Politechnika Warszawska
Wydział Geodezji i Kartografii
Informatyka Geodezyjna II

Sprawozdanie Projekt 1

TRANSFORMACJE WSPÓŁRZĘDNYCH

Stanisław Oliszewsk i Filip Grygorczuk

Numer indeksu: 325803 i 328942
Grupa: 3
Semestr: 4

Zajęcia: poniedziałek 16:15 - 17:45
Prowadzący: mgr inż. Andrzej Szeszko

Spis treści

1 Wstęp

1.1 Cel ćwiczenia

Utowrzenie skryptu jako klasy, implementującej następujące transformacje współrzędnych:

– XYZ-> BLH
Dana transformacja wykorzysuje w swoim działaniu algorytm Hirvonena, przekształca ona współrzędne ortokartezjańskie XYZ do współrzędnych geodezyjnych krzywoliniowych BLH.

– BLH -> XYZ
Dana tranformacja jest odwrotnościa do poprzedniej i zamienia współrzędne geodezyjne krzywoliniowe BLH do ortokartezjańskich XYZ.

– XYZ -> NEU
Algorytm transformuje współrzędnych ortokartezjańskich: X, Y, Z do współrzędnych topocentrycznych.

– BL -> PL2000
Algorytm transformuje współrzędne geodezyjne krzywoliniowe BL do współrzędnych w układzie PL2000 (x2000, y2000).

– BL -> PL1992
Algorytm transformuje współrzędne geodezyjne krzywoliniowe BL do współrzędnych w układzie PL1992 (x1992, y1992).

Celem napisanego skryptu jest transformacja współrzędnych pomiędzy kolejnymi dostępnymi układami. Algorytm wykorystuje bibliotekę sys w celu pobrania od użytkownika argumentów niezbędnych do transformacji. Program musi być stworzony tak aby wczytywać z folderu plik z danymi, wyknowyać transformacje i tworzyć plik wynikowy. Program powinien być przygotowany na obsługę błędów związananych z jego nieodpowiednim użyciem. W tym wypadku program powinien zwracać błąd sugerujący przyczynę błędu i/lub możliwości jego rozwiązania.

1.2 Wykorzystane narzędzia

Użyte zostały następujące narzędzia:

- ◇ Spyder 5.4.3
- ◇ Python 3.11.8
- ◇ System operacyjny: Microsoft Windows 11 oraz Windows 10

2 Etapy rozwiązywania

1. Zdefiniowanie klasy "Transformacje":
Pierwszym etapem wykonywania ćwiczenia było zadeklarowanie klasy "Transformacje". Przy użyciu funckji _init_ dokonaliśmy inicjalizacja instancji klasy podając jako argument listę parametór elipsoidy.

2. Utworzenie algorytmów dla kolejnych transformacji:
   W tym kroku zostały zdefiniowane metody klasy wymienionej w pierwszym punkcie. Odpowiadają one za transformacje współrzędnych. Do stworzenia tych algorytmów wykorzystaliśmy kod utworzony podczas zajęć z Geodezji Wyższej realizowanych w semestrze nr.3 oraz wiadomości z wykładów z tego samego przedmiotu. Do realizacji tych algorytów została użyta biblioteka python o nazwie *numpy*, która odpowiada za obliczenia numeryczne oraz macierzone podczas transformacji.

3. Warunek $ifname == main$:
   Kolejnym krokiem było zaimplementowanie warunku $ifname == main$, oraz wykorzystanie przy pomocy bibioteki sys.argv zbioru argumentów podanych przez użytkownika podczas wywołania skryptu w postaci flag. W celu interpretacji argumentów flag podanych przez użytkownika utworzone zostały 2 słowniki zawierające:

   - nazwy transformacji ubsługiwanych przez skrypt,
   - modele elipsoidy dostępne dla użytkownika

   Następnie za pomocą metody warunkowej if oraz metody *tryexcept* stworzyliśmy fragment kodu, który sprawdza czy użytkownik korzysta z programu w sposób prawidłowy tzn:

   - sprawdzenie czy użytkownik podczas wywoływania programu podał wszytkie flagi oraz każda z nich ma przypisana wartość,
   - sprawdzenie czy podane przez użytkownika nazwy elipsoidy oraz transformacji są dostępne w naszym programie

   Jeżli użytkownik popełni błąd program zwróci odpowiedni komunikat i wskaże który element wymaga poprawy.

4. Wczytywanie plików: W ostatnim kroku stworzyliśmy fragment odpowiedzlany za wczytywanie plików wejsciowych. Program zakłada strukturę pliku w której posiada on czterowierszowy nagłówek tak jak zostało to podane w przykładowym pliku na platformie Teams o nazwie *wsp_inp.txt*. Program wykona transformacje a następnie automatycznie zwróci plik z danymi wyjściowymi o nazwie "wsp_trans"w miejscu wywołania skryptu. Ponadto, kod zwóci użytkownikowi błąd gdy ten poda plik który nie istanieje lub gdy format danych w pliku będzie zły tzn:

   - plik będzie zawierał tylko jeden wiersz z danymi,
   - plik nie bedzie zawierał czterowierszowego nagłówka,
   - dane w pliku nie bede oddzielone za pomocą $'$,$'$

3 Podsumowanie

3.1 Link do repozytorium GitHub

Link do repozytorium na GitHubie

3.2 Umiejętności nabyte w trakcie wykonywania projektu

- Pisani kodu obiektowego w programie Python oraz poznanie biblioteki sys,
- Tworzenie zdalnego repozytorium do pracy zespołowej przy użyciu platformy GitHub,
- Umiejętność obsługi Python przy użyciu Terminala a nie jak do tej pory wyłącznie programu Spyder,
- Poznanie nowej formy sprawozdawczej w formie dokumentacji README,
- Współpraca w wieloosobowym zespole z wykorzystaniem systemu kontroli wersji git

3.3 Spostrzeżenia i trudności napotkane w trakcie ćwiczenia

Najczęściej napotykaną w naszym programie trudnościa stała się obsługa zdalnego repozytorium git ,która wynikała z naszego słabego doświdczenia w obsłudze terminala git bash. Podczas dodawania kolejnych fragemntów kodu doprowadziliśmy do kilku konfliktów, wynikajacych z rozbieżności stanu repozytorium lokalnego oraz zdalnego oraz niedopowiedniego użycia komedny git pull (posiadając nie z comitt'owane zmiany). Problem rozwiązaliśmy abortując komednę merge zaproponowąna przez git, a nastepnie wyrównując głowną branch master do stanu repozytorium za pomoca komendy git reset.

4 Bibliografia

- Skrypt napisany w języku Python: $geo\_v1.py$
- Plik: $proj\_1\_transformations\_requirements.txt$
- Materiały relizowane podczas zajęć z przedmiotu $'GeodezjaWyzsza'$