

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

Лабораторна робота №2  
з дисципліни «Імітаційне моделювання»  
на тему «Моделювання випадкових подій»

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

Київ 2019

## 1. МЕТА РОБОТИ

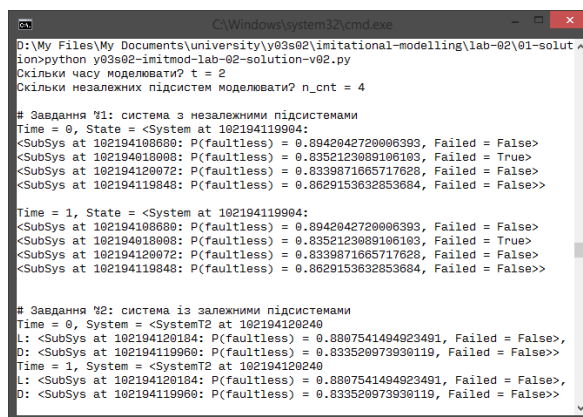
Ознайомитись з алгоритмами моделювання результатів випробувань випадкових подій; побудувати імітаційну модель функціонування системи протягом деякого часу.

## 2. ХІД РОБОТИ

Для виконання роботи задані такі завдання:

1. Задано задачу 1 загального виду: система  $S_1$  працює протягом деякого часу  $t$  і складається з  $m$  підсистем, кожна з яких незалежно одна від одної може вийти з ладу впродовж деякого часу  $t$ . Ймовірність безвідмовної роботи кожної підсистеми визначається так:  $P(n_i) = p_i$ . Побудувати імітаційну модель функціонування системи  $S_1$  протягом часу  $t$ , використовуючи генератор псевдовипадкових чисел на інтервалі  $(0; 1)$ . Усі дані вводяться користувачем.
2. Задано задачу 2 загального виду: система  $S_2$  складається з 2 підсистем, які пов'язані між собою. Ймовірність безвідмовної роботи підсистеми  $n_1$  протягом часу  $t \in P(n_1) = p_1$ . Ймовірність роботи другої підсистеми  $n_2$  така:  $P(n_i) = p_i$ . Побудувати імітаційну модель функціонування системи  $S_2$  протягом часу  $t$ , використовуючи генератор псевдовипадкових чисел на інтервалі  $(0; 1)$ .

Під час виконання роботи були розроблені імітаційні моделі для виконання поставлених завдань і реалізовані у вигляді відповідного програмного засобу (ліст. А.1). Реалізований програмний засіб був запущений на моделювання і надавав стабільний і очікуваний результат (рис. 1).



```
C:\Windows\system32\cmd.exe
D:\My Files\My Documents\university\y83s02\imitational-modelling\lab-02\01-solut
ion>python y83s02-imitmod-lab-02-solution-v02.py
Скільки часу моделювати? t = 2
Скільки незалежних підсистем моделювати? n_cnt = 4

# Завдання '1': система з незалежними підсистемами
Time = 0, State = <System at 182194119904:
<SubSys at 182194108080: P(faultless) = 0.8942042720906393, Failed = False>
<SubSys at 182194120808: P(faultless) = 0.8352123089196103, Failed = True>
<SubSys at 182194120072: P(faultless) = 0.8339871665717628, Failed = False>
<SubSys at 182194119848: P(faultless) = 0.8629153632853684, Failed = False>

Time = 1, State = <System at 182194119904:
<SubSys at 182194108080: P(faultless) = 0.8942042720906393, Failed = False>
<SubSys at 182194120808: P(faultless) = 0.8352123089196103, Failed = True>
<SubSys at 182194120072: P(faultless) = 0.8339871665717628, Failed = False>
<SubSys at 182194119848: P(faultless) = 0.8629153632853684, Failed = False>

# Завдання '2': система із залежними підсистемами
Time = 0, System = <SystemT2 at 182194120240
L: <SubSys at 182194120184: P(faultless) = 0.8807541494923491, Failed = False>,
D: <SubSys at 182194119960: P(faultless) = 0.833520973930119, Failed = False>
Time = 1, System = <SystemT2 at 182194120240
L: <SubSys at 182194120184: P(faultless) = 0.8807541494923491, Failed = False>,
D: <SubSys at 182194119960: P(faultless) = 0.833520973930119, Failed = False>
```

Рис. 1: Результат роботи реалізованого програмного засобу

### 3. ВИСНОВОК

Виконуючи дану лабораторну роботу, ми ознайомились з алгоритмами моделювання результатів випробувань випадкових подій; побудували імітаційну модель функціонування системи протягом деякого часу.

#### А. ПОВНИЙ ПОЧАТКОВИЙ КОД ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

---

Лістинг А.1: Повний початковий код програмної реалізації

```
1  #!/usr/bin/env python3
2
3  import random # random.random(), random.uniform()
4
5
6  class Subsystem(object):
7      """ Subsystem --- основний клас-підсистема для завдань №№1, 2
8      """
9      def __init__(self, p_faultless=0.0, failed=False):
10         self.p_faultless = p_faultless
11         self._failed = failed
12
13     def __str__(self):
14         """ Визначає представлення об'єкту даного класу у вигляді рядка
15         ↪ """
16         s = (
17             'return s
22
23     @property
24     def p_faultless(self):
25         """ Ймовірність безвідмовної роботи """
26         return self._p_faultless
27
28     @p_faultless.setter
29     def p_faultless(self, value):
30         """ Дозволяє задати значення ймовірності безвідмовної роботи """
31         self._p_faultless = value
32
33     @property
34     def failed(self):
35         """ Статус, що система непрацездатна """
36         return self._failed
```

---

```

36
37     @failed.setter
38     def failed(self, value):
39         self._failed = value
40
41     def _fail(self):
42         """ Робить систему непрацездатною """
43         self.failed = True
44
45     def run(self):
46         """ Запускає систему у даний момент часу.
47
48         Якщо система вже непрацездатна, завершити роботу і повернути
↪ статус
49         непрацездатності. Інакше перевірити, чи зламається система у даний
50         момент часу і також повернути статус непрацездатності.
51         """
52         if self.failed:
53             return self.failed
54         elif random.random() > self.p_faultless:
55             self._fail()
56
57         return self.failed
58
59
60 class SystemType1(object):
61     """ Система з незалежними підсистемами """
62     def __init__(self, subsystems=None):
63         self._subsystems = subsystems
64
65     def __str__(self):
66         s = ''
67         subsys_list = [str(subsys) for subsys in self.subsystems]
68         s = '\n'.join(subsys_list)
69         s = (
70             '<System at {}: \n'
71             '{}>'.format(id(self), s)
72         )
73
74         return s
75
76     @property
77     def subsystems(self):
78         return self._subsystems
79
80     def add_subsystem(self, subsystem):
81         self.subsystems.append(subsystem)
82

```

```

83     def run(self, time=1):
84         for t in range(time):
85             for s in self.subsystems:
86                 s.run()
87
88             print('Time = {}, State = {}\n'.format(t, str(self)))
89
90     def get_state(self):
91         state = ''
92         for s in self.subsystems:
93             state += '{}'.format(str(s))
94
95         return state
96
97
98 class SubsystemDependant(Subsystem):
99     """ Залежна система для завдання №2.
100
101     Успадковує та розширює незалежну підсистему
102     """
103     def __init__(self, p_faultless=0.0, p_faultless_leader_failed=0.0,
104                 failed=False):
105         self.p_faultless = p_faultless
106         self._failed = failed
107
108         self.p_faultless_leader_failed = p_faultless_leader_failed
109
110     @property
111     def p_faultless_leader_failed(self):
112         return self._p_faultless_leader_failed
113
114     @p_faultless_leader_failed.setter
115     def p_faultless_leader_failed(self, value):
116         self._p_faultless_leader_failed = value
117
118     def run_leader_faulty(self):
119         """ Метод, який описує процес роботи підсистеми у системі, де її
120             ↪ головна
121             підсистема непрацездатна
122             """
123         if self.failed:
124             return self.failed
125         elif random.random() > self.p_faultless_leader_failed:
126             self._fail()
127
128         return self.failed
129

```

```

130 class SystemType2(object):
131     """ Система із головною і залежною підсистемою для
132     завдання 2
133     """
134     def __init__(self, leader, dependant):
135         """ Конструктор
136         """
137         self.subsys_leader = leader
138         self.subsys_dependant = dependant
139
140     def __str__(self):
141         """ Визначає, як об'єкт даного класу виглядатиме у вигляді рядка
142         """
143         s = (
144             '<SystemT2 at {} \n'
145             'L: {}, \n'
146             'D: {}>'
147             .format(id(self), self.subsys_leader, self.subsys_dependant)
148         )
149         return s
150
151     def run_once(self):
152         """ Запускає систему в один момент часу """
153         # Запустити незалежну підсистему
154         self.subsys_leader.run()
155         # Якщо незалежна підсистема не зламалась, запустити залежну у
156         ↪ звичайному
157         # режимі
158         if not self.subsys_leader.failed:
159             self.subsys_dependant.run()
160         # Інакше --- у режимі зламаної головної підсистеми
161         else:
162             print('Leader failed.')
163             self.subsys_dependant.run_leader_faulty()
164
165     def run(self, time=1):
166         for t in range(time):
167             self.run_once()
168             print(
169                 'Time = {}, System = {}'
170                 .format(t, str(self))
171             )
172
173     def main():
174         runtime = int(input('Скільки часу моделювати? t = '))
175         n_cnt = int(input('Скільки незалежних підсистем моделювати? n_cnt = '))
176         ↪ '))

```

```

176     # Завдання 1
177     print('\\n# Завдання №1: система з незалежними підсистемами')
178
179     # Створюємо незалежні підсистеми
180     subsystems = [
181         Subsystem(p_faultless=random.uniform(0.8, 0.9))
182         for _ in range(n_cnt)
183     ]
184     # Створюємо систему з незалежних підсистем
185     system1 = SystemType1(subsystems)
186     # Запускаємо її
187     system1.run(time=runtime)
188
189     # Завдання 2
190     print('\\n# Завдання №2: система із залежними підсистемами')
191
192     # Створюємо систему з головною і залежною підсистемою
193     system2 = SystemType2(
194         # Головна
195         leader=Subsystem(p_faultless=random.uniform(0.8, 0.9)),
196         # Залежна
197         dependant=SubsystemDependant(
198             # Ймовірність безвідмовної роботи, коли головна система
199             ↪ справна
200             p_faultless=random.uniform(0.8, 0.9),
201             # Ймовірність безвідмовної роботи, коли головна система
202             ↪ несправна
203             p_faultless_leader_failed=random.uniform(0.6, 0.8)
204         ),
205     )
206     # Запускаємо її
207     system2.run(time=runtime)
208
209     if __name__ == '__main__':
210         main()

```

---