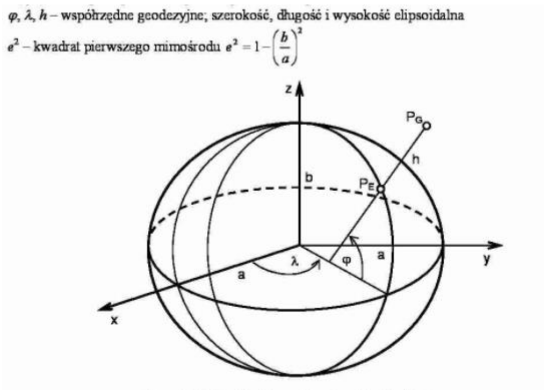
Sprawozdanie – ćwiczenie 1 – układy współrzędnych w elipsoidzie

Maria Mańk 311590

1. Najważniejsze układy odniesienia

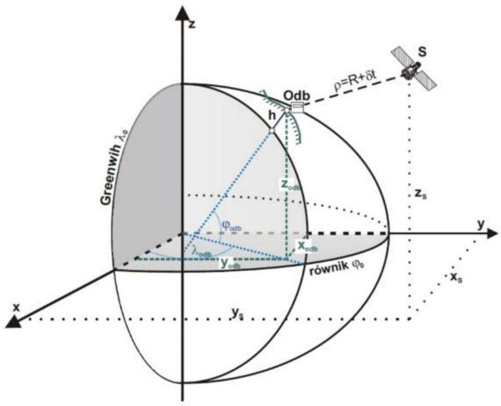
Układ współrzędnych geodezyjnych – aby określić punkt w tym układzie musimy znać trzy współrzędne φ, λ i h. φ jest to kąt między płaszczyzną równika, a prostą normalną przechodzącą przez dany punkt (prostą prostopadłą do powierzchni Ziemi), λ jest to kąt dwuścienny pomiędzy przekrojem południka początkowego i południka przechodzącego przez punkt, natomiast h jest to odległość punktu od powierzchni elipsoidy, do której się odnosimy.



Rysunek 1: Współrzędne geodezyjne, elipsoida obrotowa

Jest to wygodny dla geodetów system, ze względu na to, że wykorzystuje prostą normalną (która może być np. osią obrotu instrumentu pomiarowego). Dzięki temu jest on kompatybilny z lokalnymi układami wykorzystywanymi podczas pomiarów naziemnych.

Układ współrzędnych ortokartezjańskich – jest to pewnego rodzaju przeniesienie układu kartezjańskiego na elipsoidę ziemską, w którym punkt (0, 0, 0) znajduje się w środku masy Ziemi. Dzięki czemu można określić każdy punkt na płaszczyźnie elipsoidy, za pomocą trzech współrzędnych x, y i z.



Aby otrzymać współrzędne ortokartezjańskie z geodezyjnych stosujemy wzory:

$x = (N + h)\cos\phi \cdot cos\lambda$

$y = (N + h)\cos\phi \cdot sin\lambda$

$z = [N(1 - e^2\ ) + \mathrm{h}]sin\phi$

Gdzie:

N – promień przekroju w pierwszym wertykale

Lokalny układ współrzędnych neu – jest to lokalny układ współrzędnych, którego punkt (0, 0, 0) jest dowolnie przyłożony na powierzchni Ziemi. Jest to układ składający się z trzech osi:

n – skierowanej na północ z danego punktu,
e – skierowanej na wschód
u – skierowaną do góry – zawierająca się w prostej normalnej.

Osie n i e są prostopadłe do prostej normalnej przechodzącej przez punkt początkowy tego układu. Natomiast współrzędna u jest różnicą wysokości między punktem początkowym a dowolnym innym punktem.

2. Cel ćwiczenia

- Zrozumienie i zastosowanie algorytmu przeliczania współrzędnych geodezyjnych na lokalne współrzędne neu
- Przeliczenie współrzędnych geodezyjnych na ortokarteziańskie
- Określenie odległości skośnej, azymutu i odległości zenitalnej latającego samolotu
- Określenie punktu w którym samolot znika za horyzontem (o ile taki istnieje)
- Zwizualizować lot samolotu

3. Zastosowanie układu neu jest praktyczniejsze niż układu geodezyjnego gdyż

4. **Zastosowanie układu neu jest mniej praktyczne niż geodezyjnego gdyż** samolot pokonuje duże odległości, wręcz globalne. Co powoduje że stosowanie lokalnego układu odniesienia traci sens, gdyż został on stworzony po to by ułatwić obliczenia na terenie bliskim jego punktowi „zaczepienia”. W związku z czym wygodniejsze jest skorzystanie ze współrzędnych geodezyjnych.

5. Wybrany lot

Do ćwiczenia wybrałam lot samolotu TUI3GB z Heraklionu do Hanoveru z dnia 28.10.2021. Za punkt początkowy układu neu przyjęłam lotnisko w Heraklionie o współrzędnych geodezyjnych (φ = 35.341846, λ = 25.148254, h = 35) (nie są to dokładnie współrzędne w których wystartował samolot, a współrzędne lotniska). Ściągnęłam informacje dotyczące całej trasy lotu, następnie sformatowałam, aby pozostawić jedynie interesujące mnie dane czyli: długość, szerokość i wysokość geodezyjną (oddzielone tabulatorami). W wyniku powstał plik FlightAware.txt, który widoczny jest na załączonym wycinku.
Tak przygotowane dane poddałam obliczeniom, opisanym szerzej w dalszej części sprawozdania.

FlightAware.txt — Notatnik

Plik Edycja Format Widok Pomoc

Time	latitude	longtitude	elevation
09:58:29	35.3409	25.1590	84
09:58:49	35.3443	25.1402	373
09:59:05	35.3535	25.1339	572
09:59:21	35.3657	25.1324	770
09:59:37	35.3780	25.1303	968
09:59:54	35.3915	25.1264	1120
09:59:57	35.3941	25.1256	1196
10:00:59	35.4603	25.1014	1692
10:01:21	35.4891	25.0912	2057
10:01:44	35.5170	25.0814	2385
10:02:11	35.5533	25.0684	2903
10:02:40	35.5889	25.0555	3315
10:03:19	35.6413	25.0367	3795
10:03:50	35.6853	25.0210	4412
10:04:29	35.7379	24.9997	4900
10:05:00	35.7846	24.9777	5044
10:05:30	35.8297	24.9566	5380
10:06:00	35.8772	24.9342	5776
10:06:30	35.9269	24.9108	6096
10:07:00	35.9782	24.8866	6416
10:07:20	36.0120	24.8706	6645
10:07:40	36.0450	24.8550	6850
10:08:10	36.0988	24.8296	7148
10:08:42	36.1543	24.8033	7445
10:09:00	36.1858	24.7883	7612
10:09:21	36.2252	24.7696	7788
10:09:40	36.2605	24.7527	7940
10:10:10	36.3162	24.7261	8214
10:10:40	36.3720	24.6994	8473
10:11:21	36.4474	24.6633	8793
10:11:52	36.5049	24.6359	9014
10:12:22	36.5621	24.6084	9220
10:12:51	36.6179	24.5817	9395
10:13:21	36.6720	24.5557	9555

Link do danych lotu:
https://flightaware.com/live/flight/TUI3GB/history/20211028/0757Z/LGIR/EDDV

6. Implementacja kodu

Do zaimplementowania kodu użyłam 4 biblioteki: math, numpy, csv oraz plotly.graph_objects. Csv posłużyła mi do wygodniejszego odczytania danych z wcześniej przygotowanego pliku, a plotly umożliwiła zwizualizowanie danych (zarówno w układzie geodezyjnym jak i neu).

Funkcja przeliczające układ geodezyjny na układ ortokartezjański:

```
def geo2xyz(fi, lam, h, a, e2):
    #fi = np.deg2rad(fi)
    #lam = np.deg2rad(lam)
    N = a / (math.sqrt(1 - e2*np.sin(fi)**2))
    x = (N+h)*np.cos(fi)*np.cos(lam)
    y = (N+h)*np.cos(fi)*np.sin(lam)
    z = (N*(1-e2)+h)*np.sin(fi)
    return x, y, z
```

Funkcja geo2xyz wykorzystując wzory z pkt1 i wzór na N tworzy tablicę współrzędnych xyz. W komentarzu pozostawiłam przeliczanie fi i lambdy na radiany ze względu na to, że w funkcji kolejnej wykorzystuje tę funkcję na już przeliczonych współrzędnych. Jednak gdybym chciała przeliczać współrzędne geodezyjne na xyz, a nie wykorzystywać jedynie pośrednio w procesie przeliczania na układ neu, to pozostawiłam je w tym miejscu.

Funkcja przeliczająca układ geodezyjny na lokalny układ neu:

```
def geo2neu(F1, L1, H1, F2, L2, H2 ):
    F1 = np.deg2rad(F1)
    L1 = np.deg2rad(L1)
    F2 = np.deg2rad(F2)
    L2 = np.deg2rad(L2)
    R = np.array([[-np.sin(F1)*np.cos(L1), -np.sin(L1), np.cos(F1)*np.cos(L1)],
                  [-np.sin(F1)*np.sin(L1), np.cos(L1), np.cos(F1)*np.sin(L1)],
                  [np.cos(F1), 0, np.sin(F1)]])
    x1, y1, z1 = geo2xyz(F1, L1, H1, a, e2)
    x2, y2, z2 = geo2xyz(F2, L2, H2, a, e2)
    x = np.array([[x2 - x1, y2 - y1, z2 - z1]])
    neu = np.dot(R.transpose(), x.transpose())
    return neu
```

Na początku tworzymy macierz transformacji, następnie przeliczamy na układ ortokartezjański oba punkty, czyli punkt początkowy i punkt nas interesujący. Następnie z różnic tych współrzędnych tworzymy wektor, który następnie transponujemy i wraz ze transponowaną macierzą transformacji wykorzystujemy do otrzymania współrzędnych neu. Jako wynik otrzymujemy tablicę trzech współrzędnych n, e i u.

Dalsza część programu poświęcona jest sczytaniu danych z pliku FlightAware.txt, przedstawieniu ich, określeniu punktu w którym samolot zniknął za horyzontem i stworzenie wykresu współrzędnych neu i zapisaniu ich do pliku.

```
result = open('NEUresult.txt', "w")
result.write("Date\t\tn\t\t\te\t\tu\t\tazymut\t\todleglosc\tkat z\n")
with open('FlightAware.txt') as data:
    file = csv.reader(data, delimiter='\t')
```

Powyższe linie są liniami pozwalającymi otworzyć plik NEUreslut.txt do zapisu i FlightAware.txt do odczytu danych.

```
line_count = 0
czyzniknal = False
```

Dwie zmienne pozwalające określić w którym wierszu plików się znajdujemy i czy samolot już zniknął za horyzontem, a na koniec, czy w ogóle znikł za horyzontem.

```
for row in file:
    if line_count != 0:
        f2 = float(row[1])
        l2 = float(row[2])
        h2 = float(row[3])
        lt.insert(line_count, f2)
        lg.insert(line_count, l2)
```

Sczytanie poszczególnych współrzędnych z pliku – f2, l2, h2 to współrzędne punktu właśnie odczytywanemu z pliku. Tak sczytane współrzędne zapisujemy do tablic lt i lg (od latitude i longtitude) abyśmy potem mogli zwizualizować lot na mapie (a do tego są niezbędne współrzędne geodezyjne).

```
neu = geo2neu(f1, l1, h1, f2, l2, h2)
n = neu[0]
e = neu[1]
u = neu[2]
nt.insert(line_count, *n)
ea.insert(line_count, *e)
up.insert(line_count, *u)
```

Następnie tworzymy tablicę współrzędnych neu z tak otrzymanych danych i przepisujemy ją do trzech zmiennych n, e i u. Następnie podobnie jak współrzędne geodezyjne, zapisujemy współrzędne neu do tablic, by móc je potem zwizualizować na wykresie.

```
A = np.rad2deg(np.arctan(e / n))
s = math.sqrt(n ** 2 + e ** 2 + u ** 2)
z = np.rad2deg(np.arccos(u / s))
```

W kolejnym wierszu określamy dla danego punktu azymut, odległość skośną i kąt zenitalny.

```
line = row[0]+'\t'+str(*(np.round(n,3)))+'\t'+str(*(np.round(e,3)))+'\t'+str(*(np.round(u,3)))+'\t'+\
      str(*(np.round(A, 6)))+'\t'+str((np.round(s,3)))+'\t'+str(*(np.round(z,6)))+'\n'
result.write(line)
```

Następnie tworzę linię (wiersz) do zapisu współrzędnych neu – przybliżonych do 3 miejsc po przecinku (mm). Dodatkowo w każdym wierszu pliku będącego wynikiem programu zapisuję azymut, odległość skośną i kąt zenitalny w danym punkcie.

```
if czyzniknal == False and (z >= 90 and u <= 0):
    print('Samolot zniknął za horyzontem w punkcie o współrzędnych neu: [', *(np.round(n,3)), *(np.round(e,3)), *(np.round(u,3)), '] i kącie zenitalnym: ',
          *(np.round(z,6)),'\nI współrzędnych geodezyjnych fi, lamda, h: [', row[1], row[2], row[3], ']')
    fig = go.Figure(go.Scattermapbox(
        mode="markers",
        lon=[l2],
        lat=[f2],
        name= 'Miejsce gdzie samolot znika za horyzontem',
        marker=go.scattermapbox.Marker(
            size=6,
            color='rgb(0, 0, 0)'
        )))
    czyzniknal = True
```

Externally added files can be added to Git

Powyższa instrukcja warunkowa sprawdza czy samolot zniknął już wcześniej za horyzontem. Jeśli w poprzednim wierszu jeszcze nie znikną, ale w tym już kąt zenitalny przekroczył 90 stopni, a współrzędna u jest mniejsza od 0 to wypisuje na konsoli w jakich współrzędnych znajdował się samolot (zarówno neu jak i geodezyjne, a także kąt z). Potem zaznacza ten punkt na mapie, a następnie zmienia wartość czyzniknal – dzięki czemu instrukcja ta już nie będzie więcej wykonywana.

Dalej zamykamy plik w którym zapisywaliśmy współrzędne neu i A, s i z. Prezentuje się on tak:

Date	n	e	u	azymut	odleglosc	kat z
09:58:29	-104.904	976.91	48.924	-83.870858	983.743	87.149339
09:58:49	272.309	-732.185	337.952	-69.599154	851.151	66.605923
09:59:05	1293.187	-1304.807	536.735	-45.256269	1913.88	73.713327
09:59:21	2646.967	-1440.988	734.286	-28.56352	3101.944	76.307078
09:59:37	4011.943	-1631.662	931.526	-22.131637	4430.096	77.861698
09:59:54	5510.166	-1985.811	1082.304	-19.818676	5956.237	79.530664
09:59:57	5798.767	-2058.463	1158.024	-19.543993	6261.309	79.341831
10:00:59	13146.722	-4254.252	1641.992	-17.931474	13915.139	83.223285
10:01:21	16344.327	-5178.841	1998.895	-17.581254	17261.315	83.350112
10:01:44	19442.375	-6066.608	2317.398	-17.329499	20498.293	83.508649
10:02:11	23474.147	-7243.605	2820.571	-17.149074	24727.737	83.450297
10:02:40	27428.571	-8410.592	3215.32	-17.047433	28868.718	83.605283
10:03:19	33249.679	-10109.469	3665.097	-16.911743	34945.321	83.979695
10:03:50	38139.849	-11527.051	4252.266	-16.816454	40069.973	83.908243
10:04:29	43985.993	-13448.654	4698.786	-17.000937	46235.403	84.167107
10:05:00	49175.741	-15431.644	4800.31	-17.422172	51763.231	84.678979
10:05:30	54189.979	-17331.866	5090.717	-17.736109	57121.474	84.886965
10:06:00	59472.489	-19347.215	5433.765	-18.020416	62775.931	85.034372
10:06:30	64999.881	-21449.866	5693.005	-18.262815	68683.998	85.245474
10:07:00	70706.21	-23621.81	5944.517	-18.473646	74784.325	85.440817
10:07:20	74466.759	-25056.465	6125.173	-18.596981	78807.629	85.542299
10:07:40	78138.545	-26454.065	6280.53	-18.703701	82733.88	85.64635
10:08:10	84125.143	-28727.113	6492.42	-18.854046	89131.578	85.822827
10:08:42	90301.962	-31077.628	6693.805	-18.991019	95734.374	85.990573
10:09:00	93808.244	-32416.864	6803.455	-19.063368	99484.303	86.07864
10:09:21	98193.877	-34084.782	6904.646	-19.142767	104170.457	86.199524
10:09:40	102123.554	-35590.862	6986.616	-19.213848	108373.164	86.303685
10:10:10	108325.628	-37958.856	7144.495	-19.311194	115005.914	86.43833
10:10:40	114539.698	-40332.444	7280.208	-19.398509	121651.346	86.569092
10:11:21	122937.643	-43536.278	7422.586	-19.500735	130629.883	86.742613
10:11:52	129342.652	-45963.645	7499.71	-19.563351	137471.538	86.872697
10:12:22	135715.096	-48396.525	7555.122	-19.62643	144284.063	86.998458
10:12:51	141931.893	-50754.874	7576.293	-19.677061	150924.218	87.122582

Sprawdzenie poprawności wyników:

```
1955860.839      -1036777.543     -397040.991      -27.927507       2248986.626      100.168416
```

Powyższy wiersz, są to wyniki obliczeń dla ostatniego punktu wprowadzonego do programu, zatem dla lotniska w Hanowerze. Na tej podstawie znając położenie obu lotnisk jesteśmy stwierdzić poprawność obliczeń. Azymut i odległość skośna policzone są na podstawie współrzędnych neu, zatem jeśli na podstawie współrzędnych geodezyjnych policzymy te wartości to jesteśmy w stanie stwierdzić czy program działa poprawnie.

Azymut między lotniskami wynosi 332 stopnie (ręcznie policzone na kalkulatorze), a więc -27 stopni (dokładnie tak jak wyszło w wynikach). Natomiast odległość między ostatnim punktem a lotniskiem w Heraklionie według danych lotu wynosi 2265 km, a wynik wyszedł ok. 2249 km. Różnica wynika z tego, że odległość otrzymana w wynikach określa odległość w linii prostej. Aby otrzymać zbliżoną wartość tej odległości pomierzyłam odległość w programie Google Earth i wyszło ok. 2500 km. Świadczy to o poprawności obliczeń.

Ostatnia część kodu służy narysowaniu trasy lotu w układzie geodezyjnym i neu:

```
fig.add_trace(go.Scattermapbox(
    mode="lines",
    lon=lg,
    lat=lt,
    name='Trasa samolotu',
    marker=go.scattermapbox.Marker(
            size=8,
            color='rgb(192, 20, 20)'
        )))
```

```
fig3d = go.Figure(data=[go.Scatter3d(
    x=nt,
    y=ea,
    z=up,
    mode='markers',
    marker=dict(
        size=8,
        color="red",
        opacity=0.8
    )
)])
```

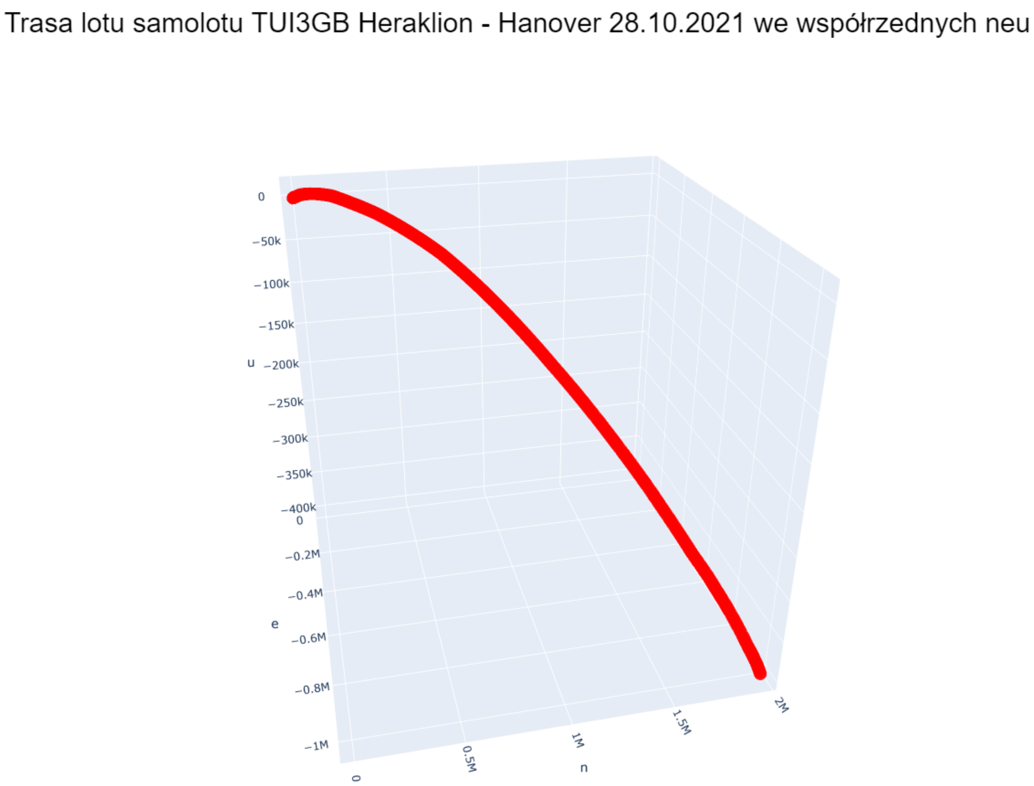
Części kodu odpowiedzialnych za wygląd mapy i wykresu nie omawiam.

Jako wynik kodu otrzymujemy na konsoli:

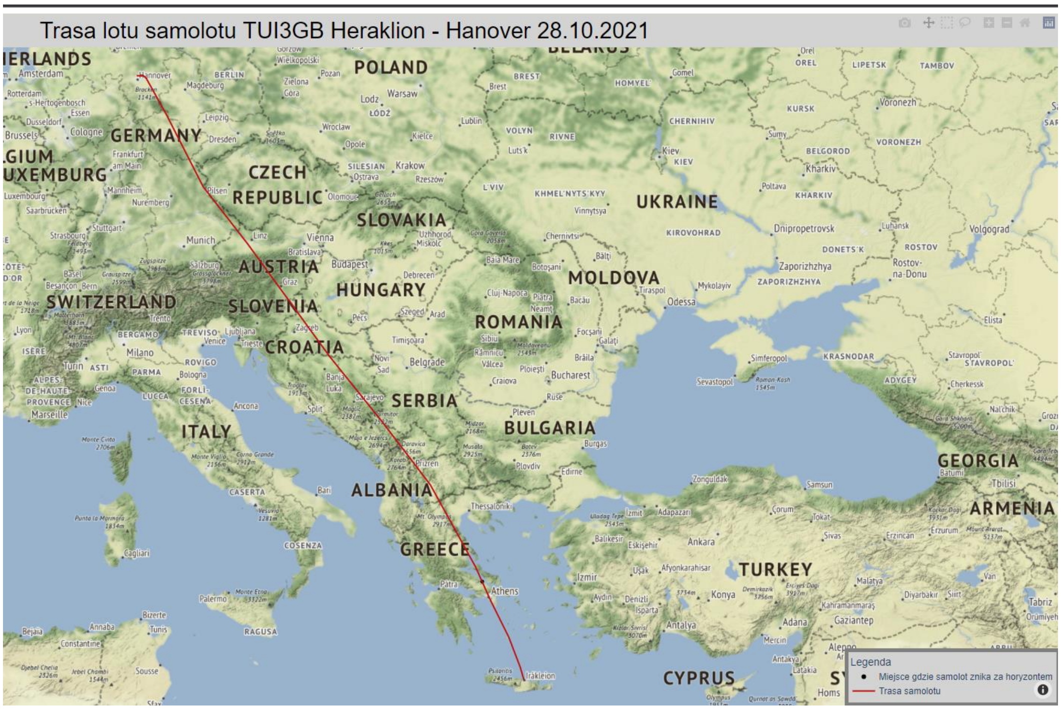
```
Samolot zniknął za horyzontem w punkcie o współrzędnych neu: [ 348209.446 -146080.751 -251.606 ] i kącie zenitalnym:  90.038177
I współrzędnych geodezyjnych fi, lamda, h: [ 38.4646 23.4771 10980 ]
Przeliczone współrzędne neu wraz z Azymutami, odlełością skośna i kątem zenitalnym zostały zapisane do pliku NEUresult.txt

Process finished with exit code 0
```

W folderze w którym został wywołany program pojawi się plik NEUresult.txt z przeliczonymi współrzędnymi i azymutami, odległościami i kątami. Program także wyświetli dwie wizualizacje w przeglądarce:



Mankamentem wizualizacji we współrzędnych neu jest to, że niestety o ile da się zmienić nazwy osi na neu, to w 'hoverlabel' współrzędne n, e, u pozostają nazwane, jako x, y, z, co jest trochę mylące. Niestety jedyne rozwiązanie które umożliwiało wypisanie współrzędnych neu bądź Azymutu, odległości i kąta zenitalnego, okazało się znacznie opóźniać działanie programu. Pomysł polegał na tym, aby podczas odczytywania pliku dodawać każdy punkt wykresu oddzielnie, bo taka operacja umożliwia nadawanie wyświetlanych informacji każdemu punktowi oddzielnie. Jednak okazało się że operacja ta jest bardzo kosztowna czasowo, w związku z czym zrezygnowałam ze zmiany wyświetlanych informacji.



Wizualizacja we współrzędnych geodezyjnych poza trasą lotu pokazuje także w którym miejscu samolot zniknął za horyzontem. To miejsce jest zaznaczone czarną kropką i w przypadku lotu 28.10.2021r znajdowało się w okolicach Aten.

7. Wnioski

Podczas wykonywania ćwiczenia obliczenia nie przysporzyły mi zbyt wielu problemów. Najtrudniejsza okazała się wizualizacja, jednak nie samo jej stworzenie, a zmienienie interfejsu otrzymanej mapy i wykresu.