Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Алгоритми та методи обчислень

Лабораторна робота №3

«Інтерполяція функцій»

Виконала:

студентка групи ІО-64

Бровченко А. В.

Залікова книжка № 6403

Номер у списку групи 3

Перевірив ст.вик. Порєв В. М.

Київ

2018 р.

**Тема:** «Інтерполяція функцій».

**Мета:** Ознайомлення з інтерполяційними формулами Лагранжа, Ньютона, рекурентним співвідношенням Ейткена, методами оцінки похибки інтерполяції.

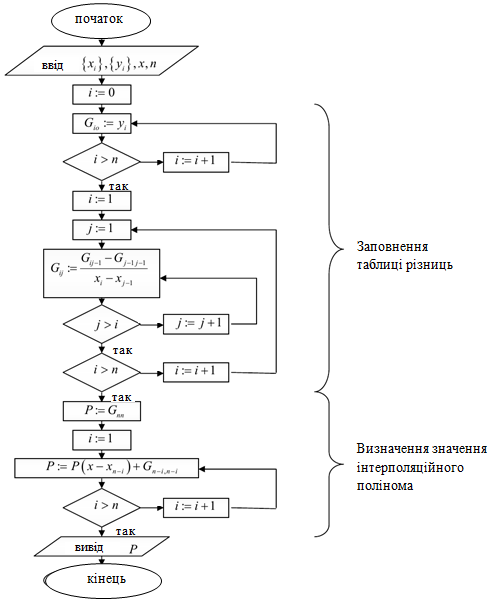
**Завдання:** Закріплення, поглиблення і розширення знань студентів при вирішенні практичних обчислювальних завдань. Оволодіння обчислювальними

методами і практичними методами оцінки похибки обчислень. Придбання умінь і навичок при програмуванні та налагодженні обчислювальних завдань на

комп'ютері.

|  |  |  |  |
| --- | --- | --- | --- |
| № Варіанта |  |  | Формула |
| 3 |  | [0.4] | 1.6 |

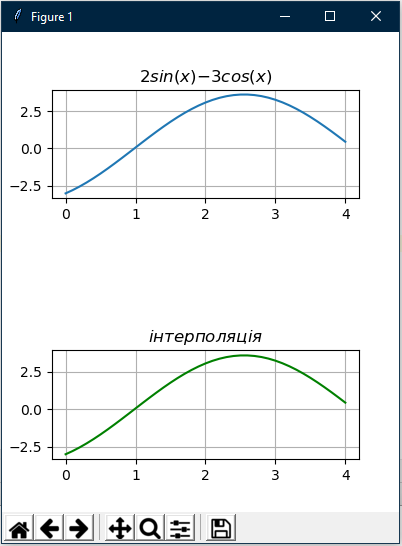
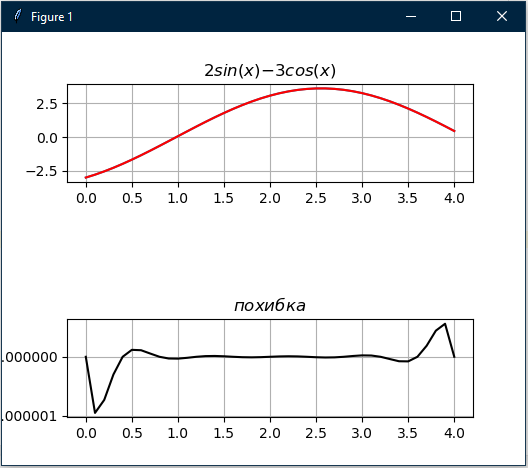
Блок-схема:



Текст програми:

**import** matplotlib  
matplotlib.use(**"TkAgg"**)  
**from** matplotlib.backends.backend\_tkagg **import** FigureCanvasTkAgg  
**from** matplotlib.figure **import** Figure  
**from** tkinter **import** \*  
**import** math  
**import** matplotlib.pyplot **as** plt  
**import** numpy **as** np  
  
root = Tk()  
root[**'bg'**] = **"yellow"**root.title(**"Лабораторна робота №3"**)  
root.geometry(**"400x220"**)  
root.resizable(width=**False**, height=**False**)  
  
font\_default = **'Arial 14'**bg\_default = **"yellow"**but\_bg\_default = **'darkorange'**but\_fg\_default = **'black'  
  
  
class** Go():  
 i = 10  
 a = 0  
 b = 4  
  
 xi = []  
 y\_f = []  
 y\_sin = []  
 list\_constants = []  
 list\_constants\_sin = []  
  
 h = (b - a)/10  
  
 **for** i **in** range(i+1):  
 xi.append(round(a+h\*i,12))  
 print(xi)  
  
 **for** i **in** range(len(xi)):  
 y\_f.append( (2\*math.sin(xi[i]))-(3\*math.cos(xi[i])) )  
 y\_sin.append( math.sin(xi[i]) )  
  
 **def** \_\_init\_\_(self):  
  
 inform = Label(root, text = **"Бровченко Анастасія Вікторівна\n"  
 "Група ІО-64\n"  
 "Номер у списку - 3\n\n"  
 "Тема:«Інтерполяція функцій"**, bg=bg\_default,  
 font=font\_default)  
 inform.pack(side = TOP, fill=BOTH, expand = **True**)  
  
 Button(root, text=**'Завдання за варіантом'**, bg=but\_bg\_default, fg=but\_fg\_default, font=font\_default,  
 command=self.condition).pack(side=BOTTOM)  
  
 **def** condition(self):  
  
 top = Toplevel(root, bg=bg\_default)  
 top.title(**"Варіант завдання"**)  
 top.geometry(**"600x280"**)  
 top.resizable(width=**False**, height=**False**)  
  
 var = Label(top, text=**"Інтерполяція функції через многочлен Ньютона\n\n"  
 "f(x)=2sin(x)−3cos(x)\n"  
 "[a,b]=[{},{}]"**.format(self.a, self.b), bg=bg\_default, font=**"Arial 16"**)  
  
 button\_f= Frame(top, bg=bg\_default)  
 button\_f1= Frame(button\_f, bg=bg\_default)  
 button\_f2= Frame(button\_f, bg=bg\_default)  
  
 empty\_label1 = Label(top, bg=bg\_default)  
 empty\_label2 = Label(button\_f, bg=bg\_default)  
  
 empty\_label3 = Label(button\_f1, bg=bg\_default)  
 empty\_label4 = Label(button\_f2, bg=bg\_default)  
  
 self.interp\_f = Button(button\_f1, text = **"Графік інтерполяції функції"**, width = 25, height = 2,  
 command = self.interpolation\_f, bg=but\_bg\_default, fg=but\_fg\_default, font=font\_default)  
 self.infelicity\_f = Button(button\_f2, text = **"Похибка інтерполяції функції"**, width = 25, height = 1,  
 command = self.infelicity\_f, bg=but\_bg\_default, fg=but\_fg\_default, font=font\_default)  
 self.interp\_sin = Button(button\_f1, text=**"Графік інтерполяції sin(x)"**, width = 25, height = 2,  
 command = self.interpolation\_sin, bg=but\_bg\_default, fg=but\_fg\_default, font=font\_default)  
 self.infelicity\_sin = Button(button\_f2, text=**"Похибка інтерполяції sin(x)"**, width = 25, height = 1,  
 command = self.infelicity\_sin, bg=but\_bg\_default, fg=but\_fg\_default, font=font\_default)  
  
 var.pack(side = TOP)  
 empty\_label1.pack(side=TOP)  
 button\_f.pack(side=TOP)  
 button\_f1.pack(side=TOP)  
 empty\_label2.pack(side=TOP)  
 button\_f2.pack(side=TOP)  
 self.interp\_f.pack(side=LEFT)  
 empty\_label3.pack(side=LEFT)  
 self.interp\_sin.pack(side=RIGHT)  
 self.infelicity\_f.pack(side=LEFT)  
 empty\_label4.pack(side=LEFT)  
 self.infelicity\_sin.pack(side=RIGHT)  
  
 top.mainloop()  
  
 **def** interpolation\_sin(self):  
 self.list\_constants\_sin = []  
  
 a = len(self.xi)-1  
 print(a)  
 **while** a >= 0:  
 constant = 0  
 **for** i **in** range(len(self.xi)-a):  
 znam = 1  
 **for** j **in** range(len(self.xi)-a):  
 **if** i != j:  
 znam \*= (self.xi[i]-self.xi[j])  
 constant += self.y\_sin[i]/znam  
 self.list\_constants\_sin.append(constant)  
 a -= 1  
  
 self.graph\_sin()  
  
 **def** result\_sin(self, x):  
 g=1  
 a = len(self.xi)-1  
 listh = [1]  
  
 **for** i **in** range(len(self.xi)):  
 g \*= (x - self.xi[i])  
 listh.append(g)  
  
 f0 = 0  
 **for** i **in** range(len(self.list\_constants\_sin)):  
 f0 += self.list\_constants\_sin[i]\*listh[i]  
 **return** f0  
  
 **def** func\_sin(self,x):  
 sin = self.result\_sin(x)  
 **return** sin  
  
 **def** graph\_sin(self):  
 self.xlist\_sin = []  
 **for** i **in** np.arange(Go().a, Go().b+0.1, 0.1):  
 self.xlist\_sin.append(i)  
  
 **def** funcy(x):  
 y = (math.sin(x))  
 **return** y  
  
 self.ylist\_sin = [funcy(x) **for** x **in** self.xlist\_sin]  
 self.flist\_sin = [self.func\_sin(x) **for** x **in** self.xlist\_sin]  
  
 plt.subplot(311)  
 plt.plot(self.xlist\_sin,self.ylist\_sin)  
 plt.title(**r'$sin(x)$'**)  
 plt.grid(**True**)  
  
 plt.subplot(313)  
 plt.plot(self.xlist\_sin,self.flist\_sin,**'g'**)  
 plt.title(**r'$інтерполяція$'**)  
 plt.grid(**True**)  
  
 plt.show()  
  
 **def** infelicity\_sin(self):  
  
 plt.subplot(311)  
 plt.plot(self.xlist\_sin,self.ylist\_sin,self.xlist\_sin,self.flist\_sin,**"g"**)  
 plt.title(**r'$sin(x)$'**)  
 plt.grid(**True**)  
  
 ylist\_sin = self.ylist\_sin  
 flist\_sin = self.flist\_sin  
  
 **def** funcy(x):  
 y = (ylist\_sin[x] - flist\_sin[x])  
 **return** y  
  
 suby = [funcy(x) **for** x **in** range(len(self.xlist\_sin))]  
 plt.subplot(313)  
 plt.plot(self.xlist\_sin,suby, **"c"**)  
 plt.title(**r'$похибка$'**)  
 plt.grid(**True**)  
  
 plt.show()  
  
 **def** interpolation\_f(self):  
 self.list\_constants = []  
  
 a = len(self.xi)-1  
 **while** a >= 0:  
 constant = 0  
 **for** i **in** range(len(self.xi)-a):  
 znam = 1  
 **for** j **in** range(len(self.xi)-a):  
 **if** i != j:  
 znam \*= (self.xi[i]-self.xi[j])  
 constant += self.y\_f[i]/znam  
 self.list\_constants.append(constant)  
 a -= 1  
  
 self.graph\_f()  
  
 **def** result\_f(self, x):  
 g=1  
 a = len(self.xi)-1  
 listh = [1]  
  
 **for** i **in** range(len(self.xi)):  
 print(x)  
 g \*= (x - self.xi[i])  
 listh.append(g)  
  
 f0 = 0  
 **for** i **in** range(len(self.list\_constants)):  
 f0 += self.list\_constants[i]\*listh[i]  
 **return** f0  
  
 **def** func\_f(self,x):  
 f = self.result\_f(x)  
 **return** f  
  
 **def** graph\_f(self):  
 slave = Toplevel(root)  
 self.xlist\_f = []  
  
 **for** i **in** np.arange(Go().a, Go().b+0.1, 0.1):  
 self.xlist\_f.append(i)  
  
 **def** funcy(x):  
 y = (2\*math.sin(x))-(3\*math.cos(x))  
 **return** y  
  
 self.ylist\_f = [funcy(x) **for** x **in** self.xlist\_f]  
 self.flist\_f = [self.func\_f(x) **for** x **in** self.xlist\_f]  
  
 f = Figure(figsize=(6, 6), dpi=100)  
 a = f.add\_subplot(311)  
 a.plot(self.xlist\_f, self.ylist\_f)  
 a.set\_title(**r'$2sin(x)−3cos(x)$'**)  
 a.grid(**True**)  
  
 b = f.add\_subplot(313)  
 b.plot(self.xlist\_f,self.flist\_f,**'g'**)  
 b.set\_title(**r'$інтерполяція$'**)  
 b.grid(**True**)  
  
 f.tight\_layout()  
  
 canvas = FigureCanvasTkAgg(f, slave)  
 canvas.show()  
 canvas.get\_tk\_widget().pack(side=BOTTOM, fill=BOTH, expand=**True**)  
  
 **def** infelicity\_f(self):  
  
 plt.subplot(311)  
 plt.plot(self.xlist\_f,self.ylist\_f,self.xlist\_f,self.flist\_f,**"r"**)  
 plt.title(**r'$2sin(x)−3cos(x)$'**)  
 plt.grid(**True**)  
  
 ylist\_f = self.ylist\_f  
 flist\_f = self.flist\_f  
  
 **def** funcy(x):  
 y = (ylist\_f[x] - flist\_f[x])  
 **return** y  
  
 suby = [funcy(x) **for** x **in** range(len(self.xlist\_f))]  
 plt.subplot(313)  
 plt.plot(self.xlist\_f, suby, **"k"**)  
 plt.title(**r'$похибка$'**)  
 plt.grid(**True**)  
  
 plt.show()  
  
  
**if** \_\_name\_\_==**"\_\_main\_\_"**:  
 go = Go()  
 root.mainloop()

Результат роботи програми:

Висновок:

Під час даної лабораторної роботи я:

закріпила знання про інтерполяції, та методи її реалізації; розробила відповідну програму на основі алгоритму інтерполяції методом Ньютона;

Результати успішної роботи програми наведені вище підтверджують правильність обраних мною рішень.