VOL. VIII No 3 DECEMBER 2011

JMR

JOURNAL OF MARITIME RESEARCH

Spanish Society of Maritime Research

VOL. VIII No.3
DECEMBER 2011

JMR JOURNAL OF MARITIME RESEARCH

JMR

Journal of Maritime Research, Vol. VIII. No. 3, pp. 51-58, 2011

ISSN: 1697-4840

USE OF ROTATION MATRICES TO PLOT A CIRCLE OF EQUAL ALTITUDE

A. Ruiz[1]

Received 03 March 2011; in revised form 10 March 2011; accepted 27 October 2011

ABSTRACT

A direct method for obtaining the points of a circle of equal altitude using the vector analysis as an alternative to the spherical trigonometry is presented, and a solution where celestial navigation and Global Navigation Satellite Systems are complementary and coexist is proposed.

Key words: Circle of Equal Altitude, Celestial Navigation, Vector Analysis, Rotation matrix, ECS.

INTRODUCTION

In celestial navigation, historically, due to the restrictions of the tools used for sailing, (paper nautical charts and their scale, sight reduction tables for solving the navigational triangle ...), only an approximation of the segment of the whole circle of position –*CoP*– near the fix is plotted. Today, electronic chart systems permit to plot the entire CoP on a nautical chart or a map in any projection. The following presents a vector method to implement this utility.

Variables and symbols

	Variable	Intervals
GHA	Greenwich Hour Angle	0 <= GHA <= 360º (W to E)
Dec	Declination	–90º (S) <= Dec <= +90º (N)
Ho	Observed altitude	0 <= Ho <= 90º
B	Latitude	–90º (S) <= B <= +90º (N)
L	Longitude	–180º (W) <= L <= +180º (E)

[1] Industrial engineer, Navigational Algorithms, Email: navigationalalgorithms@gmail.com
San Sebastian, Spain.

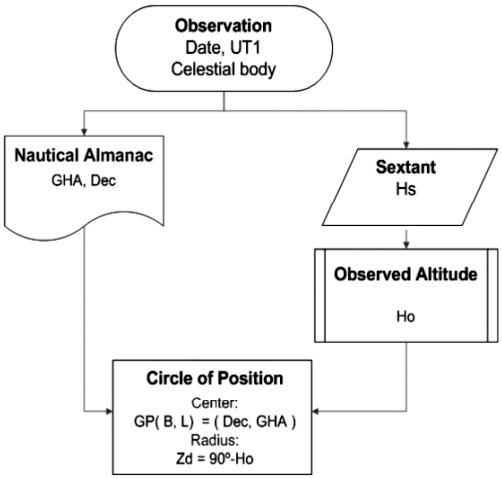

Figure 1. Circle of Equal Altitude parameters.

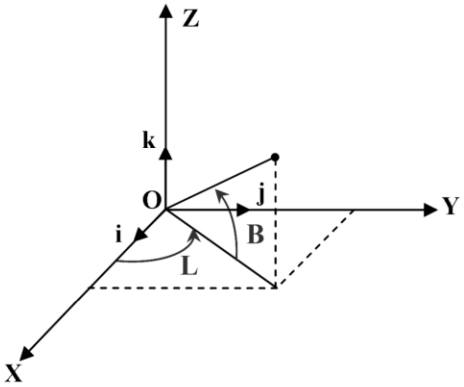

Figure 2. Coordinates and frame of reference.

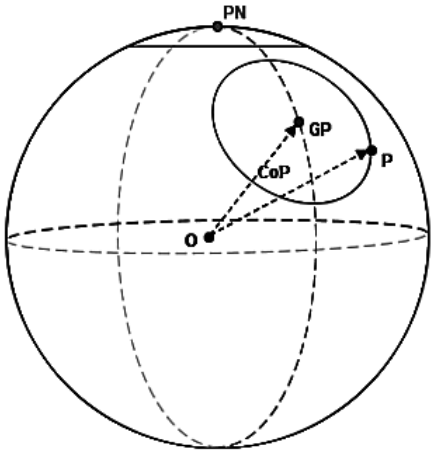

Figure 3. CoP with center at the GP, circle at the North Pole and vectors.

The Circle of Equal Altitude

Or circle of position in celestial navigation, is defined, Figure 1, by its geographical position –GP–, and the great circle distance from this point to the circle; the zenith distance of the body Zd (BOWDITCH, 1995. Ruiz, 2006 and 2008).

Coordinate Systems

To refer to the points on the surface of the Earth, a sphere of unit radius with its origin **O** in the centre of the Earth and a right-handed orthonormal basis $\{\vec{i}, \vec{j}, \vec{k}\}$ is defined as in Figure 2, constituting a ECEF –Earth Centered, Earth Fixed– reference frame. The axes of the defined Cartesian system of coordinates are:

- Z: from O to the North Pole.
- X: from O to the Greenwich meridian, included in the Earth's equatorial plane.
- Y: defined by $\vec{j} = \vec{k} \wedge \vec{i}$

The unit vector in Cartesian coordinates, (x,y,z), from the centre of the Earth to any point on the surface of the Earth with geographical coordinates (B, L,1), is:

$$\mathbf{OP} = \cos B \cos L \cdot \vec{i} + \cos B \sin L \cdot \vec{j} + \sin B \cdot \vec{k}$$

PLOTTING THE COP

For drawing a circle of position, the easy way is to place the circle with its center in the North Pole. In this fictitious position, all points of the circle has the coordinates (B,L) where B=Ho and L∈(-180,+180], Figure 3. Transforming these points to its true position on the spherical Earth, the coordinates of the points on the real CoP are obtained. *Note that only the real CoP satisfies the equation of the circle of equal altitude* (Ruiz, 2006):

$$\mathbf{OP} \bullet \mathbf{GP} = \cos(90^\circ - H_o)$$

Obtaining the points of the CoP

For each point on the Cop, its associated vector $\mathbf{OP}_i$ is obtained doing:

$$B = Ho \text{ and } L = (-180,+180]$$

The coordinates of the point are calculated by rotating this vector from its initial fictitious position to its real one using the equation:

$$\mathbf{OP} = \mathbf{R}_z(360^o - \mathrm{GHA})\mathbf{R}_y(90^o - Dec)\,\mathbf{OP}_i$$

Where the rotation matrices are (Brossard ,1994):

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \qquad \mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

These matrices represent counterclockwise rotations of a vector relative to ECEF frame, by an angle of θ. $\mathbf{R}_y$ rotates the z-axis towards the x-axis, and $\mathbf{R}_z$ rotates the x axis towards the y axis.

Transforming the Cartesian coordinates of **OP** = (x,y,z) into geographical ones, the latitude and the longitude of the point will be obtained.

$$B = ATAN2(z, \sqrt{x^2 + y^2})$$
$$L = ATAN2(y, x)$$

The algorithm is shown in a flow chart in appendix A1.

Transformation of the center of the CoP

Of course, by the rotation of the vector defined by the fictitious center of the circle, –the North Pole (B=90º,L=0,R=1): $\mathbf{OP}_N(0,0,1)$–, to the true one **OGP**, the geographical position is obtained.

CONCLUSIONS

A conceptually straightforward vector algorithm has been presented for plotting a celestial circle of position on an electronic chart. Is robust, fast and easy to implement, and the vector equation permits to map a circle of position into any projection (Mercator, gnomic, stereographic…).

Since the discovery of the line of position by Sumner (Ibáñez, et Al. 2004) in

1837, modern celestial navigation has been a valuable tool of the navigator, and now could be, or is (Kaplan, 1999), a reliable backup for a Global Navigation Satellite Systems (GNSS). An Electronic Chart Systems, ECS, including a celestial navigation module for sight reduction and running fix, with plotting capabilities like CoP and fix representation on an electronic nautical chart could be suitable for offshore navigation, a prototype is implemented with OpenCPN, appendix A3. In ship bridge design (Meck, et Al. 2009) celestial navigation could be another ingredient to take into account.

REFERENCES

BOWDITCH, N. (1995). The American Practical Navigator. Pub. Nº 9, DMA.

Brossard, J.P. (1994). Techniques de l'ingénieur - Mécanique générale, cinématique générale. INSA, Lyon.

Ibáñez Fernández, I.; Llombart Palet, J. and Iglesias Martín, M. A. (2004). "Transfer of nautical knowledge from U.S.A to Europe in the nineteenth-century: The case of the summer line position and its introduction into Spain". Journal of Maritime Research, Volume 1, Number 1, 63-79.

Kaplan, G.H. (1999). "New Technology for Celestial Navigation", Proceedings, Nautical Almanac Office Sesquicentennial Symposium, U.S. Naval Observatory, March 3–4, 1999, U.S. Naval Observatory, Washington, D.C., 239-254. Available from: http://gkaplan.us/content/NewTech.html

Meck, U.; Strohschneider, S. and Brüggemann, U. (2009). "Interaction Design in Ship Building: an Investigation Into the Integration of the User Perspective Into Ship Bridge Design". Journal of Maritime Research, Volume 6, Number 1, 15-32.

OpenCPN. Free software chart plotter and navigation software for use as an underway or planning tool. Available from: http://www.opencpn.org/

Ruiz González, A. (2006). Vectorial equation of the circle of equal altitude. Navigational Algorithms. Available from: http://sites.google.com/site/navigationalalgorithms/

Ruiz González, A. (2008). Vector Solution for the Intersection of Two Circles of Equal Altitude. The Journal of Navigation, Volume 61, Issue 02, 355-365.

APPENDIX

A1. Algorithm

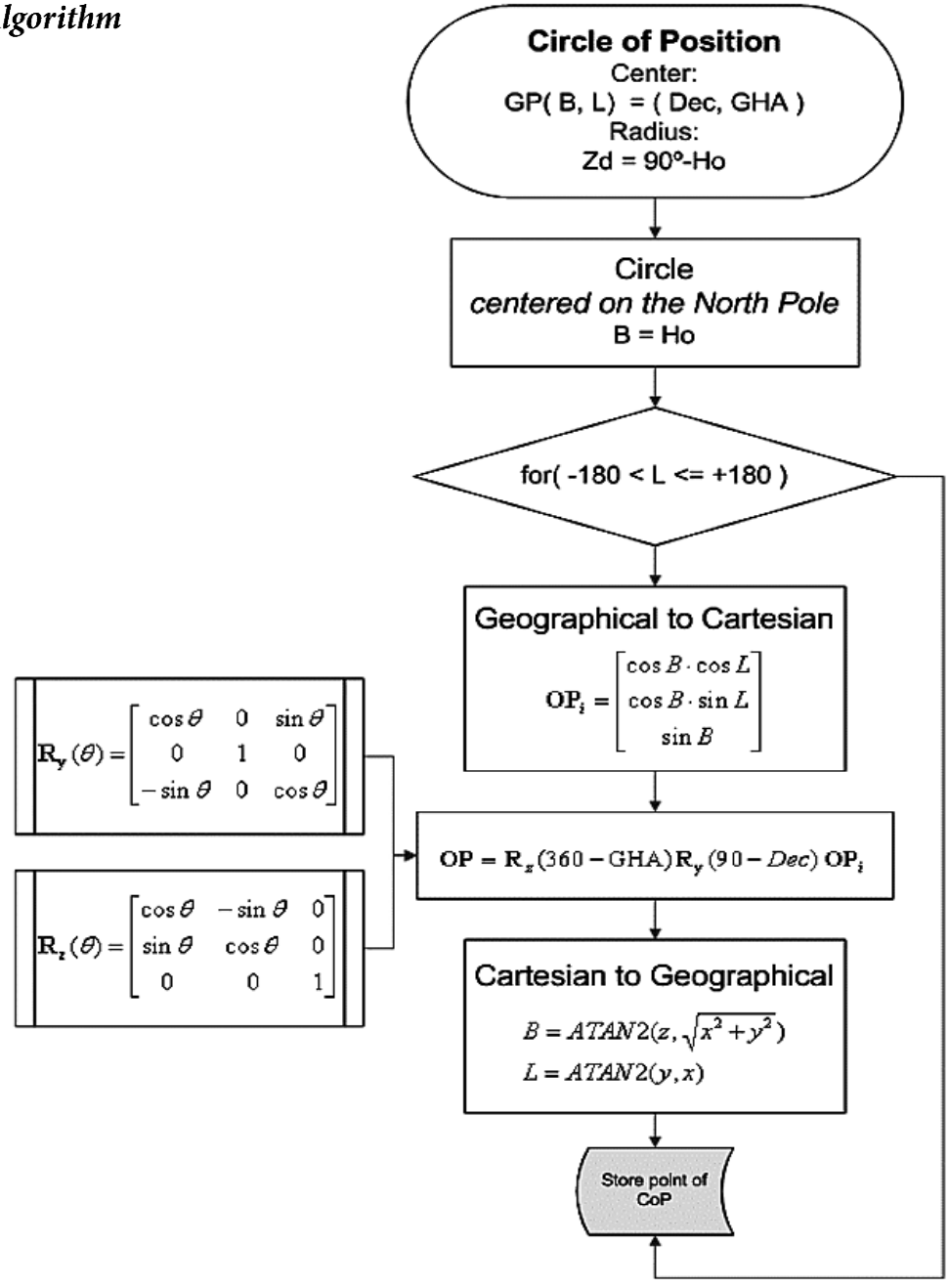

Figure 4. Points of a circle of equal altitude.

A2. Source code

Software and ANSI C source code for the algorithm is provided in an easy implementation, susceptible for being translated to other common programming languages. Available at the author's web site:

http://sites.google.com/site/navigationalalgorithms/

A3. Implementation in an ECS

This is an example of how celestial navigation serves to complement a GNSS: AstroNavigation.exe generates a *gpx* file as an input to OpenCPN.

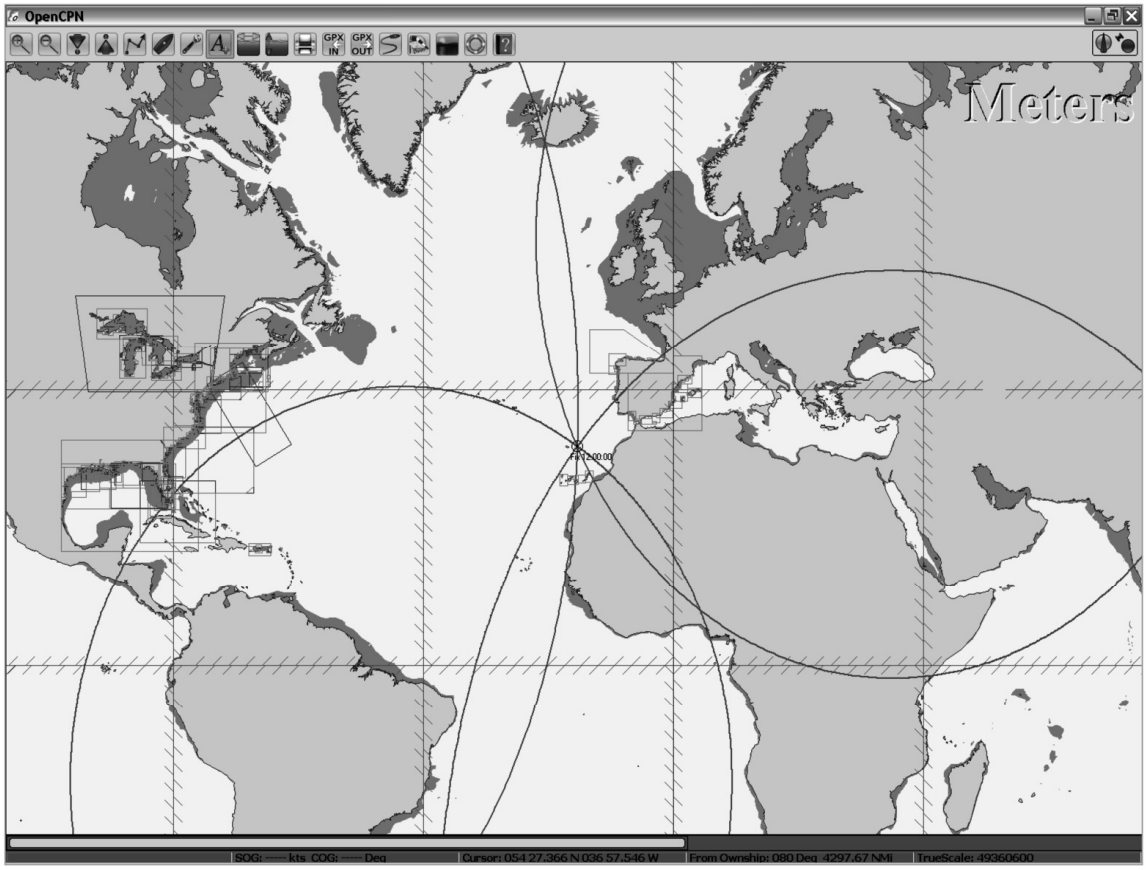

Figure 5. ECS and circles of equal altitude plotted on an electronic nautical chart. General scale.

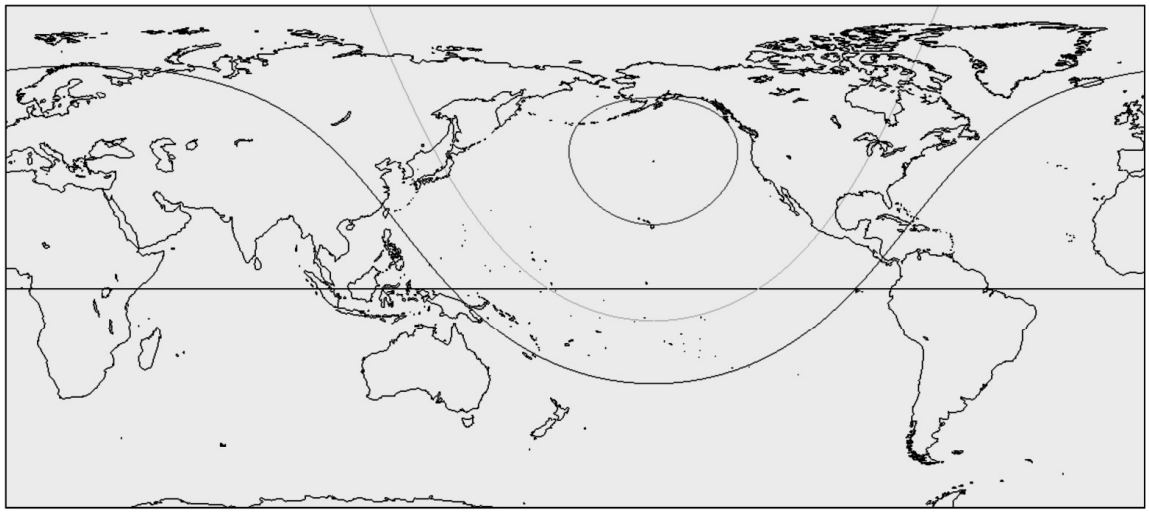

Figure 6 . Types of circles of equal altitude.

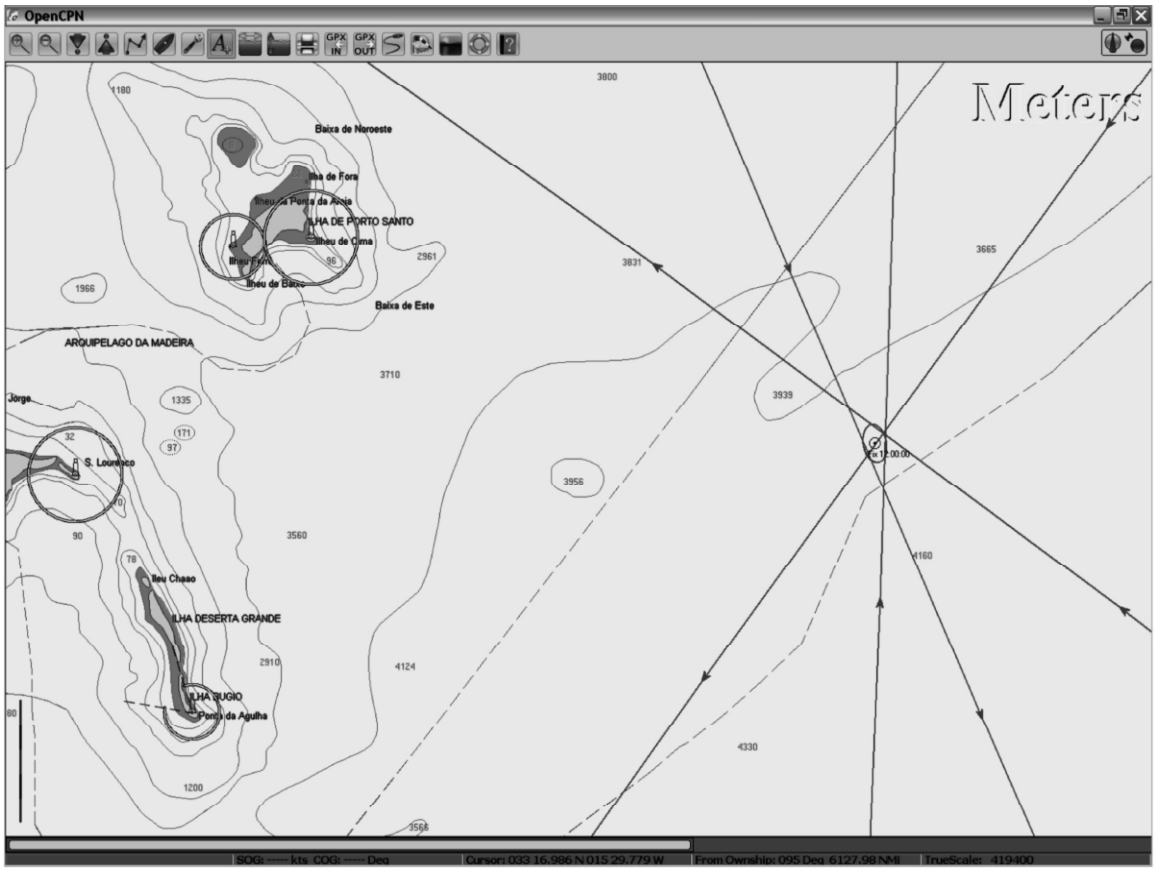

Figure 7 . ECS – Coastal scale: Fix and confidence ellipse.

```
/*
  FILE: drawCOP.h

  This file contains proprietary information of Andrés Ruiz Gonzalez
  Copying or reproduction without prior written approval is prohibited.
  Andrés Ruiz. San Sebastian - Donostia. Gipuzkoa
  Navigational Algorithms
  Copyright (c) 2006
*/



// Transformacion entre Cartesianas y Esfericas
void C2E( double x, double y, double z, double *B, double *L, double *R );
void E2C( double B, double L, double R, double *x, double *y, double *z );

// Matrices de Rotacion
void Rx( double a, double M[3][3] );
void Ry( double a, double M[3][3] );
void Rz( double a, double M[3][3] );


void MatrixVecProd( double A[3][3], double *v, double *res );


// GHA dec HO [º]
void CoPpoints( double GHA, double Dec, double Ho, double dL, WayPoint wpt[]  );
void CoPcenter( double GHA, double Dec );
```

```c
/*
  FILE: drawCOP.c

  Draw the equal altitude circle for an observation

  Centro = Polo de iluminación del astro: GP = (B, L) = ( Dec, GHA)
  Radio  = Distancia cenital: zd [nm] = 60*(90º - Ho)

  FUENTE: Use of rotation matrices to plot a circle of equal altitude

  STATUS: Finalizado
  TEST: OK
  PENDIENTE:

  This file contains proprietary information of Andrés Ruiz Gonzalez
  Copying or reproduction without prior written approval is prohibited.
  Andrés Ruiz. San Sebastian - Donostia. Gipuzkoa
  Navigational Algorithms
  Copyright (c) 2006 - 2010
*/


#include <math.h>
#include <stdio.h>

#include "..\Matematicas\angulos.h"



void C2E( double x, double y, double z, double *B, double *L, double *R )
{
  *R = sqrt( x*x+y*y+z*z);
  //*B = ASIN(z/(*R));
  *B = ATAN2( z, sqrt(x*x+y*y) );
  *L = ATAN2( y, x );
}


void E2C( double B, double L, double R, double *x, double *y, double *z )
{
  *x = R*COS(B)*COS(L);
  *y = R*COS(B)*SIN(L);
  *z = R*SIN(B);
}


void Rx( double a, double M[3][3] )
{
  M[0][0] = 1.0;
  M[1][0] = 0.0;
  M[2][0] = 0.0;
  M[0][1] = 0.0;
  M[1][1] = COS( a );
  M[2][1] = SIN( a );
  M[0][2] = 0.0;
  M[1][2] = -SIN( a );
  M[2][2] = COS( a );
}


void Ry( double a, double M[3][3] )
{
  M[0][0] = COS( a );
  M[1][0] = 0.0;
  M[2][0] = -SIN( a );
  M[0][1] = 0.0;
  M[1][1] = 1.0;
  M[2][1] = 0.0;
  M[0][2] = SIN( a );
  M[1][2] = 0.0;
  M[2][2] = COS( a );
}
```

```
void Rz( double a, double M[3][3] )
{
  M[0][0] = COS( a );
  M[1][0] = SIN( a );
  M[2][0] = 0.0;
  M[0][1] = -SIN( a );
  M[1][1] = COS( a );
  M[2][1] = 0.0;
  M[0][2] = 0.0;
  M[1][2] = 0.0;
  M[2][2] = 1.0;
}



/*
  MatrixVecProd computes the matrix product between a matrix and a vector of the dimension 3.
  The result is found in res.
*/
void MatrixVecProd( double A[3][3], double *v, double *res )
{
  int i, j;
  const int n = 3;

  for( i=0; i<n; i++ ) {
    res[i] = 0.0;
    for( j=0; j<n; j++ )
    res[i] += A[i][j]*v[j];
  }
}




void CoPpoints( double GHA, double Dec, double Ho, double dL, WayPoint wpt[]  ) // GHA dec HO [º]
{
  double B, L;
  double B0, L0;

  double vv[3], vy[3], vyz[3];
  double My[3][3], Mz[3][3];
  double RR;

  int i;

  B0 = Ho;

  // conversion entre coordenadas esfericas y (Dec, GHA)
  // GHA: 0<=GHA<=360º de W a E
  Rz( 360.0-GHA, Mz );
  Ry( 90.0-Dec, My );

  i = 0;

  for( L0 = -180.0;  L0 <= +180.0;  L0 += dL )
    {
    // 1. Coordenadas esfericas a cartesianas
    E2C( B0, L0, 1, &vv[0], &vv[1], &vv[2] );

    // 2. rotación alrededor del eje Y
    MatrixVecProd( My, vv, vy );

    // 3. rotación alrededor del eje Z (PN-PS)
    MatrixVecProd( Mz, vy, vyz );

    // 4. Coordenadas cartesianas a esfericas
    C2E( vyz[0], vyz[1], vyz[2], &B, &L, &RR );

    wpt[i].B = B;
    wpt[i].B = L;
    i++;
    }
}
```

```
void CoPcenter( double GHA, double Dec ) // GHA dec [º]
{
  double B, L;
  double vv[3], vy[3], vyz[3];
  double My[3][3], Mz[3][3];
  double RR;

  // conversion entre coordenadas esfericas y (Dec, GHA)
  // GHA: 0<=GHA<=360º de W a E
  Rz( 360.0-GHA, Mz );
  Ry( 90.0-Dec, My );

  // 1. Coordenadas esfericas a cartesianas PN(90,0,1)
  E2C( 90.0, 0.0, 1.0, &vv[0], &vv[1], &vv[2] );

  // 2. rotación alrededor del eje Y
  MatrixVecProd( My, vv, vy );

  // 3. rotación alrededor del eje Z (PN-PS)
  MatrixVecProd( Mz, vy, vyz );

  // 4. Coordenadas cartesianas a esfericas
  C2E( vyz[0], vyz[1], vyz[2], &B, &L, &RR );

  // Centro: (B, L) = (Dec, GHA)
}
```