# ЛАБОРАТОРНА РОБОТА №3

**Тема:** «Чисельні методи розв’язання нелінійних рівнянь. Уточнення кореня нелінійного рівняння»

Трубчанінов Андрій Сергійович

ФІТ 2-8

В-29

Код:

pr1.py pr2.py pr3.py

pr3.py > ...

```python
import numpy as np
from scipy.misc import derivative


def f(x):
    return 2 * pow(x, 4) + 4 * pow(x, 3) - pow(x, 2) - 3 * x - 1


def newton(a, b, eps):
    df2 = derivative(f, b, n=2)

    if f(b) * df2 > 0:
        xi = b
    else:
        xi = a

    df = derivative(f, xi, n=1)
    xi_1 = xi - f(xi) / df

    while abs(xi_1 - xi) > eps:
        xi = xi_1
        xi_1 = xi - f(xi) / df

    return print("Newton's method x=", xi_1)


def combined(a, b, eps):
    if derivative(f, a, n=1) * derivative(f, a, n=2) > 0:
        a0 = a
        b0 = b
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ

```
.0; and will be completely removed in SciPy v1.12.0. You may consider using findiff: https://github.com/maroba/findiff o
r numdifftools: https://github.com/pbrod/numdifftools
```

Python
Python

Строка 51, столбец 26 Пробелов: 4 UTF-8 CRLF Python 3.11.5 64-bit Go Live

pr1.py pr2.py pr3.py

pr3.py > ...

```python
    return print("Newton's method x=", xi_1)


def combined(a, b, eps):
    if derivative(f, a, n=1) * derivative(f, a, n=2) > 0:
        a0 = a
        b0 = b
    else:
        a0 = b
        b0 = a

    ai = a0
    bi = b0

    while abs(ai - bi) > eps:
        ai_1 = ai - f(ai) * (bi - ai) / (f(bi) - f(ai))
        bi_1 = bi - f(bi) / derivative(f, bi, n=1)
        ai = ai_1
        bi = bi_1

    x = (ai_1 + bi_1) / 2

    return print("combined method x=", x)


# Викликати функції зі своїми параметрами
newton(-2, -1 / 2, 0.001)
combined(-2, -1 / 2, 0.001)
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ

pr1.py pr2.py pr3.py

pr3.py > ...

```python
import numpy as np
from scipy.misc import derivative


def f(x):
    return 2 * pow(x, 4) + 4 * pow(x, 3) - pow(x, 2) - 3 * x - 1


def newton(a, b, eps):
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ

```
PS C:\Users\38066\Desktop\chiselni-metody> & C:/Python311/python.exe c:/Users/38066/Desktop/chiselni-metody/pr3.py
c:\Users\38066\Desktop\chiselni-metody\pr3.py:10: DeprecationWarning: scipy.misc.derivative is deprecated in SciPy v1.10
.0; and will be completely removed in SciPy v1.12.0. You may consider using findiff: https://github.com/maroba/findiff o
r numdifftools: https://github.com/pbrod/numdifftools
  df2 = derivative(f, b, n=2)
c:\Users\38066\Desktop\chiselni-metody\pr3.py:17: DeprecationWarning: scipy.misc.derivative is deprecated in SciPy v1.10
.0; and will be completely removed in SciPy v1.12.0. You may consider using findiff: https://github.com/maroba/findiff o
r numdifftools: https://github.com/pbrod/numdifftools
  df = derivative(f, xi, n=1)
c:\Users\38066\Desktop\chiselni-metody\pr3.py:18: RuntimeWarning: divide by zero encountered in scalar divide
  xi_1 = xi - f(xi) / df
c:\Users\38066\Desktop\chiselni-metody\pr3.py:6: RuntimeWarning: invalid value encountered in scalar subtract
  return 2 * pow(x, 4) + 4 * pow(x, 3) - pow(x, 2) - 3 * x - 1
Newton's method x= nan
c:\Users\38066\Desktop\chiselni-metody\pr3.py:28: DeprecationWarning: scipy.misc.derivative is deprecated in SciPy v1.10
.0; and will be completely removed in SciPy v1.12.0. You may consider using findiff: https://github.com/maroba/findiff o
r numdifftools: https://github.com/pbrod/numdifftools
  if derivative(f, a, n=1) * derivative(f, a, n=2) > 0:
c:\Users\38066\Desktop\chiselni-metody\pr3.py:40: DeprecationWarning: scipy.misc.derivative is deprecated in SciPy v1.10
.0; and will be completely removed in SciPy v1.12.0. You may consider using findiff: https://github.com/maroba/findiff o
r numdifftools: https://github.com/pbrod/numdifftools
  bi_1 = bi - f(bi) / derivative(f, bi, n=1)
combined method x= -1.9253309841128057
PS C:\Users\38066\Desktop\chiselni-metody>
```

Python
Python
Python
Python
Python

## Код зі скріншотів:

```python
import numpy as np
from scipy.misc import derivative


def f(x):
    return 2 * pow(x, 4) + 4 * pow(x, 3) - pow(x, 2) - 3 * x - 1
```

```python
def newton(a, b, eps):
    df2 = derivative(f, b, n=2)

    if f(b) * df2 > 0:
        xi = b
    else:
        xi = a

    df = derivative(f, xi, n=1)
    xi_1 = xi - f(xi) / df

    while abs(xi_1 - xi) > eps:
        xi = xi_1
        xi_1 = xi - f(xi) / df

    return print("Newton's method x=", xi_1)


def combined(a, b, eps):
    if derivative(f, a, n=1) * derivative(f, a, n=2) > 0:
        a0 = a
        b0 = b
    else:
        a0 = b
        b0 = a

    ai = a0
    bi = b0

    while abs(ai - bi) > eps:
        ai_1 = ai - f(ai) * (bi - ai) / (f(bi) - f(ai))
        bi_1 = bi - f(bi) / derivative(f, bi, n=1)
        ai = ai_1
        bi = bi_1

    x = (ai_1 + bi_1) / 2

    return print("combined method x=", x)


# Викликати функції зі своїми параметрами
newton(-2, -1 / 2, 0.001)
combined(-2, -1 / 2, 0.001)
```