Coordinate conversion from spherical to cartesian

Javier Areta
Univ. of Connecticut, ECE Dept.
Storrs, CT 06269-2157
areta@engr.uconn.edu

March 5, 2008

Notation

In general, **cartesian** coordinate vectors will be conformed by $[X\ Y\ Z]$ coordinates, in this exact order.

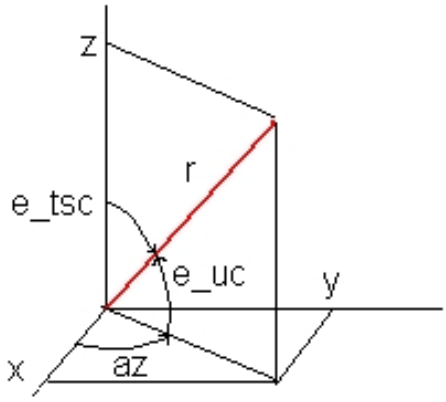
As for **Spherical** vectors, the order will be [*Range Azimuth Elevation*] ordering.
The Azimuth is measured from the X coordinate, counterclockwise.
For the Elevation there are two possibilities, one with angles measured from the Z axis (which we'll deem e_tsc), and one measured from the $X-Y$ plane, deemed e_uc. To convert from one to the other we will use

$$\text{e\_tsc} = 90 - \text{e\_uc} \tag{1}$$

if the angles are expressed in degrees and

$$\text{e\_tsc} = \pi/2 - \text{e\_uc} \tag{2}$$

if they are expressed in radians (this latter case is the most common, as usually C and matlab trigonometric functions work in radians)



The conversion for those coordinates is based on trigonometric functions:

$$\begin{array}{lcllcllcl} r & = & \sqrt{X^2+Y^2+Z^2} & X & = & r\ sin(e\_tsc)cos(az) & = & r\ cos(e\_uc)cos(az) \\ az & = & atan(Y/X) & Y & = & r\ sin(e\_tsc)sin(az) & = & r\ cos(e\_uc)sin(az) \\ e\_tsc & = & acos(Z/r) & Z & = & r\ cos(e\_tsc) & = & r\ sin(e\_uc) \end{array}$$

When noise is present in the spherical coordinates, it turns out that the resulting cartesian converted measurements are biased. To deal with this, the measurements can be scaled by a multiplicative de-biasing term which depends on the standard deviation of the noises (considered to be gaussian). Also the correlation matrix obtained should account for this. The following matlab functions deal with this problem [1]

```
function P = cart2sph(X)
 % Convert from cartesian to spherical coordinates, return angles in degrees.
r=180/pi;
for k=1:length(X)
    P(1,k) = sqrt(sum(X(:,k).^2));   % range
    P(2,k) = r*atan2(X(2,k),X(1,k)); % azimuth   - measured from x axis
    P(3,k) = r*acos(X(3,k)/P(1,k));  % elevation - measured from z axis
end

 %---------------------------------------------------------------------------

function P=sph2cart(T,sr,sb,se);
 % Convert a matrix of spherical coordinates [range azimuth elevation]
 % to cartesian coordinates [x y z] using:
 % x = range * cos(elevation) * cos(azimuth)
 % y = range * cos(elevation) * sin(azimuth)
 % z = range * sin(elevation)
 % That is, elevation measured from x-y plane and azimuth measured from x axis.
 % Angles measured in degrees, standard deviations in radians.

 le_1=exp(se^2/2);
 lB_1=exp(sb^2/2);

 P=zeros(size(T));
 P(:,1)=le_1* lB_1* T(:,1).* cos(T(:,3)*pi/180).* cos(T(:,2)*pi/180);
 P(:,2)=le_1* lB_1* T(:,1).* cos(T(:,3)*pi/180).* sin(T(:,2)*pi/180);
 P(:,3)=le_1*       T(:,1).* sin(T(:,3)*pi/180);

 %---------------------------------------------------------------------------

function R=cov(r,B,e,sr,sb,se)
 % Unbiased spherical to cartesian covariance matrix. Angles measured in
 % degrees, standard deviations in radians.
 % r - sr range and sigma range
 % B - sb azimuth and sigma azimuth
 % e - se elevation and sigma elevation

 le  = exp(-se^2/2);
 lep = exp(-2*se^2);
```

[1] taken from "Unbiased Converted Measurements for Tracking" Longbin, Bar-Shalom and others.

```
lB  = exp(-sb^2/2);
lBp = exp(-2*sb^2);

B = B*pi/180;
e = e*pi/180;

R(1,1)=1/4*lB^(-2)*le^(-2)*(r^2+2*sr^2)*(1+lBp^2*cos(2*B))*...
   (1+lep^2*cos(2*e))-1/4*(r^2+sr^2)*(1+lBp*cos(2*B))*(1+lep*cos(2*e));
R(2,2)=1/4*lB^(-2)*le^(-2)*(r^2+2*sr^2)*(1-lBp^2*cos(2*B))*...
   (1+lep^2*cos(2*e))-1/4*(r^2+sr^2)*(1-lBp*cos(2*B))*(1+lep*cos(2*e));
R(3,3)=1/2*        le^(-2)*(r^2+2*sr^2)*                  ...
   (1-lep^2*cos(2*e))-1/2*(r^2+sr^2)*                 (1-lep*cos(2*e));
R(1,2)=1/4*lB^(-2)*le^(-2)*lBp^2*(r^2+2*sr^2)*    sin(2*B)*...
   (1+lep^2*cos(2*e))-1/4*(r^2+sr^2)*   lBp*sin(2*B) *(1+lep*cos(2*e));
R(1,3)=1/2*lB     *le^(-2)*(r^2+2*sr^2)*            cos(B)*...
   lep^2*sin(2*e)-1/2*(r^2+sr^2)*   lB *cos(B)   *   lep*sin(2*e);
R(2,3)=1/2*lB     *le^(-2)*(r^2+2*sr^2)*            sin(B)*...
   lep^2*sin(2*e)-1/2*(r^2+sr^2)*   lB *sin(B)   *   lep*sin(2*e);
R(2,1)=R(1,2);
R(3,1)=R(1,3);
R(3,2)=R(2,3);
```