Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики і обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №1.1-1.2**

**з курсу: «Інтелектуальні вбудовані системи»**

*Виконав:*

студент групи ІП-84

Колмагін Д. О.

Залікова книжка №8411

*Перевірив:*

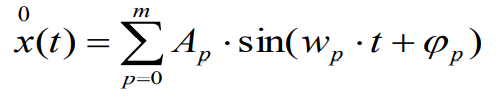
Регіда П. Г.

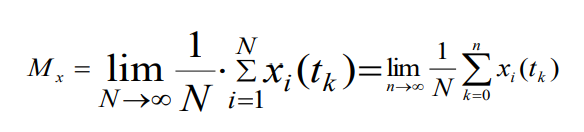
Київ 2020 р.

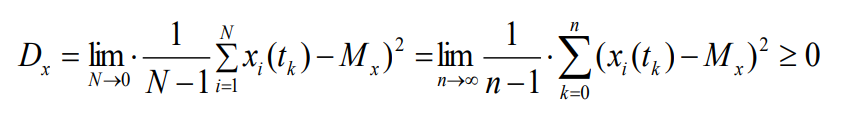
**Основні теоретичні відомості**

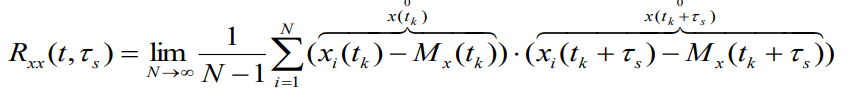
СРЧ забезпечує контроль за зміною параметрів зовнішнього середовища і в ряді випадків забезпечує управління параметрами середовища через деякі впливу на неї. Параметри середовища представляються деякою зміною фізичного середовища. При вимірах фізичного параметра ми отримуємо певний електричний сигнал на вході вимірювального датчика. Для подання такого електричного сигналу можна використовувати різні моделі. Найкращою моделлю досліджуваного сигналу є відповідна математична інтерпретація випадкового процесу. Випадковий сигнал або процес завжди представляється деякою функцією часу x(t) , значення якої не можна передбачити з точністю засобів вимірювання або обчислень, які б кошти моделі ми не використовували.

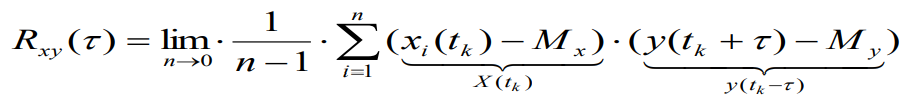
Для випадкового процесу його значення можна передбачити лише основні його характеристики: математичне сподівання M (t) x , дисперсію D (t) x , автокореляційну функцію R (t,τ ),R (t,τ ) xx xy .











**Завдання на лабораторну роботу**

Згенерувати випадковий сигнал по співвідношенню відповідно до

варіанту по таблиці і розрахувати його математичне сподівання, дисперсію і автокореляцію, а також знайти кореляцію. Розробити відповідну програму і вивести отримані значення і графіки відповідних параметрів.

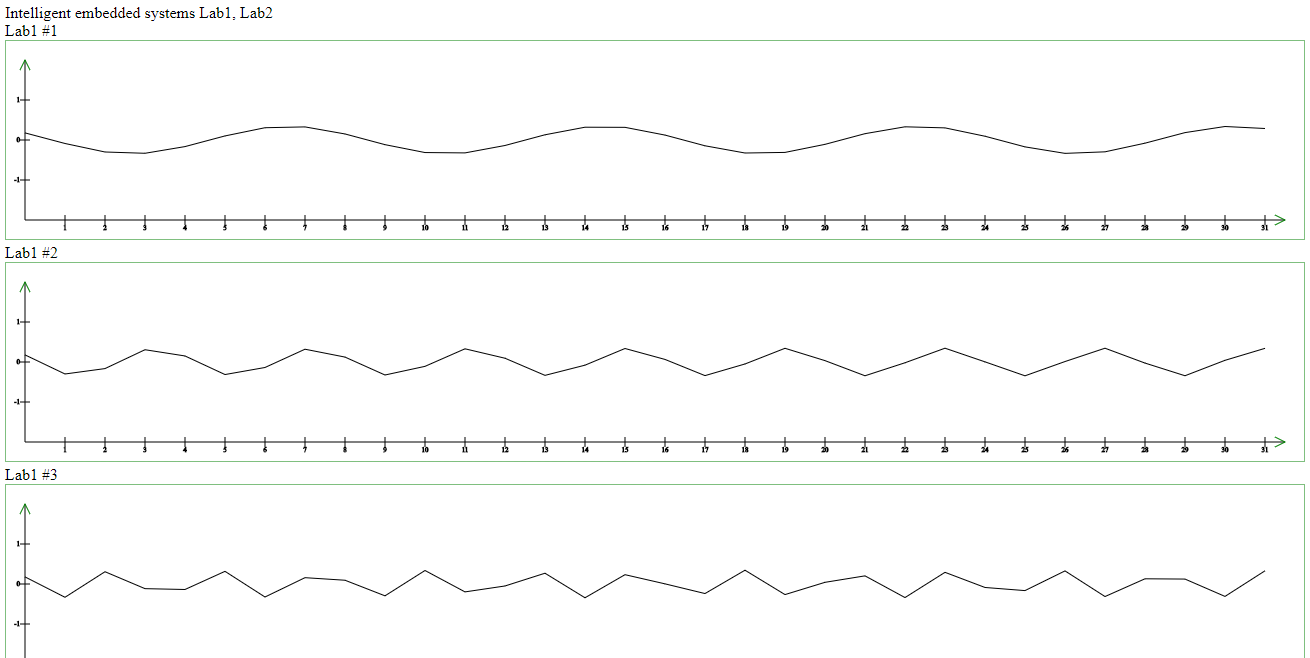
**Варіант-11**

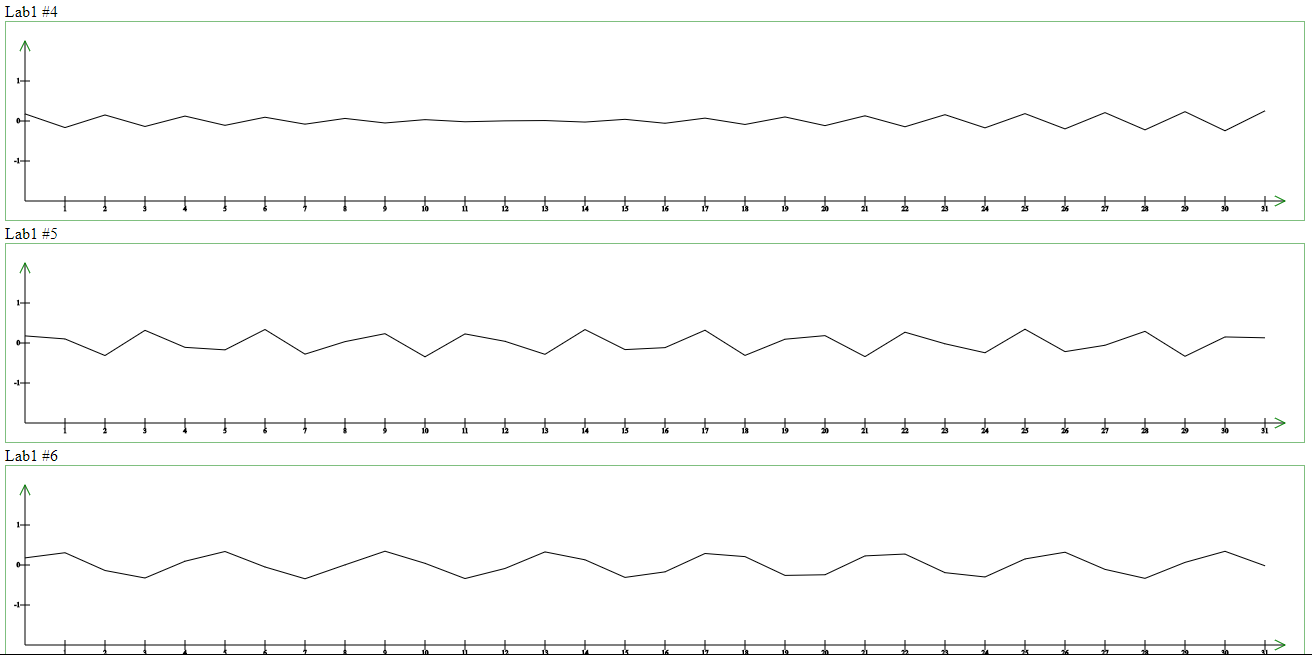
|  |  |  |
| --- | --- | --- |
| Число гармонік в сигналі n | Гранична частота, ωгр | Кількість дискретних відліків, N |
| 10 | 1500 | 256 |

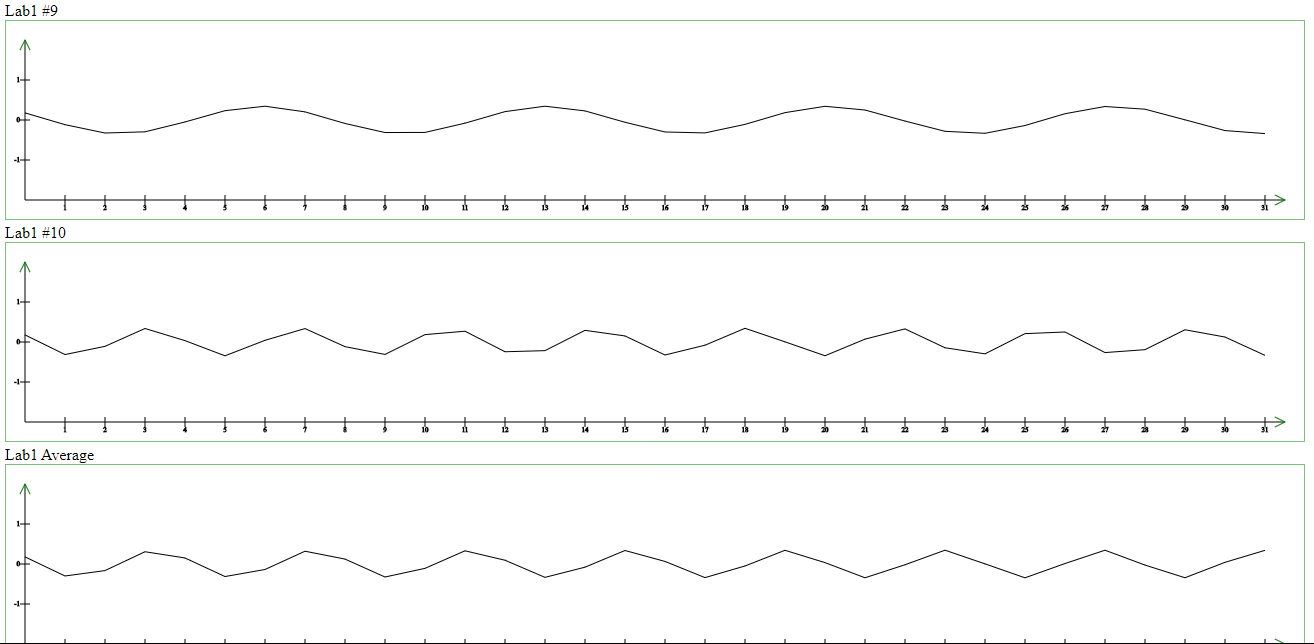
**Лістинг програми**

const getTimeCalc = (tick = () => 0, {startN, stepN, finishN, fileName = '',}) => {  
 const O = []  
 const csv = [['N', 'Time',],]  
 for (let N = startN; N <= finishN; N += stepN) {  
 let time = tick(N)  
 O.push(time)  
 csv.push(['' + N, time])  
 }  
 exportToCsv(fileName, csv)  
 return O  
}  
  
const getFuncFromArray = (array) => (x) => array[x]  
  
// Lab1  
const ***A*** = 0.345248  
const ***fi*** = 0.541387  
const ***maxW*** = 1500  
const ***n*** = 10  
const ***W*** = []  
for (let i = 1; i <= ***n***; i++) ***W***.push(***maxW*** / ***n*** \* i)  
const ***N*** = 256  
const ***arrayW*** = []  
for (let t = 0; t < ***N***; t++) ***arrayW***.push(***W***.map( w => ***A*** \* ***Math***.sin(w \* t + ***fi***) ))  
const ***xT*** = []  
for (let t = 0; t < ***N***; t++) ***xT***.push(***arrayW***[t].reduce((acc, cur) => acc + cur))  
  
const ***MX*** = ***xT***.reduce((acc, cur) => acc + cur) / ***N***const ***arrayDx*** = ***xT***.map(x => (x - ***MX***) \*\* 2)  
const ***DX*** = ***arrayDx***.reduce((acc, cur) => acc + cur) / (***N*** - 1)  
  
const ***lab1TimeCalc*** = (() => {  
 const tick = (N) => {  
 const start = new ***Date***()  
 for (let t = 0; t < N; t++) ***W***.map( w => ***A*** \* ***Math***.sin(w \* t + ***fi***) )  
 const end = new ***Date***()  
 return end.getTime() - start.getTime()  
 }  
  
 return getTimeCalc(tick, {  
 startN: 100000,  
 stepN: 10000,  
 finishN: 1000000,  
 fileName: 'Lab1.csv',  
 })  
  
 // return [  
 // 104, 130, 128, 138, 120, 144, 150, 170, 170, 181, 170,  
 // 185, 206, 204, 222, 211, 229, 245, 252, 261, 267, 289,  
 // 292, 323, 393, 350, 350, 355, 380, 366, 347, 379, 373,  
 // 391, 393, 423, 425, 420, 439, 448, 456, 471, 455, 480,  
 // 482, 494, 498, 508, 529, 563, 542, 589, 591, 568, 563,  
 // 571, 603, 587, 618, 608, 657, 622, 665, 634, 668, 682,  
 // 666, 700, 724, 701, 715, 728, 733, 743, 727, 748, 814,  
 // 775, 777, 802, 805, 830, 851, 821, 876, 848, 849, 904,  
 // 902, 868, 895  
 // ]  
})()  
  
const ***getLab1AllFunc*** = ***arrayW***[0].map((\_, i) => getFuncFromArray(***arrayW***.map((\_, newI) => ***arrayW***[newI][i])))  
const ***getLab1AverageFunc*** = getFuncFromArray(***arrayW***.map(val => val[1]))  
const ***getLab1On*** = getFuncFromArray(***lab1TimeCalc***)  
  
// Lab2  
const ***arrayRxx*** = ***xT***.map((\_, Tau) => ***xT***.map((\_, t) => (***xT***[t] - ***MX***) \* (***xT***[(t + Tau) % ***N***] - ***MX***)))  
const ***Rxx*** = ***arrayRxx***[1].reduce((acc, cur) => acc + cur) / (***N*** - 1)  
  
const ***lab2TimeCalcRxx*** = (() => {  
 const tick = (NN) => {  
 const start = new ***Date***()  
 for (let t = 0; t < NN; t++) ***xT***.map((\_, Tau) => ***xT***.map((\_, t) => (***xT***[t] - ***MX***) \* (***xT***[(t + Tau) % ***N***] - ***MX***)))  
 const end = new ***Date***()  
 return end.getTime() - start.getTime()  
 }  
 return getTimeCalc(tick, {  
 startN: 10000,  
 stepN: 1000,  
 finishN: 100000,  
 fileName: 'Lab2Rxx.csv',  
 })  
 // return [  
 // 235, 206, 150, 157, 165, 163, 177, 202, 293, 223,  
 // 244, 223, 255, 238, 238, 249, 268, 335, 279, 289,  
 // 299, 307, 317, 323, 339, 349, 357, 404, 409, 441,  
 // 491, 460, 457, 481, 469, 498, 519, 512, 556, 506,  
 // 615, 615, 529, 564, 569, 602, 624, 591, 684, 635,  
 // 641, 647, 755, 720, 928, 766, 1689, 1099, 795, 809,  
 // 871, 837, 863, 859, 852, 872, 907, 945, 940, 933,  
 // 944, 946, 1047, 956, 1050, 1065, 1051, 1187, 981, 994,  
 // 998, 1095, 1019, 1023, 1066, 947, 961, 1007, 980, 986,  
 // 994  
 // ]  
})()  
  
const generateY = () => {  
 const A = 0.690507  
 const fi = 0.730429  
 const maxW = 3000  
 const n = 3  
 const W = []  
 for (let i = 1; i <= n; i++) W.push(maxW / n \* i)  
 const N = 32  
 const arrayW = []  
 for (let t = 0; t < N; t++) arrayW.push(W.map( w => A \* ***Math***.sin(w \* t + fi) ))  
 const yT = []  
 for (let t = 0; t < N; t++) yT.push(arrayW[t].reduce((acc, cur) => acc + cur))  
 const MY = yT.reduce((acc, cur) => acc + cur) / N  
 const arrayRxy = ***xT***.map((\_, Tau) => ***xT***.map((\_, t) => (***xT***[t] - ***MX***) \* (yT[(t + Tau) % N] - MY)))  
 const Rxy = arrayRxy[1].reduce((acc, cur) => acc + cur) / (N - 1)  
 const lab2TimeCalcRxy = (() => {  
 const tick = (NN) => {  
 const start = new ***Date***()  
 for (let t = 0; t < NN; t++) ***xT***.map((\_, Tau) => ***xT***.map((\_, t) => (***xT***[t] - ***MX***) \* (yT[(t + Tau) % N] - MY)))  
 const end = new ***Date***()  
 return end.getTime() - start.getTime()  
 }  
 return getTimeCalc(tick, {  
 startN: 10000,  
 stepN: 1000,  
 finishN: 100000,  
 fileName: 'Lab2Rxy.csv',  
 })  
 // return [  
 // 295, 240, 147, 151, 164, 179, 201, 198, 210, 238,  
 // 348, 301, 245, 266, 299, 288, 307, 305, 313, 322,  
 // 332, 348, 369, 433, 473, 478, 719, 898, 462, 821,  
 // 612, 636, 699, 858, 605, 544, 545, 526, 554, 634,  
 // 650, 582, 582, 629, 599, 616, 622, 632, 673, 674,  
 // 733, 758, 761, 697, 776, 715, 778, 743, 847, 776,  
 // 799, 957, 808, 862, 832, 828, 914, 888, 946, 874,  
 // 883, 902, 1049, 974, 1017, 943, 964, 969, 1057, 1088,  
 // 992, 1008, 1016, 1022, 1078, 1048, 1059, 1144, 1157, 1092,  
 // 1101  
 // ]  
 })()  
 return {arrayRxy, Rxy, lab2TimeCalcRxy}  
}  
  
const {***arrayRxy***, ***Rxy***, ***lab2TimeCalcRxy***} = generateY()  
  
const ***getLab2RxxFunc*** = getFuncFromArray(***arrayRxx***[1])  
const ***getLab2RxyFunc*** = getFuncFromArray(***arrayRxy***[1])  
const ***getLab2OnRxx*** = getFuncFromArray(***lab2TimeCalcRxx***)  
const ***getLab2OnRxy*** = getFuncFromArray(***lab2TimeCalcRxy***)

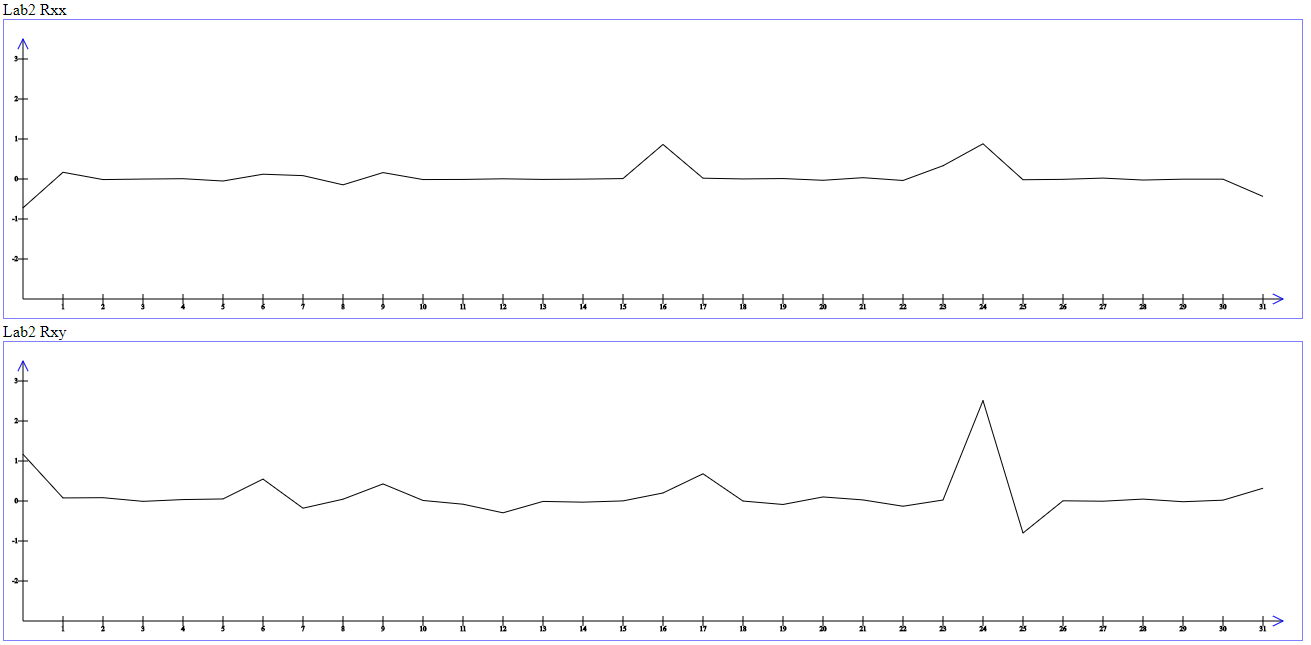
**Результати роботи програми**

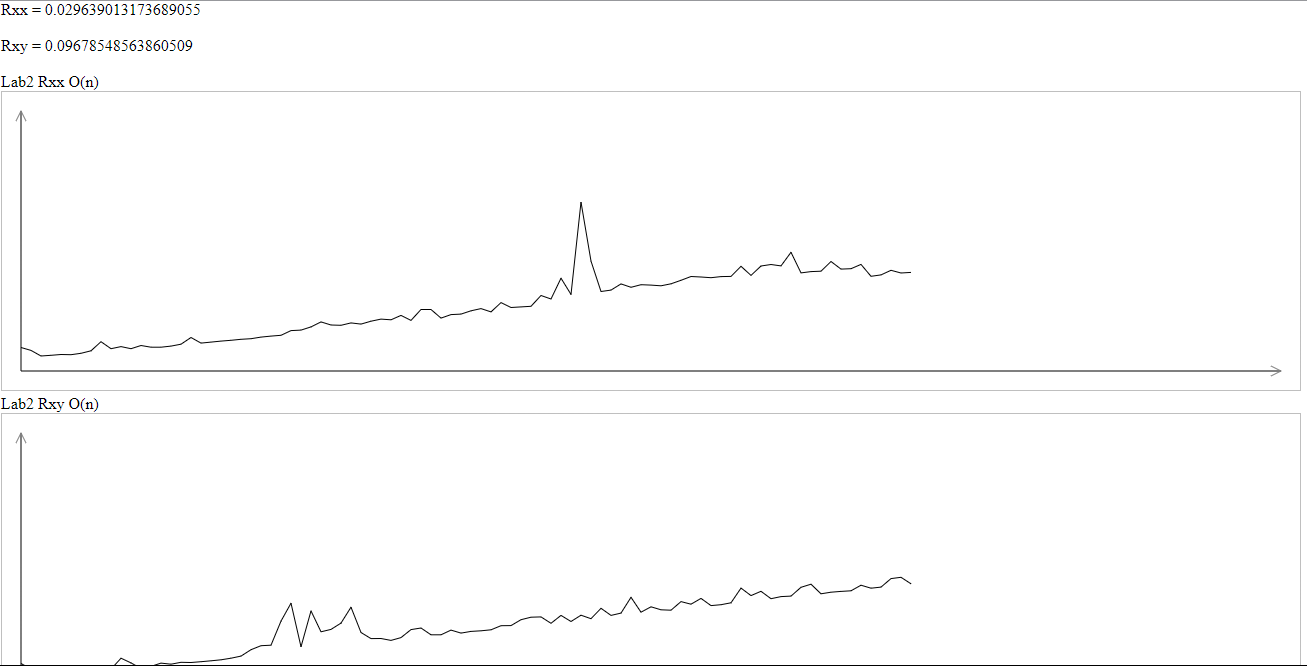
******

******

******

******

******

******

**Висновки**

В даній лабораторній роботі ми дослідили принципи генерації та побудови графіків випадкових величин. На основі знайдених даних знайшли математичне очікування і дисперсію. Побудували графіки автокореляції та кореляції, а також складності алгоритму для генерації чисел і кореляції.

Дані знання знадобляться нам у нашій подальшій кар’єрі.