Maria Mańk 311590

Zadanie 3 – Rozwiązywanie zadań na elipsoidzie obrotowej

Zadanie:

1. Wyznaczyć punkt średniej szerokości
2. Wyznaczyć punkt środkowy przy użyciu algorytmu Vincentego i Kivioji
3. Wyznaczyć różnicę odległości pomiędzy tymi punktami
4. Wyznaczyć azymuty w tych punktach.
5. Obliczyć pole powierzchni tego czworokąta wedle wzoru:

$$P = \frac{b^2(\lambda 2 - \lambda 1)}{2}\left(\frac{\sin\varphi}{1 - e^2\sin^2\varphi} + \frac{1}{2e}\ln\frac{1 + e\sin\varphi}{1 - e\sin\varphi}\right)_{\varphi 1}^{\varphi 2}$$

Dane:

Współrzędne punktów pochodzą z treści zadania. Zostały one przeliczone na radiany:

```
nr = 0
fia= 50 + 15/60 + nr*15/60 #stopni
lma= 20 + 45/60 # stopni
fib = 50 + nr*15/60
lmb = 20 + 45/60
fic= 50 + 15/60 + nr*15/60 #stopni
lmc= 21 + 15/60 # stopni
fid = 50 + nr*15/60
lmd = 21 + 15/60
```

```
fi1= np.deg2rad(45)
lb1 = np.deg2rad(17)
fi2= np.deg2rad(1)
lb2= np.deg2rad(25)
fi3= np.deg2rad(-60)
lb3= np.deg2rad(190)
```

Dodatkowo jako zmienne globalne zapisałam sobie parametry elipsoidy GRS80

```
a = 6378137 #metry
e2 = 0.00669438002290
```

Biblioteki:

Do uzyskania zamierzonego efektu wykorzystałam jedną bibliotekę. Numpy posłużyła mi do przeliczeń matematycznych.

```
import numpy as np
```

Poszczególnie funkcje znajdujące się w programie:

Od razu na początku policzyłam punkt średniej szerokości przy użyciu wzoru na średnią arytmetyczną.

```
fis = (fia + fid)/2
lms = (lma + lmd)/2
```

Pierwsza funkcja jest implementacją algorytmu Kivoji zwracająca współrzędne punktu środkowego szerokości.

```
def Kivioj (a, e2, fi, lm, s,  A): #;)
    fi = np.deg2rad(fi)
    lm = np.deg2rad(lm)
    A = np.deg2rad(A)
    n = round(s / 1000)
    ds = s / n
    for i in range(n):
        M1= (a*(1-e2))/((np.sqrt(1 - e2*np.sin(fi)**2))**3)
        N1= a / (np.sqrt(1 - e2*np.sin(fi)**2))
        fi1= (ds*np.cos(A))/M1 #pierwsze przybliżenie przyrostu szerokości
        fi12= fi+fi1/2 #średnia wartość szerokości elementu
        M2=  (a*(1-e2))/(np.sqrt((1 - e2*np.sin(fi12)**2)**3))
        N2= a / (np.sqrt(1 - e2*np.sin(fi12)**2))
        A2 = A+ ((ds/N1)*np.sin(A)*np.tan(fi))/2
        dfi = (ds*np.cos(A2))/M2
        dlm= ds*np.sin(A2)/(N2*np.cos(fi12))
        dA = ds / N2 * np.sin(A2) * np.tan(fi12)
        fi = fi + dfi
        lm = lm + dlm
        A = A + dA
        if A < 0:
            A+=np.pi*2
        elif A > 2*np.pi:
            A-=np.pi*2
    fi = np.rad2deg(fi)
    lm = np.rad2deg(lm)
    A = np.rad2deg(A + np.pi)
    return fi, lm, A
```

Algorytm Vincentego zwracający odległośc między punktami i azymuty w obie strony.

```python
def Vincenty(a, e2, fi, lm, fib, lmb):
    b = a * np.sqrt(1 - e2)
    f = 1 - (b / a)
    dlm = lmb - lm
    Ua = np.arctan((1 - f) * np.tan(fi))
    Ub = np.arctan((1 - f) * np.tan(fib))
    eps = np.deg2rad(0.000001 / 3600)
    Lp = dlm
    while True:
        sin_sigma = np.sqrt(
            (np.cos(Ub) * np.sin(Lp)) ** 2 + (
                        np.cos(Ua) * np.sin(Ub) - np.sin(Ua) * np.cos(Ub) * np.cos(Lp)) ** 2)
        cos_sigma = (np.sin(Ua) * np.sin(Ub)) + (np.cos(Ua) * np.cos(Ub) * np.cos(Lp))
        sigma = np.arctan(sin_sigma / cos_sigma)
        sin_alpha = (np.cos(Ua) * np.cos(Ub) * np.sin(Lp)) / sin_sigma
        cos2_alpha = 1 - (sin_alpha) ** 2
        cos2_sigma_m = cos_sigma - ((2 * np.sin(Ua) * np.sin(Ub)) / cos2_alpha)
        C = (((f / 16) * cos2_alpha) * (4 + f * (4 - 3 * cos2_alpha)))
        Lpo = dlm + (1 - C) * f * sin_alpha * (
                sigma + C * sin_sigma * (cos2_sigma_m + C * cos_sigma * (-1 + 2 * (cos2_sigma_m) ** 2)))
        if np.abs(Lpo - Lp) < eps:
            break
        else:
            Lp = Lpo
    u2 = ((a ** 2 - b ** 2) / b ** 2) * cos2_alpha
    A = 1 + u2 / 16384 * (4096 + u2 * (-768 + u2 * (320 - 175 * u2)))
    B = u2 / 1024 * (256 + u2 * (-128 + u2 * (74 - 47 * u2)))
    dsigma = B * sin_sigma * (cos2_sigma_m + 1 / 4 * B * (
                cos_sigma * (-1 + 2 * (cos2_sigma_m) ** 2) - 1 / 6 * B * cos2_sigma_m * ((
                    -3 + 4 * (sin_sigma) ** 2 )* (-3 + 4 * (cos2_sigma_m) ** 2))))
    sab = b * A * (sigma - dsigma)
```

Druga jego część polega na wyznaczeniu azymutów i określeniu ich ćwiartek.

```
    g1 = np.cos(Ub) * np.sin(Lpo)
    d1 = np.cos(Ua) * np.sin(Ub) - np.sin(Ua) * np.cos(Ub) * np.cos(Lpo)
    Aab = np.rad2deg(np.arctan(g1/ d1))
    if g1>0:
        if d1<0:
            Aab = 180 + Aab
    elif g1<0:
        if d1>0:
            Aab= 360 + Aab
        elif d1<0:
            Aab+=180
    if Aab > 360:
        Aab-=360
    elif Aab < 0:
        Aab+=360
    g2 = (np.cos(Ua) * np.sin(Lpo))
    d2 = (-np.sin(Ua) * np.cos(Ub) + np.cos(Ua) * np.sin(Ub) * np.cos(Lpo))
    Aba = np.rad2deg(np.arctan(g2 / d2)+ np.pi)
    if g2>0:
        if d2<0:
            Aba = 180 + Aba
    elif g1<0:
        if d1>0:
            Aba= 360 + Aba
        elif d1<0:
            Aba+=180
    if Aba > 360:
        Aba-=360
    elif Aba < 0:
        Aba+=360
    return sab, Aab, Aba
```

Funkcja licząca pole ze współrzędnych geodezyjnych:

```
def pole(a, e2, lm1, lm2, fi1, fi2):
    fi1 = np.deg2rad(fi1)
    lm1 = np.deg2rad(lm1)
    fi2 = np.deg2rad(fi2)
    lm2 = np.deg2rad(lm2)
    b = a * np.sqrt(1 - e2)
    e = np.sqrt(e2)
    a1 = (np.sin(fi1)/(1 - e2*np.sin(fi1)**2) + (1/(2*e))*np.log((1 + e*np.sin(fi1))/(1 - e*np.sin(fi1))))
    a2 = (np.sin(fi2)/(1 - e2*np.sin(fi2)**2) + (1/(2*e))*np.log((1 + e*np.sin(fi2))/(1 - e*np.sin(fi2))))
    p = b**2*(lm2 - lm1)/2*(a1 - a2)
    return p
```

```
#####Odcinek AD - długość i azymut
vin1 = Vincenty(a, e2, fia, lma, fid, lmd)
s = vin1[0]
A = vin1[1]
print('Azymut A-D:', round(A, 6))
print('odległość A-D:', round(s, 3), '[m]')
######PUNKT ŚREDNIEJ SZEROKOŚCI
print("1. punkt średniej szerokości: fi:", fis, 'lambda:', lms)
vin3 = Vincenty(a, e2, fis, lms, fid, lmd) #Azymut i odleglośc z punktu średniej szerokości do D
#######PUNKT ŚRODKOWY
kiv = Kivioj(a, e2, fia, lma, s/2, A)
fik = kiv[0]
lmk = kiv[1]
print("2. punkt środkowy szerokości (Kivioji): fi:", round(kiv[0], 6), 'lambda:', round(kiv[1], 6))
vin2 = Vincenty(a, e2, fik, lmk, fis, lms)
print("3. odległość między punktem środkowym i średniej szerokości (Vincenty): s:", round(vin2[0], 3), '[m] Azymut k - s:',
vin4 = Vincenty(a, e2, fik, lmk, fid, lmd) # Azymut i odleglośc z punktu środkowego do D\
###POLE
pol = pole(a, e2, lma, lmd, fia, fid)/1000000
print("4. pole obszaru wynosi: ", round(pol, 12), '[km^2]')
```

Wywołanie poszczególnych funkcji (powyżej) i wyświetlenie ich daje taki efekt:

```
Azymut A-D: 127.68147
odległość A-D: 45295.374 [m]
1. punkt średniej szerokości: fi: 50.125 lambda: 21.0
2. punkt środkowy szerokości (Kivioji): fi: 50.12527 lambda: 21.000651
3. odległość między punktem środkowym i średniej szerokości (Vincenty): s: 55.432 [m] Azymut k - s: 237.133178 Azymut s- k 57.132679
4. pole obszaru wynosi:  994.265196080189 [km^2]
```

Czyli odpowiedzi na pytania zadania.

Azymut A-D: 127.68147

odległość A-D: 45295.374 [m]

1. punkt średniej szerokości: fi: 50.125 lambda: 21.0

2. punkt środkowy szerokości (Kivioji): fi: 50.12527 lambda: 21.000651 – wyznaczony dzięki znajomości odległości między punktami A i D oraz azymutem, będących wynikiem wywołania algorytmu Vincentego dla tych punktów

3. odległość między punktem środkowym i średniej szerokości (Vincenty): s: 55.432 [m]
Azymut k - s: 237.133178 Azymut s- k 57.132679

4. pole obszaru wynosi: 994.265196080189 [km^2]

Wnioski:

- Punkt środkowy posiada inne współrzędne niż punkt średniej szerokości
- Możliwe jest policzenie pola powierzchni obszaru na elipsoidzie obrotowej posiadając jedynie współrzędne geodezyjne punktów.