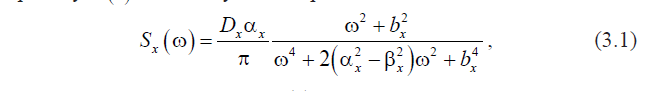
**Лабораторна робота № 3**

Моделювання стаціонарного випадкового процесу

**Мета роботи:**отримати практичні навички моделювання стаціонарного випадкового процесу з використанням комп’ютера.

**Завдання:**Спектральна щільність стаціонарного випадкового процесу *x(t)* має наступний вираз:

де *Dx* – дисперсія процесу *x(t)*; *αx* і *βx* – відповідно коефіцієнт затухання і середня частота кореляційної функції процесу *x(t)*; .

Виконати моделювання стаціонарного випадкового процесу *x(t)* на інтервалі часу [0, 300] секунд. Представити графіки *x(t)* і . Навести значення кроку інтегрування за часом Δt та точкову оцінку середньоквадратичного відхилення . Дані для моделювання: x(0) =; ε=0,05; = 0,8;

**Етапи виконання роботи:**

Виконання лабораторної роботи включає в себе наступні етапи.

1) Визначення початкового значення кроку інтегрування за часом tΔ.

2) Моделювання стаціонарного випадкового процесу *x(t)* за різницевими рівняннями. Визначення значення кроку інтегрування за часом tΔ та точкової оцінки середньоквадратичного відхилення за результатами моделювання.

3) Побудова графіків *x(t)* і .

4) Формулювання висновків за результатами виконання лабораторної роботи.

**Виконання роботи:**

Для моделювання стаціонарного випадкового процесу x(t) з використанням комп’ютера було розроблено програмне забезпечення. Текст програми представлено в додатку А.

Моделюванняпроводимо за наступними різницевими рівняннями:

|  |  |
| --- | --- |
| *;* | (3.2) |
| *,* | (3.3) |

де – ордината білого шуму в момент часу , ; значення випадкової величини з розподілом Гауса з нульовим математичним сподіванням та одиничною дисперсією, ; N0 – інтенсивність білого шуму, для наших підрахунків N0 = 1.

Точкову оцінку середньоквадратичного відхилення розрахуємо за наступною формулою:

|  |  |
| --- | --- |
|  | (3.4) |

Початкове значення кроку інтегрування за часом беремо 0,1.

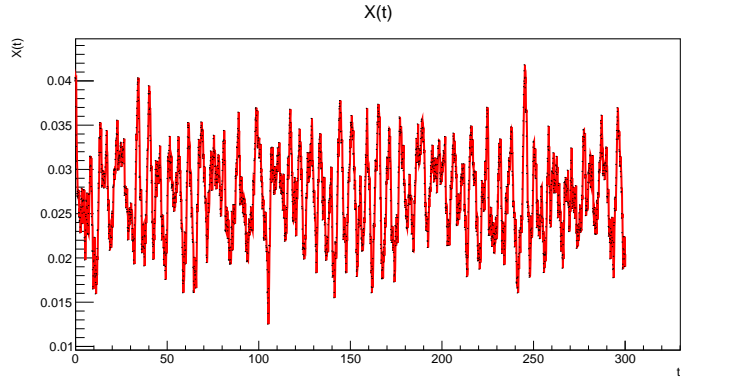
Для знаходження чисельного рішення за різницевими рівняннями (3.2) та (3.3) застосуємо такий критерій збіжності:

|  |  |
| --- | --- |
| , | (3.5) |

Де задана відносна похибка; точкова оцінка середнього квадратичного відхилення процесу *x(t).*

Графік *x(t)* представлено на рис.1.

Рисунок 1 – Графік *x(t)*



Графік представлено на рис.2.

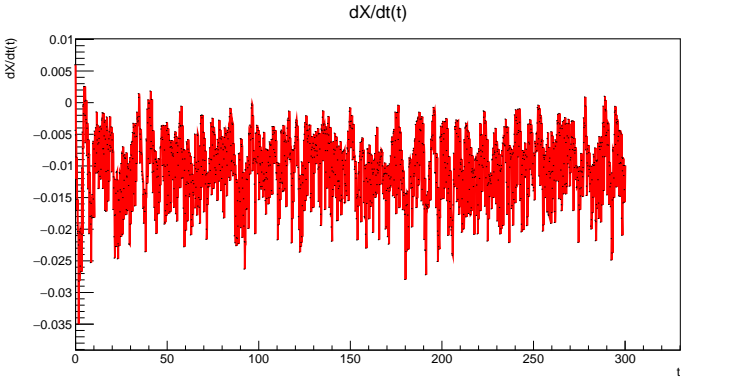


Рисунок 2 – Графік

За результатами моделювання отримали наступні значення:

* крок інтегрування за часом = 0,173;
* середньоквадратичне відхилення = 0,04198746.

**Висновки:**

Виконуючи лабораторну роботу отримав практичні навички моделювання стаціонарного процесу з використанням комп’ютера. Виконав моделювання стаціонарного випадкового процесу *x(t)* на інтервалі часу [0, 300] секунд та представив графіки *x(t)* і .

#define alpha 0.3

#define beta 0.9

#define sigma 0.04

#define epsilon 0.05

#define N0 1

#include <iostream>

#include <cmath>

#include <vector>

#include <random>

#include <fstream>

std::default\_random\_engine generator(100);

std::normal\_distribution<double> distribution(0,1.0);

double Dx = sigma\*sigma;

double x0 = sigma;

double dx0 = 0;

double b = sqrt(alpha\*alpha+beta\*beta);

double dt = 0.3;

vector<double> T = {};

vector<double> X1 = {};

vector<double> X2 = {};

vector<double> n = {};

double gaussian(double x, double mu=0, double sig=1)

{

return 1/sqrt(2\*M\_PI\*sig\*sig)\*exp(-1\*(x-mu)\*(x-mu)/2/sig/sig);

}

double mean(vector<double> A)

{

if(A.size() != 0)

{

double mean=0;

for (double a : A)

{

mean+=a;

}

return mean=mean/A.size();

}

else

{

return 0;

}

}

double stdDev(vector<double> A)

{

if(A.size() != 0)

{

double stdDev=0;

double m = mean(A);

for (double a : A)

{

stdDev+=(a-m)\*(a-m);

}

return sqrt(stdDev/(A.size()-1));

}

else

{

return 0;

}

}

void script() {

double t, X1prev, X2prev, ni;

do

{

t=0;

T.clear();

X1.clear();

X2.clear();

n.clear();

n.push\_back(gaussian(distribution(generator),0,1)\*sqrt(N0/dt));

T.push\_back(0);

X1.push\_back(x0);

X2.push\_back(dx0-sqrt(2\*Dx\*alpha)\*n.back());

while(t<=300)

{

t+=dt;

X1prev = X1.back();

X2prev = X2.back();

ni = gaussian(distribution(generator))\*sqrt(N0/dt);

n.push\_back(ni);

T.push\_back(t);

X1.push\_back(X1prev+(X2prev+sqrt(2\*Dx\*alpha)\*ni)\*dt);

X2.push\_back(X2prev+(sqrt(2\*Dx\*alpha)\*(b-2\*alpha)\*ni-2\*alpha\*X2prev-b\*b\*X1prev)\*dt);

}

dt\*=0.95;

if(fabs(sigma-stdDev(X1)/sigma)<epsilon) break;

}

while(true);

ofstream file("file.txt");

for(int i=0; i<T.size(); i++)

{

file<<T[i]<<" "<<X1[i]<<" "<<X2[i]+sqrt(2\*Dx\*alpha)\*n[i]<<endl;

}

}