Matematika pro nematematiky - Úlohy 5

Termín zadání: 23.10.2024

1 Taylorův rozvoj logaritmu

Odvoďte Taylorův rozvoj pro funkci $f(x) = \ln x$ v okolí bodu $x_0 = 1$ a pomocí Pythonu nakreslete do společného grafu jak funkci $f(x)$, tak několik prvních polynomů, obdobně jak bylo ukázáno v přednášce.

Řešení:

Jednotlivé derivace pro $f(x) = \ln x$ jsou $f'(x) = x^{-1}$, $f''(x) = -x^{-2}$, $f^{(3)}(x) = 2x^{-3}$, $f^{(4)}(x) = -2.3.x^{-4}$, $f^{(n)}(x) = \pm(n-1)!x^{-n}$. Pro $x_0 = 1$ jsou tedy derivace (včetně 0. derivace) postupně rovny 1,-1,2,-6....,$\pm(n-1)!$. $f(1) = \ln 1 = 0$. Po dosazení dostaneme

$$\begin{aligned}\ln x &\approx (x-1) - \frac{(x-1)^2}{2!} + \frac{(x-1)^3}{2!} - \frac{(x-1)^4}{2!} + \ldots \pm \frac{(n-1)!(x-1)^n}{n!} \\ &= (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \frac{(x-1)^4}{4} + \ldots \pm \frac{(x-1)^n}{n}\end{aligned}$$

```
import numpy as np
import matplotlib.pyplot as plt

o = np.linspace(0.2,4,200)

def f(o):
        return np.log(o)

def f1(o):
        return o-1

def f2(o):
        return o-1 - (o-1)**2/2

def f3(o):
        return o-1 - (o-1)**2/2 + (o-1)**3/3

def f4(o):
        return o-1 - (o-1)**2/2 + (o-1)**3/3 - (o-1)**4/4


fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(10, 10))

ax[0,0].plot(o, f(o), 'b', label = 'ln x')
```

```
ax[0,0].plot(o, f1(o), 'r', label = 'polynom 1',linestyle='-.')

ax[0,1].plot(o, f(o), 'b', label = 'ln x')
ax[0,1].plot(o, f2(o), 'r', label = 'polynom 2',linestyle='-.')

ax[1,0].plot(o, f(o), 'b', label = 'ln x')
ax[1,0].plot(o, f3(o), 'r', label = 'polynom 3',linestyle='-.')

ax[1,1].plot(o, f(o), 'b', label = 'ln x')
ax[1,1].plot(o, f4(o), 'r', label = 'polynom 4',linestyle='-.')


ax[0,0].set_xlabel('x')
ax[1,0].set_ylabel('f(x)')

ax[0,0].set_ylim([-1.5,1.5])
ax[0,1].set_ylim([-1.5,1.5])
ax[1,0].set_ylim([-1.5,1.5])
ax[1,1].set_ylim([-1.5,1.5])

ax[0,0].hlines(y=0, xmin=0, xmax=4, color='g', linestyle='--')
ax[0,1].hlines(y=0, xmin=0, xmax=4, color='g', linestyle='--')
ax[1,0].hlines(y=0, xmin=0, xmax=4, color='g', linestyle='--')
ax[1,1].hlines(y=0, xmin=0, xmax=4, color='g', linestyle='--')

ax[0,0].vlines(x=1, ymin=-1.5, ymax=1.5, color='g', linestyle='--')
ax[1,0].vlines(x=1, ymin=-1.5, ymax=1.5, color='g', linestyle='--')
ax[0,1].vlines(x=1, ymin=-1.5, ymax=1.5, color='g', linestyle='--')
ax[1,1].vlines(x=1, ymin=-1.5, ymax=1.5, color='g', linestyle='--')

ax[0,0].legend()
ax[0,1].legend()
ax[1,0].legend()
ax[1,1].legend()

plt.tight_layout()
```

Kvalitu náhrady funkce Taylorovým polynomem ukazuje obr. 1.

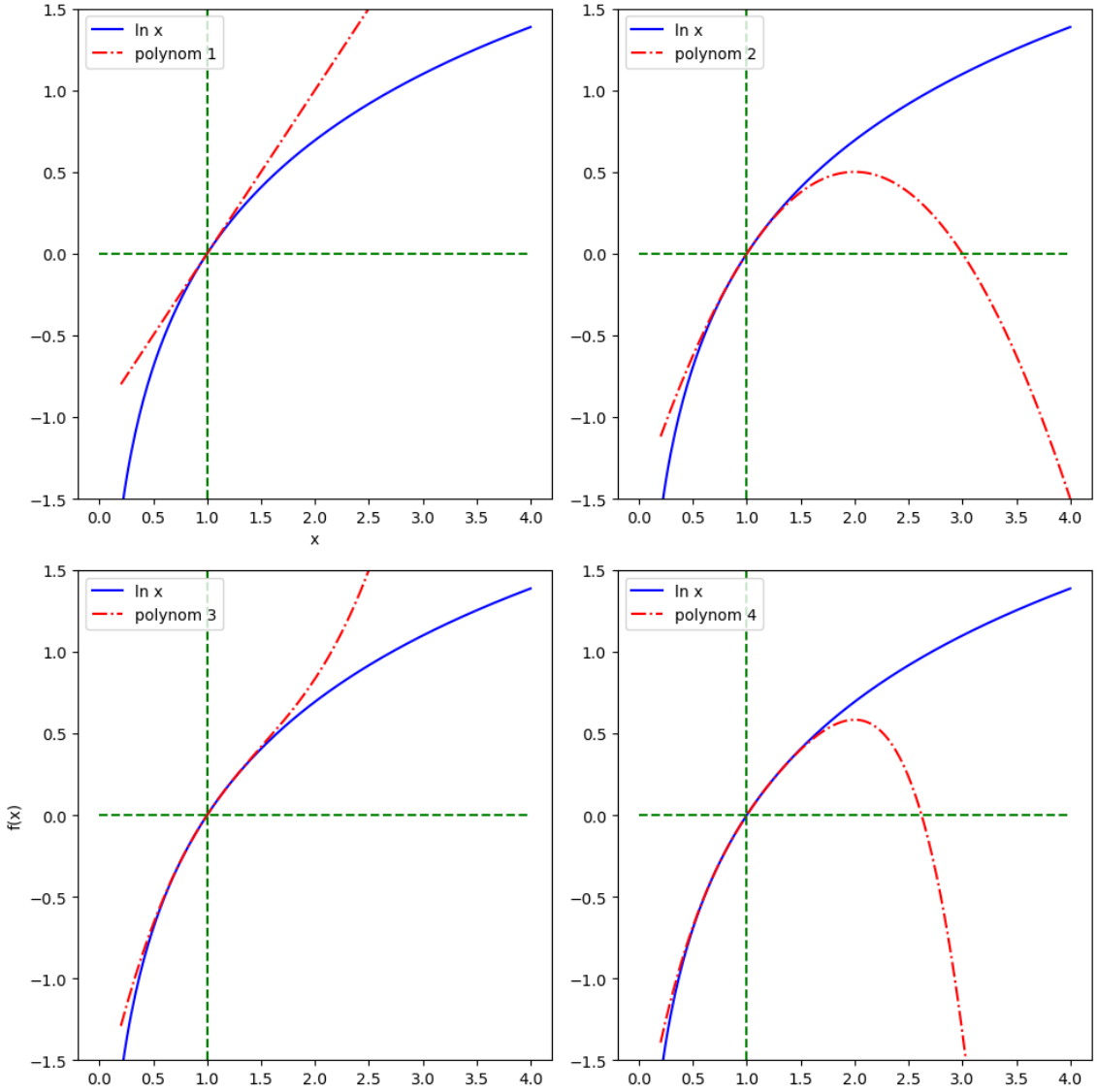
2 Taylorův rozvoj Gaussovy funkce

Odvoďte Taylorův rozvoj pro známou Gaussovu funkci $f(x) = e^{-x^2}$ v okolí bodu 0 a pomocí Pythonu nakreslete do společného grafu jak funkci $f(x)$, tak několik prvních polynomů, obdobně jak bylo ukázáno v přednášce.

Řešení:

Jednotlivé derivace pro $f(x) = e^{-x^2}$ jsou $f'(x) = -2xe^{-x^2}$, $f''(x) = (4x^2 - 2)e^{-x^2}$, $f^{(3)}(x) = (-8x^3 + 12x)e^{-x^2}$, $f^{(4)}(x) = -(16x^4 - 48x^2 + 12)e^{-x^2}$... $f^{(n)}(x) = \pm\frac{x^n}{(n/2)!}$.

Každý lichý člen je nulový. Po dosazení dostaneme



Obrázek 1: Taylorův rozvoj funkce $\ln x$

$$e^{-x^2} \approx 1 - x^2 + \frac{x^4}{2} - \frac{x^6}{6} + \frac{x^8}{24} + \dots$$

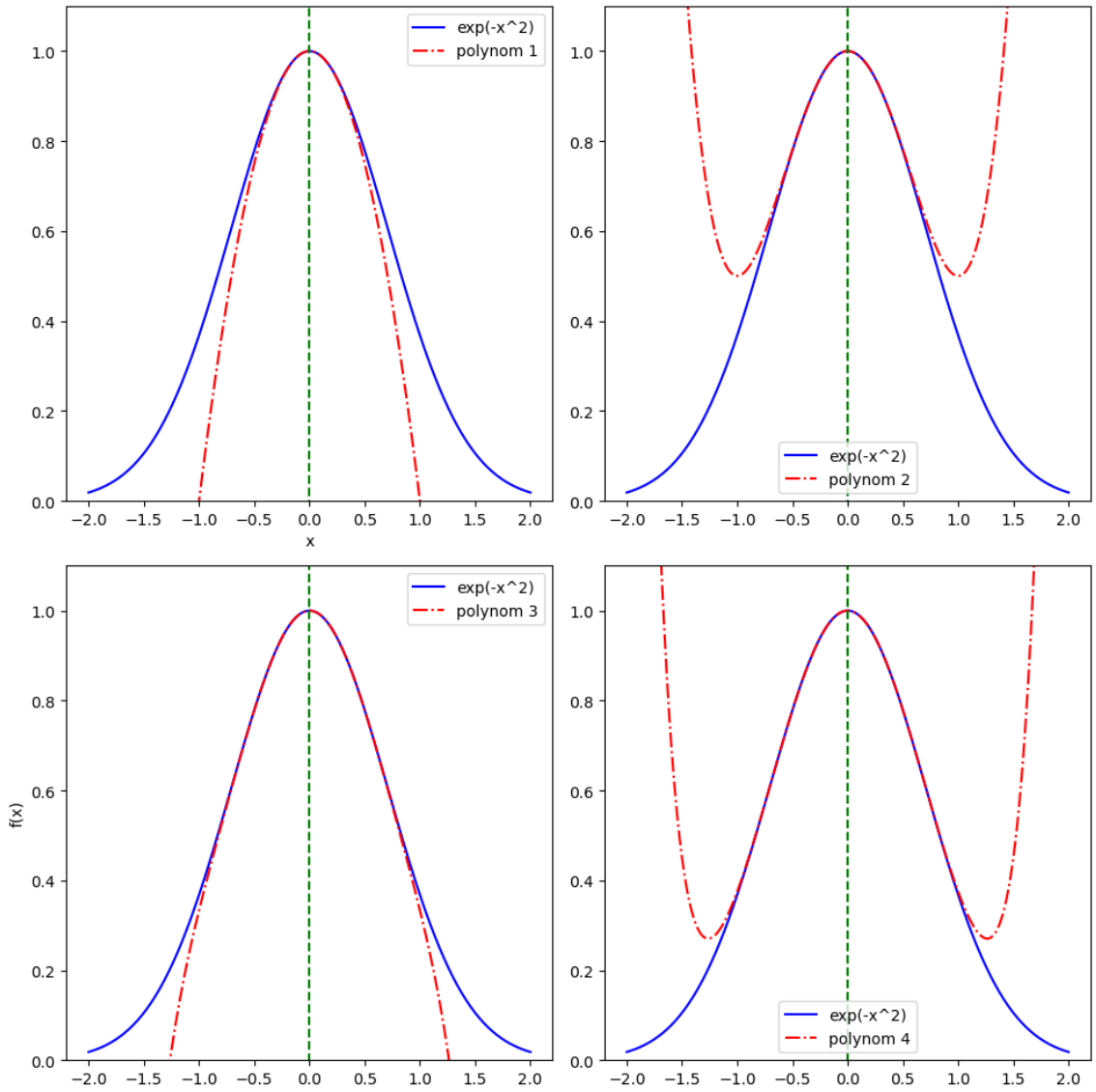
```
import numpy as np
import matplotlib.pyplot as plt

o = np.linspace(-2,2,200)

def f(o):
    return np.exp(-o**2)

def f1(o):
    return 1 - o**2

def f2(o):
```

```
    return 1- o**2 + o**4/2

def f3(o):
        return 1- o**2 + o**4/2 - o**6/6

def f4(o):
        return 1- o**2 + o**4/2 - o**6/6 + o**8/24


fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(10, 10))

ax[0,0].plot(o, f(o), 'b', label = 'exp(-x^2)')
ax[0,0].plot(o, f1(o), 'r', label = 'polynom 1',linestyle='-.')

ax[0,1].plot(o, f(o), 'b', label = 'exp(-x^2)')
ax[0,1].plot(o, f2(o), 'r', label = 'polynom 2',linestyle='-.')

ax[1,0].plot(o, f(o), 'b', label = 'exp(-x^2)')
ax[1,0].plot(o, f3(o), 'r', label = 'polynom 3',linestyle='-.')

ax[1,1].plot(o, f(o), 'b', label = 'exp(-x^2)')
ax[1,1].plot(o, f4(o), 'r', label = 'polynom 4',linestyle='-.')


ax[0,0].set_xlabel('x')
ax[1,0].set_ylabel('f(x)')

ax[0,0].set_ylim([0,1.1])
ax[0,1].set_ylim([0,1.1])
ax[1,0].set_ylim([0,1.1])
ax[1,1].set_ylim([0,1.1])


ax[0,0].vlines(x=0, ymin=-1.5, ymax=1.5, color='g', linestyle='--')
ax[1,0].vlines(x=0, ymin=-1.5, ymax=1.5, color='g', linestyle='--')
ax[0,1].vlines(x=0, ymin=-1.5, ymax=1.5, color='g', linestyle='--')
ax[1,1].vlines(x=0, ymin=-1.5, ymax=1.5, color='g', linestyle='--')

ax[0,0].legend()
ax[0,1].legend()
ax[1,0].legend()
ax[1,1].legend()

plt.tight_layout()
```

Kvalitu náhrady funkce Taylorovým polynomem ukazuje obr. 2.

3 Hodnota Eulerova čísla e

Napiště v Pythonu program, který pomocí Taylorova rozvoje určí a vypíše hodnotu Eulerova čísla e s přesností na 100 desetinných míst.



Obrázek 2: Taylorův rozvoj funkce e^{-x^2}

Řešení:

Jde o obtížnou úlohu, kderá musí vyřešit 2 technické problémy - jednak počítač ukládá číslo s limitovanou přesností, jednak opakované součty způsobují drobné chyby výpočtu.

ChatGPT uvádí e na 100 desetinných míst takto:

2.718281828459045

2353602874713526624977572470936999595749669676277240766303535475945713821785251

Následující jednoduchý program dává výsledek s přesný jen na 15 desetinných míst

2.718281828459045

534884808148490265011787414550781250000000000000000000000000000000000000000000

```
# 1+1+1/2+1/6+1/24....

delenec = 1
soucet = 1
for i in range(1,1000):
    delenec = delenec * i
```

```
    soucet += 1/delenec

print(f"{soucet:.100f}" )
```

ChatGPT navrhuje následující program:

```
from decimal import Decimal, getcontext
import math

# Nastavíme požadovanou přesnost
getcontext().prec = 110  # přesnost 110 míst, abychom zajistili 100 desetinných

# Inicializace hodnot
euler_number = Decimal(0)
factorial = Decimal(1)

# Počítáme hodnotu e s Taylorovým rozvojem
for n in range(1000):  # Vyšší počet iterací pro dosažení požadované přesnosti
        if n > 0:
        factorial *= n  # n!
        euler_number += Decimal(1) / factorial  # Přidáváme další člen do sumy

# Výsledek zaokrouhlený na 100 desetinných míst
print(f"Hodnota Eulerova čísla s přesností na 100 desetinných míst je: {euler_nu
```

Výsledek je:
2.7182818284590452353602874713526624977572470936
999595749669676277240766303535475945713821785251664274

4 Derivace Taylorova rozvoje

Zderivujte Taylorův rozvoj funkce sinus a ukažte, že výsledkem je funkce cosinus.

Řešení:

$$(\sin x)' = \left(x - \frac{x^3}{6} + \frac{x^5}{5!} + \dots\right)' = 1 - \frac{x^2}{2} + \frac{x^4}{4!} + \dots = \cos x$$

5 Limita funkce s Taylorovým rozvojem

Využijte Taylorův rozvoj funkce $\sin x$ a dokažte, že platí

$$\lim_{x \to 0} \frac{\sin x}{x} = 0$$

Nakreslete též funkci $\frac{\sin x}{x}$ pomocí matplotlib a ověřte si tak svůj výsledek.

Řešení:
Limitu nemůžeme spočítat přímo dosazením, protože tak získáme neurčitý výraz $0/0$.

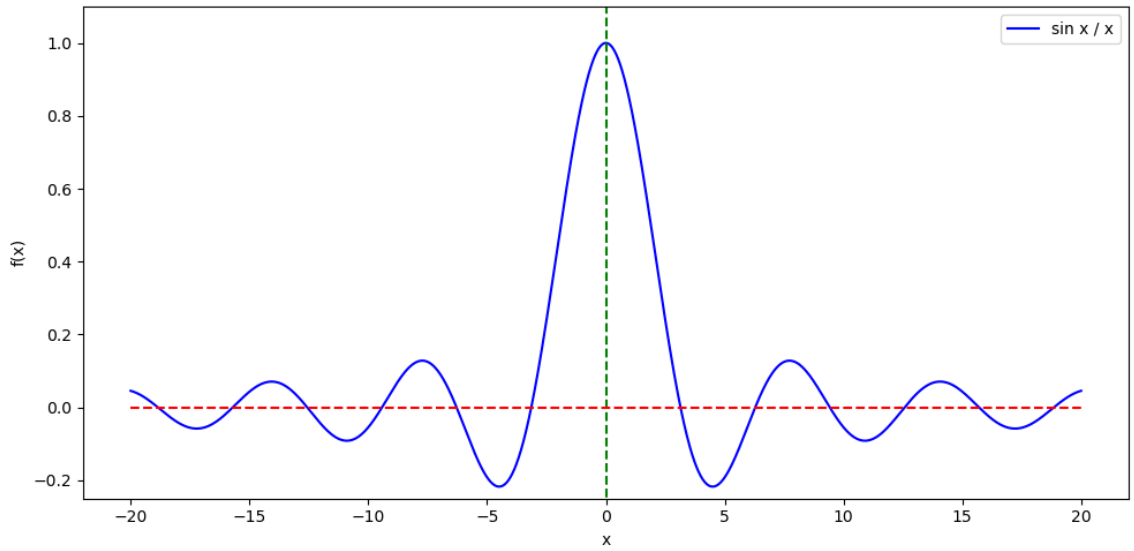
$$\lim_{x \to 0} \frac{\sin x}{x} = \frac{\sin 0}{0} = \frac{0}{0} = ??$$

Protože nás však zajímá chování v okolí bodu 0, může nám pomoci Taylorův rozvoj funkce $\sin x$ v okolí 0.

$$\lim_{x\to 0}\frac{\sin x}{x}=\lim_{x\to 0}\frac{x-\frac{x^3}{6}+\frac{x^5}{5!}+\dots}{x}=\lim_{x\to 0}\left(1-\frac{x^2}{6}+\frac{x^4}{5!}+\dots\right)=1$$

```
import numpy as np
import matplotlib.pyplot as plt

o = np.linspace(-20,20,2000)

def f(o):
        return np.sin(o)/o

fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(10, 5))

ax.plot(o, f(o), 'b', label = 'sin x / x')

ax.set_xlabel('x')
ax.set_ylabel('f(x)')

ax.set_ylim([-0.25,1.1])
ax.vlines(x=0, ymin=-1.5, ymax=1.5, color='g', linestyle='--')
ax.hlines(y=0, xmin=-20, xmax=20, color='r', linestyle='--')

ax.legend()
plt.tight_layout()
```

Funkce je patrna na obrázku 3.



Obrázek 3: Funkce $\frac{\sin x}{x}$