```python
#!/usr/bin/env python

from numpy import *

def ecef2lla(pVec):
#+==============================================================================+
# ecef2lla : Convert from a position vector in the Earth-centered, Earth-fixed
#              (ECEF) reference frame to latitude, longitude, and altitude
#              (geodetic with respect to the WGS-84 ellipsoid).
#
#
# INPUTS
#
# pVec ---- 3-by-1 position coordinate vector in the ECEF reference frame,
# in meters.
#
# OUTPUTS
#
# lat ----- latitude in radians
#
# lon ----- longitude in radians
#
# alt ----- altitude (height) in meters above the ellipsoid
#
#    1:  Fundamentals of inertial Navigation, Satellite-based Positioning and their Integration
#
# Caveats: Only good for abs(lat) > .0001 ## more clarification
#
#
#
#+------------------------------------------------------------------------------+
# References:
#    1:  Fundamentals of inertial Navigation, Satellite-based Positioning and their Integration
#        Noureldin, A; Karamat, T.B.; Gregory, J.
#        2013, XVIII, 314p. Hardcover
#        ISBN: 978-3-642-30465-1
#        Site: http://www.springer.com/978-3-642-30465-1 ### Needs clarification ##
#
# Author: Caleb North
#+==============================================================================+

    x = pVec[0,0]
    y = pVec[1,0]
    z = pVec[2,0]


    # WGS84 Values #
    a = 6378137.0            ## Semimajor axis (equatorial radius)
    f = 1/298.257223563        ## Flattening
    b = a*(1-f)                ## Semiminor axis
    e = sqrt(f*(2-f))        ## Eccentricity
    E = sqrt(a**2/b**2 -1)    ##

    p = sqrt(x**2 + y**2)
    theta = arctan(z*a/(p*b))

    # Using closed form solution #
    lat = arctan((z+E**2*b*sin(theta)**3)/(p-e**2*a*cos(theta)**3))
    lon = arctan2(y,x)
    N = a/sqrt(1-e**2*sin(lat)**2)    ## Normal Radius
    alt = p/cos(lat) - N

    ## accounting for numerical instabilities ##
    if p  < 1 and z > 0: alt =  z - b
    if p  < 1 and z < 0: alt = -z - b
    if y == 0 and x > 0: lon = 0
    if y == 0 and x < 0: lon = pi
    if p == 0 and z > 0: lat =  pi/2; lon = 0; alt =  z-b
    if p == 0 and z < 0: lat = -pi/2; lon = 0; alt = -z-b


    return (lat, lon, alt)
```