SPRAWOZDANIE

Rozwiązywanie zadań geodezyjnych na elipsoidzie obrotowej.

Autor:

Weronika Hebda 311532

Politechnika Warszawska

Wydział: Geodezji i Kartografii

Kierunek: Geoinformatyka

Przedmiot: Wybrane zagadnienia geodezji wyższej

1. Wstęp teoretyczny

Ze względu na kulisty kształt Ziemi, który przyjmujemy jako elipsoida obrotowa, istnieją różne odwzorowania kartograficzne na mapie. Jednakże nie ważne w jakim odwzorowaniu byłaby stworzona mapa, najkrótsza odległość między punktami na mapie, nie jest tak naprawdę najkrótszą odległością na elipsoidzie. Z tego względu trasa samolotu zaprezentowana na mapie wydaje się skrzywiona. Istnieją zależności i definicje tych odcinków:

- Ortodroma – jest to najkrótsza możliwa odległość między dwoma punktami na elipsoidzie.
- Loksodroma – jest to najkrótsza możliwa odległość między dwoma punktami na mapie.
- Linia geodezyjna – linia łącząca dwa punkty po najkrótszej odległości, w przypadku sfery linia geodezyjna leży na ortodromie. Każda najkrótsza odległość to linia geodezyjna, ale nie każda linia geodezyjna to najkrótsza odległość.

Chcąc wyznaczyć środek między dwoma punktami, należy posługiwać się ortodromą. Ze względu na fakt, że jest to sfera, do obliczeń należy używać trygonometrii sferycznej. Wykorzystuje się dwa główne algorytmy:

1.1. Algorytm Vincentego

Służy do wyznaczenia odległości między dwoma punktami oraz azymutu wprost i azymutu odwrotnego dwóch danych punktów. Jest to algorytm rekurencyjny.

1.2. Algorytm Kivioja

Jest to algorytm oparty na całkowaniu numerycznym. Całkowanie numeryczne polega na przybliżonym obliczaniu całek oznaczonych, kiedy klasyczne obliczenie wartości całki nie jest możliwe. Najbardziej popularną metodą jest metoda prostokątów, która polega na dzieleniu pola pod wykresem na bardzo małe prostokąty. Z tej metody korzysta algorytm Kivioja, za którego pomocą można ze sporym przybliżeniem obliczyć punkt środkowy leżący na linii geodezyjnej, poprzez wzięcie połowy długości odcinka. Jest to algorytm iteracyjny, który z każdym przejściem pętli coraz bardziej „zbliża się” do szukanego punktu końcowego.

2. Cel ćwiczenia

Celem ćwiczenia było zapoznanie się z trygonometrią sferyczną oraz wyznaczenie mają dane czterech punktów na Ziemi:

- punktu środkowego między dwoma punktami używając algorytmu Vincentego i Kivioja.
- punkt średniej szerokości.
- różnicę odległości pomiędzy tymi punktami oraz ich azymuty.
- Pole powierzchni czworokąta tworzonego przez te punkty.

Do wykonania ćwiczenia użyto języka programowania Python w wersji 3.10 w środowisku PyCharm Community Edition.

3. Przebieg ćwiczenia

Na początku należało zmienić współrzędne na stopnie dziesiętne do obliczeń oraz wprowadzić dane dla elipsoidy GRS80.

```
#wspolrzedne punktow
# A = lam1,fi1 B = lam2,fi1
#C = lam1,fi2 D = lam2,fi2
lam1 = 20.75
fi1 = 50.25
lam2 = 21.25
fi2 = 50

#bedzie liczona odleglosc punkty C i B

#dane
a=6378137
e2=0.00669437999013
```

Rysunek 1. Wprowadzenie danych.

Najpierw obliczono punkt średniej szerokości, który ma współrzędne ze średniej arytmetycznej współrzędnych czterech punktów.

```
#punkt sredniej szerokosci
fi_aryt = (fi1 + fi2)/2
lam_aryt = (lam1 + lam2)/2
```

Rysunek 2. Obliczenie punktu średniej szerokości.

Obliczenia prowadzono dla odcinka AD, posługując się bibliotekami pythona: numpy i math.

Zaimplementowano algorytm Vincentego. Należało zwrócić uwagę na zmianę współrzędnych na radiany, warunek kończący iterację oraz na obliczenia azymutów uwzględniając ćwiartki.

```
#algorytm iteracyjny vincent
def vincent(lam1,fi1,lam2,fi2):
    Ua = m.atan((1 - f) * m.tan(m.radians(fi1)))
    Ub = m.atan((1 - f) * m.tan(m.radians(fi2)))
    delta_lam = lam2 - lam1
    L = m.radians(delta_lam)
```

Rysunek 3. 1 część skryptu algorytmu Vincentego.

```
if abs(m.degrees(L1-L)) * 3600 < 0.000001:
```

Rysunek 4. 2 część skryptu algorytmu Vincentego – warunek kończący iteracje.

```
x = m.cos(Ub) * m.sin(L_last)
y = m.cos(Ua) * m.sin(Ub) - m.sin(Ua) * m.cos(Ub) * m.cos(L_last)
if (x > 0 and y > 0):
    Aab = m.atan(x / y)
elif (x > 0 and y < 0):
    Aab = m.atan(x / y) + m.pi
elif (x < 0 and y < 0):
    Aab = m.atan(x / y) + m.pi
elif (x < 0 and y > 0):
    Aab = m.atan(x / y) + 2* m.pi

c = m.cos(Ua) * m.sin(L_last)
d = -m.sin(Ua) * m.cos(Ub) + m.sin(Ub) * m.cos(Ua) * m.cos(L_last)
if (c > 0 and d > 0):
    Aba = m.atan(c / d) + m.pi
elif (c > 0 and d < 0):
    Aba = m.atan(c / d) + 2* m.pi
elif (c < 0 and d < 0):
    Aba = m.atan(c / d) + 2* m.pi
elif (c < 0 and d > 0):
    Aba = m.atan(c / d) + 3* m.pi
break
```

Rysunek 5. 3 część skryptu algorytmu Vincentego - azymuty.

Mając odległość i azymut z algorytmu Vincenta, zaimplementowano algorytm Kivioja. Aby uzyskać wystarczającą dla nas dokładność w całkowaniu numerycznym na elipsoidzie ziemskiej należało podzielić odcinki na od 1000km do 1500km. Wybrano dystans 1000km w celu uzyskaniu jak największej dokładności. Dystans między punktami należało podzielić na połowę, ponieważ chcemy uzyskać punkt środkowy. Uwzględniono również odcinek ostatni, który jest mniejszy niż 1000km. Jest to reszta z dzielenia odcinka. Do implementacji algorytmu użyto tablic.

```python
#algorytm Kivioja
def M(phi):
    return (a*(1 - e2)) / m.sqrt((1 - e2*(m.sin(m.radians(phi))**2))**3)

def N(phi):
    return a / m.sqrt(1 - e2*(m.sin(m.radians(phi))**2))

Scb = Scb/2
ds = 1000
n = int(Scb/1000)
ostatni = Scb%1000

F = []
F.append(fi1)
A = []
A.append(Aab)
L = []
L.append(lam1)

#ds to jeden fragment
for i in range(0,n+1):
    dfi = ds*m.cos(A[i])/M(F[i])
    dA = m.sin(A[i])*m.tan(m.radians(F[i]))*ds/N(F[i])
```

Rysunek 6. Część algorytmu Kivioja.

Potem obliczono pole czworokąta według wzoru.

```python
#obliczanie Pola czworokąta
e = m.sqrt(e2)
fi1 = np.deg2rad(fi1)
fi2 = np.deg2rad(fi2)
lam1 = np.deg2rad(lam1)
lam2 = np.deg2rad(lam2)
P = (b**2*(lam2-lam1)/2)*(((m.sin(fi2)/(1-e2*(m.sin(fi2)**2)))+(1/(2*e))*m.log((1+e*m.sin(fi2))/(1-e*m.sin(fi2))))
- (((m.sin(fi1)/(1-e2*(m.sin(fi1)**2)))+(1/(2*e))*m.log((1+e*m.sin(fi1))/(1-e*m.sin(fi1)))))
P = abs(P)
```

Rysunek 7.Obliczanie pola czworokąta.

Wymagana dokładność wynosi 1mm, czyli 0,00001’’. Napisano funkcję zaokrąglającą i wyświetlającą wyniki w stopniach, minutach i sekundach.

Dodatkowo obliczono punkt środkowy odcinka CB.

4. Wyniki

Używając funkcji print() wyświetlono wyniki w terminalu.

```
Punkt średniej szerokości: 50° 07' 30.0'' , 21° 00' 0.0''
Wspolrzedne punktu srodkowego AD - E: 50° 07' 30.97362'' , 21° 00' 2.34392''
Azymuty AD: 127° 40' 53.29256'' , 308° 03' 54.70041''
Odleglosc między punktem środkowym AD i punktem średniej szerokości: 55.432 m
Azymut między punktem E i D: 127° 52' 26.42329''
Azymut między punktem średniej szerokości a D: 127° 44' 28.41644''
Pole prostokąta: 994265196.074311 m²
*Wspolrzedne punktu srodkowego CB: 50° 07' 30.97363'' , 20° 59' 57.65607''
```

Rysunek 8.Wyniki skryptu.

5. Wnioski

5.1. Jest dość spora różnica między faktycznym punktem środkowym, a punktem obliczonym ze średniej arytmetycznej współrzędnych. Nie są one tym samym punktem.

5.2. Jest widoczna różnica między azymutami punktu środkowego, a punktu średniej szerokości.

5.3. Algorytmy Kiviojego i Vincentego pozwalają na dokładne wyznaczenie punktu środkowego między dwoma punktami na elipsoidzie.

5.4. Po dodatkowym sprawdzeniu punkt środkowy między punktami A i D różni się od punktu środkowego między C i B. Różnica wynika ze zmniejszania się długości równoleżników w kierunku biegunów.