Matematický software

Zápočtový dokument

Jméno:	Jakub Jelínek
Kontaktní email:	jelinekjakub@icloud.com
Datum odevzdání:	28.06.2023
Odkaz na repozitář:	https://github.com/jelinekjakub/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 alespoň polovinu z probraných témat. To prokážete vyřešením vybraných úkolů. V tomto dokumentu naleznete celkem 10 zadání, která odpovídají probíraným tématům. Vyberte si 5 zadání, vypracujte je a odevzdejte. Pokud bude všech 5 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 pro nás hodnotitele 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:
```
def prumer_python(cisla):
    return sum(cisla) / len(cisla)
def prumer_numpy(cisla):
    return np.mean(cisla)
Python: průměr: 5000000.0 Čas vykonání: 0.2372 s
Numpy: průměr: 5000000.0 Čas vykonání: 0.007 s
Rozdíl: 0.2302 s

def trideni_python(cisla):
    return sorted(cisla)
def trideni_numpy(cisla):
    return np.sort(cisla)
Python: Čas vykonání: 0.5587 s
Numpy: Čas vykonání: 0.087 s
Rozdíl: 0.4717 s

def max_python(cisla):
    return max(cisla)
def max_numpy(cisla):
    return np.max(cisla)
Python: max: 0.9999999429093457 Čas vykonání: 0.126 s
Numpy: max: 0.9999999429093457 Čas vykonání: 0.005 s
Rozdíl: 0.121 s

def soucet_python(cisla):
    return sum(cisla)
def soucet_numpy(cisla):
    return np.sum(cisla)
Python: list: 499509.0181918521 Čas vykonání: 0.0513 s
Numpy: list: 499509.0181918529 Čas vykonání: 0.001 s
Rozdíl: 0.0503 s
```

```python
def sqrt_python(cisla):
    list = []
    for item in cisla:
        list.append(item*item)
    return list
def sqrt_numpy(cisla):
    return np.square(cisla)
```

```
Python: První: 0.12296 Čas vykonání: 0.1332 s
Numpy: První: 0.12296 Čas vykonání: 0.004 s
Rozdíl: 0.1292 s
```

```python
def nasobeni_matice_python(matice1, matice2):
    vysledek = [[0 for x in range(len(matice2[0]))] for x in range(len(matice1))]
    for i in range(len(matice1)):
        for j in range(len(matice2[0])):
            for k in range(len(matice2)):
                vysledek[i][j] += matice1[i][k] * matice2[k][j]
    return vysledek
def nasobeni_matice_numpy(matice1, matice2):
    return np.matmul(matice1, matice2)
```

```
Python: Tvar matice: (250, 250) Čas vykonání: 5.4666 s
Numpy: Tvar matice: (250, 250) Čas vykonání: 0.0 s
Rozdíl: 5.4666 s
```

2. Vizualizace dat

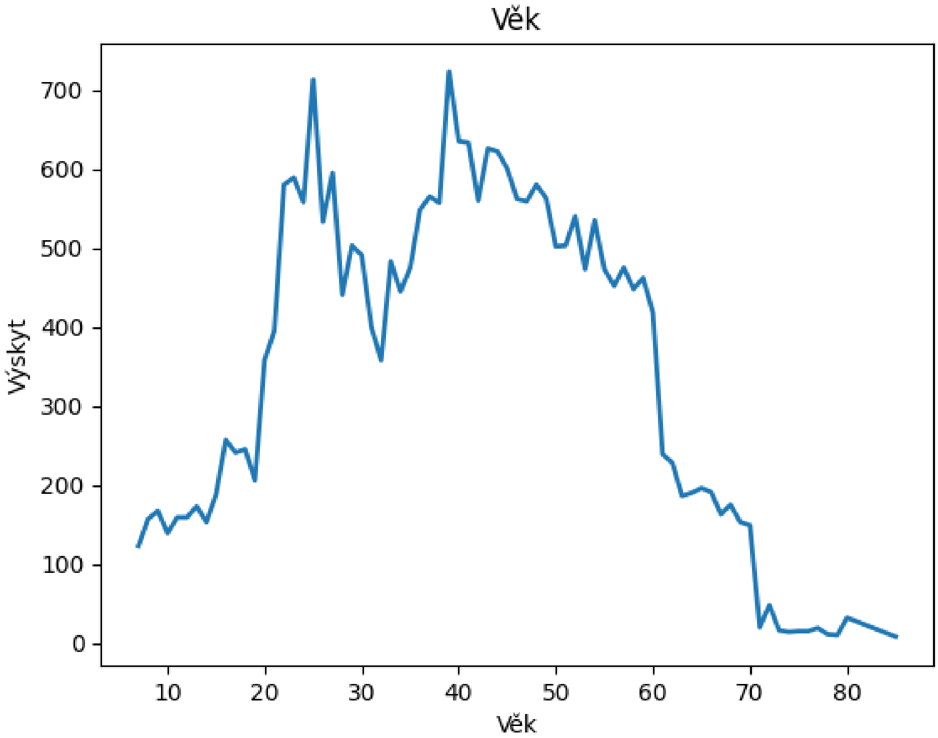
Zadání:
V jednom ze cvičení jste probírali práci s moduly pro vizualizaci dat. Mezi nejznámější moduly patří matplotlib (a jeho nadstavby jako seaborn), pillow, opencv, aj. Vyberte si nějakou zajímavou datovou sadu na webovém portále Kaggle a proveďte datovou analýzu datové sady. Využijte k tomu různé 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:
Datový set: pasažéři letatel
Z prvního grafu lze vidět, že nejvíce cestují lidé ve věku od 20 do třiceti let, tedy studenti nebo absolventi a lidé ve 40 letech věku, dále počet s narůstajícím věkem ubývá.

Z druhého grafu - lidé převážně uskutečňují cesty za prací.

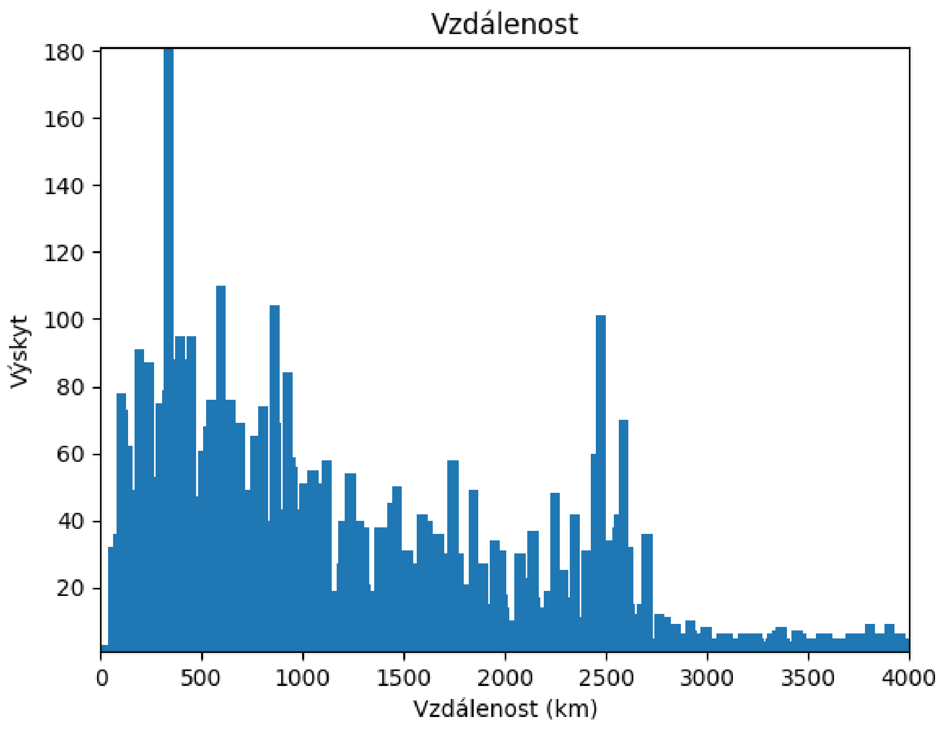
Třetí graf - nejčastější cesty jsou do 500 až 1000 kilometrů letu.

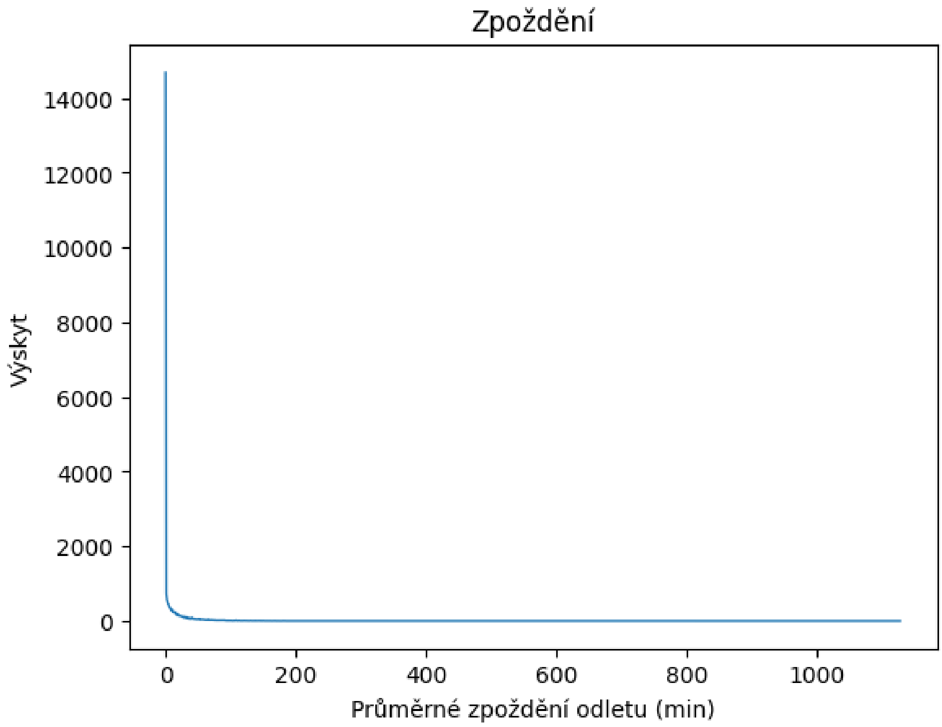
Čtvrtý graf ukazuje, že existuje vysoká pravděpodobnost že letadlo bude mít zpoždění alespoň 5 minut.
Pátý ukazuje, že se nenašel nikdo ze zkoumaných, který by nebyl spokojen s čistotou v

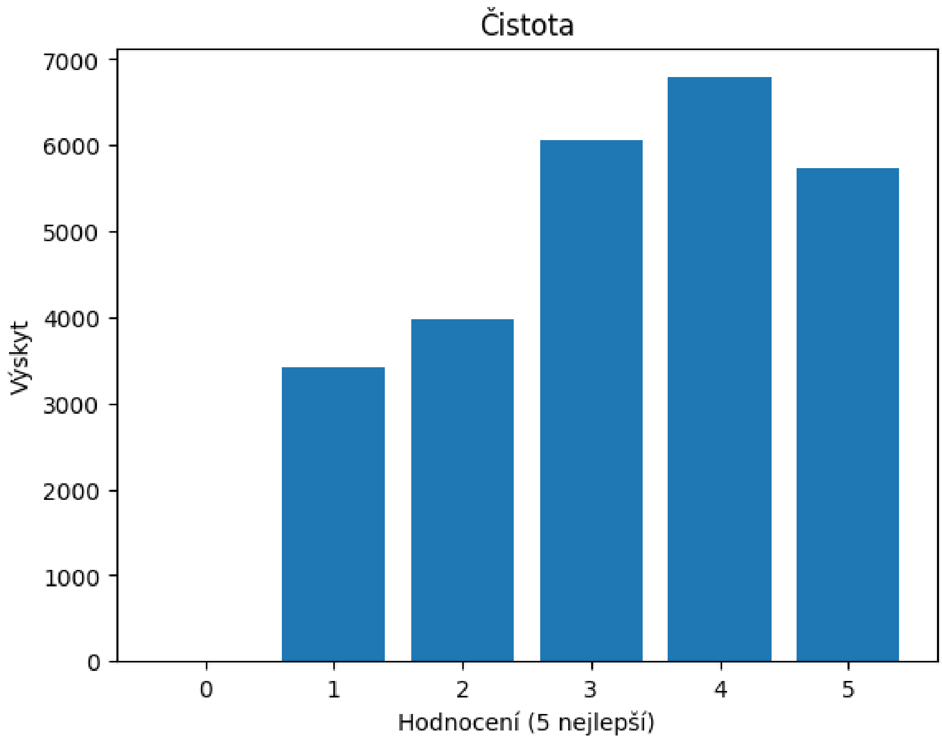


prostorách letadla.


Typ cesty
Business travel
Personal Travel


Vzdálenost
180
160
140
120
100
80
60
40
20
Výskyt
0
500
1000
1500
2000
2500
3000
3500
4000
Vzdálenost (km)


Zpoždění
Výskyt
14000
12000
10000
8000
6000
4000
2000
0
0
200
400
600
800
1000
Průměrné zpoždění odletu (min)


Čistota
Výskyt
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
Hodnocení (5 nejlepší)

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:
doplňte

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:

doplňte

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:

```
import numpy as np
def polynomial_func(x):
    return x**3 + 4*x**2 - 8
def exponential_func(x):
    return np.exp(x) - 2
def harmonic_func(x):
    return np.sin(x) + 0.5*x

from scipy.optimize import bisect, newton
def find_root_bisection(func, a, b):
    root = bisect(func, a, b)
    return root
def find_root_newton(func, x0):
    root = newton(func, x0)
    return root

# Interval pro polynom
poly_interval = (-5, 5)
# Interval pro exponenciální funkci
exp_interval = (0, 10)
# Interval pro harmonickou funkci
harmonic_interval = (-5, 5)
# Počáteční hodnota pro Newtonovu-Raphsonovu metodu
newton_x0 = 1.5
```

```
Výsledek: 1.2360679774997898
Kořen polynomu (bisekční metoda): 1.236067977500852 čas: 0.021996259689331055s
odchylka 1.0622613899613498e-12
Kořen polynomu (Newtonova metoda): 1.2360679774997898 čas:
0.18699193000793457s odchylka 0.0
```

```
Výsledek: 0.6931471805599453
Kořen exponenciální funkce (bisekční metoda): 0.6931471805603451 čas:
0.05699753761291504s odchylka 3.9979131116751887e-13
Kořen exponenciální funkce (Newtonova metoda): 0.693147180559946 čas:
0.24799609184265137s odchylka 6.661338147750939e-16

Výsledek: 0
Kořen harmonické funkce (bisekční metoda): 0.0 čas: 0.005000114440917969s
odchylka 0.0
Kořen harmonické funkce (Newtonova metoda): -4.543824632034784e-28 čas:
0.20899057388305664s odchylka 4.543824632034784e-28
```

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:

```
seed = 0
pos = []

print('Pro generování hýbejte myší')
i = 0
while len(pos) < 15:
    x, y = win32api.GetCursorPos()
    if (x,y) in pos:
        continue
    pos.append((x, y))
    i += 1
    print(f"{i}/15")
    time.sleep(0.25)
for i in pos:
    seed += np.math.factorial(sum(i) - os.cpu_count()) - abs(int(time.time()))
seed = seed // int(time.time())
seed = seed * sys.getsizeof(seed)
print(f'Seed: {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:
```
def forward_derivate(f, x0, h):
    return (f(x0+h) - f(x0))/h
def backward_derivate(f, x0, h):
    return (f(x0) - f(x0-h))/h
def central_derivate(f, x0, h):
    return (f(x0+h) - f(x0-h))/(2*h)
def adaptive_derivative(f, x0, h, threshold=1e-6):
    # Základní aproximace
    derivative = (f(x0 + h) - f(x0)) / h

    while True:
        # Snížení kroku
        h /= 2
        # Spočítá znovu
        new_derivative = (f(x0 + h) - f(x0)) / h
        # Zkontrolovat jestli rozdíl je hodnotou pod tresholdem
        if abs(new_derivative - derivative) < threshold:
            break
        derivative = new_derivative
    return derivative
```

```
Dopředná derivace - bez adaptabilního kroku: 4.100000000000001 s adaptabilním krokem 4.000001525855623
Zpětná derivace - bez adaptabilního kroku: 3.9000000000000057 s adaptabilním krokem 4.000001525855623
Centrální derivace - bez adaptabilního kroku: 4.0000000000000036 s adaptabilním krokem 4.000001525855623
```

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:
doplňte

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