Maximum Likelihood Multipath Estimation in Comparison with Conventional Delay Lock Loops

Michael Lentmaier and Bernhard Krach, *German Aerospace Center (DLR)*

BIOGRAPHY

Michael Lentmaier received the Dipl.-Ing. degree in electrical engineering from University of Ulm, Germany, in 1998, and the Ph.D. degree in telecommunication theory from Lund University, Sweden, in 2003. As a Postdoctoral Research Associate he spent 15 months at University of Notre Dame, Indiana, and four months at University of Ulm. Since January 2005, he has been with the Institute of Communications and Navigation at the German Aerospace Center (DLR).

Bernhard Krach received the Dipl.-Ing. degree in electrical engineering from University of Erlangen-Nuremberg, Germany, in 2005. Since that he has been with the Institute of Communications and Navigation at the German Aerospace Center (DLR).

ABSTRACT

The implementation of the maximum likelihood estimator for the time-delay estimation problem is practically intractable for navigation signals due to its complexity, especially when due to multipath reception several superimposed replica are taken into account. Recently it has been shown that signal compression techniques can overcome this problem, as the maximum likelihood estimator can be formulated efficiently upon a reduced data set of much smaller size compared to the original data, where the reduced data set forms a sufficient statistic for the estimated signal parameters. This paper focuses on the formulation of such a signal compression based estimator. Furthermore the integration of the estimator into navigation receivers is addressed, in particular by considering the delay lock loop architecture that is employed within conventional navigation receivers. A novel approach for integrating the efficient maximum likelihood estimator into a generic tracking loop is proposed. The performance of the proposed method is assessed by computer simulations. The results show that the conventional delay lock loop is outperformed with respect to noise performance as well as with respect to the multipath bias.

INTRODUCTION

In the absence of multipath, the delay lock loop (DLL) of a conventional navigation receiver implements an approximation of a maximum likelihood (ML) time-delay estimator. However, in reality the receiver typically has to cope with a superposition of the line-of-sight signal (LOSS) and some additional replica that are due to reflections. In this case a bias is introduced into the estimate of the DLL, resulting in a positioning error even if no noise is present.
If the reflected signals are taken into account, it is still possible to formulate an ML estimator (see e.g. [1]), now having several delays and amplitudes as parameters. Unfortunately, the resulting system of equations does no longer suggest a straightforward exact solution without dramatic increase in complexity. There are several practical difficulties in an implementation of the optimal estimator, given that the optimization problem is not only nonlinear but also multi-dimensional. One approach to reduce complexity is to break down the problem into a one-dimensional one and approximate the ML solution iteratively. An example is the MEDLL introduced in [1] and the SAGE algorithm considered in [2]. One of the latest introduced approaches to the address the multipath estimation problem is the recently introduced Vision Correlator [3].

A general framework for efficient implementation of the optimal multi-dimensional ML time-delay estimator has been given in [4]. The purpose of this work is to assess the performance of the ML estimator proposed in [4] when it is integrated into conventional navigation receiver architecture. Previous studies of the estimator have considered its open-loop performance. In computer simulations, the delays and amplitudes have been estimated for specific integration times without taking into account the dynamics of the tracking loop. The open-loop scenario has the advantage that it simplifies the

comparison with theoretical limits, such as given by the Cramer Rao lower bound (CRLB). And in the static case, when parameters do not change during the observation time, the performance is equivalent to that of a closed-loop scenario. In a dynamic situation, on the other hand, a comparison between open-loop and closed-loop performance is less straightforward. To take into account such scenarios, the delay estimator is put directly into the tracking loop as a replacement of the discriminator. In addition to the delay estimate, a phase estimate can be obtained from the complex amplitude of the ML solution.

The paper is structured as follows: At first the multipath estimation problem is treated theoretically. After the introduction of the signal model the concept of data size reduction is described. Then the efficient calculation and optimization of the cost function in the reduced space is addressed.
Secondly, practical implementation aspects are covered. An approach for integration of the estimator into a generic tracking loop is proposed. Its performance is assessed by computer simulations whose results are shown. The simulated scenarios comprise a multipath-free and a multipath scenario. The results for each scenario are discussed respectively.
Results, outcomes and findings are summarized to conclude the paper.

PROBLEM FORMULATION

Assume that the complex valued baseband-equivalent received signal is equal to

$$y(t) = \sum_{k=1}^{N_m} a_k s(t - \tau_k) + n(t) \qquad (1)$$

where $s(t)$ is the navigation signal transmitted by the satellite, N_m is the total number of paths reaching the receiver, and a_k and τ_k are their individual complex amplitudes and time delays, respectively. The signal is disturbed by additive white Gaussian noise, $n(t)$. After sampling this can be rewritten as

$$\mathbf{y} = \sum_{k=1}^{N_m} a_k \mathbf{s}(\tau_k) + \mathbf{n} = \mathbf{S}(\boldsymbol{\tau})\mathbf{a} + \mathbf{n} \qquad (2)$$

In the compact form on the right hand side the samples of the delayed signals are stacked together as columns of the matrix $\mathbf{S}(\boldsymbol{\tau})$, $\boldsymbol{\tau}=(\tau_1,\ldots,\tau_N)$, and the amplitudes are collected in the vector $\mathbf{a}=(a_1,\ldots,a_N)$.

Based on this signal model we can use techniques from standard estimation theory to attack the multipath problem.

The ML estimation is given by the set of delays and amplitudes that minimize the quadratic error:

$$\hat{\boldsymbol{\tau}} = \arg\min_{\boldsymbol{\tau}} L_C(\boldsymbol{\tau}),$$
$$L_C(\boldsymbol{\tau}) = \min_{\mathbf{a}} \left\| \mathbf{y} - \mathbf{S}(\boldsymbol{\tau})\mathbf{a} \right\|^2 \qquad (3)$$

Note that, although our interest is the delay of the first path, all delays and amplitudes are parameters in the optimization problem. As we will see, for a given set of delays, the optimal amplitudes can be derived explicitly, since their contribution to the cost function is linear. The problem that remains is to find an efficient method to determine the minimizing vector $\boldsymbol{\tau}$.
There are several practical difficulties in an implementation of the ML estimator as described above. Firstly, the optimisation problem given by (3) is not only nonlinear but also multidimensional. Such problems usually require iterative methods. Secondly, the data size in a typical navigation system is huge. To reduce the influence of noise, the received signal typically has to be observed over several codeword lengths, which can result in vectors $\mathbf{y}$ containing several millions of samples. This means that even a single numerical evaluation of the cost function $L_c(\boldsymbol{\tau})$ requires a large computational burden, making such an approach infeasible in a real-time application.
These problems can be approached by the reduced complexity techniques suggested in [4] [5]:

- **Data size reduction:**
  The large vector containing the received signal samples is transformed into a vector $\mathbf{y}_c$ of much smaller size before the actual optimization takes place. The goal is a systematic approach to achieve such a reduction with a negligible performance loss.

- **Newton-type optimization:**
  Compact symbolic expressions for the gradients and Hessians, in combination with the reduced data size result in a both efficient and robust technique. Interpolation methods allow arbitrary delay resolution independent of sampling rate.

DATA SIZE REDUCTION

As discussed above, the received vector $\mathbf{y}$ is a linear combination of signatures $\mathbf{s}(\tau_k)$ plus some additional noise. From a geometrical point of view, the signal term in $\mathbf{y}$ is inside the span of the set of signal replica $\{\mathbf{s}(\tau_1), \mathbf{s}(\tau_2),\ldots, \mathbf{s}(\tau_N)\}$. The goal in (3) is to find that vector within the signal space spanned by $\mathbf{S}(\boldsymbol{\tau})$, which is closest to the received vector. For a fixed $\boldsymbol{\tau}$, the best approximation of $\mathbf{y}$ is given by an orthogonal projection onto that signal space, as illustrated in Figure 1.

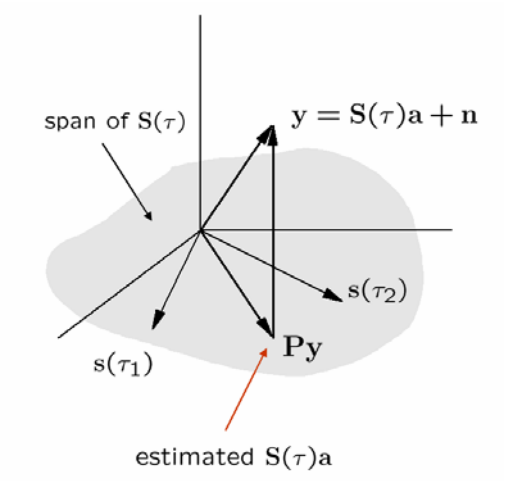


Figure 1: Projection onto the signal space

This linear projection is well-known in estimation theory and can be expressed explicitly by the projection matrix

$$\mathbf{P}(\boldsymbol{\tau}) = \mathbf{S}(\boldsymbol{\tau})\left(\mathbf{S}(\boldsymbol{\tau})^H \mathbf{S}(\boldsymbol{\tau})\right)^{-1} \mathbf{S}(\boldsymbol{\tau})^H \qquad (4)$$

Substituting the projected vector $\mathbf{P}(\boldsymbol{\tau})\mathbf{y}$ into the cost function we obtain

$$L_C(\boldsymbol{\tau}) = \left\| \mathbf{y} - \mathbf{S}(\boldsymbol{\tau})\left(\mathbf{S}(\boldsymbol{\tau})^H \mathbf{S}(\boldsymbol{\tau})\right)^{-1} \mathbf{S}(\boldsymbol{\tau})^H \mathbf{y} \right\|^2 \qquad (5)$$

In this expression the dependence on the complex amplitudes $\mathbf{a}$ has been eliminated. The cost function now depends only on the delay vector $\boldsymbol{\tau}$.

The key to reduce the data size is to find a suitable subspace of low dimension that still contains all possible signal terms. More specific, if we find a matrix $\mathbf{Q}_c$ that satisfies

$$\mathbf{Q}_c^H \mathbf{Q}_c = \mathbf{I} \quad \text{and} \quad \mathbf{Q}_c \mathbf{Q}_c^H \mathbf{s}(\tau) = \mathbf{s}(\tau) \qquad (6)$$

then it follows from the Neyman-Fisher factorization [6] that $\mathbf{Q}_c\mathbf{y}$ is a sufficient statistic for estimating the delays. In other words, there is no information loss if the delays $\boldsymbol{\tau}$ are estimated after correlation with the matrix $\mathbf{Q}_c$.

This means that the original system model can be replaced by

$$\mathbf{Q}_c^H \mathbf{y} = \mathbf{Q}_c^H \mathbf{S}(\tau)\mathbf{a} + \mathbf{Q}_c^H \mathbf{n} \qquad (7)$$

and the minimization of the cost function can be performed using the corresponding cost function, i.e.,

$$\hat{\boldsymbol{\tau}} = \arg\min_{\boldsymbol{\tau}} \left\| \mathbf{y}_c - \mathbf{P}_c \mathbf{y}_c \right\|^2 \qquad (8)$$

where

$$\mathbf{y}_c = \mathbf{Q}_c^H \mathbf{y}, \qquad \mathbf{S}_c(\boldsymbol{\tau}) = \mathbf{Q}_c^H \mathbf{S}(\boldsymbol{\tau}), \qquad (9)$$

$$\mathbf{P}_c(\boldsymbol{\tau}) = \mathbf{S}_c(\boldsymbol{\tau})\left(\mathbf{S}_c(\boldsymbol{\tau})^H \mathbf{S}_c(\boldsymbol{\tau})\right)^{-1} \mathbf{S}_c(\boldsymbol{\tau})^H \qquad (10)$$

All calculations can now be performed on this reduced size model.

For a given estimate of delays $\hat{\boldsymbol{\tau}}$, the corresponding complex amplitudes $\hat{\mathbf{a}}$ follow directly from the linear projection given in (4),

$$\hat{\mathbf{a}} = \left(\mathbf{S}_c(\hat{\boldsymbol{\tau}})^H \mathbf{S}_c(\hat{\boldsymbol{\tau}})\right)^{-1} \mathbf{S}_c(\hat{\boldsymbol{\tau}})^H \mathbf{y}_c \qquad (11)$$

If $\mathbf{Q}_c$ is a square matrix, then the transformation above is simply a rotation. In order to reduce complexity one needs to find a rectangular matrix that has a small number of columns and, hence, compresses the data size. Hereby, the conditions above should be satisfied as closely as possible if loss in performance is to be avoided

The fundamental idea is to find a subspace of small dimension in which the signal term $\mathbf{S}(\boldsymbol{\tau})\mathbf{a}$ is concentrated for any value of the parameters. Since $\mathbf{Q}_c$ compresses all columns of the signal matrix $\mathbf{S}(\boldsymbol{\tau})$ equally and all these columns have the same structure, it suffices in the selection of the subspace to consider a single signal replica $\mathbf{s}(\tau)a$. Furthermore, as the correlation with $\mathbf{Q}_c$ is a linear operation, the selection criterion is invariant in the amplitude a, which allows considering a=1 without loss of generality.

The problem can now be described as finding a compression matrix $\mathbf{Q}_c$ in such a way, that the error of reconstructing $\mathbf{s}(\tau)$ from its compressed version $\mathbf{Q}_c^H \mathbf{s}(\tau)$ is small. For a given τ the loss can be measured by the relative energy error

$$\frac{\left\| \mathbf{s}(\tau) - \hat{\mathbf{s}}(\tau) \right\|^2}{\left\| \mathbf{s}(\tau) \right\|^2} \qquad (12)$$

The reason why compression is possible is the fact that we only need to consider a limited range of possible delays. Since the most critical multipaths have delays around one chip duration or less, we may consider only τ values in a limited interval I_τ that is centered at zero. In practice the tolerated error in the choice of $\mathbf{Q}_c$ allows a trade-off between performance and complexity.

To select the compression matrix $\mathbf{Q}_c$, a two-fold data compression has been considered in [4][5], as illustrated in Figure 2.

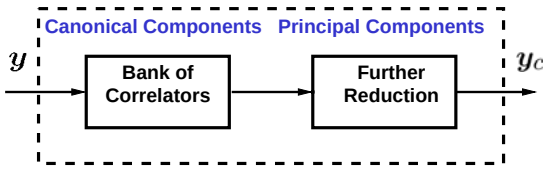


Figure 2: Two-fold data reduction

The principal components (PC) method minimizes the average reconstruction error in the desired delay range. This criterion is directly applied in the principal components method in order to select a subspace spanned by a matrix $\mathbf{Q}_p$ with orthonormal columns.

$$\mathbf{Q}_p = \arg\min_{\mathbf{Q}} \int_{I_\tau} \left\| \mathbf{s}(\tau) - \mathbf{Q}\mathbf{Q}^H \mathbf{s}(\tau) \right\|^2 \mathrm{d}\tau \qquad (13)$$

This minimizing matrix $\mathbf{Q}_p$ can be formed by the eigenvectors corresponding to the greater eigenvalues of the matrix

$$\mathbf{R}_s = \int_{I_\tau} \mathbf{s}(\tau)\,\mathbf{s}(\tau)^H \mathrm{d}\tau \qquad (14)$$

The number N_{PC} of columns in $\mathbf{Q}_p$ will determine the quality of the signal approximation. The more of the eigenvalues are close to zero, the better the achievable compression. The drawback of the PC method is that the resulting complex correlators do not posses any particular structure that might simplify a hardware implementation.

Such a structure is maintained in the canonical components (CC) method, which actually also forms the theoretical foundation behind the matched correlator techniques like the ones employed in existing navigation receivers. The CC method uses the convolutional factorization of the navigation signal into code sequence and pulse,

$$s(t) = c(t) * g(t) = \left(\sum_{i=-\infty}^{\infty} c_i \delta(t - iT_c) \right) * g(t) \qquad (15)$$

where c_i are the elements of the periodic code sequence and $g(t)$ is typically a band-limited rectangular pulse. After sampling this can be expressed by means of an N_s x N_g convolution matrix $\mathbf{C}_s$,

$$\mathbf{s} = \mathbf{C}_s \mathbf{g} \qquad (16)$$

where N_s and N_g are the lengths of the sampled signal vector $\mathbf{s}$ and the pulse vector $\mathbf{g}$, respectively. The columns of the matrix $\mathbf{C}_s$ are circularly shifted against each other according to the sample spacing T_s. With this the sampled delayed signal can be approximated by

$$\mathbf{s}(\tau) \approx \mathbf{C}(\boldsymbol{\tau}^b)\mathbf{g}(\tau) \qquad (17)$$

where the vector $\boldsymbol{\tau}^b$ of length N_{cc} defines a delay grid with spacing T_s (the Nyquist sampling period of the signal), covering the area where most energy of the pulse $g(t)$ is located. From (17) we can deduce that $\mathbf{s}(\tau)$ is within the span of $\mathbf{C}(\boldsymbol{\tau}^b)$ and, hence, can be reconstructed after correlation with the latter. From this we now want to obtain a compression matrix $\mathbf{Q}_{cc}$ that satisfies the conditions in (6) and has the same span as $\mathbf{C}(\boldsymbol{\tau}^b)$,

$$\mathbf{Q}_{cc} = \mathbf{C}(\boldsymbol{\tau}^b)\mathbf{R}_{cc}^{-1} \qquad (18)$$

where $\mathbf{R}_{cc}$ is a whitening matrix that follows from a QR or SVD decomposition of $\mathbf{C}(\boldsymbol{\tau}^b)$. This procedure has the advantage that the resulting correlators are now matched to the code $c(t)$, and the correlation procedure can thus be performed with simple integer values (-1,+1) and at the chip rate.

The outputs of the correlator bank can be expressed as

$$\mathbf{y}_{cor} = \mathbf{C}(\boldsymbol{\tau}^b)^H \mathbf{y} \qquad (19)$$

In an implementation, be it in software or hardware, the sparseness of the correlation matrix can be taken into account to reduce complexity.

It is also possible to use a bank of correlators that are matched to the navigation signal $s(t)$ directly. In this case the columns of the correlation matrix are shifted versions of the sampled navigation signal $\mathbf{s}=\mathbf{s}(\boldsymbol{\tau}=0)$, i.e.,

$$\mathbf{y}_{cor} = \mathbf{S}(\boldsymbol{\tau}^b)^H \mathbf{y} \qquad (20)$$

and

$$\mathbf{Q}_{cc} = \mathbf{S}(\boldsymbol{\tau}^b)\mathbf{R}_{cc}^{-1} \qquad (21)$$

The implementation complexity of this correlator bank may be larger, since a multiplication has to be carried out for each sample of the received vector. The difference between the two correlation operations and the resulting outputs of the correlator bank are illustrated in Figure 3 and Figure 4, respectively.

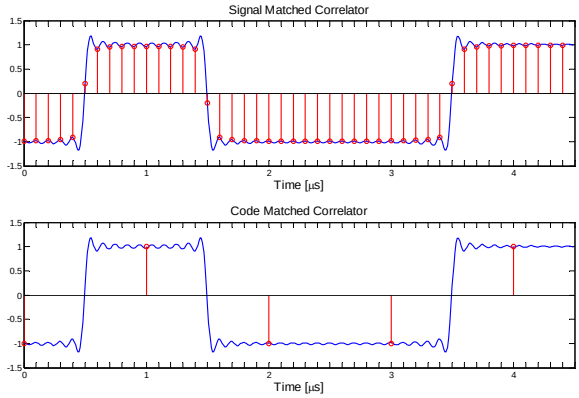


Figure 3: Illustration of signal-matched and code-matched correlation with received signal (blue).

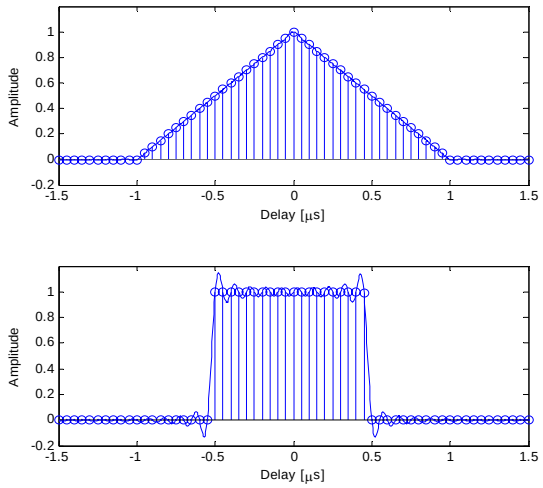


Figure 4: Output of the bank of signal-matched (top) and code-matched (bottom) correlators.

In order to achieve a better data compression, the PC method can be applied to the output of the bank of CC correlators. While both CC methods can be used equivalently, throughout this paper we will consider the code matched correlators only. Then the N_{cc} x N_{pc} PC compression matrix $\mathbf{Q}_{pc}$ is calculated from the signal

$$\mathbf{s}_{cc}(\tau) = \mathbf{Q}_{cc}^H \mathbf{s}(\tau) = (\mathbf{R}_{cc}^{-1})^H \mathbf{C}(\boldsymbol{\tau}^b)^H \mathbf{s}(\tau) \qquad (22)$$

The output after the overall compression then has the structure

$$\mathbf{y}_c = \mathbf{Q}_{pc}^H \mathbf{Q}_{cc}^H \mathbf{y} = \underbrace{\mathbf{Q}_{pc}^H (\mathbf{R}_{cc}^{-1})^H \mathbf{C}(\boldsymbol{\tau}^b)^H}_{\mathbf{Q}_c^H} \mathbf{y} \qquad (23)$$

COST FUNCTION MINIMIZATION

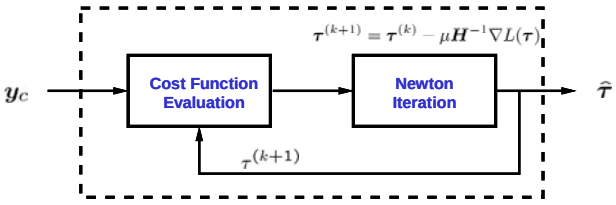


Figure 5: The Newton-type optimizer

Newton-type methods are regarded as being among the most robust and efficient techniques in unconstrained optimization. In the Newton-Raphson method, for a given cost function L(**τ**), the estimate in iteration *k* is given by

$$\boldsymbol{\tau}^{(k+1)} = \boldsymbol{\tau}^{(k)} - \mu \mathbf{H}^{-1} \nabla L(\boldsymbol{\tau}) \qquad (24)$$

where **H**=Hess L(**τ**) denotes the Hessian of L(**τ**) and μ>0 defines the step size. Alternatively, an approximation of the exact Hessian can be used, resulting in the so-called modified variable projection (MVP) method.

Because of the quadratic convergence to local minima, only a small number of iterations are required. Another advantage compared to other methods is that no special structure (e.g., Vandermonde) is required in the system model. In the considered navigation system, as shown in Figure 5, the Newton-type optimizer is applied to the reduced size vector $\mathbf{y}_c$ at the output of the data reduction. Since the number of operations per iteration is proportional to the data size, the data compression techniques described above result in a much smaller complexity in the optimization implementation.

The main drawback of this optimization technique is that the gradient and Hessian of the cost function have to be evaluated in each iteration. In general this can be a computational problem and approximations may have to be used. However, for the structure of the system model considered here, compact symbolic expressions for the exact gradients and Hessians have been derived in [4]. In combination with the data compression the Newton-type methods become thus an attractive solution for our mitigation problem.

The cost function describing the minimization problem can be written as (compare to (8))

$$L = \left\| \mathbf{y}_c - \mathbf{P}_c \mathbf{y}_c \right\|^2 = \mathrm{tr}\left\{ (\mathbf{I} - \mathbf{P}_c) \mathbf{y}_c \mathbf{y}_c^H \right\} \qquad (25)$$

Let us introduce the notation

$$\mathbf{R}_y = \mathbf{y}_c \mathbf{y}_c^H, \ \mathbf{M}_c = \left(\mathbf{S}_c^H \mathbf{S}_c \right)^{-1}, \ \mathbf{S}_c^\dagger = \mathbf{M}_c \mathbf{S}_c^H \qquad (26)$$

$$\mathbf{D} = \left(\frac{\mathrm{d}}{\mathrm{d}\tau} \mathbf{s}_c(\tau_1), \frac{\mathrm{d}}{\mathrm{d}\tau} \mathbf{s}_c(\tau_2), \ldots \right),$$
$$\mathbf{D}_2 = \left(\frac{\mathrm{d}^2}{\mathrm{d}\tau^2} \mathbf{s}_c(\tau_1), \frac{\mathrm{d}^2}{\mathrm{d}\tau^2} \mathbf{s}_c(\tau_2), \ldots \right) \qquad (27)$$

Using the symbolic method introduced in [4], the gradient of this cost function can be derived as

$$\nabla L = -2\Re\left\{ \mathrm{diag}\left\{ \mathbf{S}_c^\dagger \mathbf{R}_y (\mathbf{I} - \mathbf{P}_c) \mathbf{D} \right\} \right\} \qquad (28)$$

For the Hessian one obtains

$$\begin{aligned}\mathbf{H} = -2\Re\{ & \\ & \left(\mathbf{D}^H(\mathbf{I}-\mathbf{P}_c)\mathbf{R}_y(\mathbf{I}-\mathbf{P}_c)\mathbf{D}\right)^T \otimes \mathbf{M}_c \\ & -\left(\mathbf{S}_c^\dagger \mathbf{R}_y(\mathbf{I}-\mathbf{P}_c)\mathbf{D}\right)^T \otimes \left(\mathbf{S}_c^\dagger \mathbf{M}_c\right) \\ & -\left(\mathbf{S}_c^\dagger \mathbf{D}\right)^T \otimes \left(\mathbf{S}_c^\dagger \mathbf{R}_y(\mathbf{I}-\mathbf{P}_c)\mathbf{D}\right) \\ & -\left(\mathbf{D}^H(\mathbf{I}-\mathbf{P}_c)\mathbf{D}\right)^T \otimes \left(\mathbf{S}_c^\dagger \mathbf{R}_y \left(\mathbf{S}_c^\dagger\right)^H\right) \\ & +\mathbf{I}^T \otimes \left(\mathbf{S}_c^\dagger \mathbf{R}_y(\mathbf{I}-\mathbf{P}_c)\mathbf{D}_2\right) \ \}\end{aligned} \tag{29}$$

Here $\Re$ is the real-part operator and the symbol '$\otimes$' denotes the element-by-element (Hadamard) product of two matrices or vectors.
If the signal-to-noise ratio is large, which means that the variance σ^2 is small compared to the components of $\mathbf{R}_y$, then an approximate Hessian is often used in practice. Under this assumption we have

$$(\mathbf{I}-\mathbf{P}_c)\mathbf{R}_y \approx 0 \tag{30}$$

and the only term remaining in the Hessian results in the approximate version

$$\begin{aligned}&\mathbf{H}_{\text{appr}} = \\ &\quad 2\Re\left\{\left(\mathbf{D}^H(\mathbf{I}-\mathbf{P}_c)\mathbf{D}\right)^T \otimes \left(\mathbf{S}_c^\dagger \mathbf{R}_y \left(\mathbf{S}_c^\dagger\right)^H\right)\right\}\end{aligned} \tag{31}$$

This matrix is used in the modified variable projection (MVP) method.

During the optimization, the gradient and Hessian of the cost function have to be evaluated in every iteration for another set of possible delays $\boldsymbol{\tau}$. Since very severe multipath errors are caused by relative delays (between direct and reflected paths) of only a fraction of the chip duration T_c, the grid of the vector $\boldsymbol{\tau}$ in the optimization procedure (see (24)) needs to be sufficiently fine. This means that in the expressions (28) and (29) those matrices depending on $\boldsymbol{\tau}$ need be available in the desired resolution. All these matrices are deduced from delayed versions $\mathbf{s}(\tau)$ of the signal $\mathbf{s}$ and its derivatives. On the other hand it is desirable to achieve improved delay shift resolution without large oversampling, which again would increase complexity.

A way to achieve arbitrary resolution in $\boldsymbol{\tau}$ independent of the sampling period is the use of Fourier interpolation techniques. Since the navigation signal is composed of elementary pulses *g(t)*, we can make use the relation (17) and apply the interpolation on that pulse. Assume that the band-limited pulse *g(t)* is approximately zero outside a given time interval, and that a vector $\mathbf{g}$ of length N_g contains its sampled values with a uniform grid of separation T_s smaller than the Nyquist period. Then it is possible to calculate the samples of *g(t-τ)* for a delay τ with the interpolation formula

$$\mathbf{g}(\tau) = \mathbf{F}\,\mathrm{diag}(\mathbf{g}_F)\boldsymbol{\Phi}(\tau) \tag{32}$$

where $\mathbf{g}_F$ is the discrete Fourier transform (DFT) of $\mathbf{g}$ and

$$\begin{aligned}&\left[\mathbf{F}\right]_{q,r} = \frac{1}{N_g} e^{j2\pi(r-1)(q-1)/N_g}, \quad q,r = 1,\ldots,N_g \\ &\left[\boldsymbol{\Phi}(\tau)\right]_r = e^{-j2\pi(r-1)\tau/(N_g T_s)}, \qquad r = 1,\ldots,N_g\end{aligned} \tag{33}$$

are the N_g *x* N_g inverse DFT matrix and a length N_g Vandermonde vector, respectively.
Conventional fast Fourier transform (FFT) algorithms can be used to compute $\mathbf{g}_F$ and $\mathbf{F}$,

$$\mathbf{g}_F = \mathrm{fft}(\mathbf{g}),\ \mathbf{F}^{-1} = \mathrm{fft}(\mathbf{I}_{N_g \times N_g}) \tag{34}$$

where the FFT of a matrix is given by the transforms of its individual columns. Depending on the implementation of the transform, it may be necessary to rotate the Vandermonde vector accordingly.
Using sufficiently large N_g together with zero padding, the error in (32) becomes negligible. Since the interval N_g should also contain the non-zero part of the delayed signal *g(t-τ)*, the complexity in this interpolation is increased accordingly with the size of desired delay range. It also depends on the signal length and bandwidth, but not on the delay resolution: the Vandermonde vector allows the application of a continuous delay.
Another benefit of the Vandermonde structure in (32) is that it allows simple calculation of the derivatives of $\mathbf{g}(\tau)$ and, hence, $\mathbf{s}(\tau)$, which are required in the computation of the gradient and Hessian.
To illustrate this, an example of a band limited rectangular pulse and its shift by half a sample is given in Figure 6. The pulse has a bandwidth of 5 MHz (one sided) and is sampled with 11 samples/chip = 11 MHz to satisfy the Nyquist criterion. The samples of the curve appear at the same time instances but correspond to a shift of $\tau = T_s/2$.

The Fourier interpolation can be combined with the data compression techniques. The signal matrix and its derivatives at the desired point are computed with the signal interpolation factorization of the compressed signal $\mathbf{s}_c$

$$\mathbf{S}_c(\boldsymbol{\tau}) = \mathbf{M}_{s_c}\left(\boldsymbol{\Phi}(\tau_1),\ldots,\boldsymbol{\Phi}(\tau_{N_m})\right) \tag{35}$$

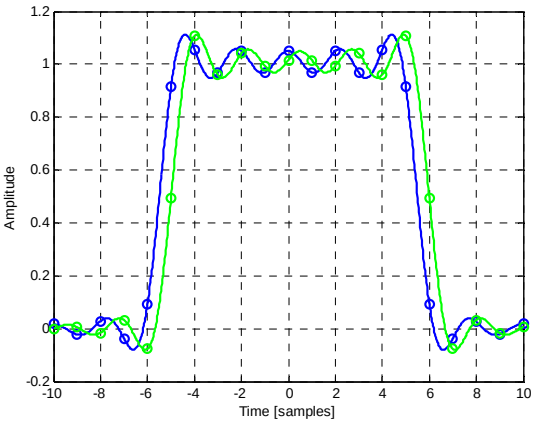


Figure 6: Example of a pulse and its shift by 1/2T$_s$

with the interpolation matrix

$$\mathbf{M}_{s_c} = \mathbf{Q}_c^H \mathbf{C}_s \mathbf{F} \mathrm{diag}(\mathbf{g}_F) \tag{36}$$

Hence, the interpolation matrix is simply multiplied with a matrix having the correspondingly shifted Vandermonde vectors as columns. The derivatives are computed accordingly, based on the easily determined derivatives of the elements of $\mathbf{\Phi}(\tau)$. With this the computations of the cost function, gradient and Hessian can be performed within the reduced signal model.

The signal interpolation also simplifies the computation of the PC matrix $\mathbf{Q}_{pc}$ [5], which is composed by eigenvectors of the correlation matrix

$$\begin{aligned} \mathbf{R}_{s_{cc}} &= E\left(\mathbf{s}_{cc}(\tau)\mathbf{s}_{cc}(\tau)^H\right) \\ &= \mathbf{M}_{s_{cc}} E\left(\mathbf{\Phi}(\tau)\mathbf{\Phi}(\tau)^H\right)\mathbf{M}_{s_{cc}}^H \end{aligned} \tag{37}$$

with

$$\mathbf{M}_{s_{cc}} = \left(\mathbf{R}_{cc}^{-1}\right)^H \mathbf{C}(\boldsymbol{\tau}^b)^H \mathbf{C}_s \mathbf{F} \mathrm{diag}(\mathbf{g}_F) \tag{38}$$

Assuming a uniform distribution of the delay τ, the mathematical expectation is calculated by integration over the corresponding delay range $I_\tau=(\tau_a,\ldots,\tau_b)$, see (14). It can be seen in (37) that the Fourier interpolation of the signal may be used to calculate $\mathbf{R}_s$ from the correlation matrix $\mathbf{R}_\Phi$ of the Vandermonde vector $\mathbf{\Phi}(\tau)$,

$$\begin{aligned} \mathbf{R}_\Phi &= E\left(\mathbf{\Phi}(\tau)\mathbf{\Phi}(\tau)^H\right) \\ &= \frac{1}{\tau_b - \tau_a} \int_{\tau_a}^{\tau_b} \mathbf{\Phi}(\tau)\mathbf{\Phi}(\tau)^H \mathrm{d}\tau \end{aligned} \tag{39}$$

This has the advantage that the elements in $\mathbf{R}_\Phi$ follow explicitly from

$$\int_{\tau_a}^{\tau_b} e^{a\tau} \mathrm{d}\tau = \begin{cases} (e^{a\tau_b} - e^{a\tau_a})/a, & a \neq 0 \\ \tau_b - \tau_a, & a = 0 \end{cases} \tag{40}$$

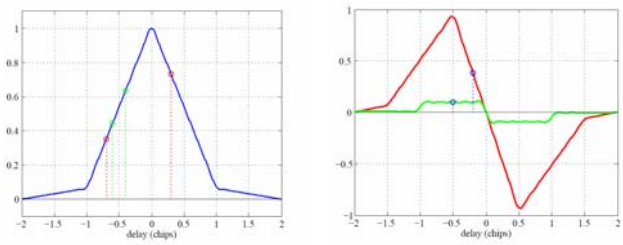
where a is a scalar value that has to be substituted accordingly for the different matrix elements. It should be noted that $\mathbf{R}_\Phi$ depends only on the number of samples in the pulse vector $\mathbf{g}$, their spacing T_s, and the delay interval I_τ.

The columns of $\mathbf{Q}_{pc}$ are then chosen as those eigenvectors of $\mathbf{R}_s$, which correspond to the N_{pc} largest eigenvalues of that matrix. Since the correlation matrix, by definition, is Hermitian, these eigenvectors are orthonormal.

INTEGRATION INTO THE TRACKING LOOP

A conventional DLL uses two correlators, matched to the navigation signal, for tracking the maximum of the autocorrelation function. If the DLL is in lock, the correlation peak is exactly in the center between the two correlators and their outputs are equal. Otherwise, the difference between the early and the late correlator indicates the direction and distance to the maximum. The discriminator curve shows this difference as a function of the current signal delay relative to the lock point, as illustrated in Figure 7 (right). It is used in the feedback of the loop in order to adjust the current local reference value of the delay and move back to the stable point at the origin. Hence, the two correlators of a DLL slide along the autocorrelation function (see left hand side in Figure 7) until their values become equal.

It can be seen in Figure 7 that the discriminator curve depends on the spacing between the two correlators. If the spacing is reduced from a standard value of 1 chip (red) to 0.1 chips (green), the linear region around the origin is reduced. Being outside this region corresponds to the case when both correlators are on the same side of the peak of the autocorrelation function. If this is the case, their output difference no longer adequately measures the distance to the origin and the performance is reduced. On the other hand, a smaller correlator spacing (commonly referred to as "narrow correlator") is known to reduce the error due to multipath.



Autocorrelation Function Early-Late Discriminator

Figure 7: Discriminator with different correlator spacings

For comparison, consider now the ML cost function $L_c(\tau)$, shown in Figure 8, which can be evaluated from the reduced data vector $\mathbf{y}_c$ with arbitrarily fine resolution in τ. The Newton optimizer estimates the delay of the incoming signal by searching the minimum of the function $L_c(\tau)$ (marked red in the figure). The current reference value τ=0 (marked green) can be used as initial value for the search.

When no multipath is present, a minimization of the cost function is equivalent to a maximization of the autocorrelation function. To track the maximum, the discriminator of a conventional DLL provides an error estimate that is used for the correction of the current delay reference. In its linear region it produces a scaled approximation of the delay value that minimizes the function $L_c(\tau)$. Consequently, the discriminator can be interpreted as an approximation of a single-path ML estimator.

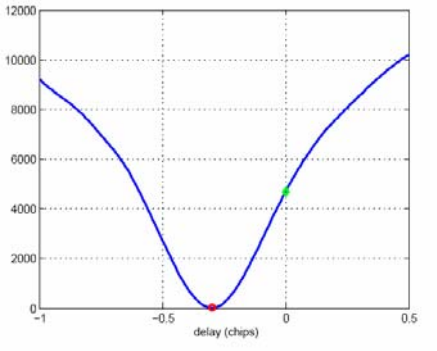


Figure 8: Cost function example

Nevertheless, as the discriminator in the DLL can be regarded as a sub-estimator element within the loop, it is the entire loop itself that finally implements the overall estimator. Thus it is difficult to compare the forward ML estimator with the DLL, which in fact implements a fully sequential estimator. Compared to a sequence of independent ML estimates the sequentially estimating DLL offers robustness against noise and transients when exposed to parameter dynamics. For this reason the incorporation of the ML estimator into the receiver becomes not straightforward, given that the ML estimator is designed to operate only on time intervals, during which the signal parameters do not change. Beside data modulation this fact restricts the possible interval for coherent integration. Hence the DLL is sometimes able to outperform the ML estimator in practical scenarios, depending on the integration interval, the noise level and the parameter variations, even when averaging is applied to the sequence of ML estimates with a filter characteristic equal to that of the DLL.

In order to overcome these shortcomings a hybrid solution with an ML estimator incorporated in a generic loop is proposed (in-the-loop MLE approach) as depicted in Figure 9. The ML estimator operates in place of the generic discriminator and provides also a phase estimate to the phase lock loop (PLL) based on (11).

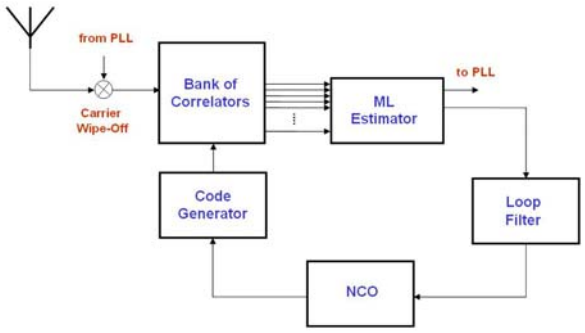


Figure 9: Tracking loop with ML estimator (in-the-loop-MLE).

PERFORMANCE WITHOUT MULTIPATH

For the performance assessment of the proposed approach in a multipath-free scenario several simulations have been carried out. The generic incoherent DLL is compared to the proposed in-the-loop MLE architecture and the performance bound that is given by the CRLB by means of the root-mean-square (RMS) tracking error in dependence on the C/N_0. The signal used within all simulations was a GPS C/A code signal of 10 MHz one-sided bandwidth. The time of coherent integration was set to 1 ms for the DLL simulations, which have been carried out for an early/late correlator spacing of 1 chip, 0.5 chips, 0.3 chips and 0.1 chips respectively. The MLE simulations have been carried out for a coherent integration time of 1 ms and 10 ms respectively, whereas the MLE assumes a single path being present, i.e., $N_m = 1$. Code matched correlators were used for the CC compression method with $N_{cc} = 41$, followed by a PC compression with $N_{pc} = 30$. The loop bandwidth for all simulations was equal to 2 Hz. The CRLB was calculated according to [6] based upon the considered received signal and the loop bandwidth.

The results, which are depicted in Figure 10, show for the DLL simulations the well-known effect of improved noise performance as the correlator spacing gets decreased, as it is covered within [7] for instance. In the figure it can be observed that the in-the-loop MLE attains the CRLB for high C/N_0, but it does diverge from the bound for low C/N_0, whereas the point and amount of divergence depends on the coherent integration time.

The phenomenon of divergence may be traced back to the fact that the ML estimator is non-linear unlike the generic early/late discriminator. For the DLL it is equivalent whether the linear gradient operation (early minus late operation) is applied before or after the linear filtering.

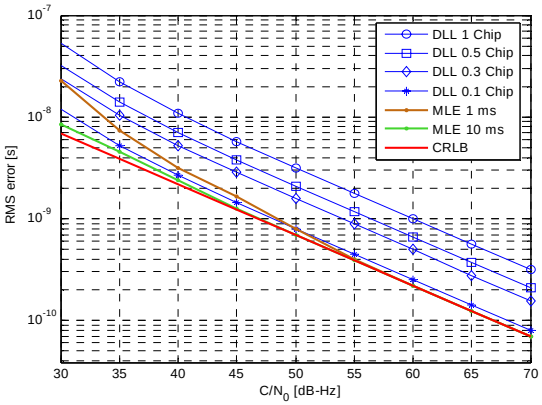


Figure 10: Simulated performance of 1-path in-the-loop MLE compared to DLL and CRLB.

The non-linear in-the-loop MLE, on the other hand, has to operate before the loop filter, since otherwise the linear character of the loop is lost. This leads to the divergence for low C/N_0, as the ML gradient operates on the data from the coherent interval only, unlike the generic linear discriminator, which operates on filtered data effectively. Hence longer coherent integration times are preferable for the in-the-loop-MLE as it is shown by the simulation results.

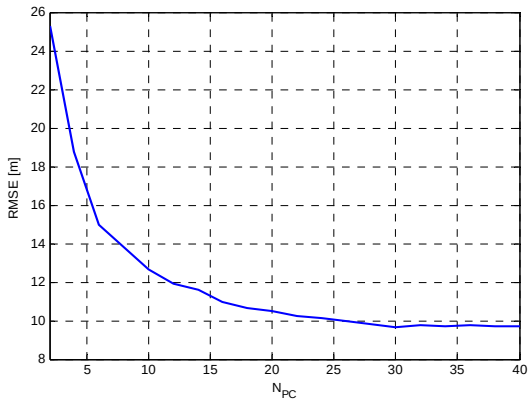


Figure 11: Performance of 1-path ML estimator as function of subspace dimension N_{pc}, N_{cc}=41.

To determine the influence of the subspace size on the performance, the MLE has been simulated as forward estimator for different N_{pc} values. The signal and MLE settings were the same as for the previous simulations, whereas the forward MLE operates on a snapshot of 1ms data at 45 dB-Hz and the resulting RMS error is obtained from a statistic of 100,000 independent snapshot estimates for each N_{pc}. The results are depicted in Figure 11 and show that for the simulated scenario of N_{cc}=*41* and C/N_0=45 dB-Hz a number of approximately 30 PC components is needed in order to avoid an observable loss due to the PC compression. It should be noted that this number strongly depends on the number of CC correlators N_{cc}. The generous choice of 41 correlators ensures that the ML estimator behaves linear in a region of ±1 chip around the in-phase correlator. This is an advantage compared to the DLL where the linear region is significantly reduced for correlator spacings below 1 chip. For the ML estimator the width of the linear region can be traded against the values N_{cc} and N_{pc}. If the deviations from the stable lock point are expected to be small, a much smaller number of correlators and PC components should be sufficient.

PERFORMANCE WITH MULTIPATH

The direct relation between cost function and correlator outputs is no longer given, if the number of paths N_m in the received signal is larger than one. The discriminator of the DLL gets distorted by multipath, and the loop locks to a value that no longer corresponds to the ML solution of the direct path.

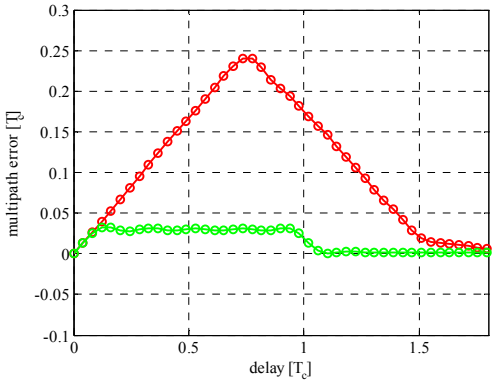


Figure 12: Multipath error envelope: advantage of narrow correlator (green).

For a single additional path, i.e., N_m=2, the error envelope shows the resulting noise-less multipath error as a function of the delay $\Delta\tau = \tau_2 - \tau_1$ between the direct and the second path. The error depends not only on the relative multipath amplitude and phase, but also on the spacing between the correlators of the DLL. Figure 12 shows the error for equal phase, amplitude ratio 1/10, and spacing of 1 chip (red) and 0.1 chips (green) considering a C/A code signal generated from the band limited example pulse shown in Figure 6. It can be seen that the narrow correlator is much less disturbed by the second path [8].

The effect of multipath on the output of the correlator bank is shown in Figure 13. The MEDLL [1] and the SAGE algorithm [2] both operate on the signal-matched correlator outputs for searching the ML solution. More recently, it was suggested to perform the estimation on filtered chip transitions instead [3][9]. Interestingly, there appears to be a close connection between the signal compression theorem in [9] and the sufficient statistics condition given in (6) for the output of the code matched correlator bank.

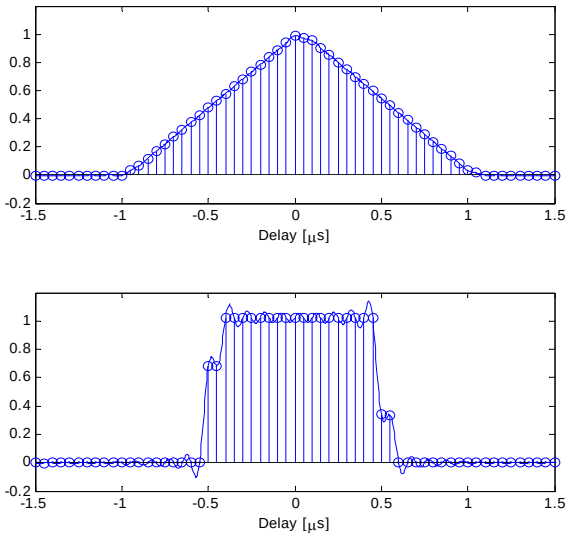


Figure 13: Effect of multipath on the signal-matched (top) and code-matched (bottom) correlator outputs.

The actual cost function of an ML estimator for the case N_m=2 is a function of two dimensions, one for each delay. An example is shown in Figure 14. Due to the linearity of the transformation into the subspace the computation of this function is independent of the data reduction method. Analogously to the one-dimensional case, the proposed multipath mitigation algorithm searches the minimum (marked red) of this cost function, starting from some given initial estimate (marked green). As before the current delay reference of the DLL can serve for the selection of the start value. For selecting the relative multipath distance of the initial value, previous estimations can be taken into account. Note, that a second minimum exists (marked blue), which corresponds to an equivalent solution after sorting. Its existence follows from the symmetry of the cost function.

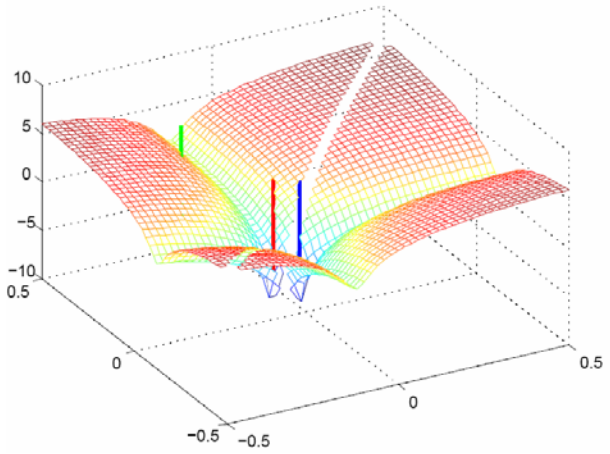


Figure 14: Multipath cost function

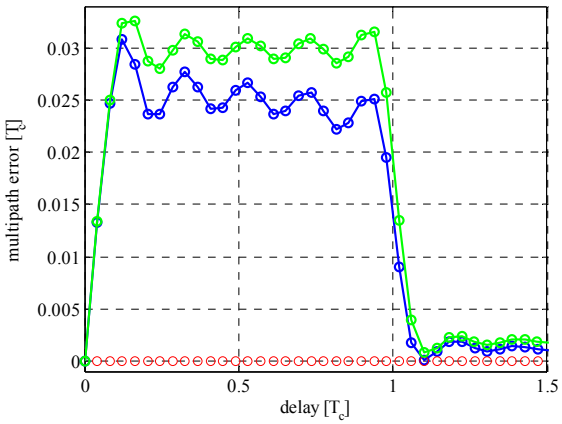


Figure 15: Multipath error envelope: narrow correlator (green) vs. ML estimator with N_m = 1 (blue) and N_m = 2 (red).

Figure 15 shows the multipath error of the narrow correlator in comparison with the ML estimation algorithm. The red curve shows that the error becomes negligible if the true cost function is used in the ML estimation.
The blue curve results if the ML estimator wrongly assumes a single path, i.e., N_m = 1. It can be seen that this curve is still slightly better than that of the narrow correlator. If the spacing of the narrow correlator is decreased further it will actually converge to the blue curve of the 1-path ML estimator. The corresponding error could be further reduced by increasing the bandwidth at the receiver input.

The multipath performance of the in-the-loop MLE in presence of noise has been simulated for the same signal and parameter settings as in the previous section. For a fixed C/N_0=50 dB-Hz and a coherent integration time of 10ms the tracking error is shown as a function of time for a relative multipath delay of $\Delta\tau = 10^{-7}$s (30 m) and $\Delta\tau = 3.33\cdot10^{-8}$s (10 m) in Figure 16 and Figure 17, respectively. The results show again that the 2-path MLE successfully mitigates the bias caused by the multipath, even for delays below 1/10 of the chip duration T_c. The figures also show that for smaller $\Delta\tau$ the variance of the ML estimator is increased. This is a well-known phenomenon that is also reflected by the corresponding CRLB, which diverges in the region of small $\Delta\tau$. Lower noise levels allow mitigation of multipath with smaller delays.

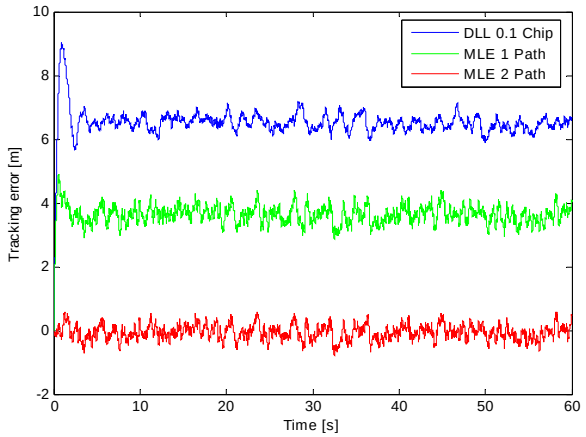


Figure 16: Simulated multipath performance of in-the-loop MLE with N_m=1 and N_m=2 compared to DLL for $\Delta\tau = 10^{-7}$s (30 m).

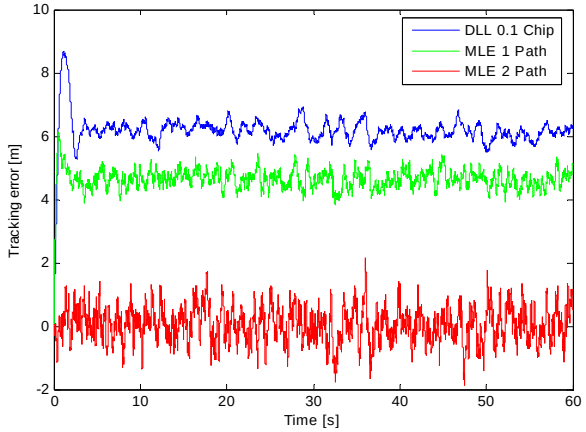


Figure 17: Simulated multipath performance of in-the-loop MLE with N_m=1 and N_m=2 compared to DLL for $\Delta\tau = 3.33 \cdot 10^{-8}$s (10 m).

CONCLUSIONS

The performance of an in-the-loop MLE has been investigated and compared to a conventional DLL. Simulation results in absence of multipath show that at high C/N_0 the MLE attains the CRLB. At medium to low C/N_0 the MLE is still capable of outperforming the narrow correlator if the coherent integration time is chosen sufficiently high. While the linear region of a DLL decreases with the correlator spacing, the one of the MLE can be adjusted by selecting the range covered by the bank of correlators. In the presence of multipath the simulation results show that the MLE is capable of mitigating the multipath bias even for multipath delays smaller than a tenth of the chip duration.

The suggested data compression and interpolation techniques are not restricted to the efficient computation of the ML solution by Newton methods but can be used in a much wider range of applications where the signal parameter likelihoods can be of interest.

REFERENCES

[1] van Nee, D.J.R., J. Siereveld, P. Fenton, and B. Townsend, "The Multipath Estimating Delay Lock Loop: Approaching Theoretical Accuracy Limits", Proceedings of the IEEE Position, Location and Navigation Symposium, Las Vegas, NV, USA, 1994.

[2] Felix Antreich, Oriol Esbri-Rodriguez, Josef A. Nossek, Wolfgang Utschick, "Estimation of Synchronization Parameters Using SAGE in a GNSS-Receiver", Proceedings of the ION GNSS 2005, Long Beach, California, USA, 2005.

[3] P. Fenton, "The Theory and Performance of NovAtel Inc.´s Vision Correlator," Proceedings of the ION GNSS 2005, Long Beach, CA, September 2005, pp. 2178-2186.

[4] Jesus Selva Vera, "Efficient Multipath Mitigation in Navigation Systems", Ph.D. thesis, DLR/ Polytechnical University of Catalunya, 2004.

[5] Jesus Selva, "Complexity reduction in the parametric estimation of superimposed signal replicas", Signal Processing, Elsevier Science Volume 84, Issue 12, December 2004, Pages 2325-2343.

[6] Kay, Steven M., Fundamentals of Statistical Signal Processing – Estimation Theory, Prentice Hall Signal Processing Series, Prentice Hall, New Jersey, 1993

[7] Weill, L., "C/A Code Pseudoranging Accuracy - How Good Can It Get?", ION GPS-94, Salt Lake City, pp. 133 - 141, 20.-23.09.94

[8] van Dierendonck. A. J., Fenton P., Ford T., "Theory and Performance of Narrow Correlator Spacing in a GPS Receiver", Journal of The Institute of Navigation, Vol. 39, No.3 Fall 1992

[9] Weill, L.R., "Achieving Theoretical Bounds for Receiver-Based Multipath Mitigation Using Galileo OS Signals," Proceedings of the ION GNSS 2006, Fort Worth, TX, September 2006.