oom_3_1_2_bisection

29 грудня 2022 р.

0.1 # 3. Розв'язування нелінійних рівнянь

Нехай маємо рівняння

$$f(x) = 0, \quad x \in [a, b], \tag{1}$$

де $f : [a, b] \to \mathbb{R}$ – неперервна на $[a, b]$ функція.

0.2 ### 3.1.2. Метод бісекції

Нехай рівняння (1) має і тільки один корінь на відрізку $[a, b]$, тобто маємо

$$f(a)f(b) < 0. \tag{2}$$

Ідея методу бісекції полягає у обчисленні послідовності $x_0, x_1, \ldots, x_n, \ldots$ за формулою $x_n = (a + b)/2$ шляхом покрокового звуження проміжку (a, b) за рахунок ділення його навпіл. На кожному кроці за новий проміжок (a, b) беруть ту частину попереднього проміжку, де локалізований корінь, тобто виконується умова (2). Поділ виконують до тих пір, коли виконається умова

$$|b - a| < eps, \tag{3}$$

де eps – задана величина (точність).

Останнє з обчислених значень x_n є наближенням кореня розв'язку рівняння з точністю eps.

Якщо для чергового значення x_n виявится, що $f(x_n) = 0$, то це значення беруть за розв'язок рівняння.

0.3 #### Пояснення до використання програмного коду

- Підготувати середовище і потрібні функції :
  1. виконати комірку для підготовки середовища
  2. виконати комірку, де визначена функція `bisection`
  3. виконати комірку, де визначена функція `plot_graphics`
  4. виконати комірку, де визначена функція `f`
- Локалізувати (графічно) корінь рівняння (1)
  1. виконати комірку, в якій задається відрізок
  2. виконати комірку, в якій будується графік
  3. виконати комірку, в якій задається звужений відрізок
  4. виконати комірку, в якій будується графік на звуженому відрізку
  5. пункти 3 і 4 можна повтороти для точнішої локалізації потрібного кореня
- Обчислити наближення локалізованого кореня

1. задати точність `eps` наближеного кореня
2. виконати комірку, де є виклик функції `bisection`
3. для знаходження іншого корення виконати дії пунктів локалізації

[1]:
```
# Підготовка середовища

# при виконанні в JupyterLab наступний рядок розкоментувати
#%matplotlib widget

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

[12]:
```
def bisection(f,a,b,eps):
    """ знаходження методом бісекції наближеного кореня рівняння (1),
        де f -- неперервна на відрізку [a,b] функція,
        на цьому відрізку рівняння має і тільки один корінь,
        eps -- задана точність
    """
    ba=np.abs(b-a)
    if ba < eps:
        return (a+b)/2
    while ba > eps:
        fa=f(a)
        x=(a+b)/2
        fx=f(x)
        if fx==0 :
            return x
        if fa*fx < 0:
            b=x
        else:
            a=x
        ba=np.abs(b-a)
    return (a+b)/2
```

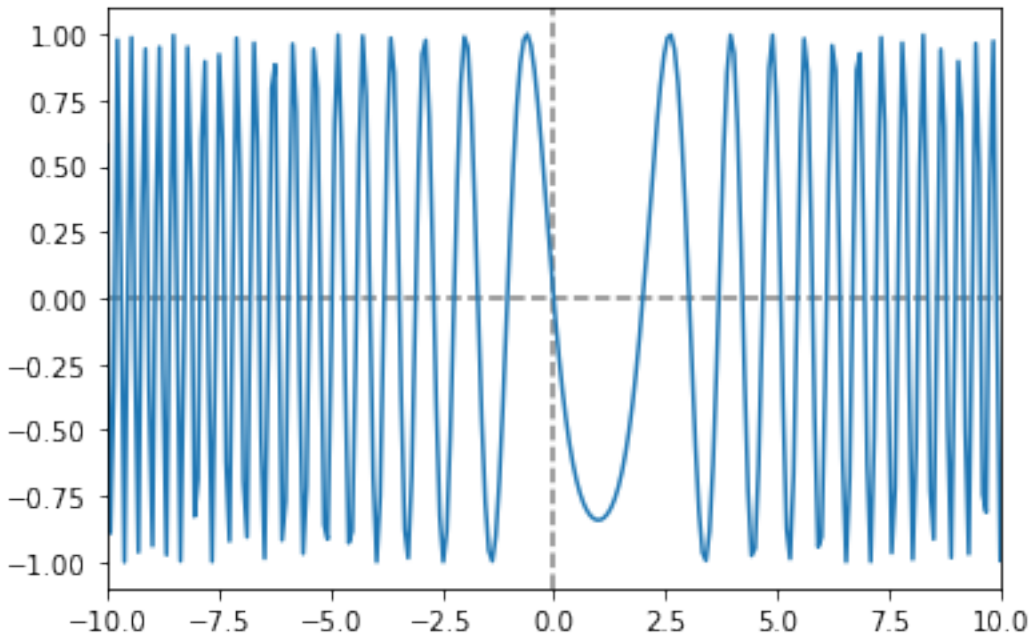
[3]:
```
def plot_graphics(f, a, b, n):
    """фуункція для побудови графіка функції f
       на відрізку [a,b] за значеннями в n точках
    """
    xarr = np.linspace(a, b, n)
    y=f(xarr)
    fig = plt.figure()
    ax = fig.gca()
```

```
    ax.plot(xarr,y)
    ax.axhline(color="grey", ls="--", zorder=-1)
    ax.axvline(color="grey", ls="--", zorder=-1)
    ax.set_xlim(a,b)
    plt.show()
```

[4]:
```
def f(x):
    """функція лівої частини рівняння (1)"""
    return np.sin(x*x-2*x)
```

[5]:
```
# задання відрізка
a=-10
b=10
```

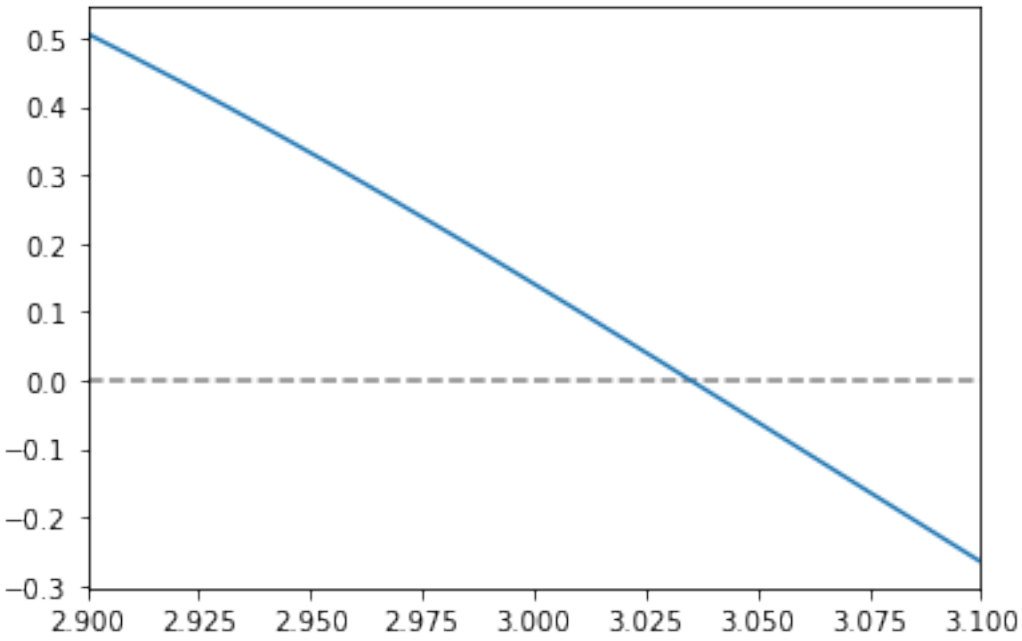
[6]:
```
plot_graphics(f, a, b, 256)
```



[7]:
```
plt.close('all')
```

[8]:
```
# задання відрізка
a=2.9
b=3.1
```

[9]:
```
plot_graphics(f, a, b, 256)
```



[10]:
```
plt.close('all')
```

[11]:
```
#знаходження наближення кореня
eps=0.001
x=bisection(f, a, b, eps)
print(f"корінь рівняння x={x} з точністю eps={eps}")
```

```
корінь рівняння x=3.034765625 з точністю eps=0.001
```

[ ]:
```

```