```
r""" This module is designed to determine the location of a satellite
given radar information

Inputs Vary

Functions:
    topo2rv
    lla2ecef
    sez2ecef
    ecef2eci

Author: Thomas J Susi       GWU         tsusi13@gwu.edu
"""
import numpy as np
import pdb
from astro import time
from kinematics import attitude

def topo2rv(rang, azm, elev, rang_r, azm_r, elev_r):
    r"""Computes the updated COEs given the time change

    Inputs:
    rang :(scalar) Given in units of km
    azm :(angle) Given in units of radians
    elev :(angle) Given in units of radians
    rang_r :(scalar) Given in units of km/s
    azm_r :(angle rate) Given in units of radians/s
    elev_r :(angle rate) Given in units of radians/s

    Outputs:
    pr_sez :(vector) Given in units of km in SEZ frame
    pv_sez :(vector) Given in units of km/s in SEZ frame

    Author: Thomas J Susi       GWU         tsusi13@gwu.edu
    """

    a1 = -rang * np.cos(elev) * np.cos(azm)
    a2 = rang * np.cos(elev) * np.sin(azm)
    a3 = rang * np.sin(elev)

    pr_sez = np.array([[a1], [a2], [a3]])

    b1 = (-rang_r * np.cos(elev) * np.cos(azm)) + (rang * elev_r *
np.sin(elev) * np.cos(azm)) + (rang * azm_r * np.cos(elev) *
np.sin(azm))
    b2 = (rang_r * np.cos(elev) * np.sin(azm)) + (-rang * elev_r *
np.sin(elev) * np.sin(azm)) + (rang * azm_r * np.cos(elev) *
np.cos(azm))
    b3 = (rang_r * np.sin(elev)) + (rang * elev_r * np.cos(elev))
```

```
    pv_sez = np.array([[b1], [b2], [b3]])


    return pr_sez, pv_sez

def lla2ecef(lat, lon, alt):
    r"""Computes the updated COEs given the time change

    Inputs:
    lat :(angle) Given in units of radians
    lon :(angle) Given in units of radians
    alt :(scalar) Given in units of km

    Outputs:
    r_ecef :(vector) Given in units of km in ECEF frame

    Author: Thomas J Susi       GWU         tsusi13@gwu.edu
    """

    N = (6378.137) / (np.sqrt(1 - ((0.08182**2)*np.sin(lat)**2)))

    a1 = (N + alt) * np.cos(lat) * np.cos(lon)
    a2 = (N + alt) * np.cos(lat) * np.sin(lon)
    a3 = (((1 - 0.08182**2)*N) + alt) * np.sin(lat)

    r_v_ecef = np.array([[a1], [a2], [a3]])

    return r_v_ecef

def sez2ecef(lat, lon, alt):
    r"""Computes the updated COEs given the time change

    Inputs:
    lat :(angle) Given in units of radians
    lon :(angle) Given in units of radians
    alt :(scalar) Given in units of km

    Outputs:
    sez2ecef :(Rotation Matrix) Unitless

    Author: Thomas J Susi       GWU         tsusi13@gwu.edu
    """

    b2ecef = np.array([[np.cos(lon),np.sin(lon),0],[-
np.sin(lon),np.cos(lon),0],[0,0,1]])

    sez2b = np.array([[np.cos(np.pi/2 - lat),0,-np.sin(np.pi/2 -
lat)],[0,1,0],[np.sin(np.pi/2 - lat),0,np.cos(np.pi/2 - lat)]])

    return sez2b, b2ecef
```

```
def ecef2eci(JD, lon):
    r"""Computes the updated COEs given the time change

    Inputs:
    JD :(time) Julian Date

    Outputs:
    ecef2eci :(Rotation Matrix) Unitless

    Author: Thomas J Susi       GWU         tsusi13@gwu.edu
    """

    gst, lst = time.gstlst(JD, lon)

    thetag = gst

    ecef2eci = np.array([[np.cos(thetag),-np.sin(thetag),0],
[np.sin(thetag),np.cos(thetag),0],[0,0,1]])

    return ecef2eci
```