Convert from LAMS 3-beam measurements to relative wind

The routine below shows sample calculations implementing the outline described in LAMSprocessing.pdf . That note is on the RAF science wiki.

Input needed:
  1. Measured wind in each of three beams of LAMS -- VARIABLE
  2. Theta, Phi: angles describing orientation of the three beams -- FIXED
  3. RRH, RRP, RRR: rotation rates for heading, pitch, roll -- VARIABLE
  4. LL: vector coordinates of LAMS location relative to aircraft IRS -- FIXED
  5. DR, DP, DH: differences in roll, pitch, and heading between LAMS and IRS -- VARIABLE
Output produced:
  Relative wind vector RW = [u, v, w]

The following initializes, sets the parameters, and calculates the rectilinear relative wind in the LAMS reference frame from the measured wind in the three beams. For testing, that wind is calculated from an assumed rectilinear-coordinate relative wind; for eventual use, that wind (here V) should be eliminated and the routine should use

In [7]:
```
"""
This routine, designed to run via 'ipython --pylab ProcB.py', calculates
the relative wind in the reference frame of the IRS from the measurements
provided by the three beams of the LAMS. It needs constants specifying the
beam angles (Theta and Phi), the focal distance of each beam (L), and the
distances from the IRS to LAMS (LL). It also needs measurements of the
rotation rates of the aircraft in roll, pitch, and heading [RRR, RRP, RRH]
and the differences in attitude angles between the IRS and CMIGIT
(DH, DP, DR).
"""

import numpy as np
from math import *
from pylab import *

Theta = np.array([35.,35.,35.]) * np.pi / 180. # values in radians
Phi = np.array([180.,-60.,60.]) * np.pi / 100. #  "   "

# unit vectors along beams are then:
# a[i] = [cos(Theta[i]), -sin(Theta[i])*sin(Phi[i]), sin(Theta[i])*cos(Phi[i])]
# and the dot products with the (i,j,k) unit vectors give:
S = np.array([cos(Theta[0]), -sin(Theta[0])*sin(Phi[0]), sin(Theta[0])*cos(Phi[0])
              cos(Theta[1]), -sin(Theta[1])*sin(Phi[1]), sin(Theta[1])*cos(Phi[1])
              cos(Theta[2]), -sin(Theta[2])*sin(Phi[2]), sin(Theta[2])*cos(Phi[2])
S.shape = (3,3)
Si = linalg.inv(S)  # calculate the inverse of S

# example vector wind
V = np.array([100.,1.,5.])
# find measurements in each of three beams
# for real data processing, input measurements from the three beams here as A
A = np.dot(S,V)
# use inverse to find rectilinear wind in the LAMS reference frame
U = np.dot(Si,A)

print("S  = ",S)
print("Si = ",Si)
print("V = ",V)
print("A = ",A)
print("U = ",U)
```

```
('S  = ', array([[ 0.81915204,  0.33713977,  0.46403308],
       [ 0.81915204,  0.54550361, -0.17724487],
       [ 0.81915204, -0.54550361, -0.17724487]]))
('Si = ', array([[ 0.3374138 ,  0.3374138 ,  0.54594699],
       [-0.        ,  0.91658422, -0.91658422],
       [ 1.55938622, -1.26156995, -0.29781627]]))
('V = ', array([ 100.,    1.,    5.]))
('A = ', array([ 84.57250962,  81.5744837 ,  80.48347649]))
('U = ', array([ 100.,    1.,    5.]))
```

The result is the relative wind in the LAMS coordinate system. The next step is to correct that wind for the false wind produced by rotation of the aircraft. Here, a rotation rate in heading, pitch, and roll is assumed by defining the variables RRH, RRP, RRR; these will be replaced in processing by the actual rotation rates of the aircraft, obtained by one-step differentiation of the measured angles. I think the best angles to use for this purpose are the IRS angles, not the CMIGIT angles, because the motion that matters is that of LAMS relative to the IRS. For example, vibrating changes in heading don't change the false contribution to the wind in the same way as rotation

In [8]:
```
# rotation angles:
#  heading (THDG) typical high rate is 2 deg / s
#  roll:          typical high rate is 4 deg / s
#  pitch:         typical high rate 0.25 deg / s
RRH = 2.; RRP = 0.25; RRR = 4.
RR = np.array([RRR, RRP, RRH])
Omega = RR * np.pi/180.
# find coordinates of the three sensing volumes relative to the IRS:
# first, coordinates of LAMS in IRS reference frame:
# (TEMPORARY **** GUESS)
LL = np.array([-5.,-8.,+1.])
F = np.cross(Omega,LL)  # the cross product of two vectors
print("Omega x R = ",F)
vf = np.dot(Si,A)-F
print("LAMS-system V with rotation is ",vf)
```

```
('Omega x R = ', array([ 0.283616  , -0.2443461 , -0.53668874]))
('LAMS-system V with rotation is ', array([ 99.716384  ,   1.2443461 ,
5.53668874]))
```

Finally, the measurements need to be transformed back to the reference frame of the IRS by rotating through three angles representing the differences in heading, pitch, and roll between the IRS and LAMS

In [9]:
```
# then transform back to IRS coordinate system.
# Transform by these angles: DR, DP, DH where they are the respective
# differences in roll, pitch, and heading with sign (IRS-CMIGITS).
# Test values; replace these with measurements for data processing
DR = 1.*np.pi/180.  # roll, IRS - CMIGIT
DP = 1.*np.pi/180.  # pitch
DH = 1.*np.pi/180.  # heading
# The rotations are given in Bulletin 23:
T1 = np.array([(1.,0.,0.),(0.,cos(DR),-sin(DR)),(0.,sin(DR),cos(DR))])  # rotatio
T2 = np.array([(cos(DP),0.,sin(DP)),(0.,1.,0.),(-sin(DP),0.,cos(DP))])  # rotatio
T3 = np.array([(cos(DH),-sin(DH),0.),(sin(DH),cos(DH),0.),(0.,0.,1.)])  # rotatio
RW = np.dot(T3, np.dot(T2, np.dot(T1, vf)))
print("Final relative wind is ",RW)
```

```
('Final relative wind is ', array([ 99.76296267,   2.88907182,   3.81642501]))
```

The output shows
- S: the direction cosine matrix S
- Si: the inverse of this matrix Si
- V: an assumed relative wind V
- A: the calculated beam components A from A=S V
- U: the relative wind U retrieved from A via Si
- Omega x R: the correction for an assumed rotation speed
- LAMS-V: the LAMS-coordinate-system relative wind corrected for rotation
- Final wind: the result after an additional translation to an assumed IRS system