Лабораторна робота 10
Вивчення інструментів моделювання в Python

10.1. Мета роботи

Ознайомлення з інструментами Python, що можуть бути застосовані у комп'ютерному моделюванні.

10.2. Методичні рекомендації щодо організації самостійної роботи

За темою лабораторної роботи студент повинен: *знати* основні закони розподілу випадкових величин та основні поняття лінійної алгебри та теорії диференціальних рівнянь; у*міти* застосовувати математичні бібліотеки Python для розв'язання задач лінійної алгебри, задачі Коші для систем звичайних диференціальних рівнянь, генерації значень випадкових величин за різними розподілами [17 – 19].

Основною бібліотекою Python для розв'язання математичних задач є бібліотека **scipy**. Розглянемо декілька розділів цієї бібліотеки.

10.2.1. Функції для обробки випадкових величин (scipy.stats)

У бібліотеці **scipy.stats** для Python є багато функцій, призначених для обробки випадкових величин. Наведемо кілька з них для законів розподілу: рівномірний, нормальний, експонентний, Пуассона.

Функції для **рівномірного закону розподілу (uniform):**

uniform.rvs(loc=0, scale=1, size=1) – повертає вектор **size** випадкових чисел, що мають рівномірний розподіл на інтервалі [**loc, loc + scale**];

uniform.cdf(x, loc=0, scale=1) – функція (від **x**) розподілу для рівномірного закону на інтервалі [**loc, loc + scale**];

uniform.pdf(x, loc=0, scale=1) – функція (від **x**) щільності розподілу ймовірності для рівномірного закону на інтервалі [**loc, loc + scale**].

uniform.ppf(q, loc=0, scale=1) – функція від **q**, обернена до функції розподілу для рівномірного закону на інтервалі [**loc, loc + scale**]..

Функції для **нормального закону розподілу (norm):**

norm.rvs(loc=0, scale=1, size=1) – повертає вектор **size** випадкових чисел, що мають нормальний розподіл (**loc** – середнє значення, **scale** > 0 – середнє квадратичне відхилення);

norm.cdf(x, loc=0, scale=1) – функція (від **x**) розподілу для нормального закону з параметрами **loc, scale**;

norm.pdf(x, loc=0, scale=1) – функція (від **x**) щільності розподілу ймовірності для нормального закону з параметрами **loc, scale.**

norm.ppf(q, loc=0, scale=1) – функція від **q**, обернена до функції розподілу для нормального закону з параметрами **loc, scale.**

Функції для **експонентного закону розподілу (expon):**

expon.rvs(loc=0, scale=1, size=1) – повертає вектор **size** випадкових чисел, що мають експонентний розподіл (**scale** = 1/lambda, **lambda** > 0 – параметр розподілу);

expon.cdf(x, loc=0, scale=1) – функція (від **x**) розподілу для експонентного закону з параметрами **loc, scale**;

expon.pdf(x, loc=0, scale=1) – функція (від **x**) щільності розподілу ймовірності для експонентного закону з параметрами **loc, scale.**

expon.ppf(q, loc=0, scale=1) – функція від **q**, обернена до функції розподілу для експонентного закону з параметрами **loc, scale.**

Функції для **закону розподілу Пуассона (poisson):**

poisson.rvs(lambda, loc=0, size=1)– повертає вектор **size** випадкових чисел, що мають розподіл Пуассона (***lambda*** > 0 – параметр розподілу (середнє значення)).

poisson.cmf(k, lambda, loc=0) – функція (від **k**) розподілу для закону Пуассона з параметрами **lambda, loc**;

poisson.pdf(k, lambda, loc=0) – функція (від **k**) щільності розподілу ймовірності для закону Пуассона з параметрами **lambda, loc**;

poisson.ppf(q, lambda, loc=0) – функція від **q**, обернена до функції розподілу для закону Пуассона з параметрами **lambda, loc.**

Корисні функції для перевірки гіпотез:

scipy.stats.nct.ppf(q, df, nc, loc=0, scale=1) – функція від **q**, обернена до функції розподілу для t-розподілу Стьюдента, де ***df*** – число степенів свободи, **nc** – параметр;

scipy.stats.chi2.ppf(q, df) – функція від **q**, обернена до функції розподілу для хі-квадрат розподілу, де ***df*** – число степенів свободи.

Приклад 1. Застосування функції **uniform.rvs**. Генеруються точки в колі радіусу 1 (рис. 10.1).

```
import scipy.stats as stats
import numpy as np
import math
import matplotlib.pyplot as plt

def Circle(n):
  r = stats.uniform.rvs(loc=0.0, scale=1.0, size=n)
  teta = stats.uniform.rvs(loc=0.0, scale=2.0*math.pi, size=n)
  x = [0]*n
  y = [0]*n
  for i in range(n):
    x[i] = r[i]*math.cos(teta[i])
    y[i] = r[i]*math.sin(teta[i])

plt.scatter(x, y)

n = 500
Circle(n)
```

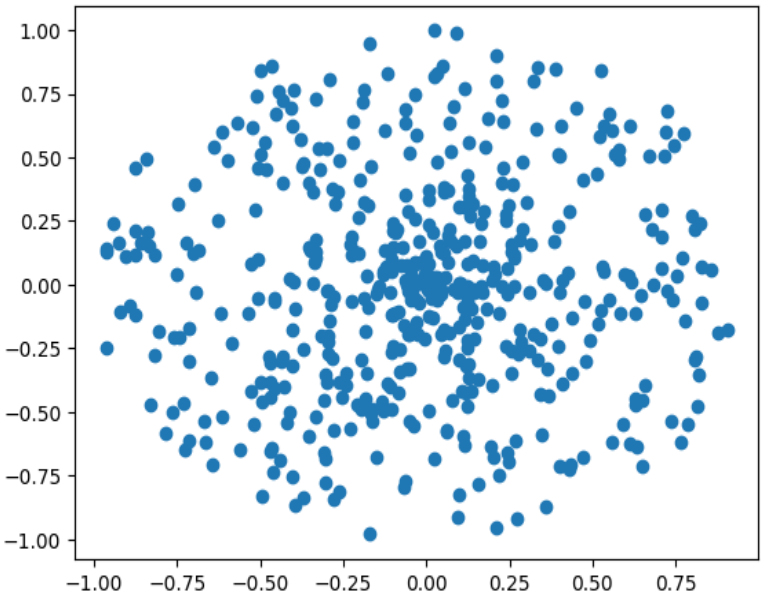

Рис. 10.1. Точки в колі радіусу 1

Приклад 2. Застосування функції **expon.rvs**. Будується графік значень випадкової величини, що підкоряється експонентному розподілу з параметром 5 (рис. 10.1).

```
import scipy.stats as stats

n = 100
y = stats.expon.rvs(loc=0.0, scale=5.0, size=n)
plt.plot(range(n), y)
```

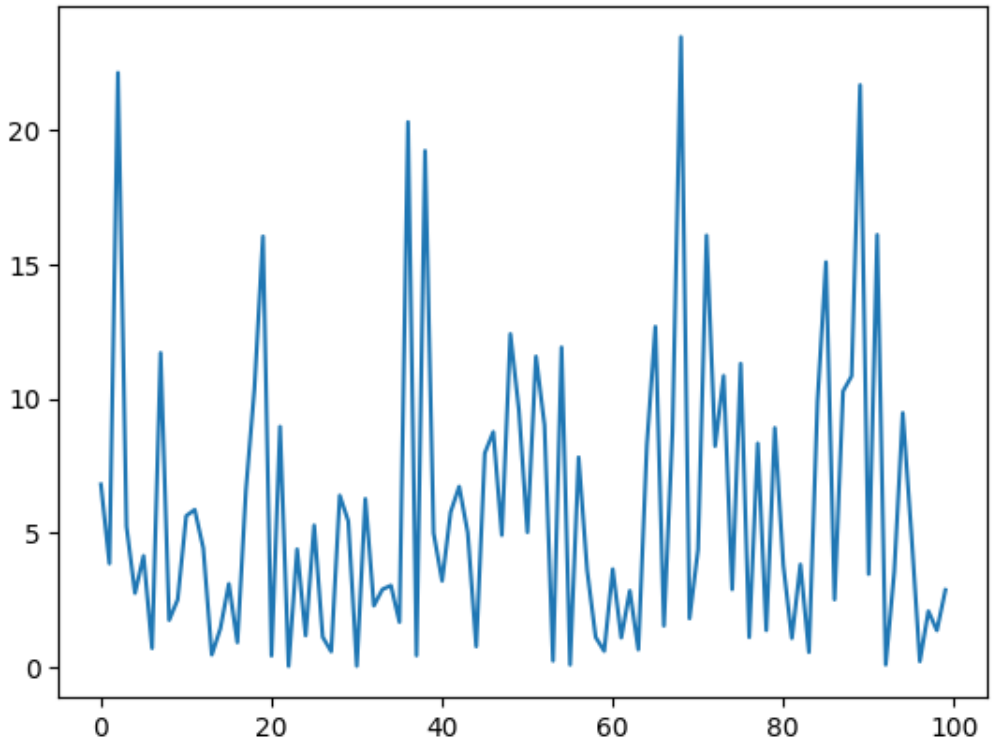

Рис. 10.2. Графік значень випадкової величини, що підкоряється експонентному розподілу

Приклад 3. Приклад застосування функцій **norm.pdf, norm.cdf**. Будуються графіки функції розподілу та функції щільності розподілу ймовірності для нормального закону розподілу з середнім значенням 3 і середнім квадратичним відхиленням 2 (рис. 10.3).

```
import scipy.stats as stats

mean = 3.
sigma = 2.
n = 21
x = np.linspace(-10, 10, n)
ypdf = [0]*n
ycdf = [0]*n
```

```
    for i in range(n):
      ypdf[i] = stats.norm.pdf(x[i], loc=mean, scale=sigma)
      ycdf[i] = stats.norm.cdf(x[i], loc=mean, scale=sigma)

    plt.plot(x, ypdf, color="red")
    plt.plot(x, ycdf, color="blue")
```

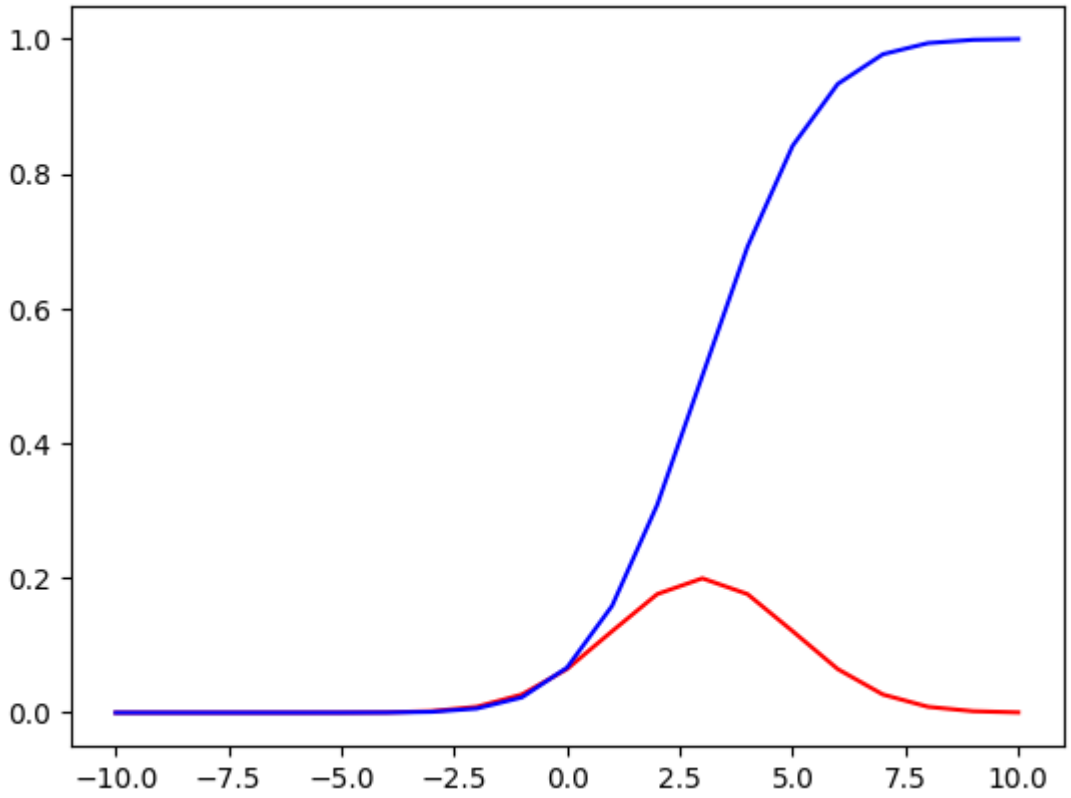

Рис. 10.3. Графік функції розподілу та функції щільності розподілу ймовірності для нормального закону розподілу

10.2.2. Функції для розв'язання задач лінійної алгебри (scipy.linalg)

Функції бібліотеки **scipy.linalg**, призначені для розв'язання задач лінійної алгебри, розподілимо на дві групи.

Функції пошуку різноманітних числових характеристик матриць:

det(A) – визначник (детермінант) матриці A;

norm(A) – обчислення норми квадратної матриці A.

Функції, що реалізують чисельні методи розв'язання задач лінійної алгебри (*scipy.linalg*):

inv(A) – знаходження оберненої матриці;

eigh(A) – обчислення власних значень та власних векторів симетричної квадратної матриці A;

solve(A, b) – розв'язання системи лінійних алгебраїчних рівнянь виду **Ax = b**.

Приклад 4. Приклад застосування функцій **eigh, solve.**

```
import numpy as np
from scipy.linalg import eigh
from scipy.linalg import solve

A = np.matrix([[7, 2, 3], [2, 5, 6], [3, 6, 9]])
print("A=", A)
LamV, Q = eigh(A)
print("Eigenvalues of the matrix A = ", LamV)
B = Q@np.diag(LamV)@Q.T
print("B=", B)

b = [3, 1, 6]
x = solve(A, b)
print("x=", x)
print("Equation residual =", A@x - b)
```

Output:

```
A= [[7 2 3]
 [2 5 6]
 [3 6 9]]
Eigenvalues of the matrix A =  [ 0.67217578  5.37131569 14.95650853]
B= [[7. 2. 3.]
 [2. 5. 6.]
 [3. 6. 9.]]
x= [ 0.16666667 -3.          2.61111111]
Equation residual = [[0.00000000e+00 0.00000000e+00 5.32907052e-15]]
```

10.2.3. Функції для розв'язання систем звичайних диференціальних рівнянь, записаних у нормальній формі

У бібліотеці **scipy.integrate** для Python є багато функцій, призначених для розв’язання задач для систем звичайних диференційних рівнянь.

Функції пакету **scipy.integrate**, що призначені для розв’язання ***задачі з початковою умовою*** (задачі Коші), розв'язують їх для **нормальних систем** звичайних диференціальних рівнянь. Задачі для диференціальних рівнянь вищих порядків зводяться до відповідних задач для нормальних систем.

Розглянемо задачу Коші:

$$y_1^{'} = f_1(t, y_1, y_2, \dots, y_k),$$

$$\begin{aligned} y_2^{'} &= f_2(t, y_1, y_2, \ldots, y_k), \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ y_k^{'} &= f_k(t, y_1, y_2, \ldots, y_k) \end{aligned} \qquad (10.1)$$

за початкових умов:

$$\begin{cases} y_1(t_0) = y_{0,1} \\ y_2(t_0) = y_{0,2} \\ \ldots \\ y_k(t_0) = y_{0,k} \end{cases}.$$

Чисельне розв'язання задачі (10.1) полягає в побудові таблиці наближених значень $y_{i,1}, y_{i,2}, \ldots, y_{i,k}$ $(\mathrm{i} = 1,2, \ldots, \mathrm{N})$. Розв'язок $y_1(t),\ y_2(t), \ldots,\ y_k(t)$ на відрізку $[\mathrm{t}_0, \mathrm{t}_\mathrm{N}]$ у точках $\mathrm{t}_1, \mathrm{t}_2, \ldots, \mathrm{t}_\mathrm{N}$, які називають вузлами сітки. Позначимо:

$$Y = \begin{pmatrix} y_1(t) \\ y_2(t) \\ \ldots \\ y_k(t) \end{pmatrix}, \qquad Y_0 = \begin{pmatrix} \mathrm{y}_{0,1} \\ \mathrm{y}_{0,2} \\ \ldots \\ \mathrm{y}_{0,\mathrm{k}} \end{pmatrix}, \qquad Y' = \begin{pmatrix} y_1'(t) \\ y_2'(t) \\ \ldots \\ y_k'(t) \end{pmatrix},$$

$$F(t, Y) = \begin{pmatrix} f_1(t, y_1, y_2, \ldots y_k) \\ f_2(t, y_1, y_2, \ldots y_k) \\ \ldots \\ f_k(t, y_1, y_2, \ldots y_k) \end{pmatrix},$$

де Y– шуканий розв'язок;

Y_0– вектор початкових умов;

$\mathrm{F}(\mathrm{t}, \mathrm{Y})$ – вектор правих частин системи.

Запишемо задачу Коші для системи диференціальних рівнянь (10.1) у векторній формі:

$$Y' = F(t, Y),\ Y(t_0) = Y_0. \qquad (10.2)$$

Із застосуванням пакету **scipy.integrate** розв'язати задачу Коші для системи (10.2) можна, наприклад, за допомогою функції **solve_inp**:

scipy.integrate.solve_ivp(*fun*, *t_span*, *y0*, *method='RK45'*, *t_eval =None, args=None*) – розв'язання задачі (10.2) на заданому відрізку методами Рунге**–**Кутта.

Тут **fun –** векторна функція $F(t, Y)$ правих частин системи (10.2);

***t_span* –** інтервал $[\mathrm{t}_0, \mathrm{t}_\mathrm{N}]$, на якому розв'язується задача Коші;

y0 – вектор початкових умов Y_0;

***method* –** варіант методу Рунге**–**Кутта;

***t_eval* –** вектор вузлів сітки $\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_\mathrm{N}$ (можна не задавати);

args – додаткові параметри векторної функції $F(t, Y)$, якщо вони є.

Приклад 7. Застосування функції **solve_ivp** з метою розв'язання задачі Коші для диференціального рівняння другого порядку**.**

Знайдемо на відрізку [0, 3] наближений розв'язок рівняння

$$y'' = exp(-t\,y),$$

що задовільнює початковим умовам

$$y(0) = 1,\ y'(0) = 1,$$

і побудуємо графік знайденого розв'язку.

Зведемо розв'язання задачі для рівняння другого порядку до задачі для еквівалентної нормальної системи першого порядку. Позначимо:

$$y_1 = y(t), \quad y_2 = y'(t).$$

Оскільки $y''(t) = (y'(t))' = y_2'(t)$, то маємо:

$$\begin{cases} y_1'(t) = y_2(t) \\ y_2'(t) = exp(-t\,y_1)' \end{cases} \quad \begin{cases} y_1(0) = 1 \\ y_2(0) = 1 \end{cases}.$$

Розв'яжемо цю задачу чисельно, застосовуючи функцію **solve_ivp** з фіксованим кроком на сітці з двадцяти рівновіддалених вузлів:

```
import numpy as np
import math
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

# Задаємо функцію правої частини диференційного рівняння
def FunRight(t, y):
  dydt = [y[0], math.exp(-t*y[0])]
  return dydt

# Задаємо відправні дані
t0 = 0         # початкова точка  відрізка;
t1 = 3         # кінцеве точка відрізка
y0 = [1, 1]    # початкові значення змінної y і її похідної

n = 100             # кількість поділень відрізка
h = (t1 - t0)/n     # шаг поділенння відрізка
t_eval = np.arange(t0, t1, h)
sol = solve_ivp(FunRight, [t0, t1], y0, t_eval=t_eval)
print(sol)
```

Output:

```
      message: The solver successfully reached the end of the integration
interval.
        success: True
         status: 0
              t: [ 0.000e+00  3.000e-02 ...  2.940e+00  2.970e+00]
              y: [[ 1.000e+00  1.030e+00 ...  1.892e+01  1.949e+01]
                  [ 1.000e+00  1.030e+00 ...  1.482e+00  1.482e+00]]
            sol: None
       t_events: None
       y_events: None
           nfev: 26
           njev: 0
            nlu: 0
```

Побудуємо графік розв’язку (рис. 10.4):

```
plt.plot(sol.t, sol.y[0])

plt.title("Solve of Equation")
plt.xlabel('t')
plt.ylabel('y')
```

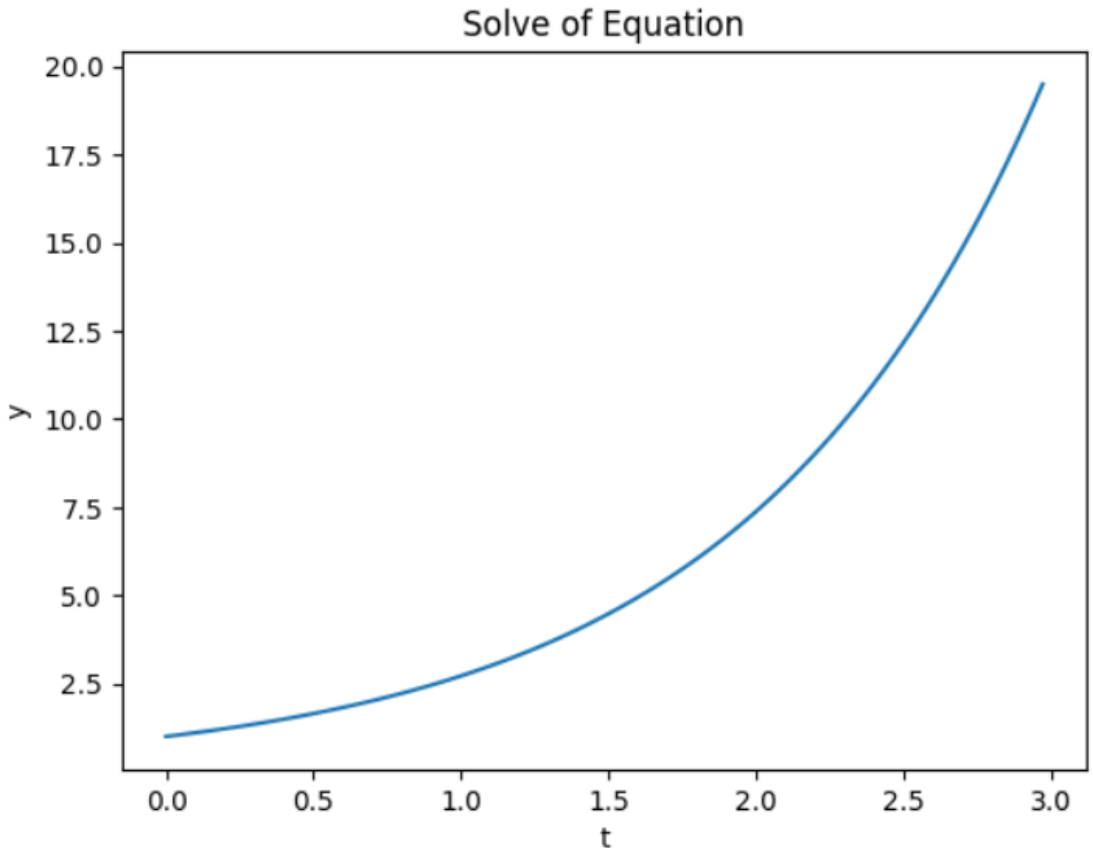

Рис. 10.4. **Графік розв'язку**

10.3. Порядок виконання роботи і варіанти завдань

10.3.1. Зміст звіту

У практичній частині роботи необхідно:
виконати всі приклади наведені в розділі 10.2;
виконати завдання, запропоновані у 10.3.2;
надати тексти складених програм і результати їх виконання.

10.3.2. Варіанти індивідуальних завдань

Виконати всі завдання:

1. Побудуйте графік значень випадкової величини, розподіленої за нормальним законом з параметрами: середнє значення – 0, середнє квадратичне відхилення – 5, відносно номера експерименту з нею (порядкового номера).

2. Побудуйте графік значень випадкової величини, розподіленої за рівномірним законом на відрізку [-5, 5], відносно номера експерименту з нею (порядкового номера).

3. Побудуйте графіки функції розподілу та функції щільності розподілу ймовірності для випадкової величини, розподіленої за рівномірним законом на відрізку [5, 15].

4. Побудуйте матрицю розмірності 5 x 4, (i, j) – елемент якої дорівнює $i^2 - j^2$.

5. Розв'яжіть задачу Коші для системи диференціальних рівнянь першого порядку на відрізку [0, 2]:

$$\begin{cases} y_1' = -11y_1 + 9y_2 \\ y_2' = 9y_1 - 11y_2 \end{cases},$$

за початкових умов

$$y_1(0) = 1, \quad y_2(0) = 0$$

і побудуйте графік знайденого розв'язку.

6. Розв'яжіть задачу Коші для диференціального рівняння другого порядку на відрізку [0, 4π]:

$$y'' + xy = \frac{x}{4\pi},$$

за початкових умов

$$y(0) = 0, \quad y'(0) = 1$$

і побудуйте графік знайденого розв'язку.

10.4. Контрольні запитання

1. Які в системі Python є засоби для генерації значень випадкових величин?
2. Які в системі Python є засоби для створення та роботи з векторами та матрицями?
3. Які в системі Python є засоби для чисельного розв’язання задач лінійної алгебри?
4. За допомогою яких інструментів Python можна розв'язати задачу Коші для звичайних диференціальних рівнянь?