§46. Лабораторная работа № 10
Задача идентификации потока экспериментальных данных

Ц е л ь р а б о т ы: изучить проблему идентификации потока экспериментальных данных

С о д е р ж а н и е р а б о т ы: осуществить компьютерную реализацию идентификации потока экспериментальных данных

Пример выполнения здания. Часть 1

Значениями случайного процесса являются случайные величины *E* с заданным распределением с математическим ожиданием *u* и дисперсией *d*. Число *d* известно, а о числе *u* известно, что во всех случаях оно равно либо *u1*, либо *u2*, причем *u2 > u1*. Требуется построить компьютерный имитатор случайной величины *E*, который начинает свою работу с равновероятного выбора между *u1* и *u2*, а затем с помощью процедуры последовательной проверки устанавливает истинное значение u.

Программный код (язык программирования Python):

```
import math
import random
import matplotlib.pyplot as plt
def math_expect(s): """ * Calculate math expectation """
return sum(s)/len(s)
def dispersion(s, M = None): """ * Calculate dispersion. * Use M - as math expectation if it known
* else calculate it """
if M is None:
M = math_expect(s)
n = len(s)
D = 0
for i in range(n):
D += (s[i] - M)**2
return D/n
```

```
def white_noize():
s = 0
for i in range(12):
s += random.random()
return s - 6
def ud_4_lognormal_dist(M, D): """ Calculate coefficients
for lognormal distribution
M - math exception
D - dispersion """ d = math.log(D / math.exp(2 * math.log(M)) + 1)
u = math.log(M) - d / 2
return u, d**0.5
def normal_dist_val(u, d):
return white_noize() * d + u
def lognormal_dist_val(u, d):
return math.exp(normal_dist_val(u, d))
def count_unten(n :float, min_count = 5)->int: """ return number tens not in number
0.1234 -> 4; 0.11 -> 2; 0.1 -> 1"""
i = 0
if n == 0:
return i
flag = False
while True:
n *= 10
mod = int(n) % 10
if not flag and mod != 0:
flag = True
if flag and mod == 0:
break
i += 1
```

```
if i >= min_count:
break
return i
def main():
M1 = M2 = 0
while M2 <= M1:
M1 = float(input("M1 = "))
M2 = float(input("M2 = "))
D = float(input("D = "))
sub = abs(M2 - M1)
e = 0.01
if sub < 1:
e = 0.1 / 10 ** count_unten(sub)
M = M1 if random.random() >= 0.5 else M2
vals = []
u, d = ud_4_lognormal_dist(M, D)
flag1 = flag2 = False
while not (flag1 or flag2) and len(vals) < 1000000:
vals += [lognormal_dist_val(u, d)]
M = math_expect(vals)
flag1 = abs(M - M1) <= e
flag2 = abs(M - M2) <= e
print("Математичское ожидание: ", end="")
if flag1:
print("M1", end="")
elif flag2:
print("M2", end="")
else:
print("undefined", end="")
print(f" = {M:f}")
```

```
if __name__ == "__main__":
main()
```

Результат выполнения программы

```
M1 = 5
M2 = 10
D = 3
Математичское ожидание: M1 = 5.002331
```

```
M1 = 1
M2 = 12
D = 6
Математичское ожидание: M2 = 11.997037
```

```
M1 = 3
M2 = 100
D = 51
Математичское ожидание: M1 = 2.998162
```

```
M1 = 6
M2 = 9
D = 12
Математичское ожидание: M2 = 8.992482
```

Пример выполнения здания. Часть 2

Основываясь на результатах лабораторной работы по теме «Задача идентификации потока экспериментальных данных» часть 1 осуществить, согласно своего варианта, компьютерный анализ зависимости времени идентификации случайного процесса от величины d. Предусмотреть программный блок, ориентированный на соответствующую обработку не имитированных, а экспериментальных данных.

Программный код (язык программирования Python):

```
import math
import random
```

```
import time
import matplotlib.pyplot as plt
def math_expect(s): """ * Calculate math expectation """
return sum(s)/len(s)
def dispersion(s, M = None): """ * Calculate dispersion. * Use M - as math expectation if it known
* else calculate it """
if M is None:
M = math_expect(s)
n = len(s)
D = 0
for i in range(n):
D += (s[i] - M)**2
return D/n
def white_noize():
s = 0
for i in range(12):
s += random.random()
return s - 6
def ud_4_lognormal_dist(M, D): """ Calculate coefficients
for lognormal distribution
M - math exception
D - dispersion """ d = math.log(D / math.exp(2 * math.log(M)) + 1)
u = math.log(M) - d / 2
return u, d**0.5
def normal_dist_val(u, d):
return white_noize() * d + u
def lognormal_dist_val(u, d):
return math.exp(normal_dist_val(u, d))
```

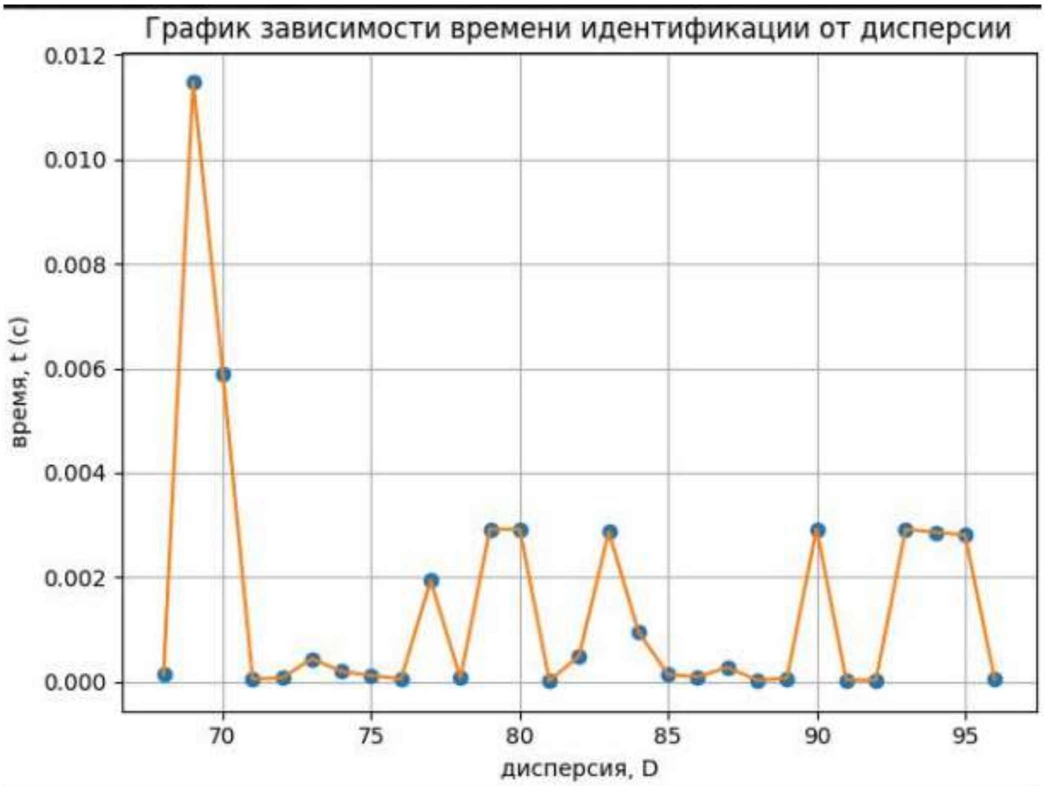
```
def count_unten(n :float, min_count = 5)->int: """ return number tens not in number
0.1234 -> 4; 0.11 -> 2; 0.1 -> 1"""
i = 0
if n == 0:
return i
flag = False
while True:
n *= 10
mod = int(n) % 10
if not flag and mod != 0:
flag = True
if flag and mod == 0:
break
i += 1
if i >= min_count:
break
return i
# ====================================== work code =============================================
def identify_random_process(M1: float, M2: float, data: list): """ M1 - math exception 1
M2 - math exception 2
return math exception for data and determine to whom to relate
ans == 0 - M is M1
ans == 1 - M is M2
ans == 2 - M is undefined"""
sub = abs(M2 - M1)
e = 0.01
if sub < 1:
```

```
e = 0.1 / 10 ** count_unten(sub)
flag1 = flag2 = False
vals = []
data = data.copy()
while not flag1 and not flag2 and len(data) > 0:
vals.append(data.pop())
M = math_expect(vals)
flag1 = abs(M - M1) <= e
flag2 = abs(M - M2) <= e
if flag1:
ans = 0
elif flag2:
ans = 1
else:
ans = 3
return ans, M
# ========================================= work code ==========================================
def input_interval(**kwargs):
a = kwargs.get("a")
b = kwargs.get("b")
text1 = kwargs.get("text1")
text2 = kwargs.get("text2")
sign = kwargs.get("sign")
if a is None:
a = 0
if b is None:
b = 0
if text1 is None:
text1 = ""
```

```
if text2 is None:
text2 = ""
if sign is None:
sign = lambda a, b: b > a
while not sign(a, b):
a = float(input(text1))
b = float(input(text2))
return a, b
def main():
M1, M2 = input_interval(text1="M1 = ", text2="M2 = ")
D1, D2 = input_interval(text1="D1 = ", text2="D2 = ")
M = M1 if random.random() >= 0.5 else M2
list_T = []
list_D = []
D = D1
while D <= D2:
list_D += [D]
u, d = ud_4_lognormal_dist(M, D)
data = [lognormal_dist_val(u, d) for i in range(1000)]
t0 = time.time()
ans, M = identify_random_process(M1, M2, data)
list_T += [time.time() - t0]
D += 1
fig, axs = plt.subplots(1, 1, constrained_layout=True)
axs.plot(list_D, list_T, 'o', list_D, list_T, '-')
axs.set_title('График зависимости времени идентификации от
дисперсии')
axs.set_xlabel('дисперсия, D')
axs.set_ylabel('время, t (c)')
axs.grid(True)
```

```
plt.show()
if __name__ == "__main__":
main()
```

Результат выполнения программы



Варианты заданий к лабораторной работе № 10

Варианты № 1 – № 5.

1. Значениями случайного процесса являются нормально распределенные случайные величины ξ с математическим ожиданием μ и дисперсией σ. Число σ известно, а о числе μ известно, что во всех случаях оно равно либо μ_1, либо μ_2, причем $\mu_2>\mu_1$. Требуется построить компьютерный имитатор случайной величины ξ, который начинает свою работу с равновероятного выбора между μ_1 и μ_2, а затем с помощью процедуры последовательной про-верки устанавливает истинное значение μ.

2. Осуществить компьютерный анализ зависимости времени идентификации случайного процесса от величины σ. Предусмотреть программный блок, ориентированный на соответствующую обработку не имитированных, а экспериментальных данных.

Варианты № 6 – № 10.

1. Значениями случайного процесса являются логнормально распределённые случайные величины ξ с математическим ожиданием μ и дисперсией σ. Число σ известно, а о числе μ известно, что во всех случаях оно равно либо μ_1, либо μ_2, причем $\mu_2>\mu_1$. Требуется построить компьютерный имитатор случайной величины ξ, который начинает свою работу с равновероятного выбора между μ_1 и μ_2, а затем с помощью процедуры последовательной проверки устанавливает истинное значение μ.

2. Осуществить компьютерный анализ зависимости времени идентификации случайного процесса от величины σ. Предусмотреть программный блок, ориентированный на соответствующую обработку не имитированных, а экспериментальных данных.

Варианты № 11 – № 15.

1. Значениями случайного процесса являются экспоненциально распределённые случайные величины ξ с параметром λ, о котором известно, что во всех случаях он равен либо λ_1, либо λ_2, причем $\lambda_2>\lambda_1$. Требуется построить компьютерный имитатор случайной величины ξ, который начинает свою работу с равновероятного выбора между λ_1 и λ_2, а затем с помощью процедуры последовательной проверки устанавливает истинное значение λ.

2. Осуществить компьютерный анализ зависимости времени идентификации случайного процесса от дисперсии величины ξ. Предусмотреть программный блок, ориентированный на соответствующую обработку не имитированных, а экспериментальных данных.

Варианты № 16 – № 20.

1. Значениями случайного процесса являются нормально распределенные случайные величины ξ с математическим ожиданием μ и дисперсией σ. Число σ известно, а о числе μ известно, что во всех случаях оно равно либо μ_1, либо μ_2, причем $\mu_1>\mu_2$. Требуется построить компьютерный имитатор случайной величины ξ, который начинает свою работу с

равновероятного выбора между μ_1 и μ_2, а затем с помощью процедуры последовательной проверки устанавливает истинное значение μ.

2. Осуществить компьютерный анализ зависимости времени идентификации случайного процесса от величины σ. Предусмотреть программный блок, ориентированный на соответствующую обработку не имитированных, а экспериментальных данных.

Варианты № 21 – № 25.

1. Значениями случайного процесса являются логнормально распределённые случайные величины ξ с математическим ожиданием μ и дисперсией σ. Число σ известно, а о числе μ известно, что во всех случаях оно равно либо μ_1, либо μ_2, причем $\mu_1>\mu_2$. Требуется построить компьютерный имитатор случайной величины ξ, который начинает свою работу с равновероятного выбора между μ_1 и μ_2, а затем с помощью процедуры последовательной проверки устанавливает истинное значение μ.

2.Осуществить компьютерный анализ зависимости времени идентификации случайного процесса от величины σ. Предусмотреть программный блок, ориентированный на соответствующую обработку не имитированных, а экспериментальных данных.

Варианты № 26 – № 30.

1. Значениями случайного процесса являются нормально распределенные случайные величины ξ с математическим ожиданием μ и дисперсией σ. Число σ известно, а о числе μ известно, что во всех случаях оно равно либо μ_1, либо μ_2, причем $\mu_1>\mu_2$. Требуется построить компьютерный имитатор случайной величины ξ, который начинает свою работу с равновероятного выбора между μ_1 и μ_2, а затем с помощью процедуры последовательной проверки устанавливает истинное значение μ.

2. Осуществить компьютерный анализ зависимости времени идентификации случайного процесса от величины σ. Предусмотреть программный блок, ориентированный на соответствующую обработку не имитированных, а экспериментальных данных.