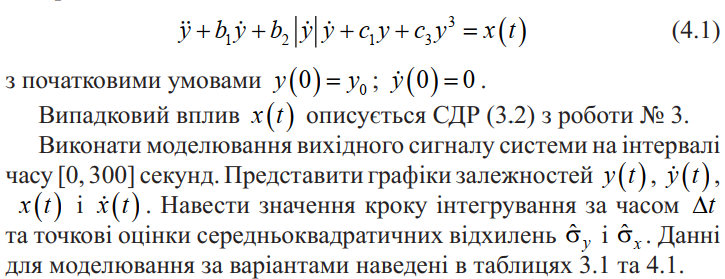
**Лабораторна** робота №4 **Моделювання нелінійної стохастичної диференціальної системи**

**Мета роботи**: отримати практичні навички моделювання нелінійної стохастичної диференціальної системи з використанням комп’ютера.

**Завдання**

Вихідний сигнал нелінійної СДС описуються наступним рівнянням



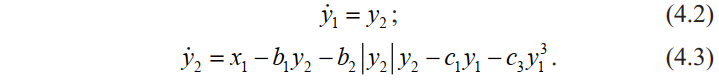


Інші параметри такі: c1 =1 і c3 = −1 (для варіантів з 1 по 14); c1 =1,1 і c3 = −1,1 (для варіантів з 15 по 28); ; ε = 0,05 . Також виконати моделювання вихідного сигналу системи на інтервалі часу [0, 300] секунд у разі, коли у якості моделі випадкового впливу використовується білий шум. Представити графіки , та . Навести значення кроку інтегрування за часом ∆t та точкову оцінку середньоквадратичного відхилення . Зробити висновки щодо отриманих результатів.

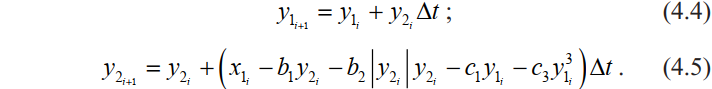
Хід роботи

Стохастичною диференціальною системою (СДС) прийнято називати таку систему, поведінка якої описується стохастичним диференціальним рівнянням (СДР). Велика кількість динамічних об’єктів у техніці є саме такими системами.

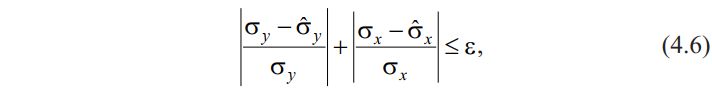
Аналітичного рішення СДР (4.1) не існує. Тому ми будемо знаходити чисельне рішення СДР (4.1). Для цього спочатку перетворимо СДР (4.1) у систему рівнянь 1-го порядку шляхом введеннядвох нових змінних = та



Тут також введена нова змінна . На основі системи рівнянь (4.2) і (4.3) будуємо систему різницевих рівнянь за методом Ейлера



Моделювання вихідного сигналу системи у разі, коли випадковий вплив описується СДР (3.2), здійснюється за різницевими рівняннями (4.4), (4.5), (3.5) і (3.6). Чисельне рішення за різницевими рівняннями (4.4), (4.5), (3.5) і (3.6) збігається у середньоквадратичному. У цьому разі може бути застосований такий критерій збіжності:



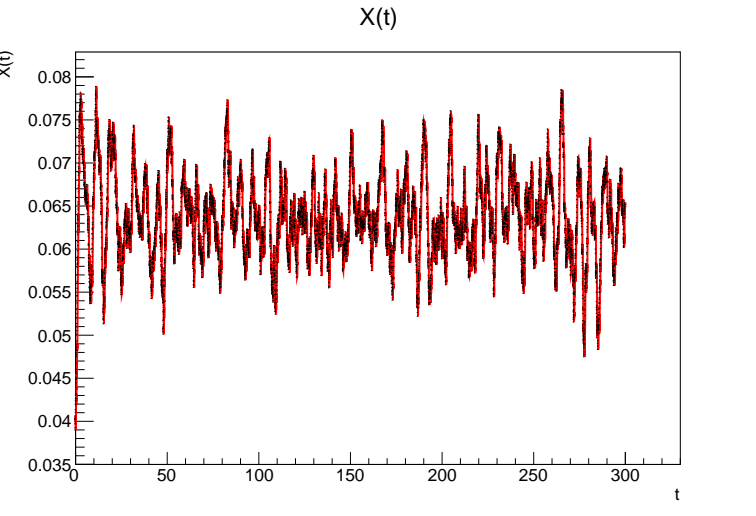
де ε – задана відносна похибка; та – точкові оцінки середньоквадратичних відхилень і процесів та відповідно.

Результатом моделювання випадкового процесу за різницевими рівняннями (4.4), (4.5), (3.5) і (3.6) буде чисельне рішення системи СДР або інакше – послідовності значень і , які задовольняють критерію (4.6). Як правило, чисельне рішення за різницевими рівняннями (4.4), (4.5), (3.5) і (3.6) вдається отримати після декількох прогонів. При цьому для кожного нового прогону слід обирати крок інтегрування за часом ∆t, меншим за попередній приблизно на 5%. Зазначимо, при першому прогоні потребує визначення початкове значення кроку інтегрування за часом ∆t. Його рекомендується обирати таким чином: ∆t повинен бути значно більшим за час кореляції та значно меншим за час постійної системи, яка моделюється.

Моделювання вихідного сигналу системи у разі, коли у якості моделі випадкового впливу використовується білий шум, здійснюється за різницевими рівняннями (4.4) і (4.5). При цьому інтенсивність білого шуму (постійну ординату його спектральної щільності) зв’язують зі спектральною щільністю (3.1) випадкового впливу , наприклад, прирівнюючи їх дисперсії (площі спектральних щільностей). Чисельне рішення за різницевими рівняннями (4.4) і (4.5) збігається у середньоквадратичному. У цьому разі також може бути застосований критерій збіжності (4.6).

За результатами проведення підбору початкового кроку було визначено наступне прийнятне значення .

За результатами моделювання отримуємо наступні графіки. Графік наведено на рисунку 1:

Рисунок 1 – Графік функції

Графік наведено на рисунку 2:

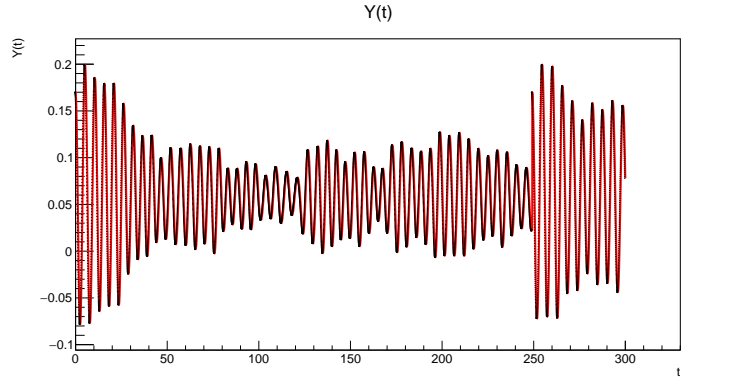


Рисунок 2 – Графік функції

Графік наведено на рисунку 3:

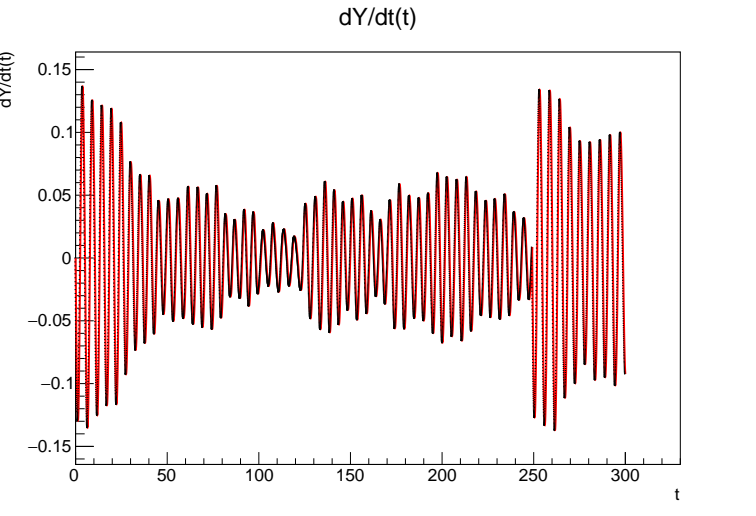


Рисунок 3 – Графік функції

При завершенні ітераційного процесу моделювання нелінійної стохастичної диференціальної системи за різницевими рівняннями отримано наступне значення кроку інтегрування за часом , значення точкової оцінки середньоквадратичного відхилення та .

**Висновки:** за результатами виконання лабораторної роботи було виконано моделювання нелінійної стохастичної диференціальної системиз використанням чисельних методів.

Отримані значення та побудовані за ними графіки дозволяють зробити висновок про успішність нелінійної стохастичної диференціальної системи.

#define alpha 0.3

#define beta 0.9

#define b1 0.04

#define b2 0.2

#define c1 1

#define c3 -1

#define sigmaX 0.04

#define sigmaY 0.17

#define epsilon 0.05

#define N0 1

#include <iostream>

#include <cmath>

#include <vector>

#include <random>

#include <fstream>

std::default\_random\_engine generator;

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

double Dx = sigmaX\*sigmaX;

double x0 = sigmaX;

double dx0 = 0;

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

double dt = 0.025;

double Y0 = sigmaY;

double dY0 = 0;

vector<double> T = {};

vector<double> X1 = {};

vector<double> X2 = {};

vector<double> Y1 = {};

vector<double> Y2 = {};

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() {

do

{

double 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());

Y1.push\_back(Y0);

Y2.push\_back(dY0);

while(t<=300)

{

double X1prev = X1.back();

double X2prev = X2.back();

double ni = gaussian(distribution(generator),0,1)\*sqrt(N0/dt);

double Y1prev = Y1.back();

double Y2prev = Y2.back();

t+=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);

Y1.push\_back(Y1prev+Y2prev\*dt);

Y2.push\_back(Y2prev+(X1prev-b1\*Y2prev-b2\*fabs(Y2prev)\*Y2prev-c1\*Y1prev-c3\*Y1prev\*Y1prev\*Y1prev)\*dt);

}

dt\*=0.95;

if(fabs(sigmaX-stdDev(X1)/sigmaX)+fabs(sigmaY-stdDev(Y1)/sigmaY)<epsilon) break;

}

while(true);

ofstream file("file.txt");

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

{

file<<T[i]<<" "<<Y1[i]<<" "<<Y2[i]<<endl;

}

}