Transformacja współrzędnych elipsoidalnych samolotu do układu lokalnego

Autor: Julia Zapała
Indeks: 325710
Numer: 25

1. Cel zadania

Celem zadania jest transformacja współrzędnych elipsoidalnych samolotu do układu lokalnego, na podstawie danych do zadania (znacznik czasu, UTC, znak wywoławczy, pozycja, wysokość, prędkość i kierunek). Współrzędne należy przeliczyć do geocentrycznych współrzędnych ortokartezjańskich, a ostatecznie do układu współrzędnych horyzontalnych.

2. Wstęp teoretyczny

Często podczas rozwiązywania zagadnień geodezyjnych ze względu na różne potrzeby pomiarów i obliczeń konieczna jest transformacja między układami współrzędnych. Każdy układ współrzędnych ma swoje własne cechy i zastosowania. Najpopularniejszymi układami są: układ współrzędnych elipsoidalnych (φ, λ i h) i układ współrzędnych ortokartezjańskich (X, Y, Z).

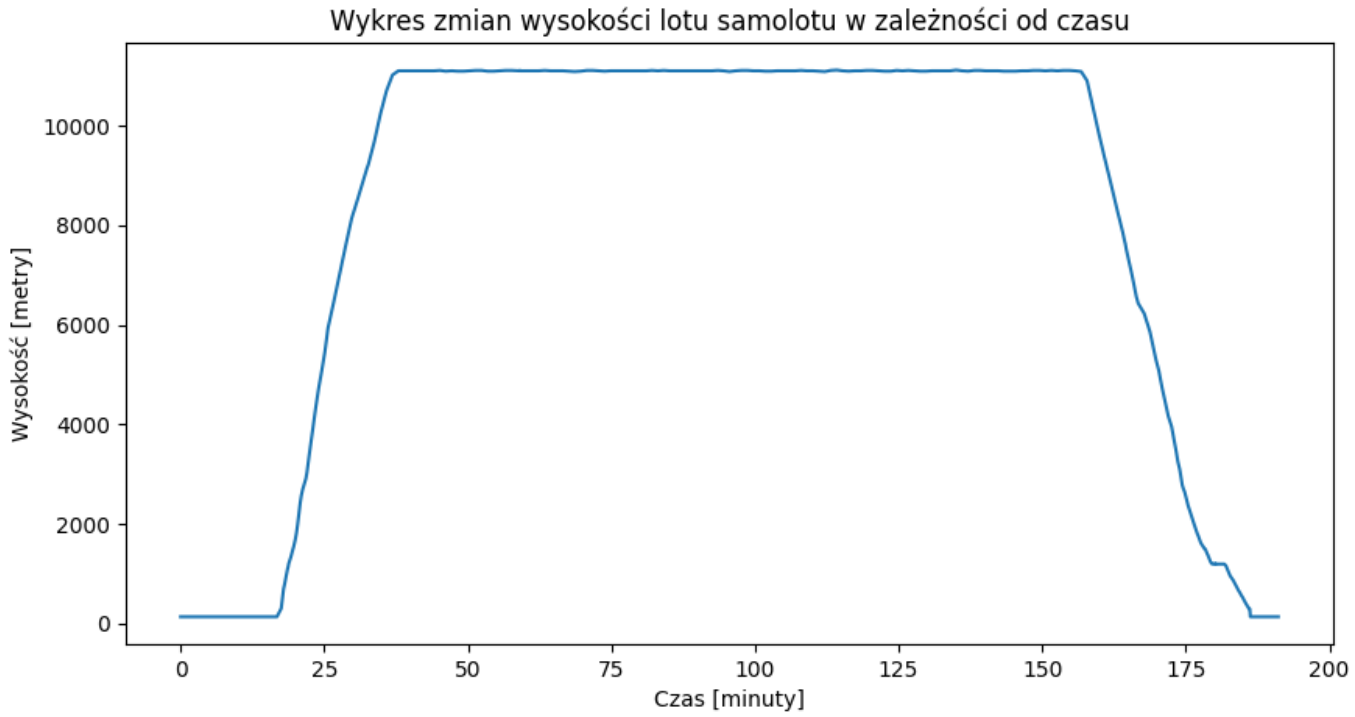
3. Realizacja zadania

Zadanie zostało wykonane za pomocą programu napisanego w python. Na początku za pomocą dostarczonej z treścią zadania funkcji odczytywane są dane z pliku lot25.csv, tworzone są odpowiednie tablice, a następnie przeliczane są współrzędne lotniska i samolotu do współrzędnych ortokartezjańskich (X, Y, Z). Obliczana jest również macierz obrotu dla danego punktu. Dla każdego położenia samolotu obliczane są współrzędne geocentryczne samolotu i azymut. Powyższe obliczenia służą stworzeniu mapy przedstawiającej trasę samolotu, z zaznaczeniem jego widoczności z lotniska startu. Na mapie zaznaczona jest również linia geodezyjna pomiędzy danymi lotniskami. Obliczenia są również wykorzystywane do stworzenia wykresów: prędkości samolotu, wysokości i odległości od lotniska.

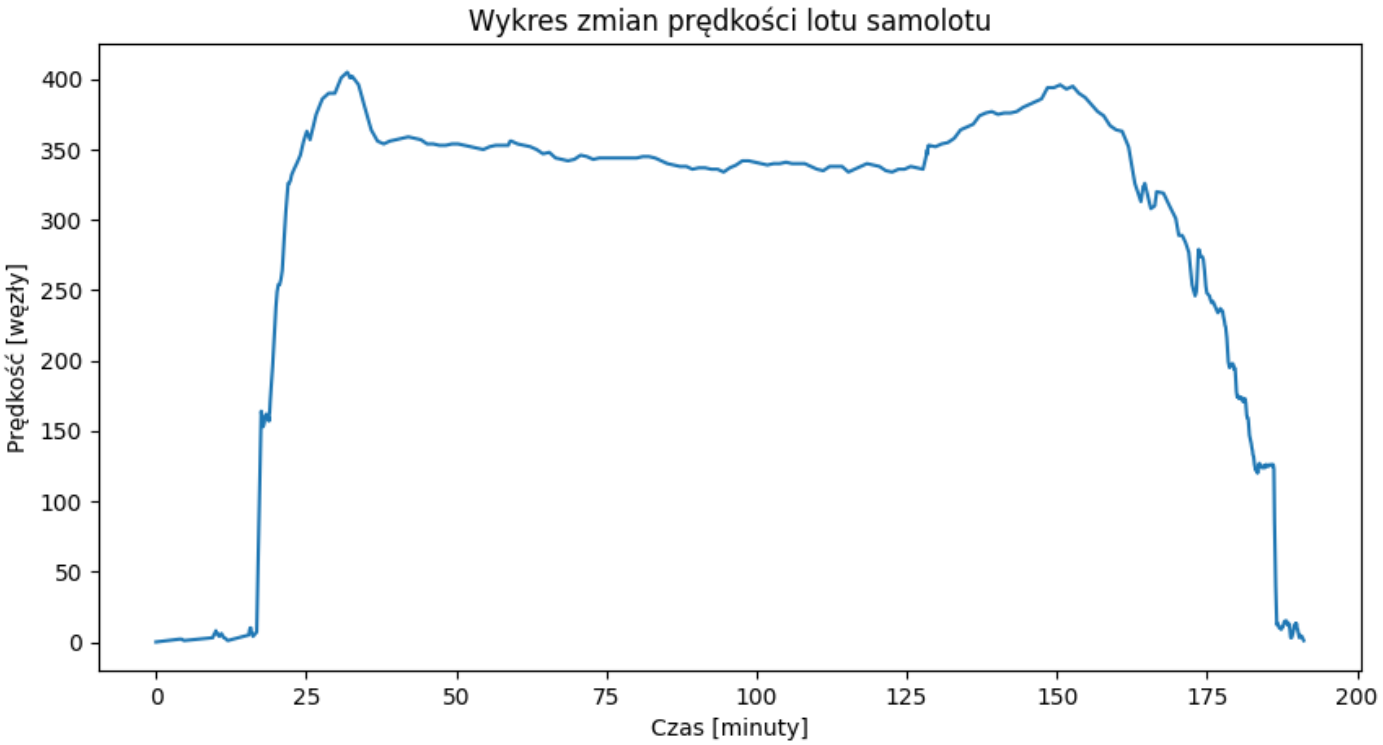
4. Wizualizacje i wnioski

Wizualizacja przedstawia mapę, na której została przedstawiona trasa samolotu. Kolorem niebieskim zaznaczony jest moment, kiedy samolot jest widoczny z lotniska startu, a kolorem czerwonym kiedy znika poniżej horyzontu. Pomarańczowa linia to linia geodezyjna, czyli najkrótsza droga pomiędzy dwoma punktami. Na podstawie wizualizacji można stwierdzić, że po starcie samolot dosyć szybko znikna za horyzontem oraz że trasa lotu nie pokrywała się z linią geodezyjną.

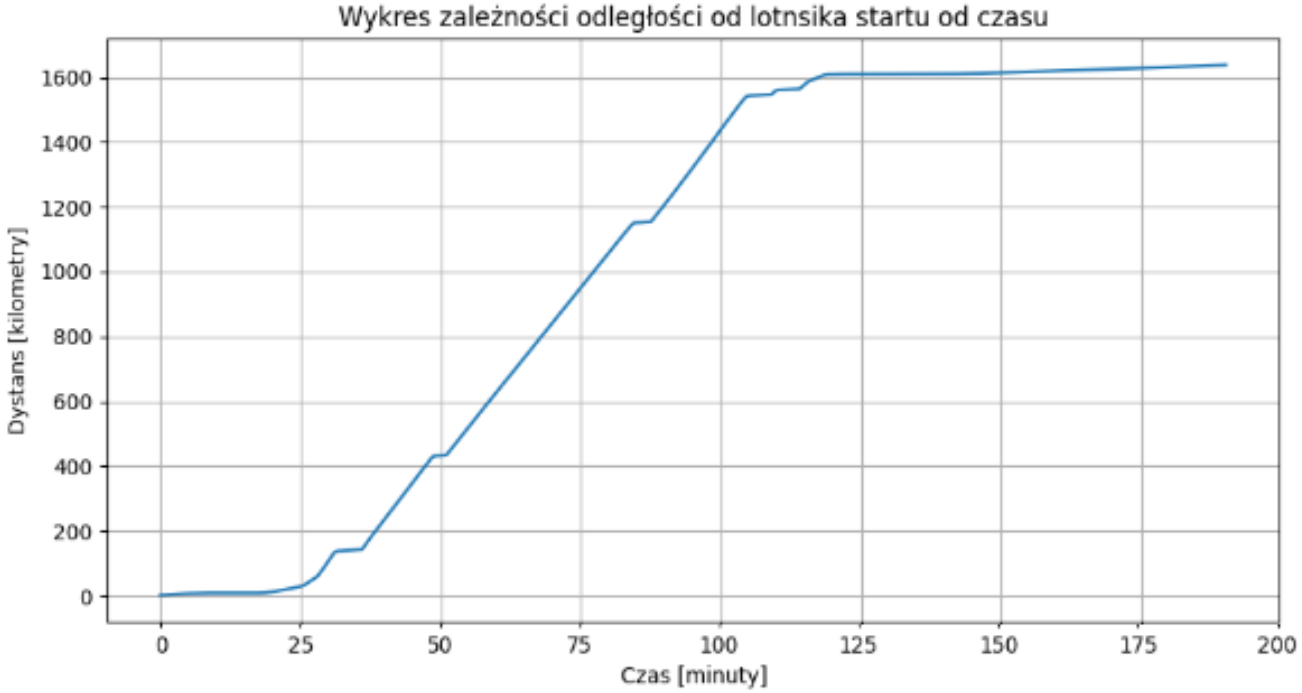
Kolejna wizualizacja przedstawia wykres zmian wysokości samolotu w zależności od czasu. Z wykresu można odczytać, że przez około 30-40 minut lotu samolot się wznosi, po czym osiąga wysokość przelotową wynoszącą ponad 10 000 metrów. Przez tę część lotu wysokość się utrzymuje. Następnie, w ostatnich 30 minutach lotu samolot zaczyna lądować, zmniejszając swoją wysokość.



Kolejny wykres przedstawia zmiany prędkości samolotu. Można zauważyć, że samolot startując osiąga swoją maksymalna prędkość, wynosząca ponad 400 węzłów. Następnie samolot zwalnia osiągając prędkość przelotową, która z lekkimi wahaniami utrzymuje się przez znaczną część lotu. Przed rozpoczęciem lądowania prędkość wzrasta, a następnie spada do 0.



Ostatni wykres przedstawia zmianę odległości samolotu od lotniska startu w zależności od czasu. Z wykresu można odczytać, że lot trwał około 190 minut, a lotnisko końcowe znajduje się w odległości około 1600 km od lotniska startu. Odległość od lotniska w trakcie lotu nie zmieniała się jednostajnie, była zależna od prędkości, z jaką leciał samolot.



5. Kod źródłowy

```
import numpy as np
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import cartopy.io.img_tiles as cimgt
import os
from geopy.distance import geodesic

def read_flightradar(file):
    '''
    Parameters
    ----------
    file : .csv file - format as downloaded from fligthradar24
        DESCRIPTION.
    Returns
    -------
    all_data : numpy array
        columns are:
            0 - Timestamp - ?
            1 - year
            2 - month
            3 - day
            4 - hour
            5 - minute
            6 - second
            7 - Latitude [degrees]
            8 - Longitude [degrees]
            9 - Altitude [feet]
            10 - Speed [?]
            11 - Direction [?]
    '''
    with open(file, 'r') as f:
        i = 0
        size= []
        Timestamp = []; date = []; UTC = []; Latitude = []; Longitude = [];
        Altitude = []; Speed = []; Direction = []; datetime_date = []
        for linia in f:
            if linia[0:1]!='T':
                splited_line = linia.split(',')
                size.append(len(splited_line))
                i+=1
                Timestamp.append(int(splited_line[0]))
                full_date = splited_line[1].split('T')
                date.append(list(map(int,full_date[0].split('-'))))
                UTC.append(list(map(int, full_date[1].split('Z')[0].split(':'))))
                Callsign = splited_line[2]
                Latitude.append(float(splited_line[3].split('"')[1]))
                Longitude.append(float(splited_line[4].split('"')[0]))
                Altitude.append(float(splited_line[5]))
                Speed.append(float(splited_line[6]))
                Direction.append(float(splited_line[7]))

    all_data = np.column_stack((np.array(Timestamp), np.array(date), np.array(UTC),
                                np.array(Latitude), np.array(Longitude), np.array(Altitude),
                                np.array(Speed), np.array(Direction)))
    return all_data

def blh2xyz(phi, lam, h):

    phi_rad = np.deg2rad(phi)
    lam_rad = np.deg2rad(lam)
    a = 6378137
    e2 = 0.00669438002290
    N = a / (np.sqrt(1 - e2 * np.sin(phi_rad) * np.sin(phi_rad)))
    X= (N + h) * np.cos(phi_rad) * np.cos(lam_rad)
    Y = (N + h) * np.cos(phi_rad) * np.sin(lam_rad)
    Z = (N * (1 - e2) + h) * np.sin(phi_rad)
    return X, Y, Z

def macierz_obrotu(phi, lam):
    R = np.array([[-np.sin(phi)*np.cos(lam), -np.sin(lam), np.cos(phi) * np.cos(lam)],
                  [-np.sin(phi) * np.sin(lam), np.cos(lam), np.cos(phi) * np.sin(lam)],
                  [np.cos(phi), 0, np.sin(phi)]])
    return R

def calculate_distances(wspolrzedne_lot, lotnisko):
    distances = [geodesic(lotnisko, (lat, lon)).kilometers for lat, lon, alt in wspolrzedne_lot]
    return distances
```

```python
if __name__ == '__main__':

    azymuty = []
    plik = 'C:/Sem3/GEODEZJA/zajecia2/dane/lot25.csv'
    dane = read_flightradar(plik)

    for lot in dane:
        wspolrzedne = dane[ :, [7, 8, 9]]
        lot = np.where(wspolrzedne[:, -1]>0)[0]
        wspolrzedne[:, -1 ] = wspolrzedne[:, -1]*0.3048 + 135.4
        wspolrzedne_lot = wspolrzedne[lot, :]
        wspolrzedne_lotniska = wspolrzedne[lot[0]-1, :]
        Speed = dane[:, 10]
        Timestamp = dane[:, 0]

        xyz_lotniska = blh2xyz(np.deg2rad(wspolrzedne_lotniska[0]),
                               np.deg2rad(wspolrzedne_lotniska[1]), np.deg2rad(wspolrzedne_lotniska[2]) )
        R = macierz_obrotu(np.deg2rad(wspolrzedne_lotniska[0]), np.deg2rad(wspolrzedne_lotniska[1]))

        for flh in wspolrzedne_lot:
            xyz_samolotu = blh2xyz(np.deg2rad(flh[0]), np.deg2rad(flh[1]), np.deg2rad(flh[2]))
            wektor_samolot_lostnisko = np.array(xyz_samolotu) - np.array(xyz_lotniska)
            neu = macierz_obrotu(np.deg2rad(flh[0]), np.deg2rad(flh[1]))
            neu = R.T@wektor_samolot_lostnisko
            az = np.arctan2(neu[1], neu[0])
            azymuty.append(az)

    request = cimgt.GoogleTiles()
    fig = plt.figure(figsize=(10, 5))
    extent = [-12, 22, 50, 57]
    ax = plt.axes(projection=request.crs)
    ax.set_extent(extent)
    ax.add_image(request, 5)
    start_point = (wspolrzedne_lotniska[0], wspolrzedne_lotniska[1])
    end_point = (dane[-1, 7], dane[-1, 8])
    num_points = 100
    lats = np.linspace(start_point[0], end_point[0], num_points)
    longs = np.linspace(start_point[1], end_point[1], num_points)
    line_points = list(zip(lats, longs))
    ax.plot(longs, lats, transform=ccrs.PlateCarree(), color='orange', linewidth=2)

    for flh, az in zip(wspolrzedne_lot, azymuty):
        if 0 <= az < np.pi:
            color = 'blue'
        else:
            color = 'red'
        ax.plot(flh[1], flh[0], transform=ccrs.PlateCarree(), color = color, marker='o', markersize=5)

    czas_od_startu = (Timestamp - Timestamp[0]) / 60

    fig2 = plt.figure(figsize=(10, 5))
    ax2 = fig2.add_subplot(1, 1, 1)
    ax2.plot(czas_od_startu, Speed)
    ax2.set_xlabel('Czas [minuty]')
    ax2.set_ylabel('Prędkość [węzły]')
    ax2.set_title('Wykres zmian prędkości lotu samolotu')

    fig3 = plt.figure(figsize=(10, 5))
    ax3 = fig3.add_subplot(1, 1, 1)
    ax3.plot(czas_od_startu, wspolrzedne[:, -1 ])
    ax3.set_xlabel('Czas [minuty]')
    ax3.set_ylabel('Wysokość [metry]')
    ax3.set_title('Wykres zmian wysokości lotu samolotu w zależności od czasu')

    distances = calculate_distances(wspolrzedne_lot, start_point)
    czas_trwania_lotu = (Timestamp[-1] - Timestamp[0]) / 60
    fig4 = plt.figure(figsize=(10, 5))
    ax4 = fig4.add_subplot(1, 1, 1)
    ax4.plot(distances)
    ax4.set_xlabel('Czas [minuty]')
    ax4.set_ylabel('Dystans [kilometry]')
    ax4.set_title('Wykres zależności odległości od lotnsika startu od czasu')
    ax4.grid(True)

    plt.show()
```