SPRAWOZDANIE

Odwzorowanie Gaussa-Krügera: układy współrzędnych płaskich stosowanych w Polsce
ćwiczenie 4

Izabella Kaim 319193

1. Cel ćwiczenia

Celem ćwiczenia było przeliczenie współrzędnych geodezyjnych czterech punktów z ćwiczenia nr 3 na współrzędne płaskie w układach PL-1992 oraz PL-2000.
Następnie wykonanie redukcji długości i azymutów obliczonych na powierzchni układu PL-2000/PL-1992 na powierzchnię elipsoidy i obliczenie pola powierzchni figury. Należało także przeliczyć współrzędne do układu PL-LAEA i obliczyć pole powierzchni.

2. Dane

φ_1 = 57° 00' 00"
λ_1 = 22° 00' 00"
φ_2 = 57° 10' 46.54"
λ_2 = 22° 00' 00"
φ_3 = 57° 10' 41.74"
λ_3 = 22° 34' 43.31"
φ_4 = 56° 59' 55.2"
λ_4 = 22° 34' 43.31"

```
Wsp punktu 1:  57.00000000000001 22.0
Wsp punktu 2:  57.179595310234625 22.0
Wsp punktu 3:  57.17826148114223 22.578698075251012
Wsp punktu 4:  56.998666132425434 22.578698075251012
```

pole figury do porównania: 701.6855901090088 $km^2 \approx 701.69\ km^2$

długości odcinków do porównania:
$d_{1\text{-}2}$: 20000 m
$d_{2\text{-}3}$: 35000 m
$d_{3\text{-}4}$: 20000 m
$d_{4\text{-}1}$: 35169.58 m

3. Wykonanie

Funkcja przeliczająca współrzędne punktów na współrzędne lokalne na płaszczyźnie Gaussa-Krügera

```
def geo2gk(f, l, lon_0):
    f = np.deg2rad(f)
    l = np.deg2rad(l)
    lon_0 = np.deg2rad(lon_0)

    a = 6378137
    e2 = 0.006694380022900
    b2 = a**2 * (1 - e2)
    ep2 = (a**2 - b2) / b2
    d_lon = l - lon_0
    A0 = 1 - e2/4 - 3*e2**2/64 - 5*e2**3/256
    A2 = (3/8)*(e2 + e2**2/4 + 15*e2**3/128)
    A4 = (15/256)*(e2**2 + 3*e2**3/4)
    A6 = 35*e2**3/3072

    N = fN(f, a, e2)
    t = np.tan(f)
    n2 = ep2 * m.cos(f)*m.cos(f)

    sigma = a*(A0*f - A2*m.sin(2*f) + A4*m.sin(4*f)-A6*m.sin(6*f))

    x = sigma + ((d_lon**2)/2) * N * m.sin(f) * m.cos(f) * ( 1 + ((d_lon**2)/12)
    y = d_lon * N * m.cos(f)
    y = y * (1 + ((d_lon**2)/6)*(m.cos(f)**2)*(1 - (t**2) + n2)+((d_lon**4)/120)

    return x, y
```

Funkcja gk2u92 przeliczająca współrzędne G-K na współrzędne układu PL-1992:

```
def gk2u92(xgk, ygk):
    x = xgk*0.9993 - 5300000
    y = ygk*0.9993 + 500000
    return x, y
```

Funkcja gk2u2000 przeliczająca współrzędne G-K na współrzędne układu PL-2000:

```
def gk2u2000(xgk, ygk, nr):
    x = 0.999923 * xgk
    y = 0.999923 * ygk + 500000 + nr*1000000
    return x, y
```

Funkcje do obliczenia R średniego promienia krzywizny

```
def R(f):
    a = 6378137
    e2 = 0.006694380022900
    M = a * (1 - e2) / np.sqrt((1 - e2 * (np.sin(f) ** 2)) ** 3)
    N =  a / np.sqrt(1 - e2 * (np.sin(f) ** 2))
    return m.sqrt(M*N)

def R2(lat1, lat2):
    a = 6378137
    e2 = 0.006694380022900
    f = lat1 + (lat2 - lat1)/2
    M = a * (1 - e2) / np.sqrt((1 - e2 * (np.sin(f) ** 2)) ** 3)
    N =  a / np.sqrt(1 - e2 * (np.sin(f) ** 2))
    return m.sqrt(M*N)
```

Funkcja obliczająca szerokość geodezyjną φ z współrzędnych G-K

```
def gk2geo(xk, yk, lon_0):
    a = 6378137
    e2 = 0.006694380022900
    b2 = a**2 * (1 - e2)
    ep2 = (a**2 - b2) / b2

    A0 = 1 - e2/4 - 3*e2**2/64 - 5*e2**3/256
    A2 = (3/8)*(e2 + e2**2/4 + 15*e2**3/128)
    A4 = (15/256)*(e2**2 + 3*e2**3/4)
    A6 = 35*e2**3/3072

    f_prev = xk/(a * A0)

    sigma = a*(A0*f_prev - A2*m.sin(2*f_prev) + A4*m.sin(4*f_prev)-A6*m.sin(6*f_prev)
    fi = f_prev + (xk - sigma)/(a * A0)

    min_dif = np.deg2rad(0.000001/3600)

    while True:
        if abs(fi - f_prev) < min_dif:
            break
        else:
            f_prev = fi
            sigma = a*(A0*f_prev - A2*m.sin(2*f_prev) + A4*m.sin(4*f_prev)-A6*m.sin(6
            fi = f_prev + (xk - sigma)/(a * A0)

    return fi
```

Funkcja fil2gamma oblicza wartość γ z współrzędnych geodezyjnych.

```
def fil2gamma(f, l, lon_0):
    f = np.deg2rad(f)
    l = np.deg2rad(l)
    lon_0 = np.deg2rad(lon_0)
    d_lon = l - lon_0
    a = 6378137
    e2 = 0.006694380022900
    b2 = a**2 * (1 - e2)
    ep2 = (a**2 - b2)/ b2
    n2 = ep2 * m.cos(f)*m.cos(f)
    t = np.tan(f)

    g = d_lon * np.sin(f) + ((d_lon**3)/3 * np.sin(f) * (np.cos(f)**2) * (1 + 3 * r

    return g
```

Funkcja obliczająca redukcję δ_{AB}

```
def redukcjaAB(xA, yA, xB, yB, f1, f2):
    a = 6378137
    e2 = 0.006694380022900

    fm = (f1 + f2) / 2
    Rm = R(fm)

    red = (xB - xA) * (2 * yA + yB) / (6 * Rm**2)

    return red
```

Przeliczenie współrzędnych na układy 2000, 1992:

```
pkt_2000 = []
for i in range(len(lon)):
    lon_0 = 21
    nr = 7
    xgk,ygk = geo2gk(lat[i], lon[i], lon_0)
    x,y = gk2u2000(xgk, ygk, nr)
    pkt_2000.append([x,y])


pkt_1992 = []
for i in range(len(lon)):
    lon_0 = 19
    xgk,ygk = geo2gk(lat[i], lon[i], lon_0)
    x,y = gk2u1992(xgk, ygk)
    pkt_1992.append([x,y])
```

Obliczenie długości odcinków w odpowiednich układach:

```
ss2000 = np.array([m.dist(pkt_2000[i], pkt_2000[i-1]) for i in range(1,len(pkt_2000))])

ss1992 = np.array([m.dist(pkt_1992[i], pkt_1992[i-1]) for i in range(1,len(pkt_1992))])

m0_2000 = 0.999923
m0_1992 = 0.9993

gk_2000 = []
for i in range(len(lon)):
    lon_0 = 21
    nr = 7
    xgk,ygk = geo2gk(lat[i], lon[i], lon_0)
    gk_2000.append([xgk,ygk])

gk_1992 = []
for i in range(len(lon)):
    lon_0 = 19
    xgk,ygk = geo2gk(lat[i], lon[i], lon_0)
    gk_1992.append([xgk,ygk])

sgk2000 = ss2000/m0_2000
Rs= []
for i in range(len(lat) - 1):
    Rs.append(R2(lat[i], lat[i+1]))

rs2000 = []
for i in range(len(ss2000)):
    r = ss2000[i] * (gk_2000[i][1]**2 + gk_2000[i][1]*gk_2000[i + 1][1] + gk_2000[i + 1][1]**2) / (6 * Rs[i]**2)
    rs2000.append(r)

ss_elip2000 = []
for i in range(len(rs2000)):
          ss_elip2000.append(sgk2000[i] - rs2000[i])


sgk1992 = ss1992/m0_1992
Rs= []
for i in range(len(lat) - 1):
    Rs.append(R2(lat[i], lat[i+1]))

rs1992 = []
for i in range(len(ss1992)):
    r = ss1992[i] * (gk_1992[i][1]**2 + gk_1992[i][1]*gk_1992[i + 1][1] + gk_1992[i + 1][1]**2) / (6 * Rs[i]**2)
    rs1992.append(r)

ss_elip1992 = []
for i in range(len(rs1992)):
          ss_elip1992.append(sgk1992[i] - rs1992[i])
```

Obliczenie kątów kierunkowych odcinków u układach:

```
katy_2000 = []
for i in range(len(gk_2000)-1):
    xA, yA = gk_2000[i]
    xB, yB = gk_2000[i+1]
    alfa = np.arctan2(yB-yA, xB-xA)
    #alfa = deg2dms(np.rad2deg(alfa))
    katy_2000.append(alfa)

print("Kąty 2000: \n", katy_2000)

katy_1992 = []
for i in range(len(gk_1992)-1):
    xA, yA = gk_1992[i]
    xB, yB = gk_1992[i+1]
    alfa = np.arctan2(yB-yA, xB-xA)
    #alfa = deg2dms(np.rad2deg(alfa))
    katy_1992.append(alfa)

print("Kąty 1992: \n", katy_1992)
```

Obliczenie zbieżności południków i redukcji kierunków δ_{AB} w odpowiednich układach:

```
gamma_2000 = []
gamma_1992 = []
for i in range(len(lat) - 1):
    gamma_2000.append(fil2gamma(lat[i], lon[i], 21))
    gamma_1992.append(fil2gamma(lat[i], lon[i], 19))

print("Zbieżności południków 1992: \n", gamma_1992)
print()
print("Zbieżności południków 2000: \n", gamma_2000)
print()

red_1992 = []
red_2000 = []
for i in range(len(gk_1992) - 1):
    xA, yA = gk_1992[i]
    xB, yB = gk_1992[i+1]
    redAB = redukcjaAB(xA, yA, xB, yB, lat[i], lat[i+1])
    red_1992.append(redAB)

print("Reduckje kierunków 1992: \n", red_1992)
print()


for i in range(len(gk_2000) - 1):
    xA, yA = gk_2000[i]
    xB, yB = gk_2000[i+1]
    redAB = redukcjaAB(xA, yA, xB, yB, lat[i], lat[i+1])
    red_2000.append(redAB)

print("Reduckje kierunków 2000: \n", red_2000)
print()
```

Obliczenie azymutów odcinków na powierzchni elipsoidy:

```
az_elip_2000 = []
az_elip_1992 = []

for i in range(len(red_1992)):
    az = katy_1992[i] + gamma_1992[i] + red_1992[i]
    az = np.rad2deg(az)
    az = deg2dms(az)
    az_elip_1992.append(az)

print("Azymuty odcinków na elipsoidzie - 1992: \n", az_elip_1992)
print()

for i in range(len(red_2000)):
    az = katy_2000[i] + gamma_2000[i] + red_2000[i]
    az = np.rad2deg(az)
    az = deg2dms(az)
    az_elip_2000.append(az)

print("Azymuty odcinków na elipsoidzie - 2000: \n", az_elip_2000)
print()
```

Obliczenie pola figury stosując tzw. wzory Gaussa dla PL-1992 i dla PL-2000:

```
punkty_92 = pkt_1992[0:4]
#pole PL-1992
PP = 0
for i in range(0, 4):
    if i == 0:
        PP += punkty_92[i][0] * (punkty_92[1][1] - punkty_92[3][1])
    elif i == 3:
        PP += punkty_92[i][0] * (punkty_92[0][1] - punkty_92[i-1][1])
    else:
        PP += punkty_92[i][0] * (punkty_92[i+1][1] - punkty_92[i-1][1])
Pole_1992 = PP/2

#kontrola
mPP = 0
for i in range(0, 4):
    if i == 0:
        mPP += punkty_92[i][1] * (punkty_92[1][0] - punkty_92[3][0])
    elif i == 3:
        mPP += punkty_92[i][1] * (punkty_92[0][0] - punkty_92[i-1][0])
    else:
        mPP += punkty_92[i][1] * (punkty_92[i+1][0] - punkty_92[i-1][0])
mPole_1992 = np.abs(mPP/2)

punkty_2000 = pkt_2000[0:4]
#pole PL-2000
PP = 0
for i in range(0, 4):
    if i == 0:
        PP += punkty_2000[i][0] * (punkty_2000[1][1] - punkty_2000[3][1])
    elif i == 3:
        PP += punkty_2000[i][0] * (punkty_2000[0][1] - punkty_2000[i-1][1])
    else:
        PP += punkty_2000[i][0] * (punkty_2000[i+1][1] - punkty_2000[i-1][1])
Pole_2000 = PP/2

#kontrola
mPP = 0
for i in range(0, 4):
    if i == 0:
        mPP += punkty_2000[i][1] * (punkty_2000[1][0] - punkty_2000[3][0])
    elif i == 3:
        mPP += punkty_2000[i][1] * (punkty_2000[0][0] - punkty_2000[i-1][0])
    else:
        mPP += punkty_2000[i][1] * (punkty_2000[i+1][0] - punkty_2000[i-1][0])
mPole_2000 = np.abs(mPP/2)
```

Obliczenie współrzędnych w układzie PL- LAEA i pole powierzchni

```
transformer = Transformer.from_crs(4326, 3035)

pkt_LAEA = []
for i in range(len(lon)-1):
    x, y = transformer.transform(lat[i], lon[i])
    pkt_LAEA.append([x,y])
    print(x,y)

PP = 0
for i in range(0, 4):
    if i == 0:
        PP += pkt_LAEA[i][0] * (pkt_LAEA[1][1] - pkt_LAEA[3][1])
    elif i == 3:
        PP += pkt_LAEA[i][0] * (pkt_LAEA[0][1] - pkt_LAEA[i-1][1])
    else:
        PP += pkt_LAEA[i][0] * (pkt_LAEA[i+1][1] - pkt_LAEA[i-1][1])

Pole_LAEA = PP/2

print("Pl-LAEA:", Pole_LAEA)
```

4. Wyniki

Współrzędne w układzie współrzędnych płaskich PL-1992

x_1 = 1019490.867000 y_1 = 682155.091595
x_2 = 1039465.643868 y_2 = 681276.249432
x_3 = 1041005.425686 y_3 = 716234.859427
x_4 = 1021035.405327 y_4 = 717283.370830

Współrzędne w układzie współrzędnych płaskich PL-2000

x_1 = 6319872.006366 y_1 = 7560766.252669
x_2 = 6339869.220333 y_2 = 7560473.214947
x_3 = 6340382.587706 y_3 = 7595469.407312
x_4 = 6320535.738646 y_4 = 7595928.613100

Dla układu PL-1992:

obliczone długości między punktami:
$d_{1\text{-}2}$: 19999.954914826347 ≈ 19999.95 m
$d_{2\text{-}3}$: 34999.92301310172 ≈ 34999.92 m
$d_{3\text{-}4}$: 19851.39524452763 ≈ 19851.40 m
$d_{4\text{-}1}$: 35168.526970829065 ≈ 35168.52 m

Długości odcinków są zbliżone do tych z poprzedniego zadania.

azymuty:
A_{1-2} = -02° 31' 09.3"
A_{2-3} = 87° 28' 40.8"
A_{3-4} = 176° 59' 40.1"
A_{1-4} = -92° 45' 34.6" = 267° 14' 25.4"

zbieżności południków:
γ_1 = 0.04392466276224204 ≈ 0.043
γ_2 = 0.044013742904797495 ≈ 0.044
γ_3 = 0.05250917411422153 ≈ 0.053
γ_4 = 0.05240374393434402 ≈ 0.052

redukcje kierunków:
δ_{1-2} = 1.4960819775185498e-05 ≈ 1.496 * 10^{-5}
δ_{2-3} = 4.561843064157874e-07 ≈ 4.562 * 10^{-7}
δ_{3-4} = -2.3403299752629727e-05 ≈ -2.340 * 10^{-5}
δ_{4-1} = -6.899296669249323e-07 ≈ -6.899 * 10^{-7}

azymuty na elipsoidzie:
A_{1-2} = 00° 00' 0.06"
A_{2-3} = 90° 00' 0.00"
A_{3-4} = 179° 59' 59.93"
A_{1-4} = -89° 45' 26.39" = 270° 14' 33.61"

pole:
698776790.1524086 m^2 ≈ 698776790.15 m^2 ≈ 698.78 km^2
kontrola: 698776790.1524057 m^2 ≈ 698776790.15 m^2 ≈ 698.78 km^2

Dla układu PL-2000:

obliczone długości między punktami:
d_{1-2} : 19999.994707690636 ≈ 19999.99 m
d_{2-3} : 34999.98763087789 ≈ 34999.99 m
d_{3-4} : 19851.447442394736 ≈ 19851.45 m
d_{4-1} : 35168.622914376276 ≈ 35168.62 m

Długości odcinków są zbliżone do tych z poprzedniego zadania.

azymuty:
A_{1-2} = 00° 50' 22.4"
A_{2-3} = 89° 09' 34.5"
A_{3-4} = 178° 40' 28.4"
A_{1-4} = -91° 04' 53.0" = 268° 55' 07.0"

zbieżności południków:
γ_1 = 0.014638006274306848 ≈ 0.015
γ_2 = 0.014667726943436495 ≈ 0.014
γ_3 = 0.023156601559354728 ≈ 0.023

γ_4 = 0.02311003719919189 ≈ 0.023

redukcje kierunków:
$\delta_{1\text{-}2}$ = 4.485260792794096e-05 ≈ 4.485 * 10^{-5}
$\delta_{2\text{-}3}$ = 3.6638914911561123e-06 ≈ 3.664 * 10^{-6}
$\delta_{3\text{-}4}$ = -5.300657531337398e-05 ≈ -5.301 * 10^{-5}
$\delta_{4\text{-}1}$ = -4.301083200210802e-06 ≈ -4.301 * 10^{-6}

azymuty na elipsoidzie:
$A_{1\text{-}2}$ = 00° 00' 0.02"
$A_{2\text{-}3}$ = 90° 00' 2.50"
$A_{3\text{-}4}$ = 179° 59' 59.98"
$A_{1\text{-}4}$ = -89° 45' 26.39" = 270° 14' 33.61"

Azymutu w poprzednim zadaniu wynosiły odpowiednio 0°, 90°, 180°, 270° 29' 04.76". Wartości obliczone tutaj są zbliżone do nich.

pole:
699071816.2450104 m^2 ≈ 699071816.24 m^2 ≈ 699.07 km^2
kontrola: 699071816.2449951 m^2 ≈ 699071816.24 m^2 ≈ 699.07 km^2

Dla układu PL-LAEA:

x_1 = 3827377.593915 y_1 = 5047046.676870
x_2 = 3847102.528161 y_2 = 5043592.165560
x_3 = 3852945.607562 y_3 = 5078054.881474
x_4 = 3833392.009730 y_4 = 5081647.766001

pole:
699072336.1029968 m^2 ≈ 699072336.10 m^2 ≈ 699.07 km^2

Pole w zadaniu 3: 701.69 km^2
Pole w układzie PL-1992: 698.78 km^2
Pole w układzie PL-2000: 699.07 km^2
Pole w układzie PL-LAEA: 699.07 km^2

Wartości pól są zbliżone do siebie, więc została spełniona wiernopolowość odwzorowań.