Matematika pro nematematiky - Úlohy 6

Termín zadání: 01.11.2024

1 Řešení algebraických rovnic numerickými metodami

Aplikujte programy, které jste zčásti vytvořili během semináře (NumerickaMatematika1.ipynb) a vyřešte následující rovnice pomocí všech 3 metod, o nichž jsme hovořili:

A) Metoda půlení intervalů
B) Metoda prosté iterace
C) Newtonova metoda tečen

Všechny 3 metody jsou též popsány ve skriptech.
Zkuste zvolit různé počáteční hodnoty x_0 a pozorujte chování programu. Stane se, že řešení někdy diverguje (t.j. nekonverguje)? Závisí rychlost konvergence, tedy počet iterací, podstatně na volbě x_0?

1. $\sin x = x$
2. $\ln x = -x^2 + 3$
3. $x^3 + 3x = 2$

Řešení: Metodou tečen mohou být rovnice vyřešeny např. takto:

```
# metoda tečen

import numpy as np

def g(x): # definice funkce
        return x - np.sin(x)
        # np.log(x) + x**2 -3
        # x**3 + 3*x - 2

def gdev(x): # definice derivace funkce
        return 1 - np.cos(x)
        # 1/x + 2*x
        # 3*x**2 + 3

iterace = 0
x0 = 1  # zvolená počáteční hodnota
x1 = x0-g(x0)/gdev(x0)
epsilon = 0.001

while abs(x0-x1) > epsilon:
        x0 = x1
```

```
        x1 = x0-g(x0)/gdev(x0)
        iterace += 1

print(x0)
print(x1)
print(iterace)
```

2 Numerická integrace

Aplikujte programy, které jste zčásti vytvořili během semináře (NumerickaMatematika1.ipynb) a vyřešte následující integrály pomocí metod, o nichž jsme hovořili:

A) Součet centrovaných obdélníků
B) Simsonovo pravidlo
C) Monte Carlo integrace

Integrály též vyřešte analyticky a porovnejte přesnost řešení.

1. $\int_0^{\pi/2} \sin x \; dx$
2. $\int_0^2 x^3 \; dx$

Řešení: Aplikací Simpsonova pravidla může být 1. integrál vypočten např. takto:

```
# simpsonovo pravidlo
import numpy as np

def g(x):
        return np.sin(x)

def simpson(a,b):   # aproximace polynomem 2. řádu
        return (b-a)*(g(a)+4*g((a+b)/2)+g(b))/6

a = 0
b = np.pi/2
dx = 0.01

x = a
S = 0

while x<b:
        S += simpson(x,x+dx)
        x += dx

print(S)
```

Při zadaném kroku $dx = 0.01$ je výsledkem $S = 1.009203$, exaktní hodnota $S = 1$.

3 Určení hodnoty π pomocí metody Monte Carlo

V Pythonu napiště program obdobný metodě Monte Carlo integrace, který spočítá plochu kruhu o poloměru 1 cm.

Řešení:

Kruh o poloměru 1 cm umístíme do čtverce o délce strany 2 cm. Náhodně generujeme body ve čtverci (se souřadnicemi x a y) a zjišťujeme, kolik z nich leží v kruhu ($x^2 + z^2 \leq 1$).

```
# výpočet obsahu kruhu pomocí Monte Carlo simulace

import numpy as np
import random # modul pro generování náhodných čísel

def g(x,y):   # vzdálenost od středu
        return x**2 + y**2

S = 0
N = 100000 # počet náhodných simulací

# Náhodné číslo
for i in range(N):
        x = random.uniform(-1, 1)  # náhodné x mezi -1 a 1
        y = random.uniform(-1, 1)  # náhodné y mezi -1 a 1
        if g(x,y) <= 1: # bod leží v kruhu
                S +=1   # sčítá případy, kdy bod padne do kruhu

S = 2*2*S/N

print(S)
```

4 Numerické derivování

V souboru NumerickaMatematika1.ipynb je uveden program pro numerickou derivaci funkce sinus pomocí metody dopředných diferencí. Narozdíl o předchozích úloh nejsou hodnoty derivace počítány postupně ve for cyklu nebo while cyklu, nýbrž "najednou"jako rozdíl vektorů numpy. Výsledek derivace společně s funkcemi sinus a cosinus jsou nakresleny v grafu.

Zkuste program přeformulovat pro derivaci metodou zpětných diferencí a metodou centrálních diferencí. ChatGPT vám pomůže.

Řešení:

Řešení může být následující:

```
# numerické derivováni

import numpy as np
import matplotlib.pyplot as plt

def f(x):
        return np.sin(x)

x = np.linspace(0,5,50)
y = f(x)
z = np.cos(x)

h = x[1] - x[0]
```

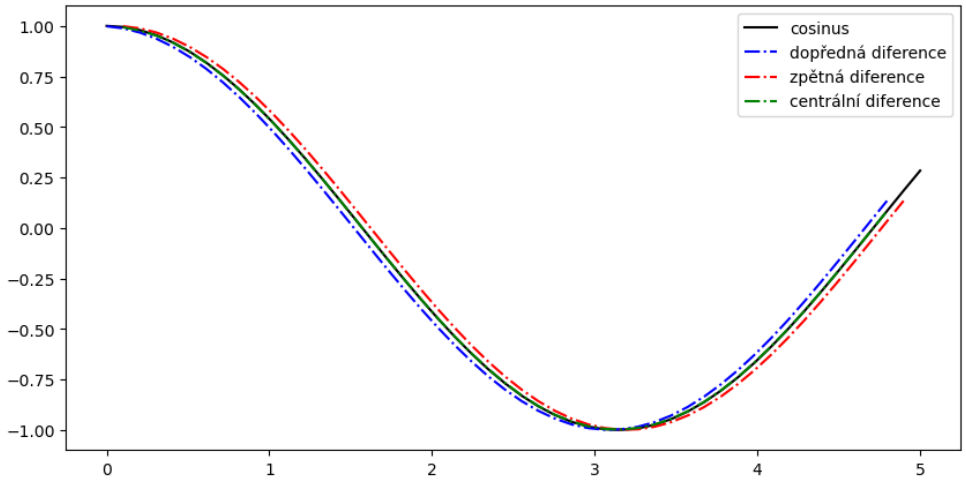
```
derivace      = (y[1:49]-y[0:48])/h
derivaceCentr = (y[2:49]-y[0:47])/(2*h)

fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(10, 10))

ax.plot(x, z, color = "black", label = 'cosinus')
ax.plot(x[0:48], derivace, 'b', label =
                    'dopředná diference', linestyle='-.')
ax.plot(x[1:49], derivace, 'r', label =
                    'zpětná diference', linestyle='-.')
ax.plot(x[1:48], derivaceCentr, 'g', label =
            'centrální diference', linestyle='-.')

ax.legend()
```

Je vidět, že metoda centrálních diferencí poskytuje téměř přesnou shodu s analytickým řešením (obr. 1).



Obrázek 1: Metody numerické derivace