**МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА**

**СПОРТУ УКРАЇНИ**

**Національний технічний університет України**

**“Київський політехнічний інститут”**

**Чисельні методи**

**Лабораторна робота №4**



Виконав:

Сочка Олександр Олександрович

**Київ 2015**

**Постановка задачі**

Розв’язати нелінійне рівняння:

**Варіант 8**

**№ = 8 | a5 = 1; a4 = 0; a3 = 0; a2 = 3; a1 = -2; a0 = -1**

**k = 2**

**a = 2**

**Рівняння:**

**a5(1+) x5 + a4x4 + a3x3 + a2x2 + a1x1 + k a0 = 0**

1) Використаємо теорему про границі усіх коренів рівняння:

2) Використаємо теорему Штурма:

**f0**:

**f1**:

**f2**:

**f3**:

**f4**:

**f5**:

Побудуємо таблицю відповідності для значень ряду Штурма:

**-4 -3/4 -1/2 0 1/4 4**

**f0 - + - - - +**

**f1 + - - - - +**

**f2 - - - + + +**

**f3 + + + 0 + +**

**f4 - - - 0 + +**

**f5 + + + + + +**

**n 5 4 3 1 1 0**

Отже, маємо 4 відрізки: [-4; -0.75]; [-0.75; -0.5]; [-0.5; 0]; [1/4;4].

Проте на деяких з цих відрізків функція не змінює знаку, це означає, що розв’язки на даних відрізках є комплексними.

Тому потрібними відрізками є: [-4; -0.75]; [-0.75; -0.5], [0.25, 4].

**Результат**

--- BISECTION ALGORITHM -----

-- Searching for roots in range [-4; -0.75]

Segment [-4.000000; -0.750000]

Segment [-2.375000; -0.750000]

Segment [-1.562500; -0.750000]

Segment [-1.156250; -0.750000]

Segment [-1.156250; -0.953125]

Segment [-1.054688; -0.953125]

Segment [-1.003906; -0.953125]

Segment [-1.003906; -0.978516]

Segment [-1.003906; -0.991211]

Segment [-1.003906; -0.997559]

Segment [-1.000732; -0.997559]

Segment [-1.000732; -0.999146]

Segment [-1.000732; -0.999939]

Segment [-1.000336; -0.999939]

Segment [-1.000137; -0.999939]

Segment [-1.000038; -0.999939]

Segment [-1.000038; -0.999989]

Segment [-1.000013; -0.999989]

Segment [-1.000001; -0.999989]

Segment [-1.000001; -0.999995]

Segment [-1.000001; -0.999998]

Segment [-1.000001; -0.999999]

Segment [-1.000000; -0.999999]

Segment [-1.000000; -1.000000]

x1 = -1.000000

-- Searching for roots in range [-0.75, -0.5]

Segment [-0.750000; -0.500000]

Segment [-0.625000; -0.500000]

Segment [-0.625000; -0.562500]

Segment [-0.593750; -0.562500]

Segment [-0.593750; -0.578125]

Segment [-0.593750; -0.585938]

Segment [-0.589844; -0.585938]

Segment [-0.587891; -0.585938]

Segment [-0.587891; -0.586914]

Segment [-0.587402; -0.586914]

Segment [-0.587402; -0.587158]

Segment [-0.587402; -0.587280]

Segment [-0.587402; -0.587341]

Segment [-0.587372; -0.587341]

Segment [-0.587372; -0.587357]

Segment [-0.587372; -0.587364]

Segment [-0.587368; -0.587364]

Segment [-0.587368; -0.587366]

x1 = -0.587367

-- Searching for roots in range [0.25, 4]

Segment [0.250000; 4.000000]

Segment [0.250000; 2.125000]

Segment [0.250000; 1.187500]

Segment [0.718750; 1.187500]

Segment [0.718750; 0.953125]

Segment [0.835938; 0.953125]

Segment [0.835938; 0.894531]

Segment [0.865234; 0.894531]

Segment [0.865234; 0.879883]

Segment [0.865234; 0.872559]

Segment [0.865234; 0.868896]

Segment [0.867065; 0.868896]

Segment [0.867065; 0.867981]

Segment [0.867523; 0.867981]

Segment [0.867752; 0.867981]

Segment [0.867752; 0.867867]

Segment [0.867752; 0.867809]

Segment [0.867781; 0.867809]

Segment [0.867781; 0.867795]

Segment [0.867788; 0.867795]

Segment [0.867791; 0.867795]

Segment [0.867791; 0.867793]

Segment [0.867792; 0.867793]

Segment [0.867793; 0.867793]

x1 = 0.867793

--- CHORDS ALGORITHM -----

-- Searching for roots in range [-4; -0.75]

Segment [-4.000000; -0.750000]

Segment [-4.000000; -0.750512]

Segment [-2.375256; -0.750512]

Segment [-1.562884; -0.750512]

Segment [-1.156698; -0.750512]

Segment [-1.156698; -0.953605]

Segment [-1.055152; -0.953605]

Segment [-1.004378; -0.953605]

Segment [-1.004378; -0.978992]

Segment [-1.004378; -0.991685]

Segment [-1.004378; -0.998032]

Segment [-1.001205; -0.998032]

Segment [-1.001205; -0.999618]

Segment [-1.000412; -0.999618]

Segment [-1.000015; -0.999618]

Segment [-1.000015; -0.999817]

Segment [-1.000015; -0.999916]

Segment [-1.000015; -0.999965]

Segment [-1.000015; -0.999990]

Segment [-1.000003; -0.999990]

Segment [-1.000003; -0.999996]

Segment [-1.000003; -0.999999]

Segment [-1.000001; -0.999999]

Segment [-1.000000; -0.999999]

x1 = -1.000000

-- Searching for roots in range [-0.75, -0.5]

Segment [-0.750000; -0.500000]

Segment [-0.604887; -0.500000]

Segment [-0.604887; -0.552443]

Segment [-0.604887; -0.578665]

Segment [-0.591776; -0.578665]

Segment [-0.591776; -0.585221]

Segment [-0.588498; -0.585221]

Segment [-0.588498; -0.586859]

Segment [-0.587679; -0.586859]

Segment [-0.587679; -0.587269]

Segment [-0.587474; -0.587269]

Segment [-0.587371; -0.587269]

Segment [-0.587371; -0.587320]

Segment [-0.587371; -0.587346]

Segment [-0.587371; -0.587359]

Segment [-0.587371; -0.587365]

Segment [-0.587368; -0.587365]

Segment [-0.587368; -0.587367]

Segment [-0.587367; -0.587367]

x1 = -0.587367

-- Searching for roots in range [0.25, 4]

Segment [0.250000; 4.000000]

Segment [0.252783; 4.000000]

Segment [0.252783; 2.126391]

Segment [0.252783; 1.189587]

Segment [0.721185; 1.189587]

Segment [0.721185; 0.955386]

Segment [0.838285; 0.955386]

Segment [0.838285; 0.896836]

Segment [0.867561; 0.896836]

Segment [0.867561; 0.882198]

Segment [0.867561; 0.874879]

Segment [0.867561; 0.871220]

Segment [0.867561; 0.869390]

Segment [0.867561; 0.868475]

Segment [0.867561; 0.868018]

Segment [0.867789; 0.868018]

Segment [0.867789; 0.867904]

Segment [0.867789; 0.867846]

Segment [0.867789; 0.867818]

Segment [0.867789; 0.867804]

Segment [0.867789; 0.867796]

x1 = 0.867793

--- NEWTON ALGORITHM -----

-- Searching for roots in range [-4; -0.75]

Current value: -2.375000

Current value: -1.925929

Current value: -1.580978

Current value: -1.325939

Current value: -1.151007

Current value: -1.048598

Current value: -1.007203

Current value: -1.000193

Current value: -1.000000

x1 = -1.000000

-- Searching for roots in range [-0.75, -0.5]

Current value: -0.625000

Current value: -0.585773

Current value: -0.587365

Current value: -0.587367

x1 = -0.587367

-- Searching for roots in range [0.25, 4]

Current value: 2.125000

Current value: 1.691383

Current value: 1.349721

Current value: 1.095532

Current value: 0.937255

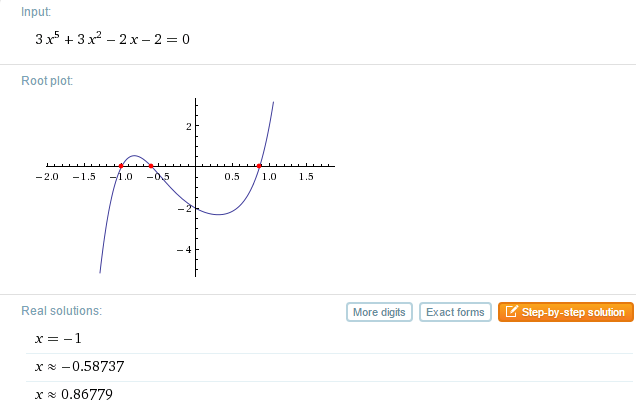
Current value: 0.876027

Current value: 0.867922

Current value: 0.867793

x1 = 0.867793

**Результат в WolframAlpha**

****

**Лістинг програми**

f = [

lambda x: 3 \* x\*\*5 + 3 \* x\*\*2 - 2 \* x - 2,

lambda x: 15 \* x\*\*4 + 6 \* x - 2,

lambda x: 60 \* x\*\*3 + 6,

lambda x: 180 \* x\*\*2,

lambda x: 360 \* x,

lambda x: 360,

]

inf = 10\*\*10

eps = 0.1\*\*6

def signs(t):

q = [ff(t) for ff in f]

for i in reversed(range(len(q))):

if q[i] == 0 or i > 0 and q[i-1] != 0 and (q[i] / abs(q[i]) == q[i-1] / abs(q[i - 1])):

q.pop(i)

return (len(q))//2

def bisect(a, b):

print('Segment [{:.6f}; {:.6f}]'.format(a,b))

c = (a + b) / 2

value = f[0](c)

if abs(value) < eps:

return c

elif (value > 0) ^ (f[0](a) > 0):

return bisect(a, c)

return bisect(c, b)

def chords(a, b):

print('Segment [{:.6f}; {:.6f}]'.format(a,b))

c = (a\*f[0](b) - b\*f[0](a)) / (f[0](b) - f[0](a))

value = f[0](c)

if abs(value) < eps:

return c

elif (value > 0) ^ (f[0](a) > 0):

return bisect(a, c)

return bisect(c, b)

def newton(a, b):

def do(arg):

print('Current value: {:.6f}'.format(arg))

value = f[0](arg)

if abs(value) < eps:

return arg

return do(arg - value / f[1](arg))

return do((a+b)/2)

print('--- BISECTION ALGORITHM -----')

print('-- Searching for roots in range [-4; -0.75]')

print('x1 = {:.6f}'.format(bisect(-4, -0.75)))

print('-- Searching for roots in range [-0.75, -0.5]')

print('x1 = {:.6f}'.format(bisect(-0.75, -0.5)))

print('-- Searching for roots in range [0.25, 4]')

print('x1 = {:.6f}'.format(bisect(0.25, 4)))

print('--- CHORDS ALGORITHM -----')

print('-- Searching for roots in range [-4; -0.75]')

print('x1 = {:.6f}'.format(chords(-4, -0.75)))

print('-- Searching for roots in range [-0.75, -0.5]')

print('x1 = {:.6f}'.format(chords(-0.75, -0.5)))

print('-- Searching for roots in range [0.25, 4]')

print('x1 = {:.6f}'.format(chords(0.25, 4)))

print('--- NEWTON ALGORITHM -----')

print('-- Searching for roots in range [-4; -0.75]')

print('x1 = {:.6f}'.format(newton(-4, -0.75)))

print('-- Searching for roots in range [-0.75, -0.5]')

print('x1 = {:.6f}'.format(newton(-0.75, -0.5)))

print('-- Searching for roots in range [0.25, 4]')

print('x1 = {:.6f}'.format(newton(0.25, 4)))

**Висновки**

В процесі лабораторної роботи було проаналізовано теореми для виділення інтервалів дійсних коренів нелінійних поліноміальних рівнянь, а також досліджено 3 методи для знаходження коренів нелінійних поліноміальних рівнянь — метод бісекції, хорд та дотичних.

Найшвидшим методом виявився метод дотичних, який і в теорії має квадратичну швидкість збіжності.

Можна використати комбінований метод хорд та дотичних, тоді результат можна отримати набагато швидше.

Метод дотичних потребує знання похідної функції, що буває неможливо отримати на практиці, оскільки дані до алгоритму можуть надходити із вимірювальних пристроїв, а функції, яка описує ці дані немає. Тому іноді використовують модифікований метод дотичних, на кожній ітерації відбувається ділення на певну константу, яка пов’язана із вхідною функцією.