SPRAWOZDANIE Z ĆWICZENIA 4:

Odwzorowanie Gaussa-Krügera: układy współrzędnych płaskich stosowanych w Polsce

Maja Kret
nr 1, gr. 2
325693

Wydział Geodezji i Kartografii
Politechnika Warszawska

Warszawa, 23 grudnia 2023

Spis treści

1 Cel ćwiczenia

Celem ćwiczenia jest transformacja współrzędnych geodezyjnych punktów do układów współrzędnych płaskich stosowanych w Polsce. Następnie należy wyznaczyć długości, azymuty odcinków oraz pole trapezu złożonego z tych odcinków.

2 Wstęp teoretyczny

2.1 Układ współrzędnych płaskich PL-2000

Odwzorowanie Gaussa-Krügera elipsoidy globalnej WGS84. Układ składa się z 4 stref numerowanych od 5 do 8. Każda ze stref obejmuje trzystopniowy pas. Stosowany na potrzeby wykonania map w skali większej niż 1:10 000. Zniekształcenia oscylują w obrębie wartości $-7,7cm/km$ do $+7cm/km$ Współrzędne analizowane w tym ćwiczeniu zawierają się w strefie 5 z południkiem osiowym 15°.

2.2 Układ współrzędnych płaskich PL-1992

Odwzorowanie Gaussa-Krügera elipsoidy lokalnej GR80. Układ ten jest jednolity i obejmuje cały obszar Polski. Południkiem środkowym jest południk 19°, a skala podobieństwa wynosi 0.9993. Zniekształcenie na południku osiowym wynosi $-70cm/km$, a an skrajnie wschodnich obszarach kraju dochodzi do $+90cm/km$. Ze względu na duże wartości zniekształceń stosowany jest do map w skali 1:10 000 i mniejszych.

2.3 Układ współrzędnych płaskich UTM

Universal Transverse Mercator - uniwersalne poprzeczne odwzorowanie Merkatora jest stosowane na całym świecie do celów nawigacyjnych i wojskowych. Jest do odwzorowanie w pasach 6°, ze skalą na południku środkowym 0.9996. Polska znajduje się w pasach 33, 34 i 35.

2.4 Układ współrzędnych płaskich LAEA

Lambert Azimuthal Equal Area - układ współrzędnych płaskich utworzony na podstawie przyporządkowania punktów na elipsoidzie GRS80 według teorii azymutalnego odwzorowania Lamberta. Układ LAEA używany jest w mapach o skalach 1:500 000 i mniejszych. Jest dobry do pomiaru długości i powierzchni, ale nieodpowiedni do pomiaru kątów i kierunków.

2.5 Układ współrzędnych płaskich LCC

Lambert Conformal Conic jest opary na odwzorowaniu stożkowym Lamberta. Stosowany jest na potrzeby wydawania map w skalach 1:500 000 i mniejszych gdy ważne jest zachowanie prawdziwego kształtu. Nadaje się do pomiarów kątów i kierunków, ale nie do pomiaru długości i powierzchni.

3 Dane do ćwiczenia

Układ	EPSG	Zakres
PL-2000	2176	Polska - strefa 5
PL-1992	2180	Polska
UTM	32633	Świat - strefa 33
LAEA	3035	Europa
LCC	3034	Europa

Tabela 1: Układy współrzędnych płaskich i ich kod EPSG

nr	φ	λ
1	53°45′00.00000″	15°15′00.00000″
2	54°06′33.75763″	15°15′00.00000″
3	54°05′58.80144″	16°46′43.41406″
4	53°44′25.04170″	16°46′43.41406″

Tabela 2: Współrzędne analizowanych punktów

4 Przebieg ćwiczenia

1. **Transformacje współrzędnych:** Wszystkie współrzędne zostały poddane transformacji do układów współrzędnych płaskich za pomocą biblioteki pyproj (kod źródłowy 2). Metoda przeliczenia bazuje na kodach EPSG układu wejściowego i wyjściowego i zwraca wynik w metrach.
2. **Redukcje długości:** Dokonano redukcji długości (kod źródłowy 3), więc dokonano obliczeń:
   (a) długości odcinków na płaszczyźnie układu PL-2000
   (b) długości odcinków na płaszczyźnie Gaussa-Krügera
   (c) redukcji długości
   (d) ostatecznej długości na elipsoidzie.
3. **Redukcje azymutów:** Dokonano redukcji azymutów (kod źródłowy 3), dokonując obliczeń:
   (a) kątów kierunkowych w współrzędnych płaskich
   (b) zbieżności południków
   (c) redukcji kierunków
   (d) ostatecznego azymutu na elipsoidzie.
4. **Pole trapezu:** Pole powierzchni zostało obliczone za pomocą wzorów Gaussa (kod źródłowy 2) dla wszystkich analizowanych układów współrzędnych.
5. **Stworzenie tabel:** Wszystkie grupy danych zostały zapisane do tabel biblioteki `pandas` w celu dodania do sprawozdania.

5 Wyniki ćwiczenia

5.1 Transformacje z biblioteką pyproj

nr	x	y
1	5516490.727	5957660.601
2	5516349.663	5997657.405
3	5616347.760	5998010.911
4	5617351.461	5958019.885

Tabela 3: Współrzędne punktów w układzie PL-2000

nr	x	y
1	252846.028	660446.270
2	254962.295	700391.989
3	354799.469	695092.080
4	353546.705	655129.270

Tabela 4: Współrzędne punktów w układzie PL-1992

nr	x	y
1	516485.400	5955736.129
2	516344.382	5995720.012
3	616310.177	5996073.405
4	617313.554	5956095.297

Tabela 5: Współrzędne punktów w układzie UTM

nr	x	y
1	4667032.566	3417280.404
2	4664091.927	3457181.890
3	4763796.950	3464409.230
4	4767591.881	3424568.613

Tabela 6: Współrzędne punktów w układzie LAEA

nr	x	y
1	4334584.794	3000070.553
2	4331838.799	3038655.338
3	4428320.820	3045476.812
4	4431864.387	3006957.453

Tabela 7: Współrzędne punktów w układzie LCC

Tabele 3 -7 przedstawiają współrzędne punktów czworokąta w układach współrzędnych płaskich obowiązujących w Polsce. Współrzędne wyrażone są w metrach i zostały obliczone za pomocą biblioteki

pyproj.

5.2 Redukcje długości

Układ	Dł. dana $[m]$	poprawka [cm/km]	poprawka względna [cm]	poprawka względna [m]
PL- 2000	40000.000	$-7,7 - +7$	$-308 - +280$	$-3.08 - +2.80$
	100000.000	$-7,7 - +7$	$-770 - +700$	$-7.70 - +7.00$
PL-1992	40000.000	$-70 - +90$	$-2800 - +3600$	$-28.00 - +36.00$
	100000.000	$-70 - +90$	$-7000 - +9000$	$-70.00 - +90.00$

Tabela 8: Oczekiwane oscylacje w współrzędnych

W tabeli 8 przedstawiono zniekształcenia współrzędnych w obrębie jednej strefy w układzie PL-2000. Analizowane punkty 3 i 4 przechodzą już na strefę 6, co oznacza, że zniekształcenie długości do i od tych punktów może być większe. Najmniejszych zniekształceń można się spodziewać przy linii 1 - 2, która całkowicie leży wewnątrz, ale nie w samym środku strefy 5.

W tabeli zawarto również zniekształcenia długości dla układu PL-2000. Pomiędzy południkami 15° a 17° spodziewamy się zniekształceń w obrębie 0 do -40 cm na km.

A-B	Dł. dane	Dł.Vincenty	PL-2000	PL-1992
1 - 2	40000.000	40000.000	40000.020	40018.352
2 - 3	100000.000	100000.000	100006.118	100019.479
3 - 4	40000.000	40000.000	40006.588	39999.022
4 - 1	100000.000	100862.551	100868.882	100882.559

Tabela 9: Długości odcinków na elipsoidzie

W tabeli 9 dokonano porównania długości linii w kolumnach przedstawiających kolejno:

1. Długości linii dane w ćwiczeniu 3
2. Długości linii zamkniętego trapezu obliczonych w ćwiczeniu 3 za pomocą algorytmu Vincentego
3. Długości zredukowane z płaszczyzny układu PL-2000 na elipsoidę
4. Długości zredukowane z płaszczyzny układu PL-1992 na elipsoidę

5.3 Redukcje azymutów

A-B	Az. dany	Az. Kivioj	Az. Vincenty A-B	Az. Vincenty B-A
1 - 2	0°00′00.00000″	0°00′00.00000″	0°00′00.00000″	180°00′00.00000″
2 - 3	90°00′00.00000″	91°14′18.34038″	90°00′00.00000″	271°14′18.34038″
3 - 4	180°00′00.00000″	180°00′00.00000″	180°00′00.00000″	360°00′00.00000″
4 - 1	270°00′00.00000″	269°59′51.09899″	271°13′49.11409″	89°59′51.09899″

Tabela 10: Azymuty odcinków z ćwiczenia 3

A-B	Azymut A-B	Azymut B-A
1 - 2	90°24′11.77172″	270°24′17.45732″
2 - 3	0°26′53.23813″	180°21′55.40451″
3 - 4	272°52′47.13538″	92°52′13.50021″
4 - 1	181°20′10.55727″	2°00′03.55518″

Tabela 11: Azymuty odcinków w układzie PL-2000

nr	Azymut A-B	Azymut B-A
1 - 2	87°10′31.05466″	267°09′58.51621″
2 - 3	357°12′31.12435″	177°07′23.71614″
3 - 4	269°38′38.92025″	89°38′27.86829″
4 - 1	178°06′26.67372″	358°46′36.23174″

Tabela 12: Azymuty odcinków w układzie PL-1992

Tabela 10 przedstawia azymuty dane w ćwiczeniu 3 oraz te wyliczone za pomocą algorytmu Kivioja i Vincentego.

W wartościach azymutów w tabelach 11 i 12 występują odstępstwa na poziomie od 30″ do 2°.

5.4 Pole trapezu

Układ	Pole $[m^2]$	Pole $[km^2]$
WGS 84	4016880873.853	4016.881
PL-2000	4016769500.134	4016.770
PL-1992	4015113362.314	4015.113
UTM	4014174886.310	4014.175
LAEA	4016817517.040	4016.818
LCC	3756510722.070	3756.511

Tabela 13: Pole powierzchni w układach współrzędnych płaskich

W tabeli 13 przedstawiono wyniki obliczeń pól powierzchni dla układów współrzędnych płaskich. Wszystkie obliczenia zostały wykonane za pomocą metody Gaussa. Pierwszy wiersz przedstawia pole obliczone w ramach ćwiczenia 3, które uzyskano z użyciem funkcji `geometry_area_perimeter`. Najbardziej zbliżone wyniki uzyskano dla układów PL-2000 i LAEA. Wynik dla układu LCC jest najbardziej odległy od pozostałych. Jest to spowodowane stożkowym sposobem odwzorowania powierzchni na płaszczyznę, które nie zachowuje powierzchni ani odległości.

6 Wnioski

Na wybór układu współrzędnych płaskich wpływa wiele czynników. Dla Polski, obowiązującym i najdokładniejszym jest układ PL-2000, ale przy mapach w skali 1:10 000 i mniejszych zalecany jest układ PL-1992. Układy LAEA i LCC są stosowane w całej Europie i stanowią ważny element systemu odniesień przestrzennych. Układ UTM jest uniwersalny i stosowany na całym świecie.

Warto zwrócić uwagę na odstępstwa w obliczonych długościach i polach powierzchni, które są spowodowane odwzorowaniem ich na płaszczyznę. Na ich dokładność będzie bezpośrednio wpływać wybór układu współrzędnych płaskich. Innym aspektem są azymuty, które przy odwzorowaniu Gaussa-Krügera mogą być zniekształcone nawet o ponad 1°. Takie błędy mogą mieć duży wpływ na pomiary kątów i kierunków.

7 Kod źródłowy

Kod źródłowy 1: Zamiany jednostek kątowych

```
degree_sign = u"\N{DEGREE SIGN}"
# Radiany na stopnie, minuty, sekundy
def rad2dms(rad):
    dd = np.rad2deg(rad)
    dd = dd
    deg = int(np.trunc(dd))
    mnt = int(np.trunc((dd-deg) * 60))
    sec = ((dd-deg) * 60 - mnt) * 60
    mnt = abs(mnt)
    sec = abs(sec)
    if sec > 59.99999:
        sec = 0
        mnt += 1
    if sec < 10:
        sec = f"0{sec:.5f}"
    else:
        sec = f"{sec:.5f}"
    if mnt < 10:
        mnt = f"0{mnt}"
    dms = (f"{deg}{degree_sign} {mnt}' {sec}''")
    return dms

# Stopnie dziesiętne na stopnie, minuty, sekundy
def deg2dms(dd):
    deg = int(np.trunc(dd))
    mnt = int(np.trunc((dd-deg) * 60))
    sec = ((dd-deg) * 60 - mnt) * 60
    mnt = abs(mnt)
    sec = abs(sec)
    if sec < 10:
        sec = f"0{sec:.5f}"
    else:
        sec = f"{sec:.5f}"
    if mnt < 10:
        mnt = f"0{mnt}"
    dms = (f"{deg}{degree_sign} {mnt}' {sec}''")
```

Kod źródłowy 2: Transformacje współrzędnych i pole powierzchni

```
import numpy as np
from pyproj import Proj, transform, CRS, Transformer, Geod
import plotly.graph_objects as go
import plotly.io as pio
import matplotlib.pyplot as plt
import pandas as pd
from shapely.geometry import LineString, Point, Polygon

# Kody układów współrzędnych
input_code = 4326
output_names = ['PL-2000', 'PL-92', 'UTM', 'LAEA', 'LCC']
output_codes = [2176, 2180, 32633, 3035, 3034]

input_proj = CRS.from_epsg(4326)
proj_2000 = CRS.from_epsg(2176)
proj_92 = CRS.from_epsg(2180)
proj_utm = CRS.from_epsg(32633)
proj_laea = CRS.from_epsg(3035)
proj_lcc = CRS.from_epsg(3034)

output_projections = [proj_2000, proj_92, proj_utm, proj_laea, proj_lcc]

# Współrzędne punktów
phis = [53.75, 54.10937712005116, 54.099667067097876, 53.74028936233757]
lambdas = [15.25, 15.25, 16.778726126645378, 16.778726126645378]
lamb0 = np.radians(15)

x1_out, x2_out, x3_out, x4_out = [], [], [], []
y1_out, y2_out, y3_out, y4_out = [], [], [], []

x_out = [x1_out, x2_out, x3_out, x4_out]
y_out = [y1_out, y2_out, y3_out, y4_out]

# Pętla transformacji
for proj in output_codes:
    for i in range(4):
        transformer = Transformer.from_crs(input_code, proj)
        y, x = transformer.transform(phis[i], lambdas[i])
        x_out[i].append(x)
        y_out[i].append(y)

        P2176 = 0
        P2180 = 0
        P32633 = 0
        P3035 = 0
        P3034 = 0

        areas_epsg = [P2176, P2180, P32633, P3035, P3034]
```

```
        P2176_con = 0
        P2180_con = 0
        P32633_con = 0
        P3035_con = 0
        P3034_con = 0
        areas_con_epsg = [P2176_con, P2180_con, P32633_con, P3035_con,
P3034_con]

        # Pętla pola powierzchni
        for u in range(5):
            for i in range(4):
                # kij
                j = (i + 1) % 4
                k = (i - 1) % 4
                # Gauss
                area = x_out[i][u] * (y_out[j][u] - y_out[k][u])
                area_con = y_out[i][u] * (x_out[j][u] - x_out[k][u])
                areas_epsg[u] += area
                areas_con_epsg[u] += area_con

        areas_epsg = [round(abs(areas_epsg[i] / 2), 3) for i in range(5)]
        areas_con_epsg = [round(abs(areas_con_epsg[i] / 2), 3) for i in range
(5)]
```

Kod źródłowy 3: Redukcje odległości i azymutów

```
# Parametry elipsoidy
a = 6378137.0
a2 = a ** 2
e2 = 0.00669438002290
b2 = a2 * (1 - e2)
e22 = (a2- b2)  / b2

# Współczynnik zniekształcenia
m2000 = 0.999923
m1992 = 0.9993

# Promienie krzywizny
def M_and_N(phi):
    sin_phi = np.sin(phi)
    M = a * (1 - e2) / (1 - e2 * sin_phi**2)**(3/2)
    N = a / np.sqrt(1 - e2 * sin_phi**2)
    return M, N

lengths_elip2000 = []
lengths_elip1992 = []

azimuths_2000 = []
azimuths_back_2000 = []
azimuths_1992 = []
```

```
    azimuths_back_1992 = []

    P_1_2000 = 0
    P_2_2000 = 0
    P_1_1992 = 0
    P_2_1992 = 0
    areas_table = [P_1_2000, P_2_2000, P_1_1992, P_2_1992]

    phis = np.deg2rad(phis)
    lambdas = np.deg2rad(lambdas)

    # Redukcje
    for i in range(4):
        # kij
        j = (i + 1) % 4
        k = (i - 1) % 4

        # Redukcja długości
        phi_m = (phis[i] + phis[j]) / 2
        M, N = M_and_N(phi_m)
        Rm = np.sqrt(M * N)

        # Długość odcinka na płaszczyźnie
        length_2000 = np.sqrt((x_out[i][0] - x_out[j][0])**2 +
      (y_out[i][0] - y_out[j][0])**2)
        length_1992 = np.sqrt((x_out[i][1] - x_out[j][1])**2 +
      (y_out[i][1] - y_out[j][1])**2)

        # Długość odcinka na płaszczyźnie Gaussa-Krugera
        length_gk2000 = length_2000 / m2000
        length_gk1992 = length_1992 / m1992

        # Redukcja długości
        r2000 = length_gk2000 /10000  * (y_out[i][0] ** 2 + y_out[i][0] *
      y_out[j][0] + y_out[j][0] ** 2) / (6 * Rm ** 2)
        r1992 = length_gk1992 /10000 * (y_out[i][1] ** 2 + y_out[i][1] *
      y_out[j][1] + y_out[j][1] ** 2) / (6 * Rm ** 2)

        # Długość odcinka na elipsoidzie
        length_elip2000 = length_gk2000 - r2000 / 100
        length_elip1992 = length_gk1992 - r1992 / 100
        lengths_elip2000.append(round(length_elip2000, 3))
        lengths_elip1992.append(round(length_elip1992, 3))

        # Redukcja azymutów
        d_lambda = lambdas[i] - lamb0
        t = np.tan(phis[0])
        eta2 = e22 * np.cos(phis[i]) ** 2

```

```
74        for u in range(2):
75            # Kąt kierunkowy
76            delta_x = x_out[j][u] - x_out[i][u]
77            delta_y = y_out[j][u] - y_out[i][u]
78            alpha_ab = np.arctan2(delta_y, delta_x)
79            delta_x = x_out[i][u] - x_out[j][u]
80            delta_y = y_out[i][u] - y_out[j][u]
81            alpha_ba = np.arctan2(delta_y, delta_x)
82
83            # Zbieżność południków
84            gamma_a = (d_lambda * np.sin(phis[i])) + ((d_lambda ** 3 / 3) *
85          np.sin(phis[i]) * np.cos(phis[i]) ** 2 * (1 + 3 * eta2 + 2 * eta2 ** 2))
86          + ((d_lambda ** 5 / 15) * np.sin(phis[i]) * np.cos(phis[i]) ** 4
87          * (2 - t ** 2))
88            gamma_b = (d_lambda * np.sin(phis[j])) + ((d_lambda ** 3 / 3) *
89          np.sin(phis[j]) * np.cos(phis[j]) ** 2 * (1 + 3 * eta2 + 2 * eta2 ** 2))
90          + ((d_lambda ** 5 / 15) * np.sin(phis[j]) * np.cos(phis[j]) ** 4
91          * (2 - t ** 2))
92
93            # Redukcja kierunków
94            delta_ab = (x_out[j][u] - x_out[i][u]) *
95          (2 * y_out[u][i] + y_out[u][j]) / (6 * Rm ** 2)
96            delta_ba = (x_out[i][u] - x_out[j][u]) *
97          (2 * y_out[u][j] +  y_out[u][i]) / (6 * Rm ** 2)
98
99            # Azymut odcinka
100           A_ab = alpha_ab + gamma_a + delta_ab
101           A_ba = alpha_ba + gamma_b + delta_ba
102           A_ab = np.degrees(A_ab)
103           A_ba = np.degrees(A_ba)
104           if A_ab < 0:
105             A_ab += 360
106           if A_ba < 0:
107             A_ba += 360
108
109           if u == 0:
110               azimuths_2000.append(A_ab)
111               azimuths_back_2000.append(A_ba)
112           elif u == 1:
113               azimuths_1992.append(A_ab)
114               azimuths_back_1992.append(A_ba)
```

Spis tabel

Spis kodów źródłowych