Matematický software

Zápočtový dokument

Jméno:	**Mykhailo Pastram**
Kontaktní email:	**michaelpastram@gmail.com**
Datum odevzdání:	**16.08.2024**
Odkaz na repozitář:	**https://github.com/pastram1905/MSW**

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:

Výpočet skalárního součinu vektoru (čistý Python):

```
from time import process_time

vector_size = 100000 # velikost vektorů
repeat = 1000 # počet opakování výpočtu
a = list(range(vector_size)) # vytvoření vektoru

# funkce pro ručný výpočet skalárního součinu
def dot_product(a, b):
  result = 0
  for i, j in zip(a, b):
    result += i * j
  return result

# měření času výpočtu
start = process_time()
for i in range(repeat):
  dot_product(a, a)
end = process_time()

dot_res_py = dot_product(a, a)
dot_time_py = round(end - start, 10)

# výpis výsledků
print(f"Standardní Python - výsledek: {dot_res_py}, použitý čas: {dot_time_py} sekund")
```

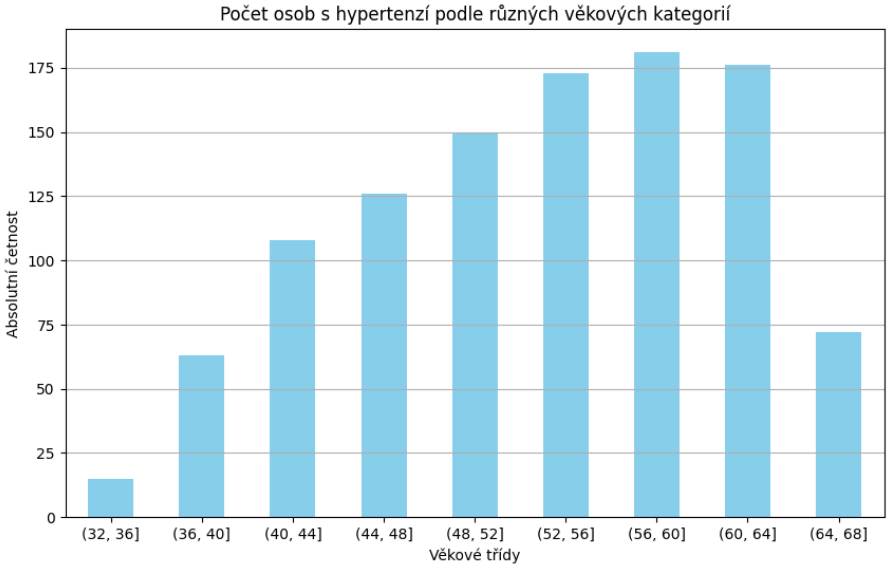
Výpočet skalárního součinu vektoru (NumPy):

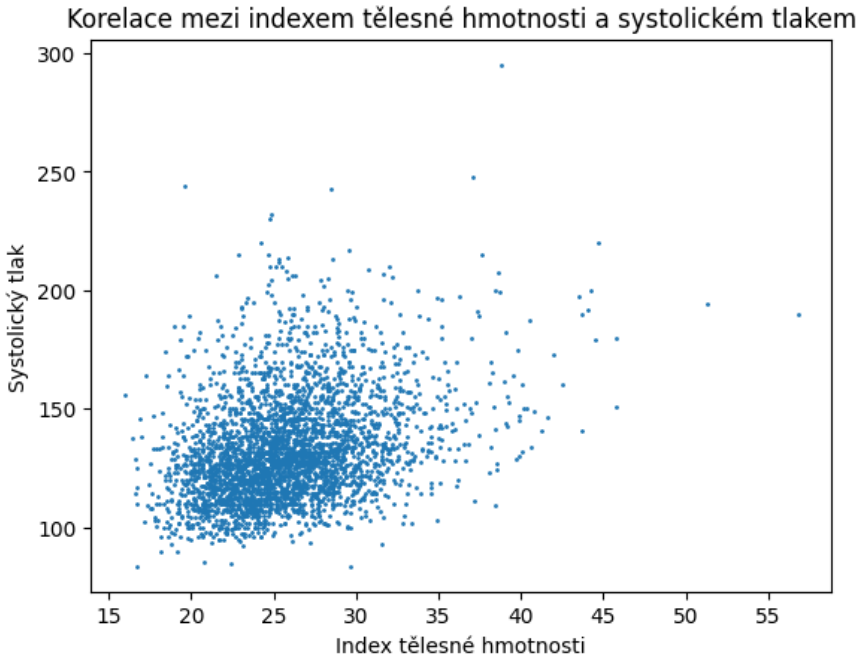
```
import numpy as np
from time import process_time

vector_size = 100000 # velikost vektorů
repeat = 1000 # počet opakování výpočtu
arr = np.arange(vector_size, dtype=np.int64) # vytvoření vektoru

# měření času výpočtu
start = process_time()
for i in range(repeat):
  np.dot(arr, arr)
end = process_time()

dot_res_np = np.dot(arr, arr)
dot_time_np = round(end - start, 10)

# výpis výsledků
print(f"NumPy - výsledek: {dot_res_np}, použitý čas: {dot_time_np} sekund")
```

2. Vizualizace dat

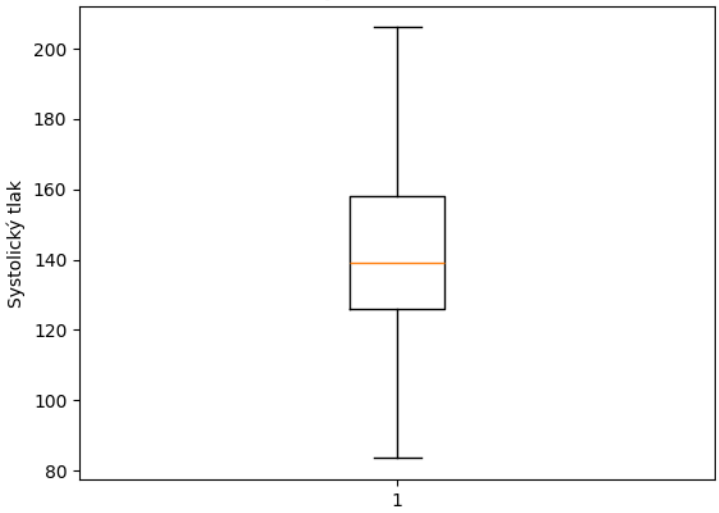
Zadání:
V jednom ze cvičení jste probírali práci s moduly pro vizualizaci dat. Mezi nejznámější moduly patří matplotlib (a jeho nadstavby jako seaborn), pillow, opencv, aj. Vyberte si nějakou zajímavou datovou sadu na webovém portále Kaggle a proveďte datovou analýzu datové sady. Využijte k tomu různé 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:

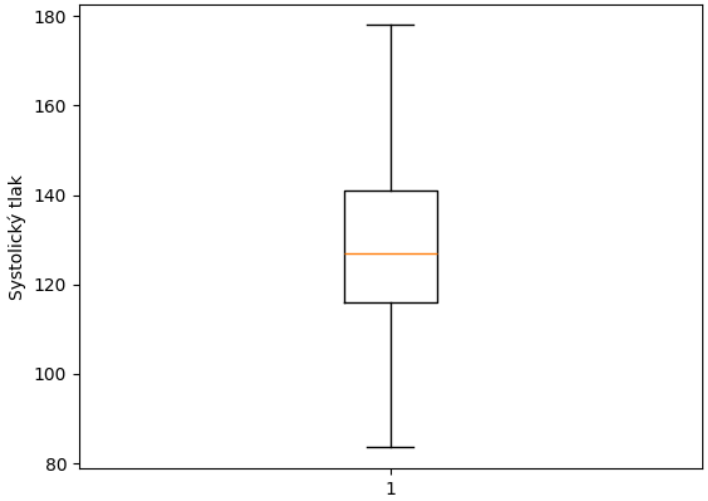




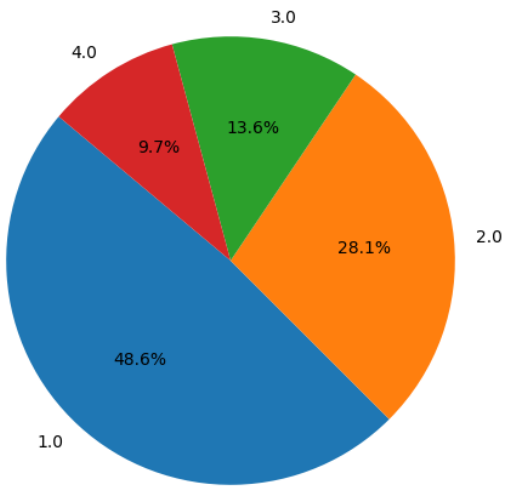
Distribuce systolického tlaku u osob ohrožených kardiovaskulárními onemocněními v příštích 10 letech



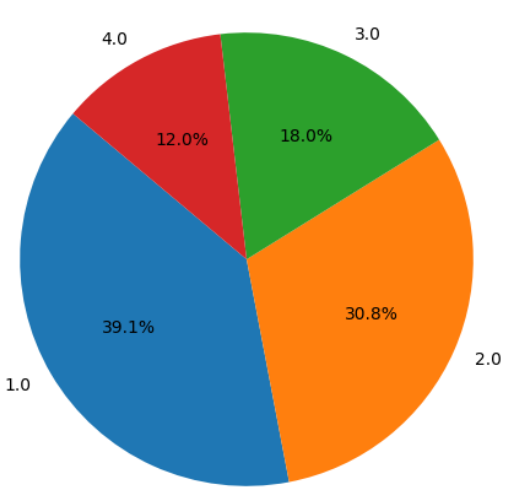
Distribuce systolického tlaku u osob neohrožených kardiovaskulárním onemocněním v příštích 10 letech



Úroveň vzdělání u osob s hypertenzí



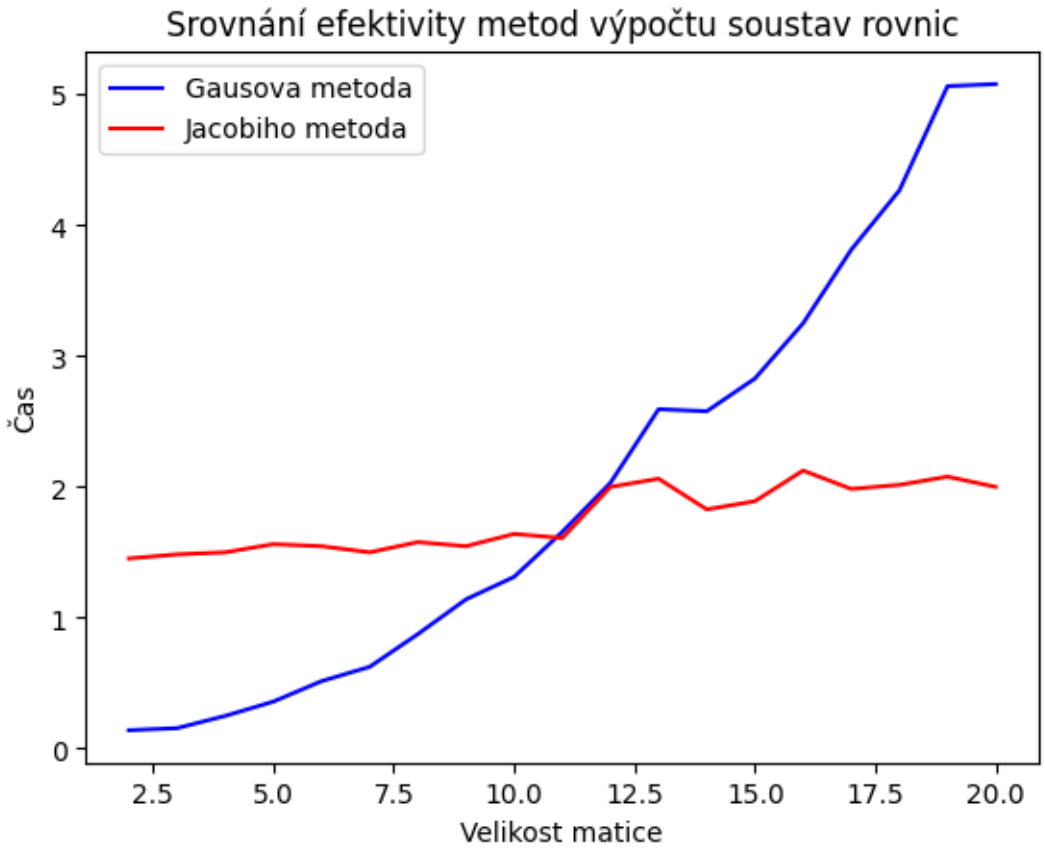
Úroveň vzdělání u osob bez hypertenze



3. Úvod do lineární algebry

Zadání:
Důležitou částí studia na přírodovědecké fakultě je podobor matematiky zvaný lineární algebra. Poznatky tohoto oboru jsou základem pro oblasti jako zpracování obrazu, strojové učení nebo návrh mechanických soustav s definovanou stabilitou. Základní úlohou v lineární algebře je nalezení neznámých v soustavě lineárních rovnic. Na hodinách jste byli obeznámeni s přímou a iterační metodou pro řešení soustav lineárních rovnic. Vaším úkolem je vytvořit graf, kde na ose x bude velikost čtvercové matice a na ose y průměrný čas potřebný k nalezení uspokojivého řešení. Cílem je nalézt takovou velikost matice, od které je výhodnější využít iterační metodu.

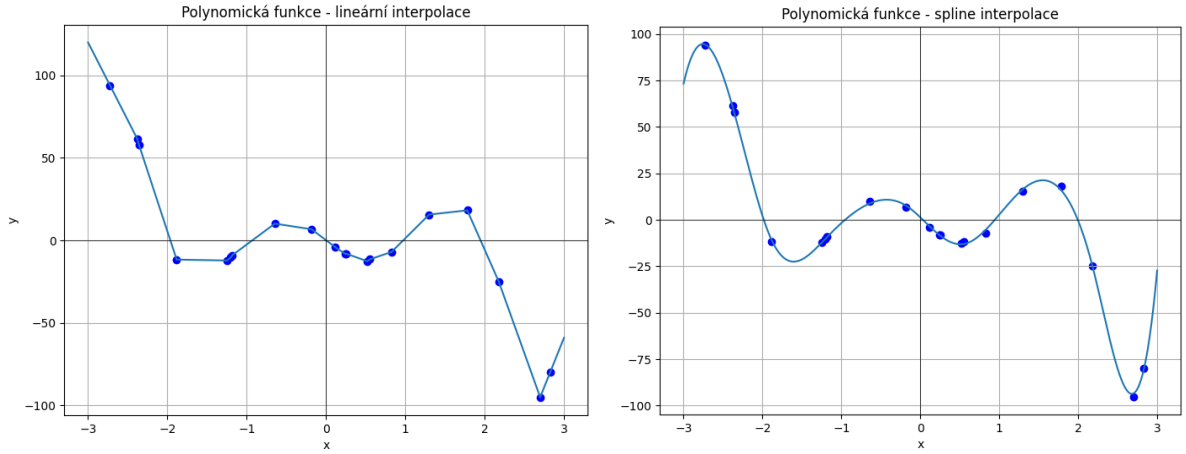
Řešení:

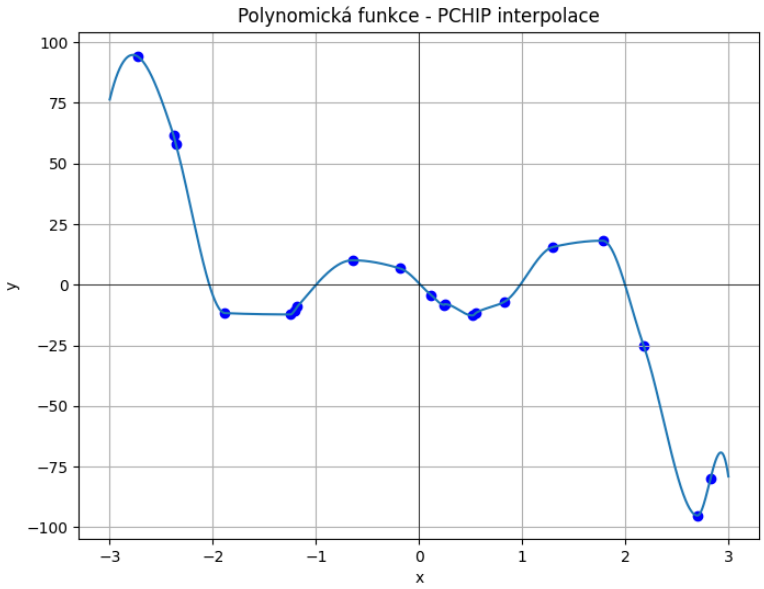


4. Interpolace a aproximace funkce jedné proměnné

Zadání:
Během měření v laboratoři získáte diskrétní sadu dat. Často potřebujete data i mezi těmito diskrétními hodnotami a to takové, které by nejpřesněji odpovídaly reálnému naměření. Proto je důležité využít vhodnou interpolační metodu. Cílem tohoto zadání je vybrat si 3 rozdílné funkce (např. polynom, harmonická funkce, logaritmus), přidat do nich šum (trošku je v každém z bodů rozkmitejte), a vyberte náhodně některé body. Poté proveďte interpolaci nebo aproximaci funkce pomocí alespoň 3 rozdílných metod a porovnejte, jak jsou přesné. Přesnost porovnáte s daty, které měly původně vyjít. Vhodnou metrikou pro porovnání přesnosti je součet čtverců (rozptylů), které vzniknou ze směrodatné odchylky mezi odhadnutou hodnotou a skutečnou hodnotou.

Řešení:





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:

```
import os
import time
import random
import hashlib
import uuid
from pynput import mouse, keyboard
from random import randint, seed


# genarátor semínka do pseudonáhodných algoritmů
def gen_seed():
    entropy_data = []

    # sběr pozice myši
    def on_move(x, y):
        entropy_data.append(f"Mouse position: {x},{y}")

    # sběr kliknuti myši
    def on_click(x, y, button, pressed):
        entropy data.append(f"Mouse click: {x},{y} {button} {'Pressed' if pressed else 
'Released'}")

    # sběr scrollu myši
    def on_scroll(x, y, dx, dy):
        entropy data.append(f"Mouse scroll: {dx},{dy}")

    # sběr náhodných úhozů klávesnice
    def on_press(key):
        entropy data.append(f"Key pressed: {key}")

    def on_release(key):
        entropy data.append(f"Key released: {key}")

    # start posluchačů pro myš a klávesnici
    mouse_listener = mouse.Listener(on_move=on_move, on_click=on_click, on_scroll=on_scroll)
    keyboard_listener = keyboard.Listener(on_press=on_press, on_release=on_release)
    mouse_listener.start()
    keyboard_listener.start()

    # časový údaj jako zdroj entropie
    entropy_data.append(f"Current time: {time.time_ns()}")

    # systémové informace jako zdroj entropie
    entropy_data.append(f"UUID: {uuid.uuid4()}")
    entropy_data.append(f"MAC Address: {':'.join(['{:02x}'.format((uuid.getnode() >> elements) 
& 0xff) for elements in range(0,2*6,2)])}")

    # ukončení posluchačů po 5 sekundách
    time.sleep(5)
    mouse_listener.stop()
    keyboard_listener.stop()

    # kombinace a hashování dat
    combined_entropy = ''.join(entropy_data)
    new_seed = int(hashlib.sha256(combined_entropy.encode('utf-8')).hexdigest(), 16)

    print(f"Generated seed: {new_seed}")

    return new_seed
```

7. Metoda Monte Carlo

Zadání:
Metoda Monte Carlo představuje rodinu metod a filozofický přístup k modelování jevů, který využívá vzorkování prostoru (například prostor čísel na herní kostce, které mohou padnout) pomocí pseudonáhodného generátoru čísel. Jelikož se jedná spíše o filozofii řešení problému, tak využití je téměř neomezené. Na hodinách jste viděli několik aplikací (optimalizace portfolia aktiv, řešení Monty Hall problému, integrace funkce, aj.). Nalezněte nějaký zajímavý problém, který nebyl na hodině řešen, a získejte o jeho řešení informace pomocí metody Monte Carlo. Můžete využít kódy ze sešitu z hodin, ale kontext úlohy se musí lišit.

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

9. Integrace funkce jedné proměnné

Zadání:
V oblasti přírodních a sociálních věd je velice důležitým pojmem integrál, který představuje funkci součtů malých změn (počet nakažených covidem za čas, hustota monomerů daného typu při posouvání se v řetízku polymeru, aj.). Integraci lze provádět pro velmi jednoduché funkce prostou Riemannovým součtem, avšak pro složitější funkce je nutné využít pokročilé metody. Vaším úkolem je vybrat si 3 různorodé funkce (polynom, harmonická funkce, logaritmus/exponenciála) a vypočíst určitý integrál na dané funkci od nějakého počátku do nějakého konečného bodu. Porovnejte, jak si každá z metod poradila s vámi vybranou funkcí na základě přesnosti vůči analytickému řešení.

Řešení:

```
import numpy as np
from scipy import integrate

# polynomická funkce
def f(x):
    return 12*x**5 + 5*x**3 - 10*x + 2

a = 0 # dolní mez
b = 2 # horní mez
dx = 0.1 # šířka vzorkování

# integrace lichoběžníkovým pravidlem
def trapezoid naive(a, b, dx):
    integral = 0
    x = a
    i = 0
    while x < b:
        integral += dx * (f(x) + f(x+dx))/2
        x += dx
    return integral

print("Lichoběžníkové pravidlo (naivní):", trapezoid_naive(a, b, dx))

# integrace ve formě Newtonových-Cotesových vzorců
def newton_integ(a, b, dx):
    n = int((b-a)//dx)+1
    integral = f(a) + f(b)
    for i in range(1, n):
        integral += 2*f(a+i*dx)
    integral *= dx/2
    return integral

print("Newtonové-Cotesové vzorce:", newton integ(a, b, dx))

# lichoběžníkové pravidlo (numpy)
x = np.arange(a, b+dx, dx)
y = f(x)

print("Lichoběžníkové pravidlo (numpy):", np.trapezoid(y, dx=dx))

# lichoběžníkové pravidlo (scipy)
print(f"Lichoběžníkové pravidlo (scipy): {integrate.trapezoid(y, x=x)}")

# simpson integrace
x = np.arange(a, b+dx, dx)
y = f(x)

print(f"Simpson integrace (scipy): {integrate.simpson(y=y, x=x)}")

# quad integrace
print(f"Quad integrace (scipy): {integrate.quad(f, a, b)}")

print("Analytické řešení: I(12*x**5 + 5*x**3 - 10*x + 2)(0, 2) = 132")
```

10. Řešení obyčejných diferenciálních rovnic

Zadání:
Diferenciální rovnice představují jeden z nejdůležitějších nástrojů každého přírodovědně vzdělaného člověka pro modelování jevů kolem nás. Vaším úkolem je vybrat si nějakou zajímavou soustavu diferenciálních rovnic, která nebyla zmíněna v sešitech z hodin a pomocí vhodné numerické metody je vyřešit. Řešením se rozumí vizualizace jejich průběhu a jiných zajímavých informací, které lze z rovnic odvodit. Proveďte také slovní okomentování toho, co lze z grafu o modelovaném procesu vyčíst.

Řešení:
doplňte