**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, wyszukiwarka Google, 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.

**Przebieg ćwiczenia:**

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 zaletej 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.

Na początek napisałam dwie definicje (funkcje) **XYZ\_to\_flh** i **flh\_to\_XYZ**, korzystając z prezentacji Pana Dominika Próchniewicza, w której był podany algorytm obliczeniowy. Następnie dodałam definicje do klasy „Transformacje” (class Transformacje).

Funkcja XYZ\_to\_flh wykonuje transformację ze współrzędnych kartezjańskich (XYZ) na elipsoidę geodezyjną (flh). 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 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.

Po dodaniu do klasy postanowiłam sprawdzić poprawne działanie kodu. Dla tego zaimportowałam bibliotekę argparse (jak było podane w zagadnieniu do ćwiczenia) i 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.

Żeby