Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра Автоматизованих Систем Обробки Інформації та Управління

Лабораторна робота №6

з дисципліни «Спеціальні розділи математики»

на тему

«Розв’язання нелінійних рівнянь»

Виконав:

студент гр. ІС-02

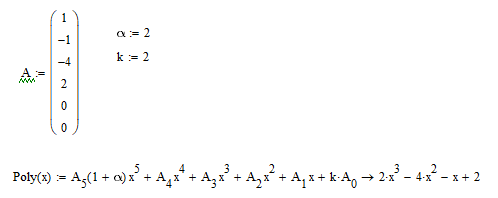
Плостак Ілля

Викладач:

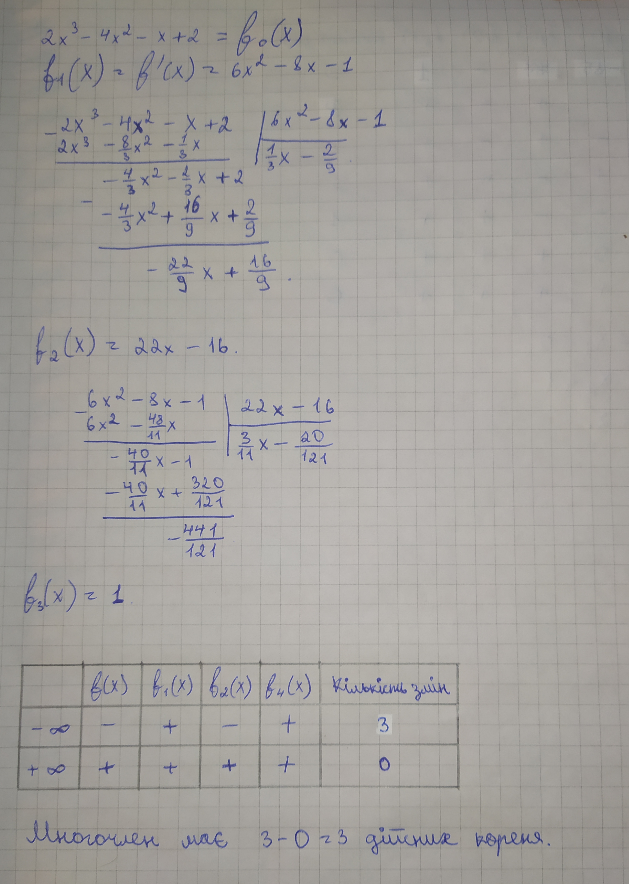
доц. Рибачук Л.В.

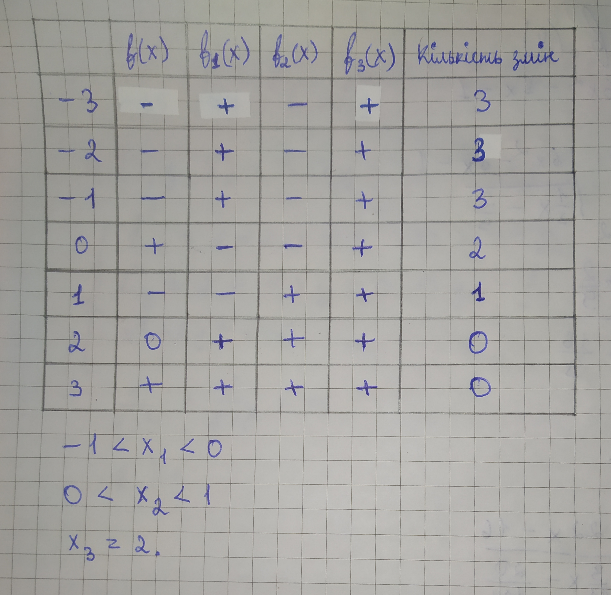
Київ – 2021

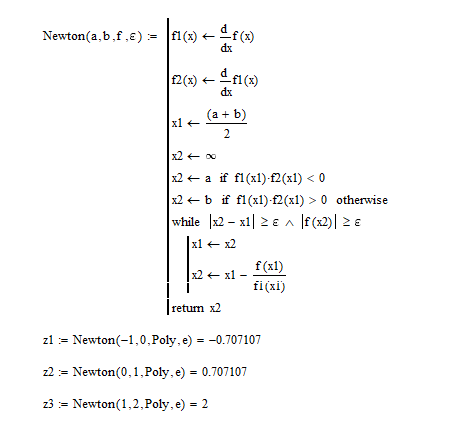
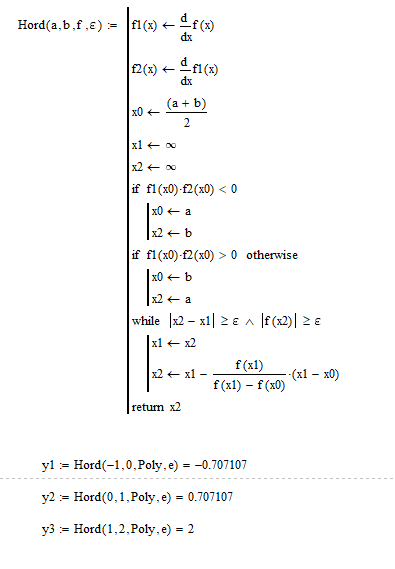
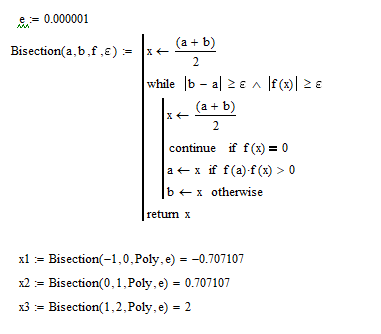
**1. Постановку задачі у вигляді вихідного рівняння**



**2. Виконання допрограмового етапу, результатом якого повинні бути проміжки, щодо яких проводиться уточнення:**





**3. Розв’язок уточнення коренів за методами бісекції, хорд, дотичних у Mathcad:**

**4. Висновки:**

В цій лабораторній роботі ми порівняли три методи уточнення коренів поліноміального рівняння: метод бісекцій, метод хорд та метод дотичних (метод Ньютона). Найдовшим виявився метод бісекцій, який дає ще й найгірший результат. На другому місці опинився метод хорд, який виконується швидше та дає більш точний результат за однієї і тієї ж точності ε. Найшвидшим з них виявився метод Ньютона (дотичних), який за тієї ж точності дає набагато точніший результат.

**5. Лістинг програми:**

[main.py](https://github.com/feedblackg44/kpilabs2/blob/master/CHM/Lab6/main.py):

from sympy import \*  
import functions as fc  
  
x = Symbol('x')  
  
f = 2\*x\*\*3-4\*x\*\*2-x+2  
e = 10\*\*-6  
  
print("Висхідна функція:\nf =", f)  
  
func = lambdify(x, f)  
  
temp1 = [0]  
temp2 = [0]  
  
bisec\_res = [fc.Bisection(-1, 0, func, e, temp1), fc.Bisection(0, 1, func, e, temp2), 2]  
bisek\_counter = [temp1[0], temp2[0], 0]  
  
hord\_res = [fc.Hord(-1, 0, f, e, temp1), fc.Hord(0, 1, f, e, temp2), 2]  
hord\_counter = [temp1[0], temp2[0], 0]  
  
newton\_res = [fc.Newton(-1, 0, f, e, temp1), fc.Newton(0, 1, f, e, temp2), 2]  
newton\_counter = [temp1[0], temp2[0], 0]  
  
print("\nКорені методу бісекцій:")  
for i in range(len(bisec\_res)):  
 print(bisec\_res[i], "\t\tКількість ітерацій: ", bisek\_counter[i], sep="")  
print("\nКорені методу хорд:")  
for i in range(len(hord\_res)):  
 print(hord\_res[i], "\t\tКількість ітерацій: ", hord\_counter[i], sep="")  
print("\nКорені методу Ньютона:")  
for i in range(len(newton\_res)):  
 print(newton\_res[i], "\t\tКількість ітерацій: ", newton\_counter[i], sep="")

[functions.py](https://github.com/feedblackg44/kpilabs2/blob/master/CHM/Lab6/functions.py):

from sympy import \*  
import math  
  
def Bisection(a, b, func, e, counter):  
 counter[0] = 0  
 x = (a + b) / 2  
 while abs(b - a) >= e and abs(func(x)) >= e:  
 counter[0] += 1  
 x = (a + b) / 2  
 if func(x) == 0:  
 pass  
 elif func(a)\*func(x) > 0:  
 a = x  
 else:  
 b = x  
 return x  
  
def Hord(a, b, func, e, counter):  
 x = Symbol('x')  
 counter[0] = 0  
  
 f1 = func.diff(x)  
 f2 = f1.diff(x)  
 f0 = lambdify(x, func)  
 f1 = lambdify(x, f1)  
 f2 = lambdify(x, f2)  
  
 x0 = (a + b) / 2  
 x1 = math.inf  
 x2 = math.inf  
  
 if f1(x0) \* f2(x0) < 0:  
 x0 = a  
 x2 = b  
 elif f1(x0) \* f2(x0) > 0:  
 x0 = b  
 x2 = a  
 while abs(x2 - x1) >= e and abs(f0(x2)) >= e:  
 counter[0] += 1  
 x1 = x2  
 x2 = x1 - f0(x1)/(f0(x1) - f0(x0)) \* (x1 - x0)  
 return x2  
  
def Newton(a, b, func, e, counter):  
 x = Symbol('x')  
 counter[0] = 0  
  
 f1 = func.diff(x)  
 f2 = f1.diff(x)  
 f0 = lambdify(x, func)  
 f1 = lambdify(x, f1)  
 f2 = lambdify(x, f2)  
  
 x1 = (a + b) / 2  
 x2 = math.inf  
  
 if f1(x1) \* f2(x1) < 0:  
 x2 = a  
 elif f1(x1) \* f2(x1) > 0:  
 x2 = b  
 while abs(x2 - x1) >= e and abs(f0(x2)) >= e:  
 counter[0] += 1  
 x1 = x2  
 x2 = x1 - f0(x1)/f1(x1)  
 return x2