Matematický software

Zápočtový dokument

**Jméno:** Ondřej Švorc

**Kontaktní email:** ondrejsvorc@email.cz​

**Datum odevzdání:** xx.xx.2024

**Odkaz na repozitář:** <https://github.com/ondrejsvorc/UJEP/tree/main/MSW/zapocet>

# Formální požadavky

**Cíl předmětu:**

Cílem předmětu je ovládnout vybrané moduly a jejich metody pro jazyk Python, které vám mohou být užitečné jak v dalších semestrech vašeho studia, závěrečné práci (semestrální, bakalářské) nebo technické a výzkumné praxi.

**Získání zápočtu:**

Pro získání zápočtu je nutné částečně ovládnout více než polovinu z probraných témat. To prokážete vyřešením vybraných úkolů. V tomto dokumentu naleznete celkem 10 zadání, která odpovídají probíraným tématům. Vyberte si 6 zadání, vypracujte je a odevzdejte. Pokud bude všech 6 prací korektně vypracováno, pak získáváte zápočet. Pokud si nejste jisti korektností vypracování konkrétního zadání, pak je doporučeno vypracovat více zadání a budou se započítávat také, pokud budou korektně vypracované.

**Korektnost vypracovaného zadání:**

Konkrétní zadání je považováno za korektně zpracované, pokud splňuje tato kritéria:

1. Použili jste numerický modul pro vypracování zadání místo obyčejného pythonu
2. Kód neobsahuje syntaktické chyby a je interpretovatelný (spustitelný)
3. Kód je čistý (vygooglete termín clean code) s tím, že je akceptovatelné mít ho rozdělen do Jupyter notebook buněk (s tímhle clean code nepočítá)

**Forma odevzdání:**

Výsledný produkt odevzdáte ve dvou podobách:

1. Zápočtový dokument
2. Repozitář s kódem

Zápočtový dokument (vyplněný tento dokument, který čtete) bude v PDF formátu. V řešení úloh uveďte důležité fragmenty kódu a grafy/obrázky/textový výpis pro ověření funkčnosti. Stačí tedy uvést jen ty fragmenty kódu, které přispívají k jádru řešení zadání. Kód nahrajte na veřejně přístupný repozitář (github, gitlab) a uveďte v práci na něj odkaz v titulní straně dokumentu. Strukturujte repozitář tak, aby bylo intuitivní se vyznat v souborech (doporučuji každou úlohu dát zvlášť do adresáře).

**Podezření na plagiátorství:**

Při podezření na plagiátorství (významná podoba myšlenek a kódu, která je za hranicí pravděpodobnosti shody dvou lidí) budete vyzváni k fyzickému dostavení se na zápočet do prostor univerzity, kde dojde k vysvětlení podezřelých partií, nebo vykonání zápočtového testu na místě z matematického softwaru v jazyce Python.

**Kontakt:**

Při nejasnostech ohledně zadání nebo formě odevzdání se obraťte na vyučujícího.

# 1. Knihovny a moduly pro matematické výpočty

**Zadání:**

V tomto kurzu jste se učili s některými vybranými knihovnami. Některé sloužily pro rychlé vektorové operace, jako numpy, některé mají naprogramovány symbolické manipulace, které lze převést na numerické reprezentace (sympy), některé mají v sobě funkce pro numerickou integraci (scipy). Některé slouží i pro rychlé základní operace s čísly (numba).

Vaším úkolem je změřit potřebný čas pro vyřešení nějakého problému (např.: provést skalární součin, vypočítat určitý integrál) pomocí standardního pythonu a pomocí specializované knihovny. Toto měření proveďte alespoň pro 5 různých úloh (ne pouze jiná čísla, ale úplně jiné téma) a minimálně porovnejte rychlost jednoho modulu se standardním pythonem. Ideálně proveďte porovnání ještě s dalším modulem a snažte se, ať je kód ve standardním pythonu napsán efektivně. ​

**Řešení:**

Ve svém řešení jsem provedl měření na **5** různých úlohách:

1. Porovnání dob provádění **skalárního součinu**

2. Porovnání dob provádění **výpočtu derivace**

3. Porovnání dob provádění **maticového součinu**

4. Porovnání dob výpočtu **faktoriálu**

5. Porovnání dob výpočtu **statistických funkcí**

V měřeních je primárně porovnáván standardní **Python** s knihovnou **NumPy**. V případě 2. měření je porovnáván **Python** s knihovnou **SymPy**. Všechna měření jsou doplněna o **grafy**, z nichž lze lehce vydedukovat rychlost Pythonu a porovnávané knihovny při různě velkých vstupech.

U téměř všech měření rychlostně zvítězila daná knihovna nad Pythonem. Při velmi malých vstupech (cca do velikosti 100) byla u 1. a 4. měření porovnávaná knihovna pomalejší, ale to se při překročení zmíněné hranice změnilo, a to již nastálo. Zajímavé bylo porovnání výpočtu statistických funkcí (tedy měření č. 5), kdy byl standardní Python při jakémkoliv vstupu nejrychlejší; pouze v případě kumulativní sumy od vstupu 1000 začala být knihovna NumPy nepatrně rychlejší. Další zajímavý poznatek se objevil při měření č. 4 (výpočet faktoriálu), kdy bylo rekurzivní řešení pomalejší než iterativní řešení; knihovna NumPy byla pomalejší než obě zmíněné metody pouze do zmíněného vstupu 100.

Veškerý kód a doprovodné grafy potvrzující má tvrzení jsou k dispozici [zde](https://github.com/ondrejsvorc/UJEP/blob/main/MSW/zadani/1.ipynb).

# 2. Vizualizace dat

**Zadání:**

V jednom ze cvičení jste probírali práci s moduly pro vizualizaci dat. Mezi nejznámější moduly patří matplotlib (a jeho nadstavby jako seaborn), pillow, opencv, aj. Vyberte si nějakou zajímavou datovou sadu na webovém portále Kaggle a proveďte datovou analýzu datové sady. Využijte k tomu různé typy grafů a interpretujte je (minimálně alespoň 5 zajímavých grafů)​. Příklad interpretace: z datové sady pro počasí vyplynulo z liniového grafu, že v létě je vyšší rozptyl mezi minimální a maximální hodnotou teploty. Z jiného grafu vyplývá, že v létě je vyšší průměrná vlhkost vzduchu. Důvodem vyššího rozptylu může být absorpce záření vzduchem, který má v létě vyšší tepelnou kapacitu.

**Řešení:**

Ve svém řešení jsem provedl měření na **7** různých úlohách:

1. Poměr výskytu seriálů a filmů na platformě Netflix (**koláčový graf**)

2. 10 nejvíce produkujících zemí na platformě Netflix (**sloupcový graf**)

3. Počet filmů, vydaných v daném roce, na platformě Netflix (**histogram**)

4. Režiséři s největším počtem titulů na platformě Netflix (**vodorovný pruhový graf**)

5. Herci vyskytující se v největším počtu titulů na platformě Netflix (**vodorovný pruhový graf**)

6. 50 filmů na platformě Netflix začínajících na písmeno 'A' vydaných mezi lety 2000–2020 (včetně) (**spojnicový graf**)

7. 50 seriálů na platformě Netflix začínajících na písmeno 'A' vydaných mezi lety 2000–2020 (včetně) (**spojnicový graf**)

# 5. Hledání kořenů rovnice

**Zadání:**

Vyhledávání hodnot, při kterých dosáhne zkoumaný signál vybrané hodnoty je důležitou součástí analýzy časových řad. Pro tento účel existuje spousta zajímavých metod. Jeden typ metod se nazývá ohraničené (například metoda půlení intervalu), při kterých je zaručeno nalezení kořenu, avšak metody typicky konvergují pomalu. Druhý typ metod se nazývá neohraničené, které konvergují rychle, avšak svojí povahou nemusí nalézt řešení (metody využívající derivace). Vaším úkolem je vybrat tři různorodé funkce (například polynomiální, exponenciální/logaritmickou, harmonickou se směrnicí, aj.), které mají alespoň jeden kořen a nalézt ho jednou uzavřenou a jednou otevřenou metodou. Porovnejte časovou náročnost nalezení kořene a přesnost nalezení.

**Řešení:**

doplňte

# 6. Generování náhodných čísel a testování generátorů

**Zadání:**

Tento úkol bude poněkud kreativnější charakteru. Vaším úkolem je vytvořit vlastní generátor semínka do pseudonáhodných algoritmů. Jazyk Python umí sbírat přes ovladače hardwarových zařízení různá fyzická a fyzikální data. Můžete i sbírat data z historie prohlížeče, snímání pohybu myší, vyzvání uživatele zadat náhodné úhozy do klávesnice a jiná unikátní data uživatelů.

**Řešení:**

doplňte

# 8. Derivace funkce jedné proměnné

**Zadání:**

Numerická derivace je velice krátké téma. V hodinách jste se dozvěděli o nejvyužívanějších typech numerické derivace (dopředná, zpětná, centrální). Jedno z neřešených témat na hodinách byl problém volby kroku. V praxi je vhodné mít krok dynamicky nastavitelný. Algoritmům tohoto typu se říká derivace s adaptabilním krokem. Cílem tohoto zadání je napsat program, který provede numerickou derivaci s adaptabilním krokem pro vámi vybranou funkci. Proveďte srovnání se statickým krokem a analytickým řešením.

**Řešení:**

doplňte