SPRAWOZDANIE

Systemy wysokości: niwelacja satelitarna – wysokości normalne
ćwiczenie 5

Izabella Kaim 319193

1. Cel ćwiczenia

Celem ćwiczenia było wykonanie profilu wysokościowego odcinka trasy w systemie wysokości normalnych, na podstawie danych współrzędnych krzywoliniowych oraz wysokości elipsoidalnych.
Następnie przedstawić przebieg trasy oraz profil na odpowiednich wizualizacjach.

2. Dane

Za moje dane wejściowe wybrałam trasę rowerową ze strony szlaków rowerowych Green Velo https://greenvelo.pl/detal/1246-greenvelo-niebieski-szlak-rowerowy-dookola-rakowa-i-jeziora-chancza. Jest to niebieski szlak rowerowy dookoła Rakowa i Jeziora Chańcza.

Dane zostały pobrane w formie pliku .gpx. Plik ten zawiera wysokości geodezyjne, szerokości geodezyjne i wysokości elipsoidalne punktów trasy.

3. Wykonanie

Przeliczenie współrzędnych geodezyjnych na współrzędne ortokartezjańskie x,y,z za pomocą funkcji:

```
def blh2xyz(f, l, h):
    a = 6378137
    e2 = 0.00669438002290
    f = np.deg2rad(f)
    l = np.deg2rad(l)
    N = a / (m.sqrt(1 - e2*np.sin(f)*np.sin(f)))
    x = (N+h)*np.cos(f)*np.cos(l)
    y = (N+h)*np.cos(f)*np.sin(l)
    z = (N*(1-e2)+h)*np.sin(f)
    return [x, y, z]
```

Aby otrzymać wysokość normalną punktów trasy, należało wykonać interpolację dwuliniową wykorzystując model quasigeoidy, PL-geoid2021, pobrany ze strony http: //www.gugik.gov.pl/bip/prawo/modele-danych.

Wykorzystałam funkcję interpn z biblioteki scipy.interpolate.

```
#interpolacja
model = np.genfromtxt('Model_quasi-geoidy-PL-geoid2021-PL-EVRF2007-NH.txt', skip_header=1)

x = model[:,0]
y = model[:,1]
z = model[:,2]

x_grid = x.reshape((len(np.unique(x)), -1))
y_grid = y.reshape((len(np.unique(x)), -1))
z_grid = z.reshape((len(np.unique(x)), -1))
```

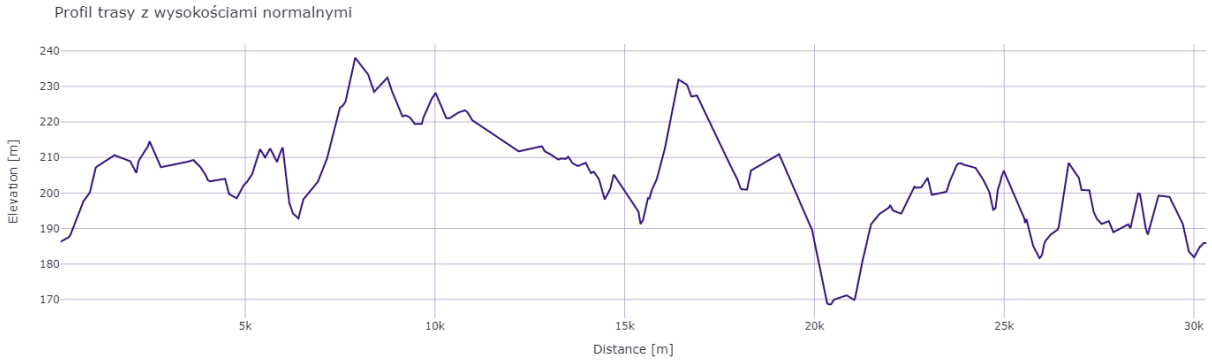
```
x_range = x_grid[:,0]
y_range = y_grid[0,:]

z_all = []

for i in range(len(route_df)):
    lat = route_df.iloc[i]['latitude']
    lon = route_df.iloc[i]['longitude']
    point = (lat, lon)
    zeta = interpn((x_range, y_range), z_grid, point)
    z_all.append(zeta)
```

Następnie odjęłam otrzymane anomalie wysokości od wysokości elipsoidalnych punktów trasy.

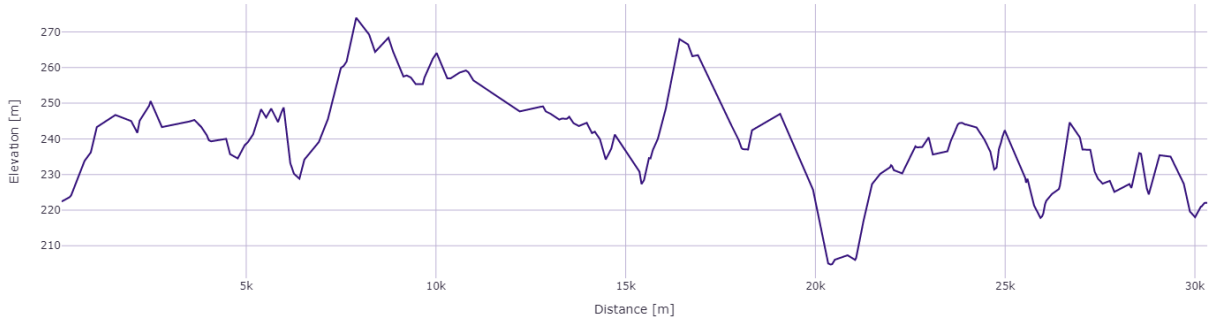
```
#wyliczenie wysokości normalnej
wys_n = []
for i in range(len(route_df)):
    h_elip = route_df.iloc[i]['elevation']
    h_n = h_elip - z_all[i][0]
    wys_n.append(h_n)

route_df['Elevation n'] = wys_n
```

4. Wyniki

Wizualizacja trasy znajduje się w aplikacji dash: http://127.0.0.1:8050/ generującej się na podstawie kodu app.py.



Profil trasy z wysokościami elipsoidalnymi



Trasa w 3D