

Projekt 2025 - část 1

předmět Zpracování a vizualizace dat v prostředí Python

Cílem projektu je získat, zpracovat a analyzovat data dostupná na internetu. První část projektu se bude týkat ověření vašeho osvojení technik pro efektivní zpracování, vizualizaci a získání dat.

Orientační struktura projektu

- Část 1 (20 b)
 - efektivní numerické výpočty
 - jednoduchá vizualizace
 - stažení a zpracování dat
- Část 2 (20 b)
 - různé pohledy na data
 - pokročilá vizualizace výsledků
 - zpracování závěrů (porozumění datům)
- Celkový projekt (60 b)
 - znázornění dat na mapě, operace nad těmito daty
 - korelace a predikce
 - automatické vytváření částí zpráv
 - spojení do analytické zprávy

Získání a předzpracování dat (20 bodů)

Vytvořte soubor `part01.py`, který bude implementovat níže uvedené metody. Pro zpracování a vizualizaci dat **není povoleno** použít pokročilých knihoven jako je **Pandas** či **Seaborn**. Kromě vestavěných knihoven (`os`, `sys`, `re`, `gzip`, `pickle`, `csv`, `zipfile`...) byste si měli vystačit s: `numpy`, `matplotlib`, `BeautifulSoup`, `requests`. Další knihovny je možné použít po schválení opravujícím. Pro instalaci virtuálního prostředí `venv` je možné použít tento soubor `requirements.txt`.

Úkol 1: Numerický výpočet inference vln (5 bodů)

Cílem této funkce je pro dvě pole (souřadnice X a Y) a n prvků zdrojů S vypočítat amplitudu Z . Pro každý bod i, j v souřadnicovém systému a pro každý zdroj s je amplituda definována pro vzdálenosti těchto bodů

$$d_s[i, j] = \sqrt{(X[i] - S_s[0])^2 + (Y[j] + S_s[1])^2}$$

Pro každý bod pak můžeme určit amplitudu na základě vzdáleností.

$$k = 2\pi/\lambda$$

$$Z[i, j] = \sum_{\forall s \in S} \frac{\cos(k \cdot d_s[i, j])}{1 + d_s[i, j]}$$

Pro získání plného hodnocení je nutné se zaměřit na efektivitu a využít možností knihovny *numpy* - t.j. vyhnout se procházení všech prvků cyklem `for`. V zadání máte k dispozici naivní verzi `wave_inference_bad`, kterou můžete použít jako referenci - funkčnost musí odpovídat. Je zakázáno použít knihovnoví funkci, která by tuto úlohu řešila sama, redukční funkce jsou samozřejmě povolené. Pro ladění vám může pomoci i funkce vizualizace z úkolu 2 při porovnání vaší a referenční implementace. Doporučuji použít funkci `np.meshgrid`, dále si udělat z 2D matice udělat 3D tensor (přidat rozměr pro různé zdroje), kdy jedna dimenze budou jednotlivá *S* (v testovacím případě tedy o rozměru 200, 200, 3) a na konci provést redukci.

Prototyp funkce:

```
Python
def wave_inference(x: NDArray[any], y: NDArray[any],
                  source : NDArray[any], wavelength: float) -> NDArray[any]:
```

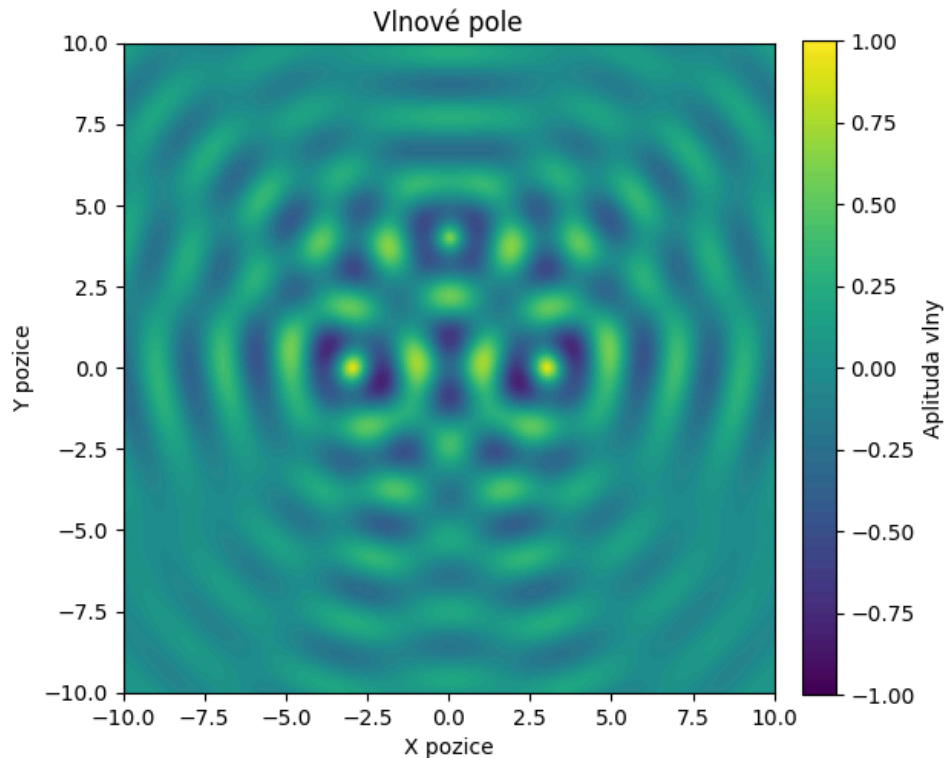
Ukázka volání:

```
Python

wave_inference(
    np.linspace(-10, 10, 200),
    np.linspace(-10, 10, 200),
    np.array([[-3, 0], [3, 0], [0, 4]]),
    2)
```

Úkol 2: Generování z vlnových délek (5 bodů)

Navrhněte funkci `plot_wave`, která bude vizualizovat výsledek z předcházející funkce (pokud ji nebudete implementovat, můžete použít referenční pomalou implementaci). Je nutné, aby graf vypadal podobně, jako je znázorněno níže - rozsahy os, zarovnání colorbaru a podobně.



Funkce `generate_graph` má další dva argumenty - boolean hodnotu `show_figure`, která určuje, zda se má graf zobrazit pomocí funkce `show()` a `save_path`, která (pokud je nastavena), určuje, kam se má graf uložit pomocí funkce `savefig()`.

Prototyp funkce

Python

```
def plot_wave(Z: NDArray[any], x: NDArray[any], y: NDArray[any],  
             show_figure: bool = False, save_path: str | None = None):
```

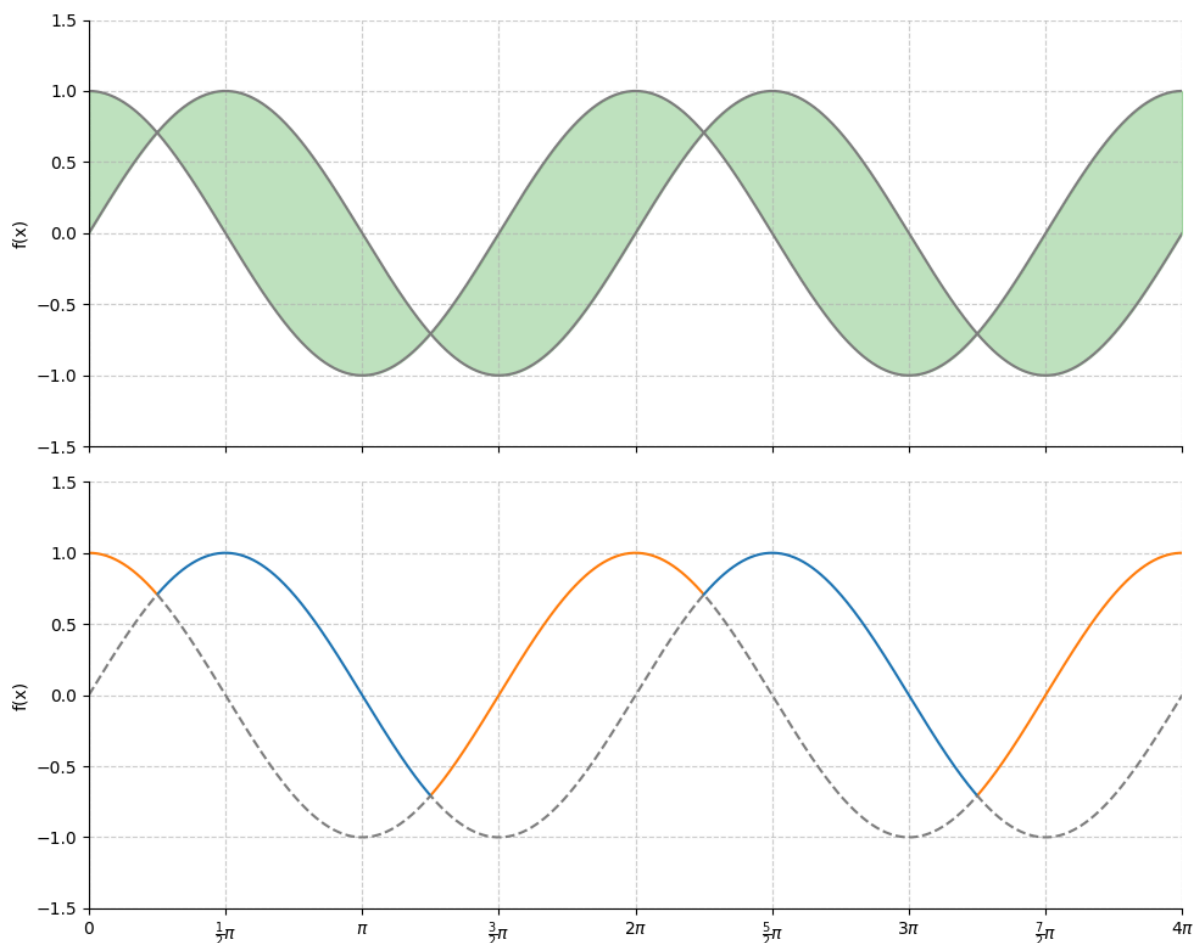
Ukázka použití

Python

```
X = np.linspace(-10, 10, 200)  
Y = np.linspace(-10, 10, 200)  
A = wave_inference_bad( # muzete pouzit i vasi implementaci samozrejme  
    X, Y, np.array([[ -3, 0], [ 3, 0], [ 0, 4]]),  
    2  
)  
plot_wave(A, X, Y, show_figure=True)
```

Úkol 3: Pokročilá vizualizace sinusového signálu (5 bodů)

Vytvořte graf se dvěma podgrafy obsahující funkce $\sin(x)$ a $\cos(x)$ v rozsahu od 0 do 4π . V prvním podgrafu jsou zobrazeny oba grafy a graficky znázorněna plocha mezi oběma funkcemi. V druhém podgrafu je čárkovaně znázorněno minimum mezi oběma funkcemi a barevně maximum. Pro výpočet minima a maxima použijte numpy funkce `min` a `max`, pro barvení vhodně numpy podmínky (oranžově tam, kde je zdrojem maximální hodnoty \cos , modře pro zdroj \sin). Upravte také body na ose tak, aby vypadaly, jako na tomto vzorovém obrázku a použijte sdílenou osu y a x , také využijte latex označení osy x . Pro argumenty `show_figure` a `save_path` platí stejné podmínky, jako v druhém úkolu.



Prototyp funkce

Python

```
def generate_sinus(show_figure: bool = False, save_path: str | None = None):
```

Úkol 4: Stažení tabulky (5 bodů)

Ze stránek <https://ehw.fit.vutbr.cz/izv/stanice.html> stáhněte meteorologických stanic. Jedná se o kopii stránek CHMI a je zakázáno se připojovat na oficiální stránky. Url, ze které

stahujete, můžete mít uloženou v kódu. Nemusíte pracovat rovnou s touto stránkou (v prohlížeči např. přes *Network panel* zjistíte, že stránky jsou řešeny poměrně “originálním” stylem). Každý sloupec bude reprezentovaný položkou ve slovníku obsahující seznam všech hodnot ve vhodném datovém typu. Příklad výstupu

JSON

```
{'positions': ['Cheb', 'Karlovy Vary'],  
 'lats': [50.0683, 50.2016],  
 'longs': [12.3913, 12.9139],  
 'heights': [483.0, 603.0]}
```

Výstupem funkce bude slovník (dict) obsahující seznamy (list) pro jednotlivé řádky. Validita výstupního formátu je základním způsobem testována i v přiloženém *unittestu*. Můžete počítat s tím, že struktura stránek se nezmění, vlastní načítání je však povinné dělat ze stránek <https://ehw.fit.vutbr.cz/izv> a je přísně zakázáno přistupovat na stránky CHMI. Stránky se při testování nebudou měnit.

Prototyp funkce

Python

```
def download_data() -> Dict[str, List[Any]]:
```

Testování

K otestování funkčnosti můžete přistoupit dvěma základními způsoby. Buď do části, která se spouští pouze pokud je zavolán celý skript (if `__name__=="__main__"`), vložíte vaši testovací sekvenci, nebo můžete použít předpřipravený *unittest* v souboru *test_part01.py* a knihovnu *pytest*. Pokud máte tuto knihovnu nainstalovanou, můžete spustit příkaz `pytest` či `python3 -m pytest`. **Pokud projdete testovacím skriptem, je to podmínka nutná (nikoliv dostačující) k dobrému hodnocení.**

Dokumentace všech částí (souborů, funkcí) bude přímo v odevzdaném souboru. Snažte se dodržovat konvenci PEP 257 [<https://www.python.org/dev/peps/pep-0257>] a PEP 8 [<https://www.python.org/dev/peps/pep-0008/>]. Pozor, Python má přímo definovaný kódovací styl (narozdíl od C), a proto kontrola bude součástí hodnocení (**1 bod**).

Odevzdávání a hodnocení

Do 26. 10. 2025 odevzdejte jeden soubor *part01.py*.

Hodnotit se bude zejména:

- správnost výsledků

- vizuální dojem z grafů
- kvalita kódu
 - efektivitu implementace a reprezentace (i rychlost v porovnání s ostatními řešeními), využívání efektivních funkcí (např. NumPy)
 - přehlednost kódu
 - dodržení standardů a zvyklostí pro práci s jazykem Python - dokumentační řetězce, splnění PEP8
 - dokumentace funkcí a souborů
 - znovupoužitelnost kódů - správná izolace potřebných částí do funkcí - kód při importu nesmí nic vykonat!

Celkem za první část můžete získat až 20 bodů, přičemž je k zápočtu nutné získat z této části minimálně 1 bod.

Dotazy a připomínky

Na mailu mrizek@fit.vutbr.cz či osobně po přednáškách.