Odwzorowanie Gaussa-Krügera: układy współrzędnych płaskich stosowanych w Polsce

Autor: Julia Zapała
Numer indeksu: 325710
grupa: GI II

1. Dane do zadania (na podstawie zadania 3)

Tabela współrzędnych punktów

Numer punktu	Szerokość geograficzna [° ' "]	Długość geograficzna [° ' "]
1	51 15 0.0	17 45 0.0
2	51 36 34.30649	17 45 0.0
3	51 36 2.36215	19 11 36.08424
4	51 14 28.0537	19 11 36.08424

2. Cel zadania

Celem zadania było przeliczenie współrzędnych geodezyjnych z zadania 3. na układ współrzędnych płaskich PL - 2000, PL-1992 i PL-LAEA, UTM i LCC, wykonanie redukcji długości i azymutu dla dwóch układów oraz obliczenie pól powstałych figur.

3. Wstęp teoretyczny

W geodezji definiujemy wiele układów odniesień, każdy z nich różni się od siebie i jest używany do różnych celów. Układ odniesienia to zbiór parametrów definiujących położenie układu współrzędnych w stosunku do bryły Ziemi poprzez wyznaczenie np. początku układu, skali i orientacji osi układu współrzędnych. Układ współrzędnych prostokątnych płaskich PL-2000 oparty jest na odwzorowaniu Gaussa-Krügera. Obszar Polski podzielony jest na 4 pasy południkowe, o szerokości 3° każdy, oznaczone odpowiednio numerami 5, 6, 7 i 8. Podobnie układ współrzędnych prostokątnych płaskich PL-1992 oparty jest na odwzorowaniu G-K. Obszar Polski objęty jest jednym pasem południkowym o szerokości 10°. Pierwszy z układów służy do opracowań w skalach większych od 1:10000 (np. mapy zasadniczej). Układ PL-1992 służy do opracowań w skali 1:10000 i skalach mniejszych oraz do tworzenia Topograficznej Bazy Danych.

4. Realizacja zadania

Zadanie zostało zrealizowane za pomocą programu napisanego w python. W celu przeliczenia współrzędnych posłużono się funkcją biblioteki *pyproj* oraz kodami EPSG (2177 dla układu PL-2000 (pas południkowy nr 6), 2180 dla układu PL-1992, 3035 dla PL-LAEA, 32633 dla UTM i 3034 dla LCC**).** W ten sposób otrzymano punkty w poszczególnych układach, które wykorzystano w dalszych obliczeniach. Dodatkowo punkty z układu PL-2000 i PL-1992 przeliczono na płaszczyznę odwzorowań G-K (przy przeliczaniu układu PL-2000 użyto pasa numer 6.) Następnie wykonano redukcję długości na powierzchni układu PL-2000 i PL-1992. Zaczęto od obliczenia długości odcinków między parami punktów za pomocą tw. Pitagorasa, dalej obliczono długości odcinka na płaszczyźnie G-K.

Obliczono redukcję, dzięki której było możliwe obliczenie długości odcinka na elipsoidzie. Wyniki porównano z odległościami z zadania 3. między punktami obliczonymi na podstawie współrzędnych geodezyjnych za pomocą zadania odwrotnego (funkcja inv). Kolejną częścią zadania było obliczenie redukcji azymutów w układzie PL-2000 i PL-1992. Zadanie rozpoczęto od obliczenia kąta kierunkowego dla par punktów. Następnie n apodstawie współrzędnych geodezyjnych policzono zbieżność południków γ. Obliczono redukcję azymutów i zsumowano otrzymane wyniki. Dodatkowo wyniki porównano z wynikami otrzymanymi w zadaniu 3. za pomocą algorytmu Vincentego. Na koniec policzono pole dla współrzędnych w każdym z omawianych układów za pomocą wzorów Gaussa. Pola porównano z polem otrzymanym w zadaniu 3.

5. Wykonanie zadania i wnioski

a) Przeliczenie współrzędnych geodezyjnych na układ współrzędnych płaskich PL-2000, PL-1992 i PL-LAEA, UTM i LCC

Przeliczenie współrzędnych rozpoczęto od wyboru odpowiednich kodów EPSG, a następnie użyto pętli do przeliczenia wartości. Otrzymane wyniki zostały przedstawione w tabelach poniżej.

	Układ 2000	
Numer	Współrzędna X	Współrzędna Y
Punkt 1	5679490.733893	6482546.413918
Punkt 2	5719487.569540	6482682.861373
Punkt 3	5719145.585229	6582676.910712
Punkt 4	5679150.585573	6583328.212653

	Układ 1992	
Numer	Współrzędna X	Współrzędna Y
Punkt 1	376664.446268	412787.872876
Punkt 2	416634.337624	413469.754734
Punkt 3	414925.599780	513387.809870
Punkt 4	374953.650138	513493.263049

	Punkt na plaszczyźnie odwzorowania gk z układu 2000	
Numer	Współrzędna X	Współrzędna Y
Punkty 1	5679928.088355	983045.608430
Punkty 2	5719928.003996	983182.066392
Punkty 3	5719585.993350	1083183.815866
Punkty 4	5679587.913842	1083835.167961

	Punkt na plaszczyźnie odwzorowania gk z układu 1992	
Numer	Współrzędna X	Współrzędna Y
Punkty 1	5680640.894895	-87273.218377
Punkty 2	5720638.784773	-86590.858867
Punkty 3	5718928.849975	13397.187902
Punkty 4	5678928.900369	13502.714949

	Układ LAEA		
Numer	Współrzędna X	Współrzędna Y
Punkt 1	3155226.958240	4860936.561155
Punkt 2	3195037.445864	4856707.067642
Punkt 3	3205643.352429	4956038.154417
Punkt 4	3165909.990623	4961052.256946

	Układ UTM		
Numer	Współrzędna X	Współrzędna Y
Punkt 1	691911.660576	5681219.462032
Punkt 2	690410.457271	5721193.230823
Punkt 3	790371.575005	5724957.662784
Punkt 4	792661.973999	5684997.506558

	Układ LCC		
Numer	Współrzędna X	Współrzędna Y
Punkt 1	2746790.990574	4521557.389394
Punkt 2	2785205.751989	4517513.593742
Punkt 3	2795305.434595	4613554.934487
Punkt 4	2756977.044134	4618348.254782

b) Wykonanie redukcji długości

Redukcję długości wykonano dla dwóch układów: PL-2000 i PL-1992. Za pomocą Tw. Pitagorasa zostały obliczone odpowiednie długości w odpowiednich układach, a następnie wykonano redukcję, której wyniki przedstawiono poniżej.

Odległość odcinka na elipsoidzie w układzie PL-200 [km]	
Odległość 1-2	39.523456
Odległość 2-3	98.680484
Odległość 3-4	39.422600
Odległość 4 -1	99.464289

Odległość odcinka na elipsoidzie w układzie PL-1992 [km]	
Odległość 1-2	39.975092
Odległość 2-3	99.937678
Odległość 3-4	39.975077
Odległość 4 -1	100.724936

Odległość odcinka z zadania 3. [km]	
Odległość 1-2	40.000000
Odległość 2-3	100.000001
Odległość 3-4	40.000000
Odległość 4 -1	100.787744

Odległości we wszystkich układach różnią się od siebie, jednak są dość zbliżone do oczekiwanych wartości. Wskazuje to na poprawność obliczeń.

c) Wykonanie redukcji azymutu

Redukcję azymutów wykonano dla dwóch układów, PL-2000 i Pl-1992. Rozpoczęto od policzenia kąta kierunkowego, γ i δ. Wyniki zsumowano. Porównano je z azymutami otrzymanymi w zadaniu 3.

Azymuty układ PL-2000	
Azymut 1-2	0° 01’ 41.77371”
Azymut 2-3	89° 54’ 30.93779”
Azymut 3-4	179° 01’ 52.45641”
Azymut 4-1	270° 11’ 36.74951”

Azymuty układ PL-1992	
Azymut 1-2	0° 46’ 47.8348”
Azymut 2-3	90° 41’ 33.69616”
Azymut 3-4	179° 50’ 36.10091”
Azymut 4-1	270° 58’ 23.31013”

Azymuty z zadania 3.	
Azymut 1-2	0° 00’ 00.0”
Azymut 2-3	90° 00’ 00.0”
Azymut 3-4	180° 00’ 00.0”
Azymut 4-1	271° 07’ 26.42735”

Otrzymane azymuty różnią się od siebie, ale różnica ta jest niewielka. Wartości są zbliżone do oczekiwanych wyników.

d) Obliczenie pola powstałych figur

Na koniec z wykorzystaniem wzoru Gaussa obliczono pola powstałych figur we wszystkich z omawianych układach odniesień.

Układ	Pole [km²]
Pole z zadania 3.	4015.461980
PL-2000	4015.019343
PL-1992	4010.044671
Płaszczyzna odwzorowań G-K dla PL-2000	4015.637728
Płaszczyzna odwzorowań G-K dla PL-1992	4015.664633
PL-LAEA	4015.387798
UTM	4018.062640
LCC	3744.454671

Pola policzone w powyższych układach różnią się od siebie, jednak różnice są niewielkie. Wskazuje to na wiernopolowość odwzorowania prawie wszystkich układów. Jedynie układ LCC nie zachowuje wiernopolowości.

6. Kod źródłowy

```
from pyproj import Proj, transform, CRS, Transformer, Geod
import numpy as np
import math
from shapely.geometry import Point, Polygon
from tabulate import tabulate
from geographiclib.geodesic import Geodesic

a = 6378137
e2 = 0.00669438002290
punkty = [(51.25, 17.75), (51.60952957952263, 17.75), (51.600656153534366, 19.193356733106977),
          (51.2411260266428, 19.193356733106977)]

# transformacje punktów między układami
punkty_pl2000 = []
input_proj = CRS.from_epsg(4326)
output_proj = CRS.from_epsg(2177)  # pas południkowy nr 6
for lon, lat in punkty:
    x, y = Transformer.from_proj(input_proj, output_proj).transform(lon, lat)
    punkty_pl2000.append((x, y))
table_data = [(f"Punkt {i}", format(x, '.6f'), format(y, '.6f')) for i, (x, y) in enumerate(punkty_pl2000, start=1)]
table_headers = ["", "Układ 2000", ""]
table_data.insert(0, ("Numer", "Współrzędna X", "Współrzędna Y"))
print(tabulate(table_data, headers=table_headers, tablefmt="pretty"))

uklad_1992 = []
output_proj1 = CRS.from_epsg(2180)
for lon, lat in punkty:
    x, y = Transformer.from_proj(input_proj, output_proj1).transform(lon, lat)
    uklad_1992.append((x, y))
table_data = [(f"Punkt {i}", format(x, '.6f'), format(y, '.6f')) for i, (x, y) in enumerate(uklad_1992, start=1)]
table_headers = ["", "Układ 1992", ""]
table_data.insert(0, ("Numer", "Współrzędna X", "Współrzędna Y"))
print(tabulate(table_data, headers=table_headers, tablefmt="pretty"))
```

```
m0_2000 = 0.999923
nr = 6
gk_2000=[]
for i in range(len(punkty_pl2000)):
    x_gk_2000 = punkty_pl2000[i][0]/m0_2000
    y_gk_2000 = punkty_pl2000[i][1]/m0_2000-nr* 1000000 + 500000
    gk_2000.append((x_gk_2000, y_gk_2000))
table_data = [(f"Punkty {i}", format(x_gk_2000, '.6f'), format(y_gk_2000, '.6f')) for i,
              (x_gk_2000, y_gk_2000) in enumerate(gk_2000, start=1)]
table_headers = ["", "Punkt na płaszczyźnie odwzorowania gk z układu 2000", ""]
table_data.insert(0, ("Numer", "Współrzędna X", "Współrzędna Y"))
print(tabulate(table_data, headers=table_headers, tablefmt="pretty"))
```

```
m0_1992 = 0.9993
gk_1992=[]
for i in range(len(uklad_1992)):
    x_gk_1992= (uklad_1992[i][0]+5300000)/0.9993
    y_gk_1992= (uklad_1992[i][1]-500000)/0.9993
    gk_1992.append((x_gk_1992, y_gk_1992))
table_data = [(f"Punkty {i}", format(x_gk_1992, '.6f'), format(y_gk_1992, '.6f')) for i,
              (x_gk_1992, y_gk_1992) in enumerate(gk_1992, start=1)]
table_headers = ["", "Punkt na płaszczyźnie odwzorowania gk z układu 1992", ""]
table_data.insert(0, ("Numer", "Współrzędna X", "Współrzędna Y"))
print(tabulate(table_data, headers=table_headers, tablefmt="pretty"))

output_proj2 = CRS.from_epsg(3035)
uklad_laea = []
for lon, lat in punkty:
    x, y = Transformer.from_proj(input_proj, output_proj2).transform(lon, lat)
    uklad_laea.append((x, y))
table_data = [(f"Punkt {i}", format(x, '.6f'), format(y, '.6f')) for i, (x, y) in enumerate(uklad_laea, start=1)]
table_headers = ["", "Układ LAEA", ""]
table_data.insert(0, ("Numer", "Współrzędna X", "Współrzędna Y"))
print(tabulate(table_data, headers=table_headers, tablefmt="pretty"))

output_proj3 = CRS.from_epsg(32633)
uklad_utm = []
for lon, lat in punkty:
    x, y = Transformer.from_proj(input_proj, output_proj3).transform(lon, lat)
    uklad_utm.append((x, y))
table_data = [(f"Punkt {i}", format(x, '.6f'), format(y, '.6f')) for i, (x, y) in enumerate(uklad_utm, start=1)]
table_headers = ["", "Układ UTM", ""]
table_data.insert(0, ("Numer", "Współrzędna X", "Współrzędna Y"))
print(tabulate(table_data, headers=table_headers, tablefmt="pretty"))

output_proj4 = CRS.from_epsg(3034)
uklad_lcc = []
for lon, lat in punkty:
    x, y = Transformer.from_proj(input_proj, output_proj4).transform(lon, lat)
    uklad_lcc.append((x, y))
table_data = [(f"Punkt {i}", format(x, '.6f'), format(y, '.6f')) for i, (x, y) in enumerate(uklad_lcc, start=1)]
table_headers = ["", "Układ LCC", ""]
table_data.insert(0, ("Numer", "Współrzędna X", "Współrzędna Y"))
print(tabulate(table_data, headers=table_headers, tablefmt="pretty"))

#readukcja długości odcinków
#układ 2000
d12 = ((punkty_pl2000[1][0] - punkty_pl2000[0][0])**2 + (punkty_pl2000[1][1] - punkty_pl2000[0][1])**2)**0.5
d23 = ((punkty_pl2000[2][0] - punkty_pl2000[1][0])**2 + (punkty_pl2000[2][1] - punkty_pl2000[1][1])**2)**0.5
d34 = ((punkty_pl2000[3][0] - punkty_pl2000[2][0])**2 + (punkty_pl2000[3][1] - punkty_pl2000[2][1])**2)**0.5
d41 = ((punkty_pl2000[0][0] - punkty_pl2000[3][0])**2 + (punkty_pl2000[0][1] - punkty_pl2000[3][1])**2)**0.5
```

```
phi12 = (punkty_pl2000[1][0] + punkty_pl2000[0][0]) / 2.0
phi23 = (punkty_pl2000[2][0] + punkty_pl2000[1][0]) / 2.0
phi34 = (punkty_pl2000[3][0] + punkty_pl2000[2][0]) / 2.0
phi41 = (punkty_pl2000[0][0] + punkty_pl2000[3][0]) / 2.0

posrednie = [phi12, phi23, phi34, phi41]

R=[]
for i in range(len(posrednie)):
    N_m = a / (1 - e2 * (np.sin(np.radians(posrednie[i]))**2))**0.5
    M_m = a * (1 - e2) / ((1 - e2 * np.sin(np.radians(posrednie[i]))**2)**1.5)
    R_m = (N_m * M_m)**0.5
    R.append(R_m)

sgk12 = d12 / 0.999923
sgk23 = d23 / 0.999923
sgk34 = d34 / 0.999923
sgk41 = d41 / 0.999923

r12 = sgk12 * (gk_2000[0][1]**2 + gk_2000[0][1] * gk_2000[1][1] + gk_2000[1][1]**2) / (6 * R[0]**2)
r23 = sgk23 * (gk_2000[1][1]**2 + gk_2000[1][1] * gk_2000[2][1] + gk_2000[2][1]**2) / (6 * R[1]**2)
r34 = sgk34 * (gk_2000[2][1]**2 + gk_2000[2][1] * gk_2000[3][1] + gk_2000[3][1]**2) / (6 * R[2]**2)
r41 = sgk41 * (gk_2000[3][1]**2 + gk_2000[3][1] * gk_2000[0][1] + gk_2000[0][1]**2) / (6 * R[3]**2)

selip12 = (sgk12 - r12) / 1000
selip23 = (sgk23 - r23) / 1000
selip34 = (sgk34 - r34) / 1000
selip41 = (sgk41 - r41) /1000

print('Długość odcinka 1-2 na elipsoidzie: {:.6f} km'.format(selip12))
print('Długość odcinka 2-3 na elipsoidzie: {:.6f} km'.format(selip23))
print('Długość odcinka 3-4 na elipsoidzie: {:.6f} km'.format(selip34))
print('Długość odcinka 4-1 na elipsoidzie: {:.6f} km'.format(selip41))
```

```python
#Uklad 1992
d12_1992 = ((uklad_1992[1][0] - uklad_1992[0][0])**2 + (uklad_1992[1][1] - uklad_1992[0][1])**2)**0.5
d23_1992 = ((uklad_1992[2][0] - uklad_1992[1][0])**2 + (uklad_1992[2][1] - uklad_1992[1][1])**2)**0.5
d34_1992 = ((uklad_1992[3][0] - uklad_1992[2][0])**2 + (uklad_1992[3][1] - uklad_1992[2][1])**2)**0.5
d41_1992 = ((uklad_1992[0][0] - uklad_1992[3][0])**2 + (uklad_1992[0][1] - uklad_1992[3][1])**2)**0.5

phi12_1992 = (uklad_1992[1][0] + uklad_1992[0][0]) / 2.0
phi23_1992 = (uklad_1992[2][0] + uklad_1992[1][0]) / 2.0
phi34_1992 = (uklad_1992[3][0] + uklad_1992[2][0]) / 2.0
phi41_1992 = (uklad_1992[0][0] + uklad_1992[3][0]) / 2.0

posrednie_1992 = [phi12_1992, phi23_1992, phi34_1992, phi41_1992]
```

```python
R_1992=[]
for i in range(len(posrednie_1992)):
    N_m_1992 = a / (1 - e2 * (np.sin(np.radians(posrednie_1992[i]))**2))**0.5
    M_m_1992 = a * (1 - e2) / ((1 - e2 * np.sin(np.radians(posrednie_1992[i]))**2)**1.5)
    R_m_1992 = (N_m_1992 * M_m_1992)**0.5
    R_1992.append(R_m_1992)

sgk12_1992 = d12_1992 / 0.999923
sgk23_1992 = d23_1992 / 0.999923
sgk34_1992 = d34_1992 / 0.999923
sgk41_1992 = d41_1992 / 0.999923

r12_1992 = sgk12_1992 * (gk_1992[0][1]**2 + gk_1992[0][1] * gk_1992[1][1] + gk_1992[1][1]**2) / (6 * R_1992[0]**2)
r23_1992 = sgk23_1992 * (gk_1992[1][1]**2 + gk_1992[1][1] * gk_1992[2][1] + gk_1992[2][1]**2) / (6 * R_1992[1]**2)
r34_1992 = sgk34_1992 * (gk_1992[2][1]**2 + gk_1992[2][1] * gk_1992[3][1] + gk_1992[3][1]**2) / (6 * R_1992[2]**2)
r41_1992 = sgk41_1992 * (gk_1992[3][1]**2 + gk_1992[3][1] * gk_1992[0][1] + gk_1992[0][1]**2) / (6 * R_1992[3]**2)

selip12_1992 = (sgk12_1992 - r12_1992) / 1000
selip23_1992 = (sgk23_1992 - r23_1992) / 1000
selip34_1992 = (sgk34_1992 - r34_1992) / 1000
selip41_1992 = (sgk41_1992 - r41_1992) / 1000

print(end='\n')
print('Długość odcinka 1-2 na elipsoidzie: {:.6f} km 1992'.format(selip12_1992))
print('Długość odcinka 2-3 na elipsoidzie: {:.6f} km 1992'.format(selip23_1992))
print('Długość odcinka 3-4 na elipsoidzie: {:.6f} km 1992'.format(selip34_1992))
print('Długość odcinka 4-1 na elipsoidzie: {:.6f} km 1992'.format(selip41_1992))
print(end='\n')

g = Geod(ellps='GRS80')
s12_inv = g.inv(punkty[0][1], punkty[0][0], punkty[1][1], punkty[1][0])[2]
s23_inv = g.inv(punkty[1][1], punkty[1][0], punkty[2][1], punkty[2][0])[2]
s34_inv = g.inv(punkty[2][1], punkty[2][0], punkty[3][1], punkty[3][0])[2]
s41_inv = g.inv(punkty[3][1], punkty[3][0], punkty[0][1], punkty[0][0])[2]
print('Długość odcinka 1-2 z zadania 3: {:.6f} km'.format(s12_inv /1000))
print('Długość odcinka 2-3 z zadania 3: {:.6f} km'.format(s23_inv / 1000))
print('Długość odcinka 3-4 z zadania 3: {:.6f} km'.format(s34_inv / 1000))
print('Długość odcinka 4-1 z zadania 3: {:.6f} km'.format(s41_inv / 1000))
print(end='\n')

def deg2dms(dd):
    deg = int(np.trunc(dd))
    minutes = int(np.trunc((dd - deg) * 60))
    seconds = round(((dd - deg) * 60 - minutes) * 60, 5)
    dms = [deg, abs(minutes), abs(seconds)]
    return dms
```

```python
def vincenty(BA,LA,BB,LB):
    '''
    Parameters
    ----------
    BA : szerokosc geodezyjna punktu A [RADIAN]
    LA : dlugosc geodezyjna punktu A [RADIAN]
    BB : szerokosc geodezyjna punktu B [RADIAN]
    LB : dlugosc geodezyjna punktu B [RADIAN]

    Returns
    -------
    sAB : dlugosc linii geodezyjnej AB [METR]
    A_AB : azymut linii geodezyjnej AB [RADIAN]
    A_BA : azymut odwrotny linii geodezyjne [RADIAN]
    '''
    b = a * np.sqrt(1-e2)
    f = 1-b/a
    dL = LB - LA
    UA = np.arctan((1-f)*np.tan(BA))
    UB = np.arctan((1-f)*np.tan(BB))
    L = dL
    while True:
        sin_sig = np.sqrt((np.cos(UB)*np.sin(L))**2 +\
                          (np.cos(UA)*np.sin(UB) - np.sin(UA)*np.cos(UB)*np.cos(L))**2)
        cos_sig = np.sin(UA)*np.sin(UB) + np.cos(UA) * np.cos(UB) * np.cos(L)
        sig = np.arctan2(sin_sig,cos_sig)
        sin_al = (np.cos(UA)*np.cos(UB)*np.sin(L))/sin_sig
        cos2_al = 1 - sin_al**2
        cos2_sigm = cos_sig - (2 * np.sin(UA) * np.sin(UB))/cos2_al
        C = (f/16) * cos2_al * (4 + f*(4 - 3 * cos2_al))
        Lst = L
        L = dL + (1-C)*f*sin_al*(sig+C*sin_sig*(cos2_sigm+C*cos_sig*(-1 + 2*cos2_sigm**2)))
        if abs(L-Lst)<(0.000001/206265):
            break
```

```
    u2 = (a**2 - b**2)/(b**2) * cos2_al
    A = 1 + (u2/16384) * (4096 + u2*(-768 + u2 * (320 - 175 * u2)))
    B = u2/1024 * (256 + u2 * (-128 + u2 * (74 - 47 * u2)))
    d_sig = B*sin_sig * (cos2_sigm + 1/4*B*(cos_sig*(-1+2*cos2_sigm**2)\
            - 1/6 *B*cos2_sigm * (-3 + 4*sin_sig**2)*(-3+4*cos2_sigm**2)))
    sAB = b*A*(sig-d_sig)
    A_AB = np.arctan2((np.cos(UB) * np.sin(L)),(np.cos(UA)*np.sin(UB) - np.sin(UA)*np.cos(UB)*np.cos(L)))
    A_BA = np.arctan2((np.cos(UA) * np.sin(L)),(-np.sin(UA)*np.cos(UB) + np.cos(UA)*np.sin(UB)*np.cos(L))) + np.pi
    return sAB, A_AB, A_BA

punkty2 = [(51.25, 17.75), (51.60952957952263, 17.75), (51.600656153534366, 19.193356733106977),
           (51.2411260266428, 19.193356733106977)]
punkty2 = np.deg2rad(punkty2)
az12 = np.rad2deg(vincenty(punkty2[0][0], punkty2[0][1], punkty2[1][0], punkty2[1][1]))[1]
az23 = np.rad2deg(vincenty(punkty2[1][0], punkty2[1][1], punkty2[2][0], punkty2[2][1]))[1]
az34 = np.rad2deg(vincenty(punkty2[2][0], punkty2[2][1], punkty2[3][0], punkty2[3][1]))[1]
az41 = np.rad2deg(vincenty(punkty2[3][0], punkty2[3][1], punkty2[0][0], punkty2[0][1]))[1]
az41 += 360
print('Azymut odcinka 1-2 z zadania 3:', deg2dms(az12))
print('Azymut odcinka 2-3 z zadania 3:', deg2dms(az23))
print('Azymut odcinka 3-4 z zadania 3:', deg2dms(az34))
print('Azymut odcinka 4-1 z zadania 3:', deg2dms(az41))

#redukcja azymutów
#Układ 2000
az12_2000 = math.atan2(gk_2000[1][1] - gk_2000[0][1], gk_2000[1][0] - gk_2000[0][0])
az23_2000 = math.atan2(gk_2000[2][1] - gk_2000[1][1], gk_2000[2][0] - gk_2000[1][0])
az34_2000 = math.atan2(gk_2000[3][1] - gk_2000[2][1], gk_2000[3][0] - gk_2000[2][0])
az41_2000 = math.atan2(gk_2000[0][1] - gk_2000[3][1], gk_2000[0][0] - gk_2000[3][0])

gamma = []
for i in range(len(punkty)):
    punkty = np.deg2rad(punkty)
    lambda_0 = np.deg2rad(18)
    b2 = a**2 * (1 - e2)
    e2_prime = (a**2 - b2) / b2
    delta_lambda = punkty[i][1] - lambda_0
    t = math.tan(punkty[i][0])
    eta2 = e2_prime * math.cos(punkty[i][0])**2
    eta = math.sqrt(eta2)
    N = a / math.sqrt(1 - e2 * math.sin(punkty[i][0])**2)
    gamma_i = delta_lambda * math.sin(punkty[i][0]) + (delta_lambda**3 / 3) * math.sin(punkty[i][0])
    * math.cos(punkty[i][0])**2 * (1 + 3 * eta**2 + 2 * eta**4)
    + (delta_lambda**5 / 15) * math.sin(punkty[i][0]) * math.cos(punkty[i][0])**4 * (2 - t**2)
    gamma.append(gamma_i)

theta12_2000 = (gk_2000[1][0]-gk_2000[0][0])*(2*gk_2000[0][1] + gk_2000[1][1]) / (6 * R[0]**2)
theta23_2000 = (gk_2000[2][0]-gk_2000[1][0])*(2*gk_2000[1][1] + gk_2000[2][1]) / (6 * R[1]**2)
theta34_2000 = (gk_2000[3][0]-gk_2000[2][0])*(2*gk_2000[2][1] + gk_2000[3][1]) / (6 * R[2]**2)
theta41_2000 = (gk_2000[0][0]-gk_2000[3][0])*(2*gk_2000[3][1] + gk_2000[0][1]) / (6 * R[3]**2)

A12_2000 = np.rad2deg(az12_2000) + np.rad2deg(gamma[0]) + np.rad2deg(theta12_2000)
A23_2000 = np.rad2deg(az23_2000) + np.rad2deg(gamma[1]) + np.rad2deg(theta23_2000)
A34_2000 = np.rad2deg(az34_2000) + np.rad2deg(gamma[2]) + np.rad2deg(theta34_2000)
A41_2000 = np.rad2deg(az41_2000) + np.rad2deg(gamma[3]) + np.rad2deg(theta41_2000)
A41_2000 += 360
print('Azymut odcinka 1-2 z układu 2000:', deg2dms(A12_2000))
print('Azymut odcinka 2-3 z układu 2000:', deg2dms(A23_2000))
print('Azymut odcinka 3-4 z układu 2000:', deg2dms(A34_2000))
print('Azymut odcinka 4-1 z układu 2000:', deg2dms(A41_2000))
```

```
#Układ 1992
az12_1992 = math.atan2(gk_1992[1][1] - gk_1992[0][1], gk_1992[1][0] - gk_1992[0][0])
az23_1992 = math.atan2(gk_1992[2][1] - gk_1992[1][1], gk_1992[2][0] - gk_1992[1][0])
az34_1992 = math.atan2(gk_1992[3][1] - gk_1992[2][1], gk_1992[3][0] - gk_1992[2][0])
az41_1992 = math.atan2(gk_1992[0][1] - gk_1992[3][1], gk_1992[0][0] - gk_1992[3][0])

theta12_1992 = (gk_1992[1][0]-gk_1992[0][0])*(2*gk_1992[0][1] + gk_1992[1][1]) / (6 * R_1992[0]**2)
theta23_1992 = (gk_1992[2][0]-gk_1992[1][0])*(2*gk_1992[1][1] + gk_1992[2][1]) / (6 * R_1992[1]**2)
theta34_1992 = (gk_1992[3][0]-gk_1992[2][0])*(2*gk_1992[2][1] + gk_1992[3][1]) / (6 * R_1992[2]**2)
theta41_1992 = (gk_1992[0][0]-gk_1992[3][0])*(2*gk_1992[3][1] + gk_1992[0][1]) / (6 * R_1992[3]**2)

A12_1992 = np.rad2deg(az12_1992) + np.rad2deg(gamma[0]) + np.rad2deg(theta12_1992)
A23_1992 = np.rad2deg(az23_1992) + np.rad2deg(gamma[1]) + np.rad2deg(theta23_1992)
A34_1992 = np.rad2deg(az34_1992) + np.rad2deg(gamma[2]) + np.rad2deg(theta34_1992)
A41_1992 = np.rad2deg(az41_1992) + np.rad2deg(gamma[3]) + np.rad2deg(theta41_1992)
A41_1992 += 360
print('Azymut odcinka 1-2 z układu 1992:', deg2dms(A12_1992))
print('Azymut odcinka 2-3 z układu 1992:', deg2dms(A23_1992))
print('Azymut odcinka 3-4 z układu 1992:', deg2dms(A34_1992))
print('Azymut odcinka 4-1 z układu 1992:', deg2dms(A41_1992))
```

```python
#Pole
punkty1 = [(51.25, 17.75), (51.60952957952263, 17.75), (51.600656153534366, 19.193356733106977),
           (51.2411260266428, 19.193356733106977)]
punkty_odwrotne = punkty1[::-1]
point_objects = [Point(lon, lat) for lat, lon in punkty_odwrotne]
polygon = Polygon(point_objects)
area, perimeter = g.geometry_area_perimeter(polygon)
area = area / 1000000
print('Pole z zadania 3: {:.6f} km²'.format(area))

def calculate_surface_area(points):
    n = len(points)
    area = 0
    for i in range(n):
        xi, yi = points[i]
        xi_minus_1, yi_minus_1 = points[i-1] if i > 0 else points[n-1]
        xi_plus_1, yi_plus_1 = points[(i+1) % n]
        area += xi * (yi_plus_1 - yi_minus_1)
    wynik = abs(area) / 2
    return wynik / 1000000

print(end='\n')
area_pl2000 = calculate_surface_area(punkty_pl2000)
print('Pole w układzie 2000: {:.6f} km²'.format(area_pl2000))
area_1992 = calculate_surface_area(uklad_1992)
print('Pole w układzie 1992: {:.6f} km²'.format(area_1992))
area_gk_2000 = calculate_surface_area(gk_2000)
print('Pole na płaszczyźnie odwzorowania gk dla układu 2000: {:.6f} km²'.format(area_gk_2000))
area_gk_1992 = calculate_surface_area(gk_1992)
print('Pole na płaszczyźnie odwzorowania gk dla układu 1992: {:.6f} km²'.format(area_gk_1992))
area_laea = calculate_surface_area(uklad_laea)
print('Pole w układzie PL-LAEA: {:.6f} km²'.format(area_laea))
area_utm = calculate_surface_area(uklad_utm)
print('Pole w układzie PL-UTM: {:.6f} km²'.format(area_utm))
area_lcc = calculate_surface_area(uklad_lcc)
print('Pole w układzie PL-LCC: {:.6f} km²'.format(area_lcc))
```