Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

NACC
Laboratory work 2
GPS positioning and Dilution of Precision

Joan Olmos

2020

Contents

1 Introduction

In this laboratory practice you will learn that the precision in the GPS positioning depends on two different terms: on one side the measured pseudoranges to the satellites have errors due to the ionosphere, relativistic effects, Doppler, etc. Those errors are modeled and the receiver tries to cancel them as much as possible, but some error still remains. On the other hand, finding the user's position entails solving a system of equations and, depending on the system matrix condition number, the obtained solution (the user position) will show an error higher than the error in the measured pseudoranges. This second effect is only due to the processing of the pseudoranges at the receiver, which "amplifies" the errors. This amplification effect is called the "dilution of precision" (DOP) and it depends only on the system matrix condition, which in turn depends on the relative positions of the involved GPS satellites with respect to the user. Knowing this, if we need to obtain the position with high precision (for topography applications for example), we can plan in advance the field measurements and take positions only when the satellites over the measurement site are in the best configuration to reduce the DOP to the minimum possible. **The tasks to perform are marked in red.**

2 LOS coverage

In the previous laboratory work (P1) you have written a set of Matlab programs to compute the ECEF coordinates of all GPS satellites at a given time. Those coordinates are independent of the position of the user. Using the programs of the previous laboratory work, **compute the ECEF coordinates of all GPS satellites at current time and date.**

If we focus on a specific user location on the Earth surface it is important to know which subset of GPS satellites are visible from the user position at a given moment. If a satellite is visible we say that we are in "Line of Sight" (LoS) coverage of that satellite. To achieve this goal we need to find the pointing angles: azimuth, elevation and distance of any GPS satellite from a given observation point. With these parameters we can decide if there is LoS between the satellite and the user.

2.1 Azimuth, elevation and distance

In this part of the work the azimuth, elevation and distance from the observation point (the user position) to the satellite will be calculated.

- The **azimuth** (β) is the horizontal angle to the satellite measured clockwise from a north base line.
- The **elevation** (α) is the angle between the horizon and the center of the satellite. Those angles are represented in Figure 2.1.

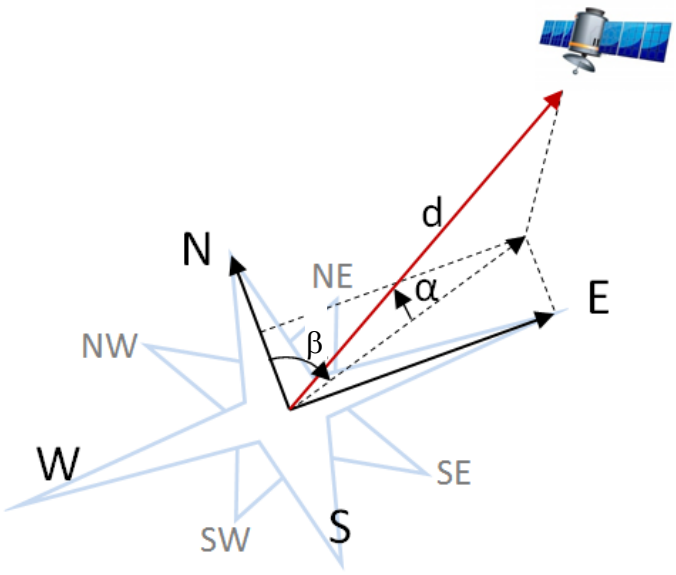


Figure 2.1. Azimuth (β), elevation (α) and distance (d)

Using the algorithms from previous work you are able to locate a satellite at ECEF (x, y, z) Cartesian coordinates which are fixed to three Earth reference axis. Also, we can specify an observation point (the user position) with its LLA coordinates. The scenario is shown in Figure 2.2. In order to compute the pointing angles and distance the next steps must be followed:

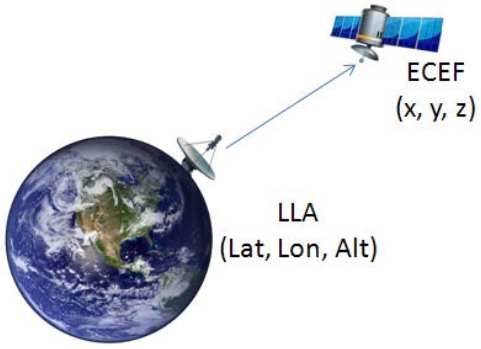
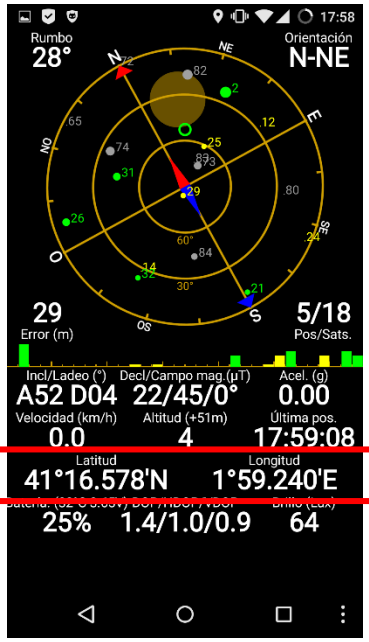


Figure 2.2. Satellite pointing scenario

Set the user position to the LLA coordinates of EETAC:

EETAC ⟹



and **write a function to convert the LLA coordinates of the user position to ECEF Cartesian (x, y, z) using the ellipsoid WGS84**. You can use the expressions:

$$x = \left(\frac{a}{\sqrt{1 - e^2 \operatorname{sen}^2 \phi}} + h \right) \cos\phi \cdot \cos\lambda$$

$$y = \left(\frac{a}{\sqrt{1 - e^2 sen^2 \phi}} + h \right) \cos\phi \cdot sen\lambda$$

$$z = \left(\frac{a\left(1 - e^2\right)}{\sqrt{1 - e^2 sen^2 \phi}} + h \right) sen\phi$$

where λ means geodetic longitude, ϕ means geodetic latitude, h means the height above the ellipsoid and a, e are the ellipsoid parameters.

Compute the vector that points from the user position to each of the satellites by subtractions of vectors. This is simple if both vectors are in the ECEF coordinate system.

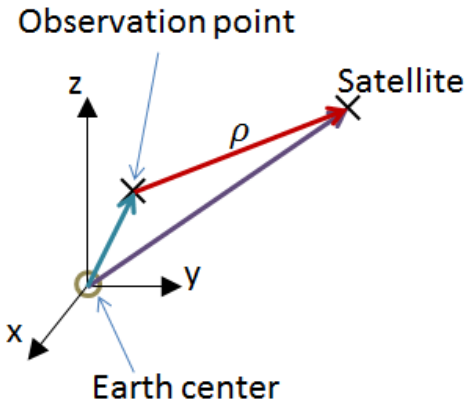


Figure 2.3. Pointing vector

Computing the vector ρ is achieved by subtracting the vectors of the user position (observation point) and the satellite ECEF coordinates:

$$\bar{\rho} = \overline{Sat} - \overline{Obs}$$

Now we know the vector ρ that points from the user position to the satellite but its coordinates are given in a system parallel to ECEF centered at the user position. So we must rotate the axes to transform the vector to NED (North, East, Down) coordinates with respect to the user position.

Write a MATLAB functions to rotate the axes from Cartesian ECEF to Cartesian NED coordinates and use them to convert the pointing vector coordinates into NED coordinates relative to the user position. This conversion is a multiplication of the ECEF vector by a 3x3 matrix whose coefficients depend on the latitude and longitude of the observation point. To rotate from ECEF to NED we must apply the following matrix product (where λ and ϕ are, respectively, the geodetic longitude and latitude of the user):

$$\begin{bmatrix} N \\ E \\ D \end{bmatrix} = \begin{bmatrix} -sin\emptyset\, cos\lambda & -sin\emptyset\, sin\lambda & cos\emptyset \\ -sin\lambda & cos\lambda & 0 \\ -cos\emptyset\, cos\lambda & -cos\emptyset\, sin\lambda & -sin\emptyset \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

The elevation and azimuth angles, and also the distance, from NED Cartesian coordinates can be computed easily. Figure 2.4 represents the vector from the observation point to the satellite represented in NED coordinates. In this case the azimuth is denoted as β and the elevation as α.

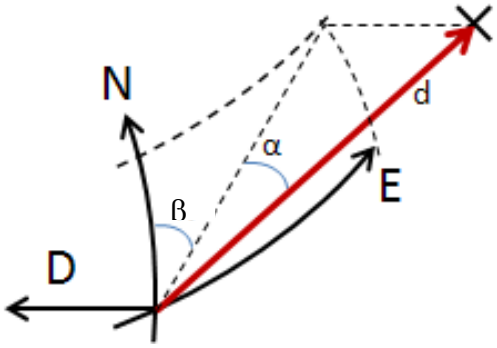


Figure 2.4. From NED coordinates to elevation and azimuth

From Figure 2.4 we can **obtain the pointing angles and the distance to each of the satellites**:

$$d = \sqrt{N^2 + E^2 + D^2}$$

$$\alpha = \text{asin}\left(-\frac{D}{d}\right)$$

$$\beta = \text{atan}\left(\frac{E}{N}\right)$$

Once you have implemented those expressions it's easy to find out which satellites are visible from the user position at a given time. You just have to **check which satellites have an elevation higher than a minimum value and plot them in a polar representation using the provided Matlab function "viewSat" (see ANNEX I: Available satellite mapping function)**. We will assume that a satellite must be at least **10**˚ above the horizon to provide a useful GPS signal. At this point you must **compare your result with the polar plot generated with the tools recommended at section 5 of this document**.

3 The GPS navigation system of equations

This section describes how the satellite pseudoranges are related to the user position and the method for finding the user position by solving the navigation system of equations.

Ignoring all the error sources except the time errors, the pseudorange between the user and one of the satellites can be expressed as:

$$\rho_j = \sqrt{(x_j - x_u)^2 + (y_j - y_u)^2 + (z_j - z_u)^2} - c \cdot t_u = f(x_u, y_u, z_u, t_u)$$

where x_j, y_j and z_j are the ECEF coordinates of the satellite, x_u, y_u and z_u are the ECEF coordinates of the user, c is the speed of light and t_u the difference between the time error of the satellite (known) and the time error of the user's clock (unknown).

Since this expression contains 4 unknowns, we need at least 4 independent equations to solve the system. Therefore, a minimum of 4 pseudoranges to 4 different satellites have to be measured to find the position and the clock error (notice that the time error is common to the 4 equations). The pseudorange is measured at the receiver by computing the correlation between the received PRN sequences and their local version. It's called a pseudorange because it depends on the range (distance) to the satellite but also on the time error and other errors (mainly the ionosphere error, since the pseudorange assumes that the signal travels at the speed of light, which is not true inside the ionosphere).

Although the pseudorange equations are not linear, the equation system can be solved using a linearization method. In order to obtain a linear system, the pseudorange ρ_j is expressed as:

$$\rho_j = f(x_u, y_u, z_u, t_u) = f(\hat{x}_u + \Delta x_u, \hat{y}_u + \Delta y_u, \hat{z}_u + \Delta z_u, \hat{t}_u + \Delta t_u)$$

where $\hat{x}_u$, $\hat{y}_u$, $\hat{z}_u$ and $\hat{t}_u$ are the user estimated position and time (known) and Δx_u, Δy_u , $\Delta z_u$and Δt_u are the increments (unknown) with respect to the estimated values.

By expressing the function as a Taylor series (centered at the estimated values) and keeping only the linear terms we get the approximated expression:

$$\rho_j \cong f(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u) + \frac{\partial f(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u)}{\partial \hat{x}_u}\Delta x_u + \frac{\partial f(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u)}{\partial \hat{y}_u}\Delta y_u + \frac{\partial f(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u)}{\partial \hat{z}_u}\Delta z_u + \frac{\partial f(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u)}{\partial \hat{t}_u}\Delta t_u$$

where all the partial derivatives can be computed to give:

$$\frac{\partial f(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u)}{\partial \hat{x}_u} = -\frac{x_j - \hat{x}_u}{\hat{r}_j} \equiv -a_{xj}$$

$$\frac{\partial f(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u)}{\partial \hat{y}_u} = -\frac{y_j - \hat{y}_u}{\hat{r}_j} \equiv -a_{yj}$$

$$\frac{\partial f(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u)}{\partial \hat{z}_u} = -\frac{z_j - \hat{z}_u}{\hat{r}_j} \equiv -a_{zj}$$

$$\frac{\partial f(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u)}{\partial \hat{t}_u} = -c$$

and where $\hat{r}_j$ is the estimated range to the satellite (which can be computed based on the user estimated position):

$$\hat{r}_j = \sqrt{\left(x_j - \hat{x}_u\right)^2 + \left(y_j - \hat{y}_u\right)^2 + \left(z_j - \hat{z}_u\right)^2}$$

Notice that the vector $\vec{a}_j = \left(a_{xj}, a_{yj}, a_{zj}\right)$ is the unitary vector pointing from the estimated user position to the satellite. Then, the estimated pseudorange, which can be computed based on the user estimated position and clock error, is given by:

$$\hat{\rho}_j = \sqrt{\left(x_j - \hat{x}_u\right)^2 + \left(y_j - \hat{y}_u\right)^2 + \left(z_j - \hat{z}_u\right)^2} - c \cdot \hat{t}_u = f(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u)$$

Notice that estimated pseudorange is not measured, but computed; so it contains no errors. Based on $\hat{\rho}_j$ we can write the approximated pseudorange as:

$$\rho_j \cong \hat{\rho}_j - a_{xj} \cdot \Delta\mathrm{x_u} - a_{yj} \cdot \Delta\mathrm{y_u} - a_{zj} \cdot \Delta\mathrm{z_u} - \mathrm{c} \cdot \Delta\mathrm{t_u}$$

And defining $\Delta\rho_j = \hat{\rho}_j - \rho_j$ we get:

$$\Delta\rho_j = a_{xj} \cdot \Delta\mathrm{x_u} + a_{yj} \cdot \Delta\mathrm{y_u} + a_{zj} \cdot \Delta\mathrm{z_u} + \mathrm{c} \cdot \Delta\mathrm{t_u}$$

In case of measuring the pseudorange to 4 different satellites we get the following equation system:

$$\begin{aligned}
\Delta\rho_1 &= a_{x1} \cdot \Delta\mathrm{x_u} + a_{y1} \cdot \Delta\mathrm{y_u} + a_{z1} \cdot \Delta\mathrm{z_u} + \mathrm{c} \cdot \Delta\mathrm{t_u} \\
\Delta\rho_2 &= a_{x2} \cdot \Delta\mathrm{x_u} + a_{y2} \cdot \Delta\mathrm{y_u} + a_{z2} \cdot \Delta\mathrm{z_u} + \mathrm{c} \cdot \Delta\mathrm{t_u} \\
\Delta\rho_3 &= a_{x3} \cdot \Delta\mathrm{x_u} + a_{y3} \cdot \Delta\mathrm{y_u} + a_{z3} \cdot \Delta\mathrm{z_u} + \mathrm{c} \cdot \Delta\mathrm{t_u} \\
\Delta\rho_4 &= a_{x4} \cdot \Delta\mathrm{x_u} + a_{y4} \cdot \Delta\mathrm{y_u} + a_{z4} \cdot \Delta\mathrm{z_u} + \mathrm{c} \cdot \Delta\mathrm{t_u}
\end{aligned}$$

which is a linear system that can be also written in matrix notation:

$$\mathbf{\Delta\rho} = \mathbf{H} \cdot \mathbf{\Delta u}$$

In that system the known data are: the incremental pseudorange vector:

$$\mathbf{\Delta\rho} = \begin{bmatrix} \Delta\rho_1 \\ \Delta\rho_2 \\ \Delta\rho_3 \\ \Delta\rho_4 \end{bmatrix}$$

and the unitary vectors from the user estimated position (matrix $\mathbf{H}$):

$$\mathbf{H} = \begin{bmatrix} a_{x1} & a_{y1} & a_{z1} & 1 \\ a_{x2} & a_{y2} & a_{z2} & 1 \\ a_{x3} & a_{y3} & a_{z3} & 1 \\ a_{x4} & a_{y4} & a_{z4} & 1 \end{bmatrix}$$

While the unknown vector is the user incremental position:

$$\mathbf{\Delta u} = \begin{bmatrix} \Delta x_u \\ \Delta y_u \\ \Delta z_u \\ c \cdot \Delta t_u \end{bmatrix}$$

When the number of available satellites is exactly 4 the system can be solved by inverting matrix $\mathbf{H}$:

$$\mathbf{\Delta u} = \mathbf{H}^{-1} \cdot \mathbf{\Delta \rho}$$

but when more than 4 satellites are available, which is the usual situation, the system becomes over specified (more equations than unknowns) and it has not an exact solution. In this case we must assume that the pseudorange vector contains errors and solve the system in the sense of the least square error. The least square error solution is based on the pseudo-inverse matrix and can also be applied when only 4 satellites are available:

$$\mathbf{\Delta u} = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}\mathbf{H}^{\mathrm{T}} \cdot \mathbf{\Delta \rho}$$

This solution must be computed repeatedly until the magnitude of $\mathbf{\Delta u}$ becomes sufficiently small (to ensure that the Taylor series approximation is realistic).

Set the user position to the LLA coordinates of EETAC and compute matrix $\mathbf{H}$ using all the GPS satellites with elevation higher than 10° at the current time and date. You don't need to solve the navigation system of equations. Notice that matrix $\mathbf{H}$ is (*N x 4*), since it has *4* columns and as many rows as satellites are taken into consideration (*N*). Also notice that the rows of $\mathbf{H}$ (except the last column which is always equal to 1) are simply the 3 Cartesian coordinates of the unitary vectors pointing from the user to each of the satellites.

4 Dilution of precision

In the previous equations we did not take into account explicitly the error terms inside the measured pseudoranges. Remember that those errors can be reduced by using adequate modelling, but cannot be reduced to zero. If we admit that there are some unknown (not modelled) errors in the measured pseudoranges the positioning equation becomes:

$$\mathbf{\Delta \rho'} = \mathbf{\Delta \rho} + \mathbf{\eta} = \mathbf{H} \cdot \mathbf{\Delta u} + \mathbf{\eta}$$

where $\mathbf{\Delta \rho'}$ is the measured pseudorange vector (with errors), $\mathbf{\Delta \rho}$ is the theoretical pseudorange vector (including only the time errors) and $\mathbf{\eta}$ is the pseudorange error vector. The least squares solution of the previous equation is:

$$\mathbf{\Delta u}' = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}\mathbf{H}^{\mathrm{T}} \cdot \mathbf{\Delta \rho}' = \mathbf{\Delta u} + \left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}\mathbf{H}^{\mathrm{T}} \cdot \boldsymbol{\eta} = \mathbf{\Delta u} + \boldsymbol{\varepsilon}$$

where $\mathbf{\Delta u}'$ is the obtained incremental position of the user. This means that we cannot obtain the user position without errors. The error vector affecting the obtained position is given by:

$$\boldsymbol{\varepsilon} = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}\mathbf{H}^{\mathrm{T}} \cdot \boldsymbol{\eta}$$

and we see that the pseudorange error vector is amplified by a factor which depends only on the system matrix. This is the factor that originates the DOP.

We can assume that all the components of $\boldsymbol{\eta} = [\eta_1\ \eta_2\ \eta_3\ \eta_4]^{\mathrm{T}}$ are independent zero mean random variables with the same variance: σ^2. Ignoring for the moment the matrix $(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}$, the vector that produces the positioning errors in terms of distance if contributed by the multiplication of the first 3 rows of matrix $\mathbf{H}^{\mathrm{T}}$ by vector $\boldsymbol{\eta}$, that is:

$$\begin{bmatrix} a_{x1} \cdot \eta_1 + a_{x2} \cdot \eta_2 + a_{x3} \cdot \eta_3 + a_{x4} \cdot \eta_4 \\ a_{y1} \cdot \eta_1 + a_{y2} \cdot \eta_2 + a_{y3} \cdot \eta_3 + a_{y4} \cdot \eta_4 \\ a_{z1} \cdot \eta_1 + a_{z2} \cdot \eta_2 + a_{z3} \cdot \eta_3 + a_{z4} \cdot \eta_4 \end{bmatrix}$$

The squared magnitude of this error vector is:

$$(a_{x1} \cdot \eta_1 + a_{x2} \cdot \eta_2 + a_{x3} \cdot \eta_3 + a_{x4} \cdot \eta_4)^2 + \left(a_{y1} \cdot \eta_1 + a_{y2} \cdot \eta_2 + a_{y3} \cdot \eta_3 + a_{y4} \cdot \eta_4\right)^2 + (a_{z1} \cdot \eta_1 + a_{z2} \cdot \eta_2 + a_{z3} \cdot \eta_3 + a_{z4} \cdot \eta_4)^2$$

Assuming that all the components of vector $\boldsymbol{\eta}$ are independent random variables, the average of this squared magnitude is:

$$(a_{x1}^2 + a_{y1}^2 + a_{z1}^2) \cdot \sigma^2 + (a_{x2}^2 + a_{y2}^2 + a_{z2}^2) \cdot \sigma^2 + (a_{x3}^2 + a_{y3}^2 + a_{z3}^2) \cdot \sigma^2 + (a_{x4}^2 + a_{y4}^2 + a_{z4}^2) \cdot \sigma^2 = 4\sigma^2$$

That is the same average of the squared magnitude of vector $\boldsymbol{\eta}$, so we conclude that the product of vector $\boldsymbol{\eta}$ by matrix $\mathbf{H}^{\mathrm{T}}$ does not increase the distance error.

The time error component is contributed by the product of the last row of matrix $\mathbf{H}^{\mathrm{T}}$ by vector $\boldsymbol{\eta}$, that is $\eta_1 + \eta_2 + \eta_3 + \eta_4$, which has an average squared value of $4\sigma^2$ (again the same as the average squared magnitude of vector $\boldsymbol{\eta}$), so we conclude that the product by matrix $\mathbf{H}^{\mathrm{T}}$ does not produce any increase in the positioning errors.

It is only the product by matrix $\mathbf{Q} \equiv (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}$ what contributes to the increase of the error variance. Notice that matrix $\mathbf{Q}$ is calculated from the estimated user position. Matrix $\mathbf{Q}$ is always a (4 x 4) square matrix (even if more than 4 satellites are considered for $\mathbf{H}$).

Assuming that all the components of $\boldsymbol{\eta}$ are independent zero mean random variables with the same variance (σ^2), only the diagonal elements of $\mathbf{Q}$ are relevant for the DOP calculation. This can be shown by computing the covariance matrix of vector $\boldsymbol{\varepsilon}$:

$$\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^{\mathrm{T}} = \mathbf{Q}\mathbf{H}^{\mathrm{T}} \cdot \boldsymbol{\eta}\boldsymbol{\eta}^{\mathrm{T}} \cdot \mathbf{H}\mathbf{Q}^{\mathrm{T}} = \mathbf{Q}\mathbf{H}^{\mathrm{T}} \cdot \sigma^2\mathbf{I} \cdot \mathbf{H}\mathbf{Q}^{\mathrm{T}} = \sigma^2 \cdot \mathbf{Q} \cdot \mathbf{H}^{\mathrm{T}}\mathbf{H} \cdot \mathbf{Q}^{\mathrm{T}} = \sigma^2\mathbf{Q}^{\mathrm{T}} = \sigma^2\mathbf{Q}$$

For our purposes matrix $\mathbf{Q}$ can be written as:

$$\mathbf{Q} = \begin{bmatrix} Q_{xx} & Q_{yx} & Q_{zx} & Q_{ctx} \\ Q_{xy} & Q_{yy} & Q_{zy} & Q_{cty} \\ Q_{xz} & Q_{yz} & Q_{zz} & Q_{ctz} \\ Q_{xct} & Q_{yct} & Q_{zct} & Q_{ctct} \end{bmatrix}$$

Since vector $\boldsymbol{\varepsilon}$ has been computed in Cartesian ECEF coordinates, σ_x, σ_y and σ_z are the error standard deviations referred to the ECEF axis. If we want to obtain the variance components of the error vector referred to the local coordinate system (σ_N, σ_E and σ_D) we must rotate vector $\boldsymbol{\varepsilon}$ to North, East, Down Cartesian coordinates. This will allow us to decompose the DOP into HDOP and VDOP. The rotated error vector can be expressed as $\boldsymbol{\delta} = \mathbf{R}\boldsymbol{\varepsilon}$, with $\mathbf{R}$ given by:

$$\mathbf{R} = \begin{bmatrix} -sin\emptyset\, cos\lambda & -sin\emptyset\, sin\lambda & cos\emptyset & 0 \\ -sin\lambda & cos\lambda & 0 & 0 \\ -cos\emptyset\, cos\lambda & -cos\emptyset\, sin\lambda & -sin\emptyset & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Notice that matrix $\mathbf{R}$ rotates the spatial components of $\boldsymbol{\varepsilon}$ from ECEF to NED and does not change the time component of $\boldsymbol{\varepsilon}$. The covariance matrix of $\boldsymbol{\delta}$ is:

$$\boldsymbol{\delta}\boldsymbol{\delta}^{\mathrm{T}} = \mathbf{R} \cdot \boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^{\mathrm{T}} \cdot \mathbf{R}^{\mathrm{T}} = \mathbf{R} \cdot \sigma^2\mathbf{Q} \cdot \mathbf{R}^{\mathrm{T}} = \sigma^2\mathbf{Q}'$$

$$\mathbf{Q}' = \mathbf{R} \cdot \mathbf{Q} \cdot \mathbf{R}^{\mathrm{T}}$$

$$\mathbf{Q}' = \begin{bmatrix} q_{NN} & q_{NE} & q_{ND} & q_{Nct} \\ q_{EN} & q_{EE} & q_{ED} & q_{Ect} \\ q_{DN} & q_{DE} & q_{DD} & q_{Dct} \\ q_{ctN} & q_{ctE} & q_{ctD} & q_{ctct} \end{bmatrix}$$

The Horizontal DOP (HDOP) is the multiplication factor that increases the distance errors in the horizontal plane. It is given by:

$$HDOP = \sqrt{q_{NN} + q_{EE}}$$

The HDOP is usually the most relevant parameter for positioning applications on the Earth surface. The vertical DOP (VDOP) is the multiplication factor that increases the distance error in the vertical plane (altitude error). It is given by:

$$VDOP = \sqrt{q_{DD}}$$

The Position DOP (PDOP) is the multiplication factor that increases the 3D distance error. It is given by:

$$PDOP = \sqrt{Q_{xx} + Q_{yy} + Q_{zz}} = \sqrt{q_{NN} + q_{EE} + q_{DD}} = \sqrt{HDOP^2 + VDOP^2}$$

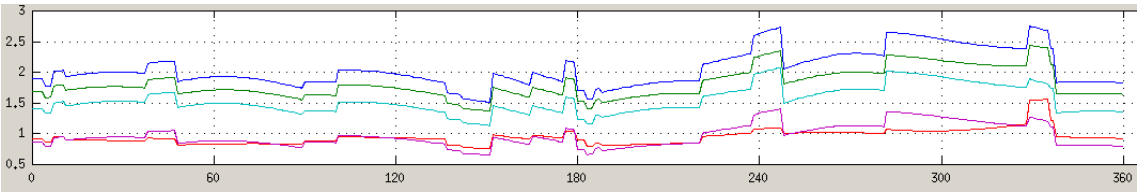
The Time DOP (TDOP) is the multiplication factor that increases the time error. It is given by:

$$TDOP = \sqrt{q_{ctct}}$$

The Geometric DOP (GDOP) takes into account the position and time error increase factor:

$$GDOP = \sqrt{PDOP^2 + TDOP^2}$$

Write a Matlab function to compute the GDOP, PDOP, HDOP, VDOP and TDOP. You must previously compute matrices Q and Q' from matrix H. Add a time loop to your program to repeat the DOP computations for each minute within the next 6 hours interval. You will get a plot similar to this:



The following pseudo-code summarizes the explained steps (where functions "LLA2ECEF", "CONSTELLATION", "LoS" and "POINTING" or equivalent code must be written by you or reused from previous lab. work):

```
xyzSatellites = CONSTELLATION(time);% find ECEF coordinates of GPS sats.

sta_lat = lat(EETAC)     % set user position to EETAC
sta_lon = lon(EETAC)

xyztUserPosition = LLA2ECEF([sta_lat, sta_lon]); % find ECEF user position

time = now();    % Set time

xyzLocalSatellites = LoS(xyzSatellites); % find visible GPS satellites

H = POINTING(xyzLocalSatellites, xyztUserPosition);% Compute matrix H

Q ≡ (H^T H)^-1 % Compute matrix Q

Q' ≡ RQR^T % Compute matrix Q'

GDOP = sqrt(q_NN+ q_EE + q_DD + q_ctct);    % Geometric DOP
PDOP = sqrt(q_NN+ q_EE + q_DD);             % Position DOP
```

```
HDOP = sqrt(q_NN+ q_EE);                % Horizontal plane DOP
VDOP = sqrt(q_DD);                          % Vertical DOP
TDOP = sqrt(q_ctct);                        % Time DOP
```

5 Test your results with an online service or with your smart phone

You can use an online calculator to compare the obtained results about DOP and satellite visibility. Check this URL:

http://gnssmissionplanning.com/

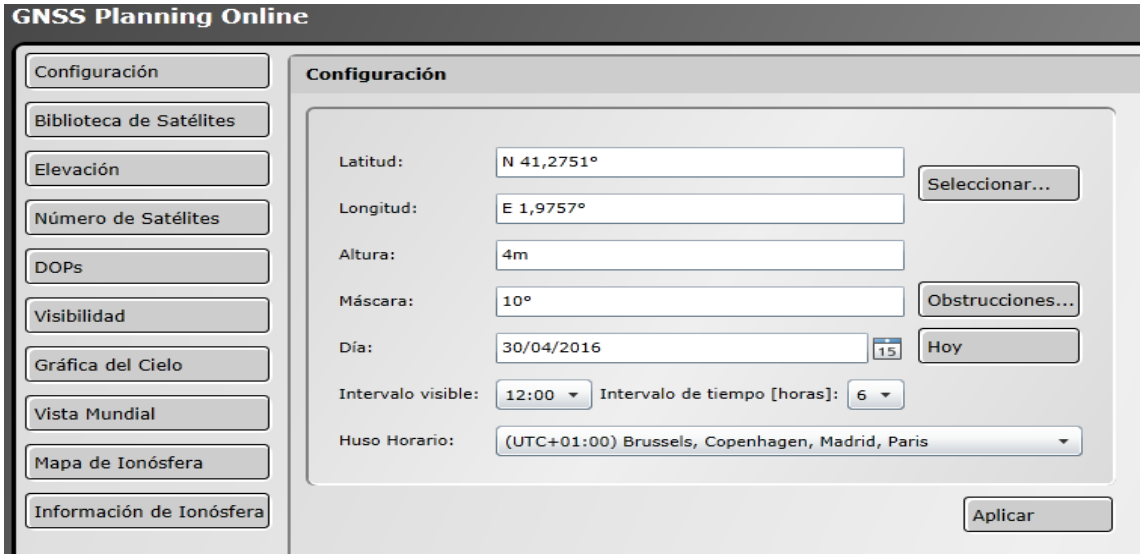
Another online resource for computing DOP and satellite visibility is:

http://www.trimble.com/GNSSPlanningOnline/

But this web page will ask you to install the (free) Microsoft Silverlight extension. For this reason http://gnssmissionplanning.com seems more friendly.

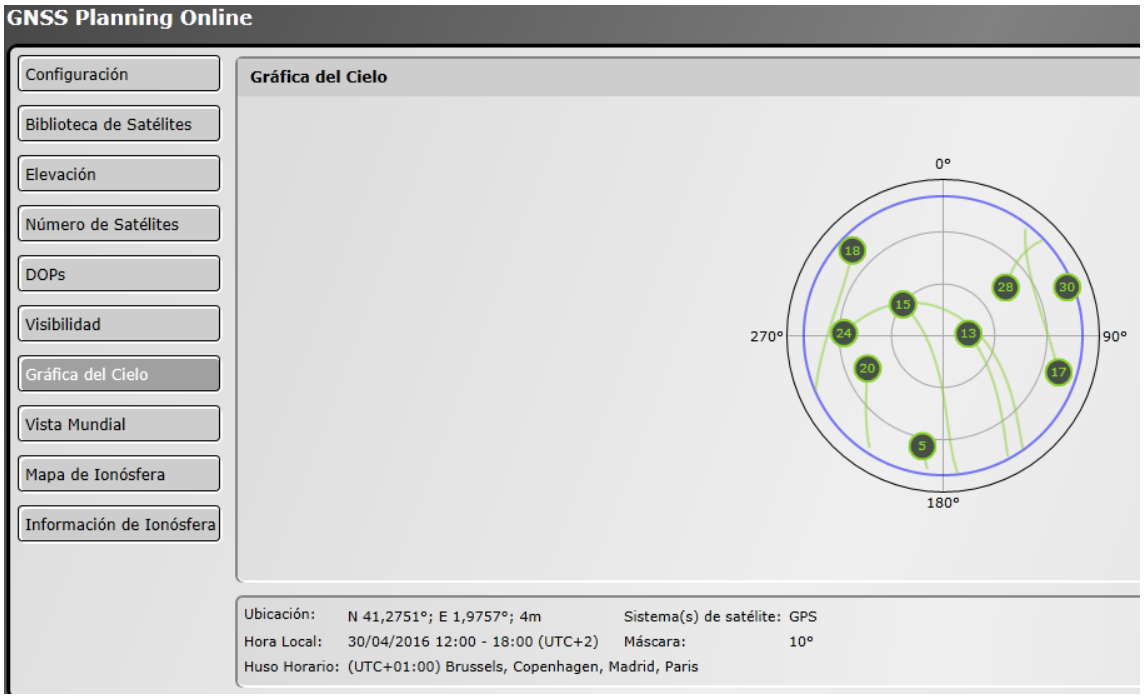
These instructions are for http://www.trimble.com/GNSSPlanningOnline/ but the use of http://gnssmissionplanning.com is quite similar.

First you must set up your location (EETAC, Castelldefels) at this screen (set the mask angle to 10º):

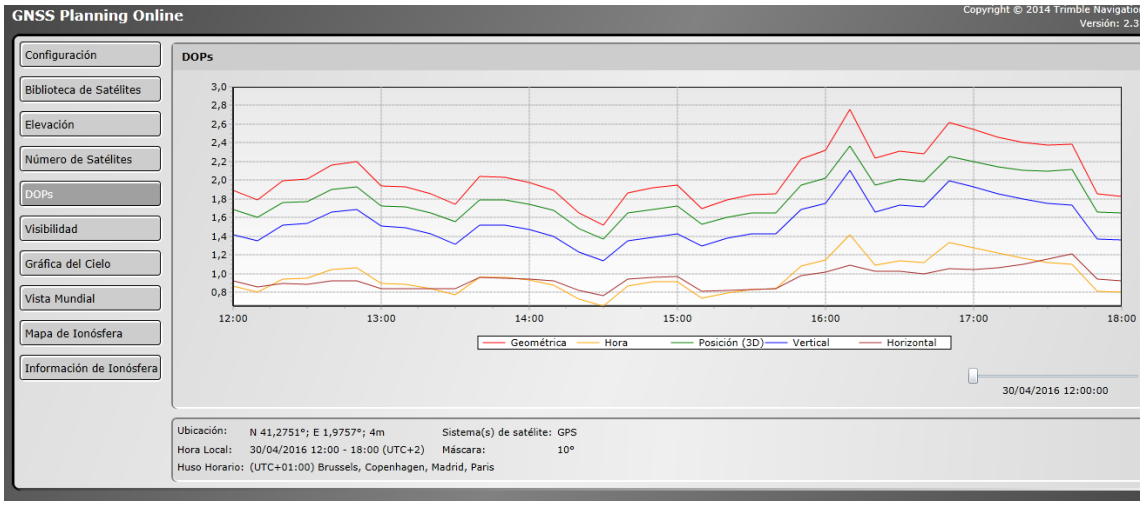


Then select the GPS satellite constellation only:

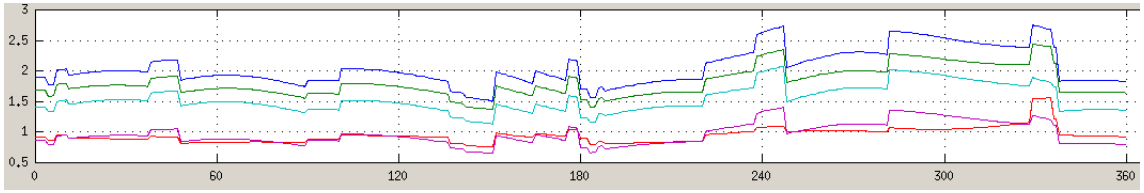
In the next screens you can check which GPS satellites are currently visible and their azimuth and elevation as seen from your current location:



This screen shows you the evolution of the different DOP values at your location using a time step of 10 minutes:



Notice that this tool computes the DOP with low time resolution. With your Matlab program you can compute the DOPs every minute:



In this way you can notice the stepwise changes in DOPs due to the appearance or disappearance of GPS satellites over the horizon.

You can download several applications for smart phones which give you a polar map showing the elevation and azimuth angles to visible GPS satellites and the actual values of DOP from your current position.

One of these applications for Android is the “GPS status and toolbox”. It can be downloaded from Google Play:

https://play.google.com/store/apps/details?id=com.eclipsim.gpsstatus2&hl=es

6 References

1. **U.S. Department of Homeland Security.** *GPS NANUS, ALMANACS, & OPS ADVISORIES.* [Online] http://www.navcen.uscg.gov/?pageName=gpsAlmanacs.

2. **Kleder, Michael.** *Convert earth-centered, earth-fixed (ECEF) coordinates to latitude, longitude, and altitude.* [Online] April 2006. http://www.mathworks.es/matlabcentral/fileexchange/7941-convert-cartesian-ecef-coordinates-to-lat-lon-alt/content/ecef2lla.m.

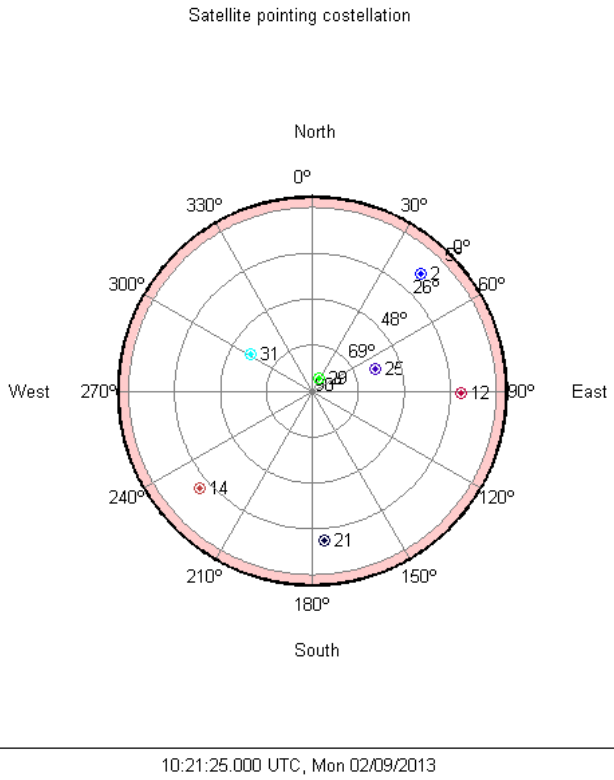
3. **Subirana, J. Sanz, Zornoza, J.M. Juan and Hernández-Pajares, M.** *Transformations between ECEF and ENU coordinates.* [Online] 2011. http://www.navipedia.net/index.php/Transformations_between_ECEF_and_ENU_coordinates.

4. **[Online] http://leapsecond.com/java/gpsclock.htm.**

5. U.S. Department of Homeland Security. ***Definition of YUMA almanac.*** **[Online] http://www.navcen.uscg.gov/?pageName=gpsYuma.**

7 ANNEX I: Available satellite mapping function

This Matlab function represents the available GPS satellites in a polar plane centered at the user position. The angle represents the azimuth and the radius represents the elevation. Both magnitudes are in degrees.



The function is called as follows:

```
viewSat(AZIMUTH, ELEV, ID, min_elev, max_elev, downstr)
```

The inputs are the following:

- AZIMUTH: Array with the values of azimuth of the available satellites (in degrees).
- ELEV: Array with the values of elevation of the available satellites (in degrees).
- ID: PRN sequence identifier of the available satellites.
- Min_elev: miminum elevation in degrees.
- Max_elev: maximum elevation in degrees.
- Downstr: text which is showed at the bottom of the figure.