**Cel ćwiczenia:**

Stworzenie skryptu implementującego następujące transformacje:

* (XYZ TO flh) Transformacja współrzędnych geocentrycznych X, Y, Z na współrzędne elipsoidalne (φ, λ, h) fi, lambda, h.
* (flh TO XYZ) Transformacja współrzędnych elipsoidalnych (φ, λ, h) fi, lambda, h na współrzędne geocentryczne X, Y, Z.
* (XYZ TO NEU) Transformacja współrzędnych X,Y,Z do układu topocentrycznego Northing, Easting, Up.
* (FL [GRS80, WGS84, ew. Krasowski] to 2000) Transformacje współrzędnych elipsoidalnych () fi, lambda z systemów odniesienia GRS80, WGS84, ew. Krasowski do układy PL2000.
* (FL [GRS80, WGS84, ew. Krasowski] to 1992) Transformacje współrzędnych elipsoidalnych () fi, lambda z systemów odniesienia GRS80, WGS84, ew. Krasowski do układy PL1992.

**Wykorzystane narzędzia i materiały potrzebne do replikacji ćwiczenia:**

Podczas wykonania ćwiczenia korzystaliśmy z takich programów: Spyder 5.4.3 (Anakonda3), Python 3.10.11, Git BASH, TeXstudio (LaTEX editor), JabRef, SO Windows 11.

Wykorzystując takie materiały pomocnicze: udostępnione przez prowadzącego zajęcia linki, własne napisane wcześniej kody, przykłady kodów udostępnione na innych zajęciach, materiały z ćwiczeń z Informatyki I i Geodezji Wyższej I.  
Należy pamiętać że aby wywołać funkcję w wierszu poleceń bezpośrednio przez polecenie „python”, trzeba mieć ustawione środowisko zmienne.

**Przebieg ćwiczenia: (duży tytuł)**

**wstęp i przygotowania do napisania programu (podtytuł)**

Zdecydowałyśmy pisać skrypt w środowisku programistycznym Spyder (ostatniej mogła być aktualizowana w trakcie wykonania ćwiczenia) dla języka Python, które jest dostępne w pakiecie Anaconda, ponieważ oferuje wiele funkcji, takich jak edytor kodu z podświetlaniem składni, przeglądarkę zmiennych, system pomocy oraz wiele innych, które ułatwiają tworzenie i testowanie programów w języku Python.

Następnym krokiem było zaimportowanie biblioteki NumPy. Głównymi zaletami której są łatwe wykonywanie operacji arytmetycznych, algebraicznych, logicznych i innych operacji matematycznych na wielowymiarowych tablicach oraz łatwe indeksowanie i przetwarzanie danych, co ułatwia operacje na dużych zbiorach danych.

**Kolejność pisania kodu**

Na początku została utworzona klasa „Transformacje” (class Transformacje). W tej klasie umieszczone będą kolejne metody obliczania transformacji współrzędnych oraz inne funkcje przydatne w tym procesie.

**Funkcje transformacji FLH do XYZ i powrotne**

Pierwsze napisane definicje (funkcje) to **XYZ\_to\_flh** i **flh\_to\_XYZ**. Napisane zostały przy korzystaniu z materiałów z przedmiotu Geodezja Wyższa, który mieliśmy w semestrze 3. W materiałach do przedmiotu był podany algorytm obliczeniowy.

Funkcja XYZ\_to\_flh wykonuje transformację ze współrzędnych kartezjańskich (XYZ) na elipsoidę geodezyjną (flh) dla współrzędnych początkowych podanych przez użytkownika. Wewnątrz funkcji, a i e2 reprezentują wielkość i spłaszczenie elipsoidy geodezyjnej, p jest odległością punktu od osi obrotu elipsoidy, f jest szerokością geodezyjną, a h jest wysokością geodezyjną. Funkcja wykorzystuje iteracyjny algorytm do rozwiązania równań geodezyjnych i obliczenia wartości f i h dla danych wartości X, Y i Z. Funkcja flh\_to\_XYZ wykonuje transformację odwrotną: z elipsoidy geodezyjnej (flh) na współrzędne kartezjańskie (XYZ). Wewnątrz funkcji, a i e2 reprezentują wielkość i spłaszczenie elipsoidy geodezyjnej, N reprezentuje promień krzywizny normalnej w danym punkcie, a X, Y i Z reprezentują współrzędne kartezjańskie. Funkcja wykorzystuje wzory do obliczenia wartości X, Y i Z dla danych wartości f, l i h.

**Biblioteka argparse**

Po dodaniu powyższych metod do klasy zostało sprawdzone poprawne działanie kodu. Potrzebna do tego była biblioteka argparse (jak było podane w zagadnieniu do ćwiczenia). Następnie utworzyłam skrypt, który akceptuje argumenty z wiersza poleceń i zwraca wynik transformacji. Kod sprawdza, czy skrypt jest uruchamiany jako plik główny (czyli bezpośrednio, a nie importowany jako moduł) poprzez sprawdzenie, czy name == 'main'. Następnie tworzony jest obiekt ArgumentParser, który pozwala zdefiniować argumenty wiersza poleceń. Argumenty te są zdefiniowane jako podprogramy, w tym przypadku 'XYZ' i 'flh'. Każdy podprogram definiuje argumenty, które są wymagane dla danej transformacji. Następnie funkcja parse\_args() jest wywoływana, aby przetworzyć argumenty wiersza poleceń. W zależności od wybranej operacji, transformacja jest wykonywana przez obiekt Transformacje, a wynik jest drukowany na ekranie. Należy pamiętać, że aby wywołać funkcję w wierszu poleceń bezpośrednio przez polecenie „python”, trzeba mieć ustawione zmienne środowiskowe.

**Środowisko git hub i branche w repozytorium**

Po napisaniu powyższej części kodu zauważałam, że jest domyślnie ustawiony branch „main”, więc utworzyłam branch „master”. Branch „main” nie został usunięty, co może być mylące, ponieważ reszta commitów będzie się znajdowała na innej gałęzi.

**Utworzenie funkcji transformacji do układów PL-2000 i PL-1992**

Następnie zostały napisane 4 funkcje: z fl na elipsoidzie GRS80 do układu PL-2000, z fl na elipsoidzie WGS84 do układu PL-2000, z fl na elipsoidzie GRS80 do układu PL-1992 oraz z fl na elipsoidzie WGS84 do układu PL-1992. Nazywają się one odpowiednio: fl\_80\_2\_2000, fl\_84\_2\_2000, fl\_80\_2\_1992, fl\_84\_2\_1992. Funkcje te obliczają dla podanych w Command Window współrzędnych transformacje.

Kolejność obliczeń dla przeliczeń ze współrzędnych elipsoidalnych do układów PL-2000 oraz PL-1992:

najpierw współrzędne są przeliczane na odwzorowanie Gaussa-Krugera. W zależności od wybranej elipsoidy w obliczeniach są uwzględniane ich parametry. Do przeliczenia jest potrzebny południk zerowy (u nas nazwany lambda0). Do wyznaczenia tego południka dla układu 2000 została napisana funkcja lambda0\_2000, która dopasowuje ten południk do współrzędnych dla których obliczamy transformacje. Szczegółowy opis funkcji w rozdziale **Funkcje pomocnicze**. Dla układu 1992 południk zerowy to zawsze 19°.

**Funkcje pomocnicze**

W różnych momentach wykonywania zadania tworzone zostały funkcje pomocnicze, których algorytmy opiszemy w tym rozdziale.

Funkcja **lambda0\_2000** na podstawie podanej wartości współrzędnej l – lambda wyznacza południk zerowy. Jest to 15°, 18°, 21° lub 24°. Funkcja oblicza któremu z podanych południków najbliższa jest wartość południka podanego i dla niego zapisuje zmienną l0 oraz nr\_strefy. Numer strefy przydatny jest w przeliczeniu z odwzorowania Gaussa-Krugera do układu PL-2000.

Funkcja **rad\_to\_dms**, która przelicza wartość z radianów na stopnie, minuty, sekundy. Funkcja przelicza podaną wartość w radianach na żądany format, a następnie printuje wynik w formacie stopnie, minuty, sekundy z dokładnością do 0,00001. Do napisanych definicji została dodana klauzula \_\_name\_\_, na przykładzie tego samego algorytmu.

**Utworzenie funkcji transformacji NEU**

Korzystając z wymienionych wcześniej materiałów z Geodezji Wyższej, została dodana funkcja XYZ\_to\_neu, która przyjmuje cztery argumenty: dX, X, Y i Z. Funkcja wykorzystuje te argumenty do obliczenia macierzy transformacji R, która przekształca wektor przesunięcia dX, wyrażony w układzie współrzędnych kartezjańskich z punktem początkowym w (X, Y, Z), na lokalny układ współrzędnych płaszczyzny stycznej zdefiniowany przez wektor normalny w punkcie (X, Y, Z). Funkcja została dodana do klasy oraz dopisałyśmy do niej część \_\_name\_\_.

**Modyfikacje argumentów funkcji transformacji**

Wszystkie wyżej wymienione funkcje zotały napisane w sposób pozwalający użytkownikowi wpisać komendę oraz współrzędne do Command Window i od razu otrzymać wynik. Jest to szybki i wygodny sposób na obliczenie transformacji dla pojedynczych punktów. Jednak kiedy mamy listę współrzędnych obliczanie ich pojedynczo zajęłoby za dużo czasu. Dlatego wszystkie funkcje stworzone w klasie zostały zmodyfikowane tak, żeby można było obliczyć transformacje dla serii współrzędnych pobranych z pliku tekstowego.  
Zmiany w funkcjach zwykle polegały na: stworzeniu list lub tablic Numpy Array z podanych kolumn pliku, zmianie typu danych na float, zamianie działań arytmetycznych na takie działające dla list lub macierzy.  
Dla niektórych transformacji metody zostały zduplikowane: jest oddzielna dla pojedynczych argumentów podanych przez użytkownika oraz oddzielna dla opcji czytania współrzędnych z pliku.

**Stworzenie i obliczenia dla funkcji pobierającej dane początkowe z pliku txt**

Utworzyłyśmy metodę służąca do pobrania współrzędnych z pliku txt. O formacie współrzędnych w pliku w rozdziale **Użytkowanie programu**. Funkcja **pobranie\_wsp** ma za argumenty ścieżkę do pliku ze współrzędnymi oraz rodzaj transformacji. Użytkownik musi podać oba argumenty przy wywoływaniu funkcji. Następnie funkcja wybiera wybraną przez użytkownika transformację i stosuje ją poprzez wywołanie zmodyfikowanych wcześniej metod transformacji.

Wszystkie funkcje w metodzie pobranie\_wsp zwracają dane w formacie tablicy. Wyniki formatuje metoda z biblioteki Nympy: column\_stack. Współrzędne są formatowane w taki sposób, że w każdym wierszu są współrzędne jednego punktu, w kolejności X, Y, Z (lub odpowiadające f, l ,h itp.) Funkcja **pobranie\_wsp** nie wyświetla wyników lecz zapisuje je do pliku z wynikami używając do tego metody **zapisz**.

**Zapisywanie wyników do pliku**

Ostatnią funkcją jaka została napisana jest funkcja zapisująca wyniki w osobnym pliku tekstowym. Funkcja ta nazywa się **zapisz** i jest prostą funkcją używającą write i np.savetxt. Funkcja ta zaokrągla również wyniki do 5 miejsc po przecinku. Argumentami są: wynik transformacji – tablica współrzędnych punktów, filename – nazwa pod jaką plik zostanie zapisany, header – krótki opis zapisany w pierwszej linijce pliku z wynikami. Funkcja zapisuje plik txt w miejscu w którym znajduje się program. Funkcja ta jest użyta wewnątrz metody pobieranie\_wsp, co oznacza że dane są zapisywane do pliku automatycznie.

**Użytkowanie programu**

Program obsługuje dwa podstawowe rodzaje operacji: „szybkie” operacje dla pojedynczego punktu, o współrzędnych podanych przy wywołaniu komendy przez użytkownika, oraz operacje na współrzędnych pobranych z pliku tekstowego.

Obsługa programu ma miejsce w CommandWindow. Program należy otworzyć otwierając folder w którym dany program się znajduje. Należy pamiętać o włączeniu zmiennych środowiskowych.  
Przed każdą z dwóch dróg wpisujemy w CommandWindow „python skrypt.py”

Do „szybkiego” obliczenia transformacji po wpisaniu „python skrypt.py”, należy wprowadzić nazwę żądanej operacji. Nazwę operacji wybieramy z podanych:

„XYZ\_to\_flh”

„flh\_to\_XYZ”

„XYZ\_to\_neu”

“fl\_GRS80\_to\_2000”

“fl\_GRS80\_to\_1992”

“fl\_WGS84\_to\_2000”

„fl\_WGS84\_to\_1992”.

A następnie podać żądane argumenty, na koniec wcisnąć enter.  
W wypadku błędnego wprowadzenia danych program python wyświetli informacje których danych zabrakło.

Do obliczenia transformacji na danych współrzędnych w pliku txt, należy mieć współrzędne w odpowiednim formacie.  
Program przyjmuje plik tekstowy podający współrzędne w 3 lub 4 kolumnach (4 dla neu), współrzędne powinny być oddzielone przecinkami, bez spacji pomiędzy nimi.  
W CommandWindow najpierw należy wpisać komendę „pobierz\_dane”, następnie wpisać ścieżkę do pliku txt, a następnie wpisać jedną z komend podanych powyżej – wybrać rodzaj transformacji i nacisnąć enter.  
Program zwróci informację „Wyniki zostały zapisane w pliku o nazwie: <nazwa pliku>” i tam właśnie będzie można zobaczyć wyniki transformacji. Program nie wyświetla ich, ponieważ w wypadku gdy plik wyjściowy miałby współrzędne 100 różnych punków, wypisywanie wyników zajęłoby dużo miejsca oraz mogłoby prowadzić do zawieszenia się komputera???

**Dokumentacja README**

Następnym krokiem było napisanie dokumentacji do kodu w pliku README.md, który będzie się wyświetlał na GitHub. Jest to plik tekstowy, który zawiera podstawowe informacje na temat projektu. W naszym przypadku to: nazwa, opis metod transformacji (do czego służy, dane wejściowe i wyjściowe) , sposób instalacji, przykłady używania funkcji i inne ważne informacje, które mogą być przydatne dla użytkowników.

Przy napisaniu wystąpił problem ze śledzeniem plików, ponieważ dodałam do repozytorium plik .png, który wykorzystałam w dokumentacji. Przy rozwiązaniu tego problemu stworzył się w śledzonym folderze clon naszego repozytorium, co powodowało zapisywanie zmian do nowego pliku. Za pomocą prowadzącego ten problem został rozwiązany, w wyniku czego nie było negatywnych skutków.

.idk czy to niżej potrzebne

Żeby pisać sprawozdanie z projektu przy pomocy programu LaTeX był stworzony pomocniczy plik, zawierający informacje na temat zadania, czyli cel ćwiczenia, przebieg i tp. Ponieważ nie miałyśmy jeszcze do czynienia z tym programem i chciałyśmy widzieć, jak musi wyglądać ostateczny plik.