Міністерство освіти і науки України
Національний авіаційний університет
Навчально-науковий інститут комп'ютерних інформаційних технологій
Кафедра комп'ютеризованих систем управління

# Лабораторна робота №5
з дисципліни «Імітаційне моделювання»
на тему «Моделювання процесу функціонування системи за принципом $\Delta t$»

Виконав:
студент ННІКІТ
групи СП-325
Клокун В. Д.
Перевірила:
Марченко Н. Б.

Київ 2019

## 1. Мета роботи

Ознайомитись з методами імітаційного моделювання та принципами побудови моделі процесу функціонування системи; побудувати імітайційну модель процесу функціонування системи в часі за принципом $\Delta t$.

## 2. Хід роботи

Для виконання роботи поставлені такі завдання:

1. Побудувати імітаційну модель процесу функціонування системи в часі.
2. На основі отриманих даних в створеній програмі побудувати траєкторію процесу функціонування системи.
3. Знайти ймовірність того, що неперервна випадкова величина $X$ прийме значення, яке належить інтервалу $[a; b]$.

ЗАВДАННЯ ЗА ВАРІАНТОМ У резервуар, що містить $m$ кг солі на $V_1$ л суміші, кожну хвилину поступає $v$ л води та витікає $V_2$ л суміші. Процес концентрації розчину відбувається за законом:

$$x(t) = \frac{V_1(v - V_2)}{(m + t)^2}.$$

Імітувати процес концентрації розчину в часі з кроком $\Delta t$, якщо при $t = 0$ значення $x = 10$. Значення $\Delta t$ вибирається з інтервалу $(0; 1)$ за допомогою генератора псевдовипадкових чисел. Визначити, яка кількість солі залишиться в резеруварі через $t$ хвилин, припускаючи, що суміш миттєво змішується. Значення змінних $V_1$, $V_2$, $v$ та $m$ вводяться користувачем.

Під час виконання роботи була розроблена імітаційна модель для виконання поставлених завдань і реалізована у вигляді відповідного програмного засобу (ліст. А.1). Реалізований програмний засіб був запущений на моделювання і надавав стабільний і очікуваний результат (рис. 1). В результаті програма будує графік траєкторії процесу функціонування системи (рис. 2).

## 3. Висновок

Виконуючи дану лабораторну роботу, ми ознайомились з методом оберненої функції імітації неперервних випадкових величин; побудували імітаційну модель отримання системи неперервних випадкових величин (СНВВ).

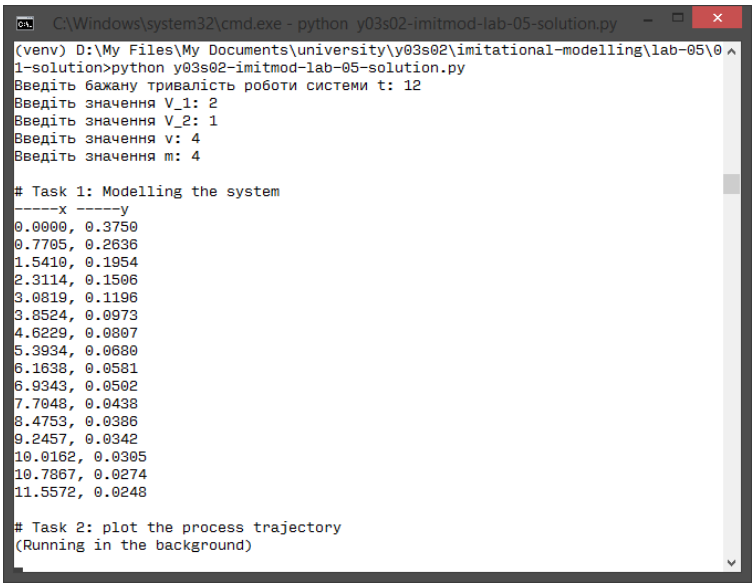


Рис. 1: Результат роботи програми: вікно терміналу

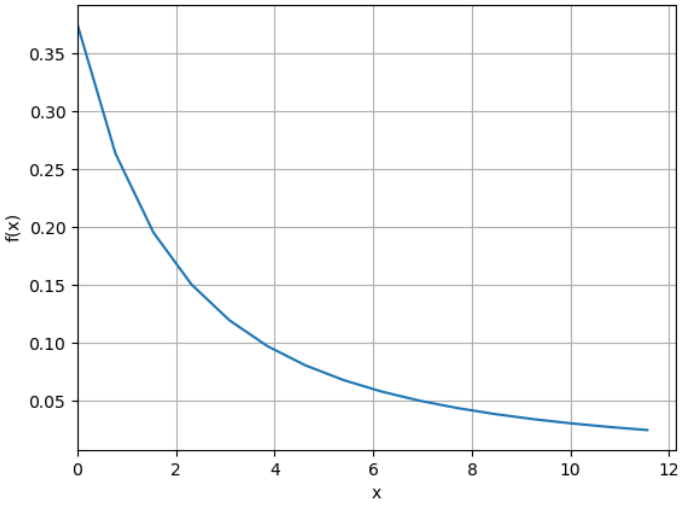


Рис. 2: Графік траєкторії процесу функціонування системи

## А. Повний початковий код програмної реалізації

Лістинг А.1: Повний початковий код програмної реалізації

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import random
import multiprocessing as mp
import matplotlib.pyplot as plt


class Model(object):
    """ A class that represets a process model
    """

    def __init__(self, functional_law, delta_t_min, delta_t_max,
                 delta_t=None, **kwargs):
        """
        :param functional_law: A function that describes how a model
            behaves in time
        :type functional_law: func
        :param delta_t_min: lower bound for randomly generating delta_t
        :type delta_t_min: float
        :param delta_t_max: upper bound for randomly generating delta_t
        :type delta_t_max: float
        :param delta_t: explicit delta_t value. Overrides generated one.
        :type delta_t: float
        """
        # The functional law itself
        self.functional_law = functional_law
        # Variables neccessary for the functional_law to be computed
        self.functional_law_vars = kwargs
        # Time step
        self.delta_t = random.uniform(delta_t_min, delta_t_max)
        # If delta_t is explicitly passed, override it
        if delta_t:
            self.delta_t = delta_t

    def run(self, t):
        """
        Run the model at time point "T"

        :param t: time point value
        :type t: float

        :return: time point t and the value of a system in time point t
        :rtype: tuple
```

```
44          """
45          return t, self.functional_law(**self.functional_law_vars, t=t)
46
47      def run_for(self, time):
48          """
49          A generator function that runs a model for a given duration `time`
50
51          :param time: time duration how long a model should be run
52          :type time: float
53          """
54          t = 0
55          while t < time:
56              yield self.run(t)
57              t += self.delta_t
58
59
60  def split_list_of_points(lofp):
61      """ Splits a list of points of form [(x1, y1), (x2, y2), (x3, y3),
        ↪  ...]
62      into [x1, x2, x3, ...], [y1, y2, y3, ...]
63      """
64      lx, ly = zip(*lofp)
65      return lx, ly
66
67
68  def print_table(t, colh1='x', colh2='y'):
69      print('{:->6} {:->6}'.format(colh1, colh2))
70      for x, y in t:
71          print('{:02.4f}, {:02.4f}'.format(x, y))
72
73
74  def plot_func(x, y, xlabel='x', ylabel='f(x)'):
75      fig = plt.figure(1)
76
77      ax = fig.add_subplot(111)
78      ax.plot(x, y)
79      ax.set_xlabel(xlabel)
80      ax.set_ylabel(ylabel)
81      ax.set_xlim(0.0, None)
82      plt.grid()
83      plt.show()
84      return
85
86
87  def main():
88      # Declare the function described in the task
89      def f(t, V_1, V_2, v, m): return (V_1 * (v-V_2)) / (m+t)**2
90
```

```
91      # Read the time duration
92      t = float(input('Введіть бажану тривалість роботи системи t: '))
93
94      # Read the needed parameters from stdin
95      V_1 = float(input('Введіть значення V_1: '))
96      V_2 = float(input('Введіть значення V_2: '))
97      v = float(input('Введіть значення v: '))
98      m = float(input('Введіть значення m: '))
99
100     # Run the system
101     # Pass every needed variable as a keyword argument
102     print('\n# Task 1: Modelling the system')
103
104     sys = Model(f, 0, 1, V_1=V_1, V_2=V_2, v=v, m=m)
105     simulation_table = [x for x in sys.run_for(time=t)]
106     print_table(simulation_table)
107     x, y = split_list_of_points(simulation_table)
108
109     # Task 2: Plot process trajetory
110     print('\n# Task 2: plot the process trajectory'
111           '\n(Running in the background)')
112
113     p = mp.Process(target=plot_func, args=(x, y))
114     p.start()
115
116
117 if __name__ == '__main__':
118     main()
119
```