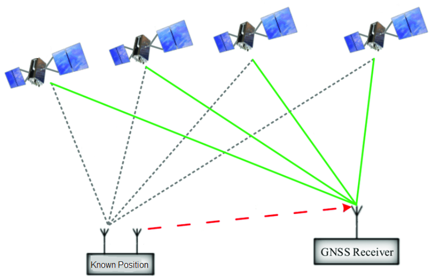
Maximum likelihood estimation

Cristian Iñiguez Rodriguez, 1566514@uab.cat

***Abstract*—The main objective is to obtain the Maximum likelihood estimate of the carrier phase ϕ assuming that the carrier frequency f_0 is know. A study on the bias and the variance of the ML estimator will also be carried out to calculate it's mean square error (MSE) and compare it with the best estimator with the Cramér-Rao bound.**

Introduction

In the report we will evaluate the maximum likelihood estimator of the carrier phase of a GNSS (Global Navigation Satellite System) received signal. GNSS refers to a set of satellites orbiting the Earth that were designed to broadcast signals with global coverage whose processing provides the user's positioning. The procedure is based on calculating the travel time of those signals going from the satellites to the user's receiver, and with triangulation techniques estimates the position.



GNSS signals contain so-called navigation messages that allow the user to determine the position of satellites at a given time. A satellite could be identified given the cross-correlation of the received signal with a replica of the code from such a satellite and avoid interferences with other satellites thanks to their inter-orthogonal properties.

Once the GNSS signal impinges onto the receiver antenna, the signal is filtered and digitalized; sometimes before that, the signal is down-converted to an intermediate frequency. It results in to real-valued and discrete-time band-pass signal.

$$r_k(n) = AC_k(n-\tau)D_k(n-\tau)\cos(2\pi f_0 n + \phi) + w(n)$$

where n is the discrete-time sample index, A is the received signal amplitude, τ the propagation delay, phi is the carrier phase offset and f_0 is the received carrier frequency, that is composed of two therms F_{IF}, which is known, and F_d which is the Doppler frequency, offset due to the relative motion between satellite and user's receiver, it usually unknown and because is expressed in lower case, which indicates that it is a normalized frequency. Finally, $w(n)$ is the white Gaussian noise, $w(n) \sim \mathcal{N}(0, \sigma^2)$.

In order to process the received signal, the user needs to identify the three unknown synchronization parameters τ, f_d, ϕ. We will assume D_k constant so it can be ignored henceforth, and PRN code is wiped off due to the fact that $C_k(n)C_k(n) = 1$ for simplicity we will assume it's our case henceforth. So the signal becomes

$$r(n) = A\cos(2\pi f_0 n + \phi) + w(n) \tag{1}$$

I. Answer to Question 1

Considering only one sample of the random signal $r(n)$ in (1) and that the carrier phase ϕ is unknown but deterministic, we can write down the corresponding PDF (Probability density function) such as

$$f(r(n)) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(r(n)-A\cos(2\pi f_0 n+\phi))^2}{2\sigma^2}} \tag{2}$$

The mean can be computed, step by step, by doing the following.

$$\mu(n) = E[r(n)] = E[A\cos(2\pi f_0 n + \phi) + w(n)]$$

$$= E[A\cos(2\pi f_0 n + \phi)] + E[w(n)]$$

$$\mu(n) = A\cos(2\pi f_0 n + \phi) \tag{3}$$

As the result is $\mu(n)$ in (3), and we know that phi is deterministic we can say that $A\cos(2\pi f_0 + \phi)$ doesn't depend on the time sample; however, the n is multiplying inside the cosinus, so bringing about that the mean depends on the discrete time sample n.

II. Answer to Question 2

Considering N samples of the random signal $r(n)$ in (1) that are stacked into a $(N \times 1)$ vector $\mathbf{r} = \begin{bmatrix} r(0) \\ \vdots \\ r(N-1) \end{bmatrix}$ the probability density function is the following.

$$f(\mathbf{r}) = \frac{1}{(\sigma^2 2\pi)^{\frac{N}{2}}} e^{\frac{-1}{2\sigma^2}\sum_{n=0}^{N-1}(r(n)-A\cos(2\pi f_0 n+\phi))^2} \tag{4}$$

The mean can be computed, step by step, by doing the following.

$$\boldsymbol{\mu} = E[\mathbf{r}] = \begin{bmatrix} E[r(0)] \\ E[r(0)] \\ \vdots \\ E[r(N-1)] \end{bmatrix}$$

$$= \begin{bmatrix} E[A\cos(\phi)] \\ E[A\cos(2\pi f_0 + \phi)] \\ \vdots \\ E[A\cos(2\pi f_0 (N-1) + \phi)] \end{bmatrix} + \begin{bmatrix} E[w(0)] \\ E[w(1)] \\ \vdots \\ E[w(N-1)] \end{bmatrix}$$

$$\boldsymbol{\mu} = \begin{bmatrix} A\cos(\phi) \\ A\cos(2\pi f_0 + \phi) \\ \vdots \\ A\cos(2\pi f_0 (N-1) + \phi) \end{bmatrix} \tag{5}$$

In both questions we assume that the noise is white Gaussian noise with $w(n) \sim \mathcal{N}(0, \sigma^2)$.

III. Answer to Question 3

The ML estimate is the value of the unknown parameter to be estimated that maximizes the log-likelihood function; that is, nothing but the PDF, of the received samples. As such, the ML estimator is obtained as

$$\hat{\theta_{ML}} = \arg\max_{\theta} \ln f(x;\theta)$$

For our problem, we need to maximize our (log)-likelihood function by finding the value of ϕ for which the derivative of the (log)-likelihood is equal to zero. Giving as a result

$$\hat{\phi_{ML}} = -\arctan \frac{\sum_{n=0}^{N-1} r(n)\sin(2\pi f_0 n)}{\sum_{n=0}^{N-1} r(n)\cos(2\pi f_0 n)} \tag{6}$$

A close look at the ML estimator shows a resemblance to phase and quadrature modulation. If we look closely at our signal r(n) we could see that is almost equal to, in the most general sense, a QAM. QAM can be represented as [1]

$$x_n(t) = A_n g(t)\cos(2\pi f_c t + \phi_n)$$

where $g(t)$ in our case it would be constant evaluated to 1.

IV. Answer to Question 4

Once we know that the ML estimator declaration is (6) implementing it in Matlab is as easy as:

```
    r_ =  @(n, phi) cos(n + phi);
    ml =  @(r, n) -atan(sum(r .* ...
        sin(n))/sum(r .* cos(n)));
```

To confirm that the ML estimator that estimates correctly any value of ϕ we generate signal samples of $r(n) = \cos(2\pi f_0 n + \phi)$ with different values of ϕ, and then we compute the absolute bias for each one and we store it for plot the results.

```
n  =  (0:10000) .* 2 * pi * 0.49;
index = 1;
for i = 0:0.25:pi/2
   r  = r_(n,i);
   v(index) = abs(i - ml(r, n));
   index = index + 1;
end
```

Theoretically it should be 0, but we get a little error, as can it be seen here.

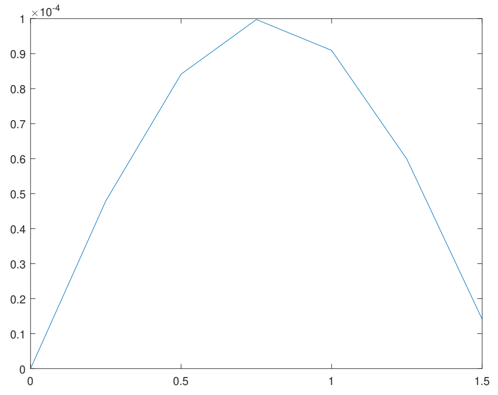


Fig. 1. Absolute bias value of ML estimator

V. Answer to Question 5

To correctly implement in Matlab the code of $r(n)$ in (1) we need first to implement the white Gaussian noise with 0.1 noise power. Once we know how to calculate the noise is as easy as adding to $r(n) = \cos(2\pi f_0 n + \phi)$ to properly get (1). The main idea is to take advantage of Matlab and generate a matrix with random values; it would be the noise, with 0.1 noise power, and then adding it to r_k matrix, that is just the signal repeated k times. In Matlab it would be the following implementation

```
awgn = @(n) sqrt(0.1) .* randn(1000, n);
rk = @(n, phi, k) repmat(r_(n, phi),1000,1) ...
     + awgn(length(n));
```

The next step is to calculate the bias of r_k, to do it properly we need, firstly, calculate the ML of each vector; once we obtain it, we get a vector with the ML of each vector on r_k. Finally, we need to compute the mean of that vector and then we can compute the bias.

```
function [out, out2] = mle_phi(rk, n, phi)
wrap = @(r) -atan(sum(r.' .* sin(n))/sum(r.' ...
     .* cos(n)));
m   = num2cell(rk',1);
ml_vector = cellfun(wrap, m);
meanml = mean(ml_vector);
out    = meanml - phi;
out2   = mean((ml_vector-meanml).^2);
end
```

The following code is just for each case of $10, 100, 1000$

```
phi = 0.1;
n    = (1:10) .* 2 * pi * 0.3;         % N = 10
[b1, v1] = mle_phi(rk(n, phi), n, phi);
n    = (1:100) .* 2 * pi * 0.3;        % N = 100
[b2, v2] = mle_phi(rk(n, phi), n, phi);
n    = (1:1000) .* 2 * pi * 0.3;       % N = 1000
[b3, v3] = mle_phi(rk(n, phi), n, phi);
```

For a better representation, we plot the absolute value of bias.

```
bias     = abs([b1 b2 b3]);
semilogy([10 100 1000], bias);
```

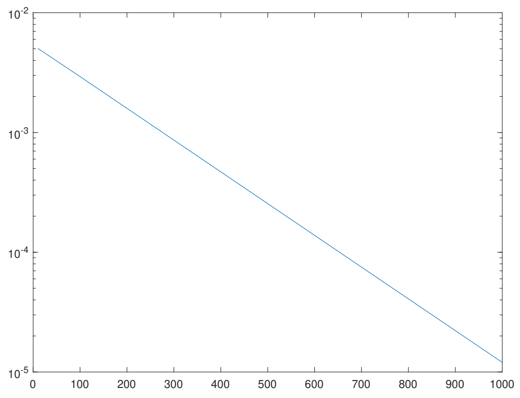


Fig. 2. Bias values 0.005037, 0.0029297, 1.2029e-05

VI. Answer to Question 6

Thanks to the way that our Matlab code computes the bias, calculating the variance is as easy as extending our code with the following code.

```
    out2   = mean((ml_vector-meanml).^2);
```

Then, to plot the variance we just need to do the following.

```
varianza = [v1 v2 v3];
semilogy([10 100 1000], varianza);
```

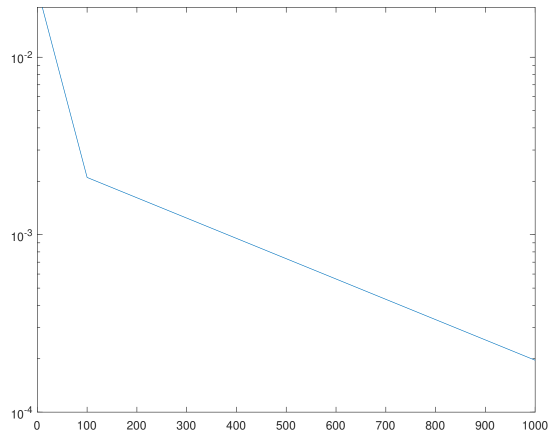


Fig. 3. variance values 0.019119, 0.002102, 0.00019611

VII. Answer to Question 7

From a qualitative point of view, an estimator is efficient when it uses the data it has in the best possible way following the philosophy of the MVU (unbiased minimal variance). This can be evaluated using the MSE measurement, which covers both parameters.

$$MSE\{\hat{\theta_{ML}}\} \doteq bias^2\{\hat{\theta_{ML}}\} + var\{\hat{\theta_{ML}}\}$$

and we want it to be as close to zero as possible. The way to check the optimality of an estimator is to compare its MSE with the well-known Cramér-Rao Bound (CRB), which is defined as the best accuracy that an unbiased estimator can achieve.

$$var\{\hat{\theta_{ML}}\} \geq CRB\{\hat{\theta_{ML}}\}$$

Its easy to see that an unbiased estimator would imply that their $MSE\{\hat{\theta_{ML}}\} = var\{\hat{\theta_{ML}}\}$. However, (when doing simulations) is important to make sure that the actual performance does fulfill the following inequality:

$$MSE\{\hat{\theta_{ML}}\} \geq CRB\{\hat{\theta_{ML}}\}$$

The CRB (Cramér–Rao bound) expresses a lower bound on the variance of unbiased estimators of a deterministic (fixed, though unknown) parameter, the variance of any such estimator is at least as high as the inverse of the Fisher information.

$$CRB\{\theta\} = \frac{1}{I(\theta)}$$

where the Fisher information $I(\theta)$ is defined by

$$I(\theta) = -E[\frac{\partial^2 ln(f(\theta|x))}{\partial\theta^2}]$$

After operating mathematically we obtain that the CRB of our sign is

$$CRB(\phi) \approx \frac{2\sigma_w^2}{NA^2}$$

The next Matlab code is the implementation of the MSE and CRB, For a given noise power and computing the MSE with a finer sweep of N from $N = 1$ to $N = 1000$ in steps of 10.

```
index = 1;
for  i = 1:10:1000
    n   = (1:i) .* 2 * pi * 0.4;
    [b, v] = mle_phi(rk(n, phi), n, phi);
    MSE(index) = b^2 + v;
    index = index + 1;
end

figure;
n = 1:10:1000;
CRB = (2*0.1)./(n);
s3 = semilogy(n, MSE, n, CRB);
```

We can see that the estimator does a great job. Although it is not totally unbiased, since it has a small value, at the time of calculating the MSE when that unbiased should be important, and we make the comparison with the Cramer-Rao limit, we can see that it has the same shape, so we can say that it is an efficient estimator. It is also consistent because every time the number of samples grows the MSE becomes closer to zero.

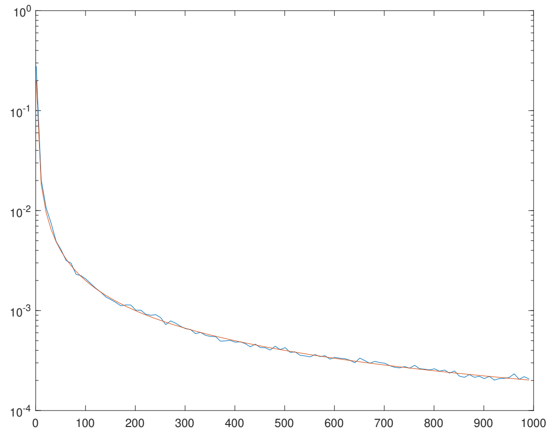


Fig. 4. CRB (orange) and MSE (blue)

Conclusion

We have seen how from a simplified GNSS signal that we will call r (n), which is similar to a QAM signal, how to eliminate the noise, that is, the random signal; the way to do it is as easy as calculating the expectation or the also called mean. The phase estimator is calculated, taking into account that phi is the only unknown. And with the bias and variance of the estimator, we calculate the MSE; after comparing it with the CRB, we see that our estimator is efficient and consistent. Overall, we saw some tools to evaluate our estimators in the real world and know how good they are.

References

[1] Si Chen and Alexander M. Wyglinski. “Chapter 3 - Digital communication fundamentals for cognitive radio”. In: *Cognitive Radio Communications and Networks*. Ed. by Alexander M. Wyglinski, Maziar Nekovee, and Y. Thomas Hou. Oxford: Academic Press, 2010, pp. 41–83.