Міністерство освіти і науки, молоді та спорту України
Національний технічний університет України
«КПІ ім. Ігоря Сікорського»
Фізико-технічний інститут

Лабораторна робота №1. Обчислення значень функції

Група ФФ-83
Виконав: Попівчак Б.П.
Перевірив(ла): Гордійко Н.О.

Київ, 2020 р.

1 Хід роботи

1.1 Задача 13.а

Заданий многочлен $P(x) = 0.22x^5 - 3.27x^4 - 2.74x^3 + 2.81x^2 - 3.36x + 2$. *Визначити значення* $P(x)$ *де* $x = 0.8 + 0.05k$, $(k = 16, 17, ..., 20)$.

Теоретичне підгрунтя для вирішення задачі

Нехай є многочлен n-ого степеня $P(x) = a_0x^n + a_1x^{n-1} + ... + a_n$ з дійсними коефіцієнтами $a_k(k = 0, 1, ..., n)$, і нехай потрібно визначити значення цього многочлена при $x = \xi$:

$$P(x) = a_0\xi^n + a_1\xi^{n-1} + ... + \xi_n \tag{1}$$

Обчислення $P(\xi)$ найзручніше проводити наступним чином. Представимо вираз (1) у вигляді вкладених множень $P(\xi) = (...(((a_0\xi + a_1)\xi + a_2)\xi + a_3)\xi + ... + a_n)$.

Якщо ввести числа

$$\left.\begin{aligned} b_0 &= a_0, \\ c_1 = b_0\xi,\ b_1 &= a_1 + c_1, \\ c_2 = b_1\xi,\ b_2 &= a_2 + c_2, \\ \dots \qquad &\qquad \dots \\ c_n = b_{n-1}\xi,\ b_n &= a_n + c_n, \end{aligned}\right\} \tag{2}$$

то $b_n = P(\xi)$

Отже, обчислення значення многочлена P(x) при $x = \xi$ зводиться до повторення таких елементарних операцій: $c_k = b_{k-1}\xi, b_k = a_k + c_k, (k = 0, 1, 2, \dots, n)$.

Код програми

```
koefs = [0.22, -3.27, -2.74, 2.81, -3.36, 2]
first_dodanok =0.8
koef_pry_k = 0.05
znach_k = [i for i in range(16,21)]

def Gorner_scheme(k):#Realizaciya sxemy Gornera dlya pevnogo znachennya k
        b = koefs[0]#Pochatkove znachennya sxemy Gornera dorivnyuye koefu pry x z najbilshym stepenem
        for koef  in range(1,len(koefs)):
            b = (b*(first_dodanok + koef_pry_k * k)) + koefs[koef]# first_dodanok + koef_pry_k * ka - ce znachennya x
        return (b)

def Gorner_scheme_cycle() :#zapusayemo cykl sxemy Gornera dlya vsix znachen k
    znach_poly_pry_k = []#Spysok v yakyj budemo dodavaty znachennya polinoma pry riznyx k
    for ka in znach_k:
        p = Gorner_scheme(ka)
        znach_poly_pry_k.append(p)#pry kozhnomu novomu k dodayemo znachennya polinoma v spysok
    return (znach_poly_pry_k)

#Adding to csv table for printing in document
znach_x = []
for i in znach_k:
    m = round(first_dodanok + koef_pry_k*i , 2)
    znach_x.append(m)
```

```
table=[['x', 'znach']]
for i in range(0,len(znach_x)):
    l = Gorner_scheme_cycle()
    print(str(znach_x[i])+'  :  '+str(l[i]))
    s = [znach_x[i], l[i]]
    table.append(s)
outfile = open('znach_1.csv', 'w', newline="")
import csv
writer = csv.writer(outfile)
for row in table:
    writer.writerow(row)
outfile.close()
```

Вихідний результат

В таблиці 1 наведено результат роботи програми.

x	$P(x)$
1.60	-26.5288448
1.65	-29.748899668750013
1.70	-33.24040160000001
1.75	-37.017207031249995
1.80	-41.0933824

Табл. 1: Значення полінома $P(x)$ при заданих значення x.

1.2 Задача 7.б

Користуючись розкладом функції cosx в степеневий ряд, скласти таблицю її значень з точністю до 10^{-15} *для вказаного значення x, якщо:* $x = 1.75 + 0.01k,\ (k = 0, 1, 2, ..., 15)$.

Теоретичне підгрунтя для вирішення задачі

Для обчислення функції *cosx* користуємося користуємось степеневим розкладом:

$$cosx = \sum_{k=0}^{\infty}(-1)^k \frac{x^{2k}}{(2k)!}, \qquad (-\infty < x < \infty). \tag{3}$$

Данний ряд при великих x збігається повільно, але, враховуючи періодичність функції *cosx* і формули зведення тригонометричних функцій, легко зробити висновок, що достатньо вміти обчислювати *cosx* для проміжку $0 \le x \le \frac{\pi}{4}$. При цьому можна використовувати такі рекурентні формули :

$$\left.\begin{aligned} cosx &= \sum_{k=1}^{n} \nu_k + R_n(x), \\ \nu_1 = 1, \quad \nu_{k+1} &= -\frac{x^2}{(2k-1)2k}\nu_k, \quad (k = 1, 2, .., n-1) \end{aligned}\right\} \tag{4}$$

Код програми

```
"""zadayemo znachennya iz umovy‘"""
a = 1.75
b = 0.01
znach_k = [i for i in range(0,16)]

#Funkciya dlya obrahunku sumy v_i ,
#yaki zaneseni v spysok znach_v
def listsum(numList):
    theSum = 0
    for i in numList:
        theSum = theSum + i
    return round(theSum,15) #Okruglyayemo otrymane znachennya do 15 znaku

#Vyznachaye znachennya vsih v_i ta zanosyt yix u spysok
def cosx(k):
    # Ve pershe za umovoyu dorivnyuye 1
    v = 1
    znach_v = [1]
    for i in range(1,100): #diapazon znachennya i mozhna braty dovilnyj, golovne, shhob vykonalos dostatnye chyslo iteracij
        if abs(v) < pow(10,-15): #umova pry yakij nastupne ve ite menshe dopustymoyi poxybky
            # Oskilky odne znachennya v_i ,yake menshe za dopustymu poxybku zanosytsya v spysok , treba jogo vydalyty
            del znach_v[len(znach_v)-1]
            return (listsum(znach_v))
            break

        else :
            v = -v*pow((a+(b*k)),2)/(((2*i)-1)*2*i)
            znach_v.append(v) #Dodayemo vsi v_i v spysok

#Obchyslyuye znachennya kosynusu dlya riznyx k
def znach_cos():
    znach_cos = []
    for k in znach_k:
        t = cosx(k)
        znach_cos.append(t)
    return (znach_cos)


#Adding to csv table for printing in document
znach_x = []
for i in znach_k:
    m = round(a + b*i , 2)
    znach_x.append(m)
table=[['x', 'znach']]
for i in range(0,len(znach_x)):
    l = znach_cos()
    print(str(znach_x[i])+'  :  '+str(l[i]))
    s = [znach_x[i], l[i]]
    table.append(s)
outfile = open('znach_2.csv', 'w', newline="")
import csv
writer = csv.writer(outfile)
for row in table:
    writer.writerow(row)
outfile.close()
```

Вихідний результат

В таблиці 2 наведено результат роботи програми.

x	$cosx$
1.75	−0.178246055649492
1.76	−0.188076838892880
1.77	−0.197888814609109
1.78	−0.207681001608783
1.79	−0.217452420681364
1.80	−0.227202094693087
1.81	−0.236929048684674
1.82	−0.246632309968834
1.83	−0.256310908227522
1.84	−0.265963875608980
1.85	−0.275590246824512
1.86	−0.285189059245020
1.87	−0.294759352997260
1.88	−0.304300171059832
1.89	−0.313810559358882
1.90	−0.323289566863503

Табл. 2: Значення функції $cosx$ при заданих значення x.

1.3 Задача 7.в

Користуючись многочленним наближенням та схемою Горнера, скласти таблицю значень функції з точністю до ε для вказаного значення , якщо:

$$y = \frac{1}{2\sqrt{\pi}} e^{-\frac{x^2}{2}}, \qquad x = 1.75 + 0.01k \ (k = 0, 1, 2, ..., 15), \qquad \varepsilon = 10^{-4}.$$

Теоретичне підгрунтя для вирішення задачі

Обчислення за допомогою рядів Тейлора дає досить швидку збіжність, взагалі-то, лише при малих значеннях $|x - x_0|$. Однак часто буває потрібно за допомогою многочлена порівняно невисокого степеня підібрати наближення, яке давало б достатню точність для всіх точок заданого відрізка. У цих випадках застосовуються розклади функцій, отримані за допомогою поліномів Чебишева на заданому відрізку.Для обчислення значень многочлена можна використовувати схему Горнера.

Для обчислення значень показникової функції на відрізку $[-1, 1]$. Користуємося такими многочленним наближенням:

$$e^x \approx \sum_{k=0}^{7} a_k x^{7-k} \qquad (|x| \le 1), \qquad \varepsilon = 2 \cdot 10^{-7} \tag{5}$$

$a_7 = 0.9999998,\ a_6 = 1.0000000,\ a_5 = 0.5000063,\ a_4 = 0.1666674,\ a_3 = 0.0416350,\ a_2 = 0.0083298,\ a_1 = 0.0014393,\ a_0 = 0.0002040.$

Код програми

```
import math
znach_a = [ 0.0002040 , 0.0014393, 0.0083298,
           0.0416350, 0.1666674, 0.5000063, 1.0 ,0.9999998]
a = 0.4
koef_pry_k = 0.002
alpha = 1/(2*math.sqrt(math.pi))
znach_k = [k for k in range(0,16)]

def Gorner_scheme_exp(k):#Realizaciya sxemy Gornera dlya pevnogo znachennya k
    b = znach_a[0]#Pochatkove znachennya sxemy Gornera dorivnyuye koefu pry x z najbilshym stepenem
    for koef in range(1, len(znach_a)):
        # -pow((a + koef_pry_k * k),2)/2) - ce znachennya stepeniu exponenty
        b = (b * (-pow((a + koef_pry_k * k),2)/2)) + znach_a[koef]
    return round(b*alpha,4)



def Gorner_scheme_cycle():#zapusayemo cykl sxemy Gornera dlya vsix znachen k
    znach_poly_pry_k = []#Spysok v yakyj budemo dodavaty znachennya polinoma pry riznyx k
    for ka in znach_k:
        p = Gorner_scheme_exp(ka)
        znach_poly_pry_k.append(p)#pry kozhnomu novomu k dodayemo znachennya polinoma v spysok
    return (znach_poly_pry_k)

#Adding to csv table for printing in document
znach_x = []
for i in znach_k:
    m = round(a + koef_pry_k*i , 3)
    znach_x.append(m)
table=[['x', 'znach']]
for i in range(0,len(znach_x)):
    l = Gorner_scheme_cycle()
    print(str(znach_x[i])+'  :  '+str(l[i]))
    s = [znach_x[i], l[i]]
    table.append(s)
outfile = open('znach_3.csv', 'w', newline="")
import csv
writer = csv.writer(outfile)
for row in table:
    writer.writerow(row)
outfile.close()
```

Вихідний результат

В таблиці 3 наведено результат роботи програми.

x	$\frac{1}{2\sqrt{\pi}}e^{-\frac{x^2}{2}}$
0.400	0.2604
0.402	0.2602
0.404	0.2600
0.406	0.2598
0.408	0.2596
0.410	0.2594
0.412	0.2591
0.414	0.2589
0.416	0.2587
0.418	0.2585
0.420	0.2583
0.422	0.2581
0.424	0.2578
0.426	0.2576
0.428	0.2574
0.430	0.2572

Табл. 3: Значення функції $y = \frac{1}{2\sqrt{\pi}}e^{-\frac{x^2}{2}}$ при заданих значення x.

1.4 Задача 7.г

Користуючись методом ітерацій, скласти таблицю значень функції y з точністю до 10^{-15}*, якщо:* $y = \sqrt[3]{x}, \qquad x = 3 + k, \ (k = 0, 1, 2, ..., 15)$.

Теоретичне підгрунтя для вирішення задачі

Будь-яку функцію $y = f(x)$ можна різними способами задавати неявно, тобто деяким рівнянням

$$F(x, y) = 0 \tag{6}$$

Чисельний метод, в якому відбувається послідовне, крок за кроком, уточнення початкового наближення, називається *ітераційним методом.*

Часто буває, що розв'язання рівняння (5) відносно y якимось ітераційним методом зводиться до однотипних операцій, які легко реалізувати на комп'ютері. Тоді, вочевидь, доцільно застосувати **метод ітерацій**.

Один з можливих ітераційних процесів для обчислення $y(x)$ можна побудувати таким чином. Нехай y_n наближене значення y. Застосувавши формулу Лагранжа, отримаємо $F(x, y_n) = (y_n - y)F'_y(x, \overline{y_n})$, де $\overline{y_n}$ деяке проміжне значення між y_n та y. Звідси $y = y_n - \frac{F(x, y_n)}{F'_y(x, \overline{y_n})}$, при чому значення $\overline{y_n}$ нам не відоме.

Вважаючи наближено, що $\overline{y_n} \approx y_n$, отримаємо наступну формулу дл обчислення $y \approx y_{n+1}$:

$$y_{n+1} = y_n - \frac{F(x, y_n)}{F'(x, y_n)} \qquad (n = 0, 1, 2, ...) \tag{7}$$

Якщо $F'_y(x,y)$ та $F''_y(x,y)$ існують і зберігають постійні знаки в розглянутому інтервалі, що містить корінь $y(x)$, то ітераційний процес збігається до $y(x)$.

Процес ітераційний продовжується доти, поки в границях заданої точності два послідовних значення y_n та y_{n+1} не співпадуть між собою, після чого наближено вважають, що $y(x) \approx y_{n+1}$.

Обчислення кубічного кореня.

Нехай маємо $y = \sqrt[3]{x}$. Застосувавши формулу 7 до рівняння $F(x,y) \equiv y^3 - x = 0$, отрмаємо ітераційну формулу для обчислення кубічного кореня у вигляді

$$y_{n+1} = \frac{1}{3}\left(\frac{2y_n^3 + x}{y_n^2}\right)$$

Початкове наближення $y_0 = 2^{E\left(\frac{2}{3}\right)}$, де $x = 2^m x_1$, де m - це ціле число і $\frac{1}{2} \le x_1 < 1$.

Код програми

```
a = 3
znach_k = [i for i in range(0,16)]

znach_x = [] #stvoryuyemo spysok iz znachennyamy iksiv dlya podalshogo drukuvannya
for i in znach_k:
    iks = a + i
    znach_x.append(iks)


def pochatkove_nabl(x): #Obraxovuye pochatkovi nablyzhennya dlya iksiv
    m = 0
    for i in range(1,12 ):
        if 1/2 <= (x/pow(2, i)) < 1:
            m = m+1
            return pow(2, int(m/3))
            break
        else:
            m = i

znach_y_0=[]
for i in znach_x: #Zanosymo pochatkovi nablyzhennya dlya vsix x u spysok
    y_0 = pochatkove_nabl(i)
    znach_y_0.append(y_0)



#obchyslyuye korin kubichnyj dlya kozhnogo okremogo x
# k - poryadkovyj nomer elementa zi spysku iksiv , x - jogo znachennya
def kub_korin(k,x):
    znach_y = []
    y = znach_y_0[k]
    for i in range(0,100):
        y = (1/3)*(((2*pow(y,3))+x)/pow(y,2))
        znach_y.append(y)
        if i>1 and abs(znach_y[i]-znach_y[i-1])<=pow(10,-15): #umova prypynennya iteracij
            return round(znach_y[len(znach_y)-1],15)
            break
```

```
38
39 #Vyznachayemo korin dlya kozhnogo iksa ta zanosymo v spysok
40 def all_kub_korin():
41     all_znach_y = []
42     for i in range(0,len(znach_x)) :
43         r = kub_korin(i, znach_x[i])
44         all_znach_y.append(r)
45     return all_znach_y
46
47
48 #Adding to csv table for printing in document
49 table=[['x', 'znach']]
50 for i in range(0,len(znach_x)):
51     l = all_kub_korin()
52     print(str(znach_x[i])+'  :  '+str(l[i]))
53     s = [znach_x[i], l[i]]
54     table.append(s)
55 outfile = open('znach_4.csv', 'w', newline="")
56 import csv
57 writer = csv.writer(outfile)
58 for row in table:
59     writer.writerow(row)
60 outfile.close()
```

Вихідний результат

В таблиці 4 наведено результат роботи програми.

x	$\sqrt[3]{x}$
3	1.442249570307408
4	1.587401051968199
5	1.709975946676697
6	1.817120592832140
7	1.912931182772389
8	2.000000000000000
9	2.080083823051904
10	2.154434690031883
11	2.223980090569315
12	2.289428485106664
13	2.351334687720757
14	2.410142264175230
15	2.466212074330470
16	2.519842099789746
17	2.571281590658235
18	2.620741394208896

Табл. 4: Значення функції $y = \sqrt[3]{x}$ при заданих значеннях x.

Дякую за увагу!