low* means that they are not). The integer parameters remain resolved for $P_c > 0.8$ or after 85 seconds. (*Bottom:*) $\|x - x_{\mathrm{ML}}\|$ as a function of time. Note that at times that the integer parameters are resolved correctly this error almost drops to zero.

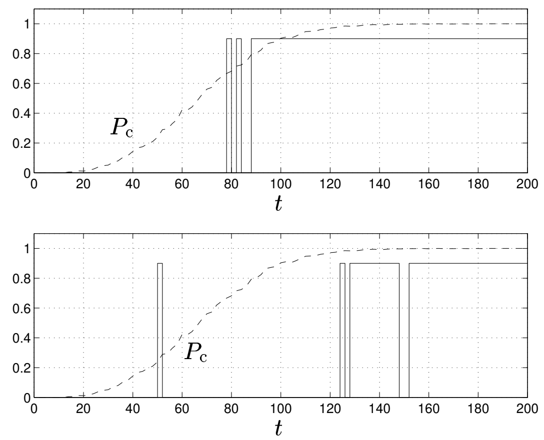

Figure 11: (*Top:*) Same as Figure 10; repeated here for convenience: The times at which the integer parameters are resolved using global optimization. (*Bottom:*) The times at which the integer parameters are resolved using *simple rounding* (without the LLL algorithm). Clearly, it takes a longer time for the integer parameter estimates to settle in this case than in the global optimization case (150 as compared to 85 seconds).

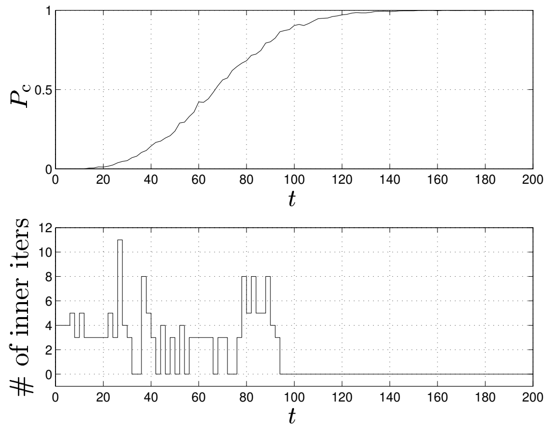

Figure 12: (*Top:*) P_c as a function of time. (*Bottom:*) Number of inner iterations in the global optimization algorithm for finding the integer parameter estimates as a function of time. For high P_c, the *initial guess* for the global minimizer is *guaranteed* to be the *global minimizer* using a *very cheaply computed* lower bound on $d_{\min}$ (using the Gram-Schmidt method) which is a byproduct of the verification step. Therefore, the global optimization algorithm becomes extremely efficient for high P_c.

$d_{\min}$ using the Gram-Schmidt method). It is very interesting to note that for high P_c ($P_c > 0.9$ or after 95 seconds), the only case for which we are really interested in z_{ML}, this always happens and therefore the global optimization algorithm becomes extremely efficient. According to the discussion of §5.2 this makes perfect sense since as P_c gets larger so does the probability that the suboptimal algorithm gives the optimal solution. Note that the Gram-Schmidt lower bound is very easy to compute and is a byproduct of the verification step.

Another example of the application of the methods described in this paper for GPS can be found in [CRA+97].

8 Conclusions

In this paper we considered parameter estimation and verification in linear models subject to additive Gaussian noise when some of the parameters are known to be integers. The integer nature of these parameters results in high computational complexity for finding the maximum likelihood estimates. The main computational effort in the estimation step is the solution to an integer least-squares problem (NP-hard). Verifying the maximum likelihood estimates is also as hard as the estimation problem. The probability of correct integer parameter estimation P_c was used to verify the estimates, which is a quantity that is very difficult to compute numerically (e.g., just bounding P_c using $d_{\min}$ is conjectured to be NP-hard).

The estimation and verification ($d_{\min}$) problems are closely related to solving simultaneous Diophantine equations (non-homogeneous and homogeneous respectively). Methods for solving simultaneous Diophantine equations (optimal up to a factor of c^q) are chiefly based on a polynomial time algorithm due to Lenstra, Lenstra and Lovász (LLL algorithm), and therefore, the LLL algorithm is very useful in finding relatively efficient algorithms for solving the estimation and verification problems. It should be noted that there are different alternatives to the LLL basis reduction algorithm in the literature (cf. [Kan83, Sch85]) that (almost arbitrarily) tradeoff the complexity of the algorithm with its performance (i.e., for example, how well it can be used to approximate $d_{\min}$). A more complex basis reduction algorithm can be used to get a better lower bound on $d_{\min}$ and hence a better lower bound on P_{c}. Also, such an algorithm can be used to get a better initial guess for z_{ML}, that can possibly, result in a better overall efficiency for the global optimization algorithm of §5.4.

In practice, the proposed global optimization algorithm for solving the estimation problem is very fast and almost instantly solves least-squares problems with a few tens of integer variables (larger problem sizes than those typically encountered in applications such as GPS) on a typical personal computer. Moreover, the global optimization algorithm becomes even more efficient for high P_{c} or when the estimates become reliable[6] (according to §5.2 as P_{c} gets larger then so does the probability that the suboptimal algorithm gives the optimal solution). As for the verification problem, it is possible to compute bounds on P_{c} very efficiently that become exact as P_{c} approaches unity, and therefore, we are not too conservative in bounding P_{c} when the estimates become reliable.

Simulation results show that we will get very poor estimates if we neglect the integer nature of the parameters, i.e., treat these parameters as being real, compute the estimates by solving a standard least-squares problem and then round them off to the nearest integers.

All these observations suggest the following general outline for integer parameter estimation in linear models (e.g., in GPS applications):

General outline for integer parameter estimation in linear models. Under the setup of §1

- Update G and $\tilde{y}$ after every measurement, say recursively, as in §6.2.
- Compute $P_{\mathrm{c,up}}$ in (18), the upper bound on P_{c}. This bound is very tight in practice as seen in the simulations. While P_{c} is small, the reliability in the estimates is low (so it is not necessary to compute lower bounds on P_{c} or even z_{ML}).
- When $P_{\mathrm{c,up}}$ becomes large (say > 0.99), compute the lower bound $P_{\mathrm{c,low}}$ in which the Gram-Schmidt method is used to provide a lower bound on $d_{\min}$.
- When $P_{\mathrm{c,low}}$ becomes high (say > 0.99), the estimate z_{ML} is reliable and for practical purposes it can be assumed that $z_{\mathrm{ML}} = z$. If only reliable estimates are required, only at this step solve

[6]This property is specially useful in communications applications where the system is always designed to achieve high P_{c}. In such cases, the suboptimal methods of §5.2 for computing the least-squares solution would yield z_{ML} that can be checked using condition (25) (cf. [MTHBC98]).

for $z_{\rm ML}$ using the algorithm of §5.4. Since $P_{\rm c}$ is close to unity, the algorithm of §5.4 for computing $z_{\rm ML}$ is very efficient.

- Once the integer parameter z is resolved, plug $z = z_{\rm ML}$ into the equations, and use standard methods to estimate or verify the real parameter x (see §5.5 and §4.4).

References

[AM79] B. D. O. Anderson and J. B. Moore. *Optimal Filtering.* Prentice Hall, Englewood Cliffs, NJ, 1979.

[Bab86] L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6:1–13, 1986.

[Boa81] P. van Emde Boas. Another NP-complete partition and the complexity of computing short vectors in a lattice. Technical Report # 81-04, Mathematical Institute, University of Amsterdam, Amsterdam, 1981.

[Bur74] C. D. Burnside. *Electromagnetic Distance Measurement.* Crosby Lockwood Staples, London, 1974.

[CRA+97] T. Corazzini, A. Robertson, J. C. Adams, A. Hassibi, and J. P. How. GPS sensing for spacecraft formation flying. In *Proc. of the Institute of Navigation GPS-97 Conf.*, Kansas City, MO, September 1997.

[GLS88] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Algorithms and Combinatorics. Springer-Verlag, New York, 1988.

[HB95] A. Hassibi and S. Boyd. Integer parameter estimation in linear models with GPS applications. Technical report, Information Systems Laboratory, Stanford University, Stanford, February 1995.

[HB96] A. Hassibi and S. Boyd. Integer parameter estimation in linear models with GPS applications. In *Proc. IEEE Conf. on Decision and Control*, Kobe, Japan, 1996.

[Hur89] J. Hurn. *GPS: A Guide to the Next Utility.* Trimble Navigation, 1989.

[Hur93] J. Hurn. *Differential GPS Explained.* Trimble Navigation, 1993.

[HWLC97] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System, Theory and Practice.* Springer-Verlag, 4th edition, 1997.

[JT96] P. J. de Jonge and C. C. J. M. Tiberius. *The LAMBDA method for integer ambiguity estimation: Implementation aspects*, volume 12 of *LGR-Series.* Delft Geodetic Computing Centre, 1996.

[Kan83] R. Kannan. Improved algorithms for integer programming and related problems. In *23rd IEEE Symposium on Foundations of Computer Science*, 1983.

[Koc88] K. R. Koch. *Parameter Estimation and Hypothesis Testing in Linear Models.* Springer-Verlag, 1988.

[KT96] A. Kleusberg and P. J. G. Teunissen. *GPS for Geodesy.* Lecture Notes in Earth Sciences. Springer-Verlag, 1996.

[Lei95] A. Leik. *GPS Satellite Surveying.* John Wiley & Sons, 1995.

[Len81] H. W. Lenstra, Jr. Integer programming with a fixed number of variables. Technical Report 81-03, University of Amsterdam, Dept. of Mathematics, 1981.

[Len83] H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[LLL82] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[MTHBC98] A. Maleki-Tehrani, Arash Hassibi, Stephen Boyd, and John Cioffi. An implementation of discrete multi-tone over slowly time-varying multiple-input/multiple-output channels. *Submitted to IEEE GLOBECOM*, 1998.

[OtR85] A. M. Odlyzko and H. te Riele. Disproof of the Mertens conjecture. *J. reine angew. Math.*, 357:138–160, 1985.

[PS96] B. W. Parkinson and J. J. Spilker, Jr. *Global positioning system : theory and applications.* American Institute of Aeronautics and Astronautics, Washington, DC, 1996.

[SB97] G. Strang and K. Borre. *Linear Algebra, Geodesy, and GPS.* Wellesley-Cambridge Press, October 1997.

[Sch85] C. P. Schnorr. A hierarchy of polynomial time basis reduction algorithms. In L. Lovász and E. Szemer, editors, *Theory of Algorithms*, pages 375–386. North-Holland, Amsterdam, 1985.

[Teu93] P. J. G. Teunissen. *Least-squares estimation of the ineger GPS ambiguities.* Number 6 in LGR-Series. Delft Geodetic Computing Centre, 1993.

[Teu94] P. J. G. Teunissen. A new method for fast carrier phase ambiguity estimation. In *Proc. IEEE Position, Location and Navigation Symp.*, Las Vegas, NV, 1994.

[Teu95a] P. J. G. Teunissen. The invertible GPS ambiguity transformations. *Manuscripta Geodaetica*, 20(6):489–497, September 1995.

[Teu95b] P. J. G. Teunissen. The least-squares ambiguity decorrelation adjustment: a method for fast GPS integer ambiguity estimation. *Journal of Geodesy*, 70(1-2):65–82, November 1995.

[Teu97a] P. J. G. Teunissen. A canonical theory for short GPS baselines, part IV: Precision versus reliability. *Journal of Geodesy*, 71:513–525, 1997.

[Teu97b] P. J. G. Teunissen. Closed form expressions for the volume of the GPS ambiguity search spaces. *Artificial Satellites*, 32(1):5–20, 1997.

[TJ96] P. J. G. Teunissen and P. J. de Jonge. The volume of the GPS ambiguity search space and its relevance for integer ambiguity resolution. In *Proc. of the Institute of Navigation GPS-97 Conf.*, pages 889–898, 1996.

[TT97] C. C. J. M. Tiberius and P. J. G. Teunissen. Kinematic GPS: performance and quality control. In *Proc. KIS97*, 1997.

A Lenstra, Lenstra, Lovász (LLL) algorithm

Suppose a lattice $L = \mathbf{L}(G)$ with $G = [g_1\ g_2 \cdots g_q]$ is given. A *reduced* basis for L can be obtained as follows [GLS88]

- Perform the *Gram-Schmidt orthogonalization* procedure on the vectors $g_1, g_2, \dots, g_q$, i.e., compute the vectors $g_1^*, g_2^*, \dots, g_q^*$ through the recursion (21). $\left\{g_1^*, g_2^*, \dots, g_q^*\right\}$ will be an orthogonal basis for the subspace spanned by $g_1, g_2, \dots, g_q$. From (21), it is trivial that any vector g_j can be expressed as a linear combination of the vectors $g_1^*, g_2^*, \dots, g_q^*$ as

$$g_j = \sum_{i=1}^{j} \mu_{ji} g_i^*, \tag{40}$$

where $\mu_{ji} = g_j^T g_i^* / \|g_j^*\|^2$ for $i = 1, 2, \dots, j-1$ and $\mu_{jj} = 1$.

- For $j = 1, 2, \ldots, q$, and, given j, for $i = 1, 2, \ldots, j-1$, replace g_j by $g_j - \lceil \mu_{ji} \rfloor g_i$, where $\lceil \mu_{ji} \rfloor$ is the integer nearest to μ_{ji}. Update the g_j^*s and μ_{ji}s with the new g_js according to (21) and (40).
- If there is a subscript j violating

$$\|g_{j+1}^* + \mu_{(j+1)j} g_j^*\|^2 \geq \frac{3}{4}\|g_j^*\|^2, \tag{41}$$

  then interchange g_j and g_{j+1} and return to the first step[7], otherwise, stop. $G = [g_1\, g_2 \cdots g_q]$ is the *reduced* generator matrix of the lattice L.

The choice of 3/4 in (41) as the allowed factor of decrease is arbitrary: any number between 1/4 and 1 would do just as well. The most natural choice (giving a best upper bound on the product $\|g_1\|\|g_2\| \cdots \|g_q\|$) would be 1 instead of 3/4; but the polynomiality of the resulting algorithm cannot be guaranteed.

[7]A particularly efficient strategy for choosing which j is described in [LLL82].