Przeniesienie współrzędnych geodezyjnych na powierzchni elipsoidy obrotowej

Autor: Michał Ambroży

Numer indeksu: 328934 Numer grupy: 1, Numer w dzienniku: 1

Ćwiczenie ma na celu zastosowanie algorytmu Kivioja lub biblioteki pyproj do przeliczenia współrzędnych geodezyjnych punktów na elipsoidzie. WYkorzystałem bibliotekę pyproj, lecz podaję niżej również algorytm Kivioja. Wykorzystałem podane długości i azymuty linii geodezyjnych, aby obliczyć współrzędne punktów 2, 3, 4. Sprawdziłem, czy zamknięcie 'trapezu' utworzy figurę zamkniętą, a następnie wyznaczyłem odległość i azymut z punktu 4 do punktu 1. Ostatecznie, przedstawiłem położenie punktów na mapie i obliczyłem pole powierzchni figury. Poniżej znajduje się niewykorzystany algorytm Kivioja:

In [ ]:
```
# algorytm Kivioja
import numpy as np
import folium as fl

def Np(phi):
    a = 6378137 # wielka polos elipsoidy
    b = 6356752.3142 # mala polos elipsoidy
    e2 =  0.00669438002290 # kwadrat pierwszego mimośrodu elipsoidy
    N = a / np.sqrt(1-(e2*np.sin(phi)**2)) # promien krzywizny N
    return N

def Mp(phi):
    a = 6378137 # wielka polos elipsoidy
    b = 6356752.3142 # mala polos elipsoidy
    e2 =  0.00669438002290 # kwadrat pierwszego mimośrodu elipsoidy
    M = a * (1 - e2) / (1 - e2 * np.sin(phi)**2)**(3/2) # promien krzywizny M
    return M
```

```
def kivioj(phi, lam, s, az, m):
    # 1. Podzielenie linii geodezyjnej na n elementów ds.
    n = round(s / 1000)
    ds = s / n

    for i in range(n):
        # 2. Obliczamy główne promienie krzywizny N i M w punkcie wyjściowym P1 oraz stałą c linii geodezyjnej.
        N_i = Np(phi)  # Promień krzywizny N
        M_i = Mp(phi)  # Promień krzywizny M

        # 3. Pierwsze przybliżenie przyrostu szerokości i azymutu:
        dphi_i = ds * np.cos(az) / M_i
        dA_i = ds * np.sin(az) / (N_i * np.cos(phi))

        # 4. Obliczenie szerokości i azymutu w punkcie środkowym (m) odcinka, na podstawie przyrostów:
        phi_im = phi + dphi_i / 2
        az_im = az + dA_i / 2

        # 5. Obliczenie promieni krzywizny w kierunkach głównych w punkcie m:
        N_im = Np(phi_im)
        M_im = Mp(phi_im)

        # 6. Ostateczne przyrosty szerokości, długości i azymutu:
        dphi_i = ds * np.cos(az_im) / M_im
        dlam_i = ds * np.sin(az_im) / (N_im / np.cos(phi_im))
        dA_i = ds * np.sin(az_im) / (N_im * np.cos(phi_im))

        # 7. Obliczamy współrzędne końca odcinka ds oraz azymut na końcu odcinka linii geodezyjnej.
        phi = phi + dphi_i
        az = az + dA_i
        lam = lam + dlam_i

        # 8. Rysujemy punkt odcnikowy na mapie
        fl.Marker(location=[np.rad2deg(phi), np.rad2deg(lam)], popup='Punkt ' + str(i + 2), icon=fl.Icon(color='red', icon='ok

        # 9. Rysujemy odcinek na mapie
        fl.PolyLine([[phi - dphi_i, lam - dlam_i], [phi, lam]], color="red", weight=2.5, opacity=1).add_to(m)

        # 10. Obliczamy azymut odwrotny
```

```
        az_odw = az + np.pi
        if az_odw > 2 * np.pi:
            az_odw = az_odw - 2 * np.pi

    return phi, lam, az_odw
```

Obliczenie oraz przedstawienie na mapie puntków 1, 2, 3 ,4, 5 (1*)

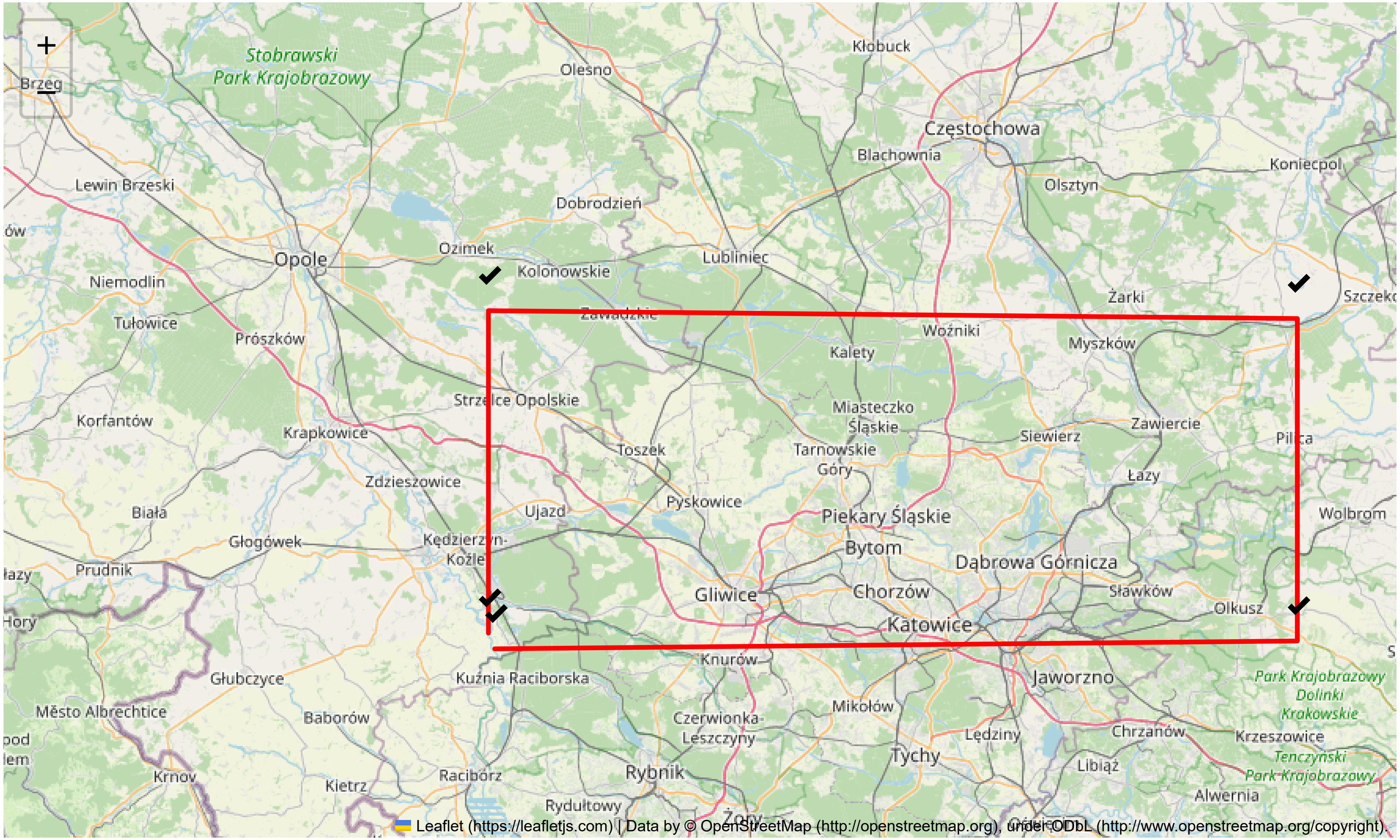
Punkty zostały podane w tabelce.

In [ ]:
```
import pyproj
from pyproj import Geod
import folium as fl

geod = pyproj.Geod(ellps='WGS84')
# Współrzędne punktu 1 dla numeru 1 w stopniach:
phi_1 = 50 + 15/60
lambda_1 = 18 + 15/60

# Kolejne długości i azymuty linii geodezyjnych:
# długość s [m] azymut A [◦]
s1_2 = 40000
s2_3 = 100000
s3_4 = 40000
s4_1 = 100000

A1_2 = 0
A2_3 = 90
A3_4 = 180
A4_1 = 270

# Tworzymy mapę folium
fig = fl.Figure(width=1000, height=1000)
m = fl.Map(location=[phi_1, lambda_1], zoom_start=9)

# definicja elipsoidy WGS84, transformacja współrzędnych i obliczenie współrzędnych punktów
geod = pyproj.Geod(ellps='WGS84')
p2 = geod.fwd(lambda_1,phi_1,A1_2,s1_2)
```

```
p3 = geod.fwd(p2[0],p2[1],A2_3,s2_3)
p4 = geod.fwd(p3[0],p3[1],A3_4,s3_4)
p5 = geod.fwd(p4[0],p4[1],A4_1,s4_1)

# wyświetl współrzędne punktów w stopniach w tabeli
print('Nr      \tSzerokość geo  \t\tDlugość geo')
phi_deg = int(phi_1)
phi_min = (phi_1 - phi_deg)*60
phi_sec = (phi_min - int(phi_min))*60
lam_deg = int(lambda_1)
lam_min = (lambda_1 - lam_deg)*60
lam_sec = (lam_min - int(lam_min))*60
print(f'1\t\t{phi_deg}° {int(phi_min)}\' {phi_sec:.5f}"\t{lam_deg}° {int(lam_min)}\' {lam_sec:.5f}"')
points = [p2,p3,p4,p5]
for i,p in enumerate(points):
    phi_deg = int(p[1])
    phi_min = (p[1] - phi_deg)*60
    phi_sec = (phi_min - int(phi_min))*60
    lam_deg = int(p[0])
    lam_min = (p[0] - lam_deg)*60
    lam_sec = (lam_min - int(lam_min))*60
    if i == len(points)-1:
        print(f'1*\t\t{phi_deg}° {int(phi_min)}\' {phi_sec:.5f}"\t{lam_deg}° {int(lam_min)}\' {lam_sec:.5f}"')
    else:
        print(f'{i+2}\t\t{phi_deg}° {int(phi_min)}\' {phi_sec:.5f}"\t{lam_deg}° {int(lam_min)}\' {lam_sec:.5f}"')

# za pomocą biblioteki folium rysujemy punkty na mapie
fl.Marker(location=[phi_1, lambda_1], popup='Punkt 1', icon=fl.Icon(color='red', icon='ok')).add_to(m)
fl.Marker(location=[p2[1], p2[0]], popup='Punkt 2', icon=fl.Icon(color='red', icon='ok')).add_to(m)
fl.Marker(location=[p3[1], p3[0]], popup='Punkt 3', icon=fl.Icon(color='red', icon='ok')).add_to(m)
fl.Marker(location=[p4[1], p4[0]], popup='Punkt 4', icon=fl.Icon(color='red', icon='ok')).add_to(m)
fl.Marker(location=[p5[1], p5[0]], popup='Punkt 5', icon=fl.Icon(color='red', icon='ok')).add_to(m)

# za pomocą biblioteki folium rysujemy odcinki na mapie
fl.PolyLine([[phi_1,lambda_1],[p2[1],p2[0]],[p3[1],p3[0]],[p4[1],p4[0]],[p5[1],p5[0]]], color='red').add_to(m)

m
```

```
Nr              Szerokość geo          Dlugość geo
1               50º 15' 0.00000"       18º 15' 0.00000"
2               50º 36' 34.52938"      18º 15' 0.00000"
3               50º 36' 3.69855"       19º 39' 45.19830"
4               50º 14' 29.16725"      19º 39' 45.19830"
1*              50º 13' 58.73456"      18º 15' 39.25758"
```

Out[ ]:



In [ ]:

```python
# obliczmy teraz odległość między punktami 1 i 5
# ustalamy elipsoidę WGS84
```

```
geod = Geod(ellps='WGS84')
az12,az21,dist = geod.inv(lambda_1,phi_1,p5[0],p5[1]) # obliczamy odległość między punktami 1 i 5 przy pomocy biblioteki pypro
print(f'Odległość między punktami 1 i 5 wynosi {dist/1000:.3f} km') # wyświetlamy wynik w kilometrach
```

```
Odległość między punktami 1 i 5 wynosi 2.047 km
```

Wniosek:

Po obliczeniu kolejnych wieżchołków 'trapezu' na podstawie podanych obserwacji otrzymana figura się nie zamknie. Odległość początkowego punktu 1 oraz końcowego 5 (1*) w naszym przypadku wynosi 2.047km. Różnica spowodowana jest sferycznością Ziemii, a zatem im większe dlugośći boku naszego 'trapezu', tym różnica w położeniu punktu 1 oraz 5 (1*) będzie również rosła.

Obliczenie pola powierzchni figury powstałej z obliczonych puntków

Pole powierzchni wyniosło 4110.305 km2.

In [ ]:
```
# pole powierzchni powstałej figury
# ustalamy elipsoidę WGS84
geod = Geod(ellps='WGS84')
# obliczamy pole powierzchni powstałej figury przy pomocy polygon_area_perimeter z biblioteki pyproj, która oblicza pole powie
area = geod.polygon_area_perimeter([lambda_1,p2[0],p3[0],p4[0],p5[0]],[phi_1,p2[1],p3[1],p4[1],p5[1]])
print(f'Pole powierzchni powstałej figury wynosi {area[0]/1000000:.3f} km2') # wyświetlamy wynik w kilometrach kwadratowych
```

```
Pole powierzchni powstałej figury wynosi -4110.305 km2
```

Zadanie Odwrotne

W podanym kodzie użyto biblioteki pyproj z elipsoidą WGS84 do obliczenia odległości, azymutu z punktu 4 do punktu 1, wykorzystując funkcję geod.inv. Następnie wyniki przedstawiono w kilometrach, stopniach, minutach i sekundach, co pozwala odczytać odległość i azymut geodezyjny między punktami na powierzchni elipsoidy. Odległość między punktami 4 i 1 wyniósł 100.760 km, natomiast azymut z punktu 4 do punktu 1 wyniósł -5094^0, -28', -18.59652"

In [ ]:
```
# ustalamy elipsoidę WGS84
geod = Geod(ellps='WGS84')
# obliczamy odległość i azymut z punktu 4 do punktu 1 przy pomocy biblioteki pyproj i funkcji geod.inv, która oblicza odległoś
az41,az14,dist = geod.inv(p4[0],p4[1],lambda_1,phi_1)
print(f'Odległość między punktami 4 i 1 wynosi {dist/1000:.3f} km') # wyświetlamy wynik w kilometrach
# wysiwetlamy azymut z punktu 4 do punktu 1 w stopniach, minutach i sekundach
az_deg = int(np.rad2deg(az41))
az_min = (np.rad2deg(az41) - az_deg)*60
az_sec = (az_min - int(az_min))*60
print(f'Azymut z punktu 4 do punktu 1 wynosi {az_deg}°, {int(az_min)}\', {az_sec:.5f}"') # wyświetlamy wynik w stopniach, minu
```

```
Odległość między punktami 4 i 1 wynosi 100.760 km
Azymut z punktu 4 do punktu 1 wynosi -5094°, -28', -18.59652"
```