МIНIСТЕРСТВО ОСВIТИ I НАУКИ УКРАЇНИ

НАЦIОНАЛЬНИЙ ТЕХНIЧНИЙ УНIВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛIТЕХНIЧНИЙ IНСТИТУТ»

Кафедра прикладної математики

Звіт

з лабораторної роботи №2

з дисципліни «Алгоритми і системи комп’ютерної математики-2.

Програмні засоби»

на тему:

«Експоненційна регресія з трьома параметрами»

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студент групи КМ-63 | *Старший викладач Бай Ю.П.* |
| *Вовченко І.В..* |  |

KM-6304.docx

KM-6304.py

KM-6304.m

Київ — 2020

# **ЗМІСТ**

[ЗМІСТ 1](#_Toc33517742)

[1 ВСТУП 2](#_Toc33517743)

[2 ОСНОВНА ЧАСТИНА 3](#_Toc33517744)

[2.1 Постановка задачі 3](#_Toc33517745)

[2.2 Описання методу 3](#_Toc33517746)

[2.3 Порядок виконання роботи 5](#_Toc33517747)

[2.4 Контрольний приклад 6](#_Toc33517748)

[2.5 Контрольні запитання 7](#_Toc33517749)

[2.6 Опис програмних засобів 9](#_Toc33517750)

[3 ВИСНОВКИ 12](#_Toc33517751)

[4 СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 13](#_Toc33517752)

[ДОДАТКИ 14](#_Toc33517753)

[Додаток А (код програми GNU Octave) 14](#_Toc33517754)

[Додаток Б (код програми Python) 15](#_Toc33517755)

[Додаток B (результати виконання ) 18](#_Toc33517756)

# **1 ВСТУП**

Темою для даної лабораторної роботи є «Експоненційна регресія з трьома параметрами». Головною метою для виконання лабораторної роботи являється розробка програмного забезпечення. Програмне забезпечення має бути реалізовано з використанням мови Python, а також за допомогою програмного пакету математичних обчислень GNU Octave. Програмне забезпечення необхідно реалізувати в двох варіантах: з використанням вбудованих функцій бібліотек Python та GNU Octave, а також власноруч реалізувати метод.

# **2 ОСНОВНА ЧАСТИНА**

# **2.1 Постановка задачі**

За допомогою Python та GNU Octave точковим методом найменших квадратів апроксимувати експоненційною функцією з трьома параметрами наступну дискретно задану функцію. Побудувати множину заданих точок та знайдену функцію. Визначити максимальну за модулем похибку апроксимації.

Варіант №12:

|  |  |
| --- | --- |
| ***x*** | ***y*** |
| 0 | 1,2 |
| 1 | 3,7 |
| 2 | 4,8 |
| 3 | 14,2 |
| 4 | 24,1 |
| 5 | 58,5 |

# **2.2 Описання методу**

Метод найменших квадратів (МНК) дозволяє за експериментальними даними підібрати таку аналітичну функцію, що проходить настільки близько до експериментальних точок, наскільки це можливо. У загальному випадку завдання можна сформулювати наступним чином.

Нехай в результаті експерименту була отримана деяка експериментальна залежність , представлена у вигляді множини точок:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***x*** |  |  |  | … |  |  |
| ***y*** |  |  |  | … |  |  |

Необхідно побудувати аналітичну залежність , що найбільш точно описує результати експерименту. Для побудови параметрів функції будемо використовувати метод найменших квадратів. Ідея методу найменших квадратів полягає в тому, що функцію необхідно підібрати таким чином, щоб сума квадратів відхилень виміряних значень від обчислених значень була найменшою:

Для вирішення задачі необхідно за результатами експерименту визначити вигляд шуканої залежності, а також підібрати коефіцієнти залежності . Математична задача підбору параметрів залежності зводиться до визначення коефіцієнтів .

В межах даної лабораторної роботи необхідно підібрати параметри для експоненційної функції з трьома параметрами, що апроксимує дискретно задану множину точок. Тобто необхідно визначити коефіцієнти функції . Це можна зробити за допомогою наступних кроків:

Зводимо до лінійної регресії:

Заміна:

Зводимо до наступного вигляду:

Функція відхилення:

# **2.3 Порядок виконання роботи**

1. Записуємо вхідні дані відповідно умови варіанту, тобто множину точок X та Y
2. За допомогою формул наведених вище складаємо систему лінійних рівнянь відносно невідомих параметрів *a* і *b*. Система має наступний вигляд:
3. Знаходимо розв’язок утвореної системи, тобто знаходимо шукані параметри *a* і *b* апроксимуючої експоненційної функції.
4. Знаходимо третій необхідний параметр *c* за формулою:

Отримуємо рівняння апроксимуючої експоненційної функції з трьома параметрами . Також знаходимо невідомі параметри за допомогою вбудованих функцій.

1. Будуємо графік знайденої апроксимуючою функції за допомогою алгоритму, а також графік функції, що знайдена з використанням вбудованих методів. На обох графіках позначаємо точки початкової дискретно заданої функції.
2. Обчислюємо похибку апроксимації за формулою:

Представлену вище послідовність дій необхідно реалізувати у вигляді програмного забезпечення двома способами: використовуючи мову Python, а також за допомогою програмного пакету математичних обчислень GNU Octave.

# **2.4 Контрольний приклад**

Точковим методом найменших квадратів апроксимувати експоненційною функцією з трьома параметрами наступну дискретно задану функцію:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***x*** | 0 | 1 | 2 | 3 | 4 |
| ***y*** | 0 | 1,18 | 1,9 | 2,33 | 2,59 |

Складаємо систему за формулою:

Маємо:

Знаходимо невідомі коефіцієнти:

Отримуємо шукані параметри *A* і *B*:

Далі за формулою знаходимо третій невідомий параметр *C*:

Отже, визначили рівняння апроксимуючої експоненційної функції з трьома параметрами:

# **2.5 Контрольні запитання**

1. В якому вигляді шукають апроксимуючу функцію?

Функцію можна апроксимувати поліномом n-го степеня, степеневою функцією, зокрема експоненційною регресією, логарифмічною функцією, тригонометричною функцією тощо.

1. Який метод використовується для оптимального вибору параметрів апроксимуючої функції?

Наприклад, можна використовувати метод найменших квадратів.

1. Який принцип роботи алгоритму методу найменших квадратів?

Цей метод оснований на мінімізації суми квадратів відхилень деяких функцій від шуканих змінних та дозволяє підібрати аналітичну функцію, що проходить настільки близько до експериментальних точок, наскільки це можливо

1. Які існують види експоненційної регресії?

Експоненційна регресія з двома параметрами:

та експоненційна регресія з трьома параметрами:

1. Яка вбудована функція бібліотек Python дозволяє отримати коефіцієнти експоненційної регресії?

Функція *curve\_fit()* бібліотеки SciPy, що в якості параметрів приймає вигляд апроксимуючої функції, а також вектори X та Y.

1. Яка вбудована функція GNU Octave дозволяє отримати інтерполяцію функції степенним поліномом степені

, де X та Y – початкові точки дискретно заданої функції (n точок), а – степінь інтерполюючого полінома

1. Як оцінити відносну похибку отриманої апроксимації?

За допомогою формули:

# **2.6 Опис програмних засобів**

В процесі реалізації поставленої задачі були використані наступні вбудовані функції бібліотек numpy (np), scipy, matplotlib мови Python:

1. - повертає числове значення, що відповідає кількості елементів вектора ;
2. – додає елемент до масиву;
3. *–* повертає значення натурального логарифму числа ;
4. - повертає вектор, що є розв’язком системи ;
5. - повертає значення експоненти в степені
6. – виконує побудову точкового графіку векторів X та Y;
7. – виконує побудову графіку функції, що передається параметром *func()* на відрізку, що заданий параметром *np.arange()*;
8. – повертає коефіцієнти експоненційної регресії, в якості параметрів приймає вигляд апроксимуючої функції, а також початкові точки X та Y.

В процесі реалізації поставленої задачі були використані наступні вбудовані функції GNU Octave:

1. – повертає числове значення, що відповідає кількості елементів вектора ;
2. *–* повертає значення натурального логарифму числа ;
3. - повертає вектор, що є розв’язком системи

;

1. – повертає значення експоненти в степені
2. – повертає модуль числа;
3. – дозволяє виконати побудову графіку функції.
4. , - повертає інтерполяційний поліном, де X та Y – початкові точки дискретно заданої функції (n точок), а – степінь інтерполюючого полінома

# **3 ВИСНОВКИ**

В даній лабораторній роботі було розроблено програмне забезпечення, що реалізує апроксимацію дискретно заданої функції експоненційною функцією з трьома параметрами за допомогою методу найменших квадратів. В результаті виконання лабораторної роботи було створено дві програми з використанням мови Python, а також за допомогою програмного пакету математичних обчислень GNU Octave. В кожному з цих середовищ розробки метод був реалізований як з використанням вбудованих функцій, так і за допомогою самостійно написаного алгоритму. В обох програмах обчислюється відносна похибка апроксимації, а також оцінюється правильність розрахунків за допомогою контрольного прикладу.

# **4 СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. [Линник Ю. В](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D0%BD%D0%B8%D0%BA,_%D0%AE%D1%80%D0%B8%D0%B9_%D0%92%D0%BB%D0%B0%D0%B4%D0%B8%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B8%D1%87_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA)). Метод наименьших квадратов и основы математико-статистической теории обработки наблюдений. — 2-е изд. — М., 1962.
2. Лоран, П. Ж. Аппроксимация и оптимизация. — М.: Мир, 1975. — 496 с.
3. Е. Р. Алексеев, О. В. Чеснокова. Введение в Octave для инженеров и математиков — С.326-335.

# **ДОДАТКИ**

# **Додаток А (код програми GNU Octave)**

#control example

##X = [0 1 2 3 4];

##Y = [0 1.18 1.9 2.33 2.59];

X = [0 1 2 3 4 5];

Y = [1.2 3.7 4.8 14.2 24.1 58.5];

n = length(X);

SumFunction = 0;

SumXt = 0;

SumXt2 = 0;

SumXF = 0;

SumY = 0;

SumA = 0;

for i=1:(n-1)

F(i) = log((Y(i+1)-Y(i))/(X(i+1)-X(i)));

SumFunction = SumFunction + F(i);

Xt(i) = (X(i+1)+X(i))/2;

SumXt = SumXt + Xt(i);

Xt2(i) = (Xt(i))^2;

SumXt2 = SumXt2 + Xt2(i);

XF(i) = Xt(i)\*F(i);

SumXF = SumXF + XF(i);

end;

M = [(n-1) SumXt; SumXt SumXt2];

m = [SumFunction SumXF];

x = inv(M)\*m';

disp('k exp regression:')

B = -x(2)

A = exp(x(1))/B

for i=1:n

Yt(i) = A\*(1-exp(-B\*X(i)));

SumY = SumY + (Y(i)-Yt(i));

end;

C = SumY/n

y = A\*(1 - exp(-B\*X)) + C;

%r = 0;

%mistake

for i=1:n

SumA = SumA + abs((Y(i)-y(i))/Y(i));

Ab(i) = abs((Y(i)-y(i))/Y(i));

% if Ab(i) > r

% r = Ab(i);

% m = i

% end;

end;

%r

%m

disp('mistake(%):')

Abs = (SumA/(n-1))\*100

##%graphic of function

##wind = figure();

##x = 0:0.1:5;

##z = A\*(1 - exp(-B\*x)) + C;

##plot(X, Y,'\*',x, z);

##pause(5);

##delete(wind);

##

##%graphic build with internal function

##wind2 = figure();

##l = polyfit(X, Y, (n-1));

##y1 = polyval(l, x);

##plot(X, Y, '\*', x , y1);

##pause(10);

##delete(wind2);

wind3 = figure();

x = 0:0.1:5;

z = A\*(1 - exp(-B\*x)) + C;

l = polyfit(X, Y, (n-1));

y1 = polyval(l, x);

plot(X, Y, '^' ,x , z, 'm', x, y1, 'g')

xlabel('X');

ylabel('Y');

legend("margin - external, \ngreen - internal");

grid on;

# **Додаток Б (код програми Python)**

**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**from** scipy.optimize **import** curve\_fit  
*# #  
# Y = [0, 1.18, 1.9, 2.33, 2.59]  
# X= [0, 1, 2, 3, 4]*X = [0, 1, 2, 3, 4, 5]  
Y = [1.2, 3.7, 4.8, 14.2, 24.1, 58.5]  
  
n = len(X)  
  
F = []  
SumF = 0  
  
**for** i **in** range(n-1):  
 F.append(np.log((Y[i+1]-Y[i])/(X[i+1]-X[i])))  
 SumF = SumF + F[i]  
print(F)  
  
Xt = []  
SumXt = 0  
  
**for** i **in** range(n-1):  
 Xt.append((X[i+1]+X[i])/2)  
 SumXt = SumXt + Xt[i]  
print(Xt)  
  
Xt2 = []  
SumXt2 = 0  
  
**for** i **in** range(n-1):  
 Xt2.append((Xt[i])\*\*2)  
 SumXt2 = SumXt2 + Xt2[i]  
print(Xt2)  
  
XF = []  
SumXF = 0  
  
**for** i **in** range(n-1):  
 XF.append(Xt[i]\*F[i])  
 SumXF = SumXF + XF[i]  
print(XF)  
  
A = np.array([[n-1, SumXt], [SumXt, SumXt2]])  
b = np.array([SumF, SumXF])  
  
solve = np.linalg.solve(A, b)  
print(solve)  
  
Bk = -solve[1]  
print(Bk)  
  
Ak = np.exp(solve[0])/Bk  
print(Ak)  
  
Yt = []  
SumY = 0  
  
**for** i **in** range(n):  
 Yt.append(Ak\*(1 - np.exp(-Bk\*X[i])))  
 SumY = SumY + (Y[i] - Yt[i])  
  
  
Ck = SumY/n  
print(Ck)  
  
**def** exp\_func(x, a, b, c):  
 **return** a \*(1 - np.exp(-b \* x)) + c  
  
Y\_exp = []  
**for** i **in** X:  
 Y\_exp.append(exp\_func(i, Ak, Bk, Ck))  
  
  
SumError = 0  
**for** i **in** range(1, n):  
 SumError = SumError + np.fabs((Y[i]-Y\_exp[i])/Y[i])  
Abs\_error = 100\*SumError/(n-1)  
print(**'Approximation error: '**, round(Abs\_error,5),**'%'**)  
  
plt.scatter(X,Y)  
plt.plot(np.arange(0, 5.1, 0.1), exp\_func(np.arange(0, 5.1, 0.1), Ak, Bk, Ck), color = **'g'**, label=**"external"**)  
*# plt.show()***def** func\_exp(x, a, b, c):  
 *# c = 0* **return** a \* np.exp(b \* x) + c  
  
  
  
**if** n==6:  
 **def** exponential\_regression(x\_data, y\_data):  
 popt, pcov = curve\_fit(func\_exp, x\_data, y\_data)  
 *# popt, pcov = curve\_fit(func\_exp, x\_data, y\_data, p0=(-1, 0.01, 1)* print(popt)  
 puntos = plt.plot(x\_data, y\_data, **'x'**, color=**'xkcd:maroon'**, label=**"data"**)  
 curva\_regresion = plt.plot(x\_data, func\_exp(x\_data, \*popt), color=**'xkcd:teal'**,  
 label=**"internal: {:.3f}, {:.3f}, {:.3f}"**.format(\*popt))  
 plt.legend()  
 plt.show()  
 **return** func\_exp(x\_data, \*popt)  
  
  
 exponential\_regression(np.array(X), np.array(Y))  
  
**else**:  
 **def** exponential\_regression1(x\_data, y\_data):  
 popt, pcov = curve\_fit(func\_exp, x\_data, y\_data, p0=(-1, 0.01, 1))  
 print(popt)  
 puntos = plt.plot(x\_data, y\_data, **'x'**, color=**'xkcd:maroon'**, label=**"data"**)  
 curva\_regresion = plt.plot(x\_data, func\_exp(x\_data, \*popt), color=**'xkcd:teal'**,  
 label=**"internal: {:.3f}, {:.3f}, {:.3f}"**.format(\*popt))  
 plt.legend()  
 plt.show()  
 **return** func\_exp(x\_data, \*popt)  
 exponential\_regression1(np.array(X), np.array(Y))

# **Додаток B (результати виконання )**

Octave:

>> k exp regression:

B = -0.74408

A = -1.2866

C = 2.3646

mistake(%):

Abs = 33.010

A close up of a device

Description automatically generated

Python:

Approximation error: 13.60069 % A close up of a map

Description automatically generated