Министерство образования и науки РФ

Федеральное государственное автономное образовательное учреждение высшего образования

Севастопольский государственный университет

Институт информационных технологий и управления в технических системах

Кафедра «Информационных Систем»

Отчет

по лабораторной работе №1

Определение вероятности безотказной работы.

Интенсивность отказов.

Среднее время наработки на отказ.

Выполнил:

ст. гр. ИС22-д

Серегин А.В.

Проверил:

Колесников

Севастополь

2016

1. Цель работы

Определить показатели надежности и произвести расчеты на С++ партии изделий при проведении испытаний в течении заданного промежутка времени.

2. Вариант задания

№ 21:

N = 1000; t = 7000; n­­1 = 230; n2 = 100; ∆t = 150.

3. Код программы.

Реализация библиотеки RIS:

#include "RIS.h"

#include <cmath>

double RIS::reliabilities(const uint64\_t& \_count, const uint64\_t& \_failsCount, const double& \_deltaTime)

{

return \_failsCount / double(\_count \* \_deltaTime);

}

double RIS::reliabilities(const uint64\_t& \_count, const uint64\_t& \_countLeft)

{

return (double)\_countLeft / (double)\_count;

}

double RIS::reliabilities(const double& \_failureIntensity, const double& \_workTime)

{

return std::exp(-\_failureIntensity \* \_workTime);

}

double RIS::failureRate(const uint64\_t& \_count, const uint64\_t& \_failsCount)

{

return 1.0 - double(\_failsCount) / double(\_count);

}

double RIS::failureIntensity(const double& \_failureRate, const double& \_reliabilities)

{

return \_failureRate / \_reliabilities;

}

double RIS::failureIntensity(const uint64\_t& \_count, const uint64\_t& \_failsCount, const double& \_deltaTime)

{

return 2.0 \* double(\_failsCount) / double(\_count + \_count - \_failsCount) / double(\_deltaTime);

}

double RIS::averageWorkTime(const uint64\_t& \_count, const uint64\_t& \_failsCount, const double& \_deltaTime)

{

return 1.0 / failureIntensity(\_count, \_failsCount, \_deltaTime);

}

double RIS::averageWorkTime(const double& \_failureIntensity)

{

return 1.0 / \_failureIntensity;

}

Main.cpp:

#include <QString>

#include <QDebug>

#include <RIS.h>

int main()

{

uint64\_t count = 1000;

uint64\_t time = 7000;

uint64\_t deltaTime = 150;

uint64\_t failsT = 230;

uint64\_t failsDT = 100;

double reliabilitiesDT = RIS::reliabilities(count, failsDT, deltaTime);

qDebug() << "Reliabilities for the dt =" << QString("%1 (1/hour)").arg(reliabilitiesDT, 0, 'f', 7);

double failureRateT = RIS::failureRate(count, failsT);

qDebug() << "Failure rate for the t =" << QString("%1").arg(failureRateT, 0, 'f', 7);

double failureRateTDT = RIS::failureRate(count, failsT + failsDT);

qDebug() << "Failure rate for the t + dt =" << QString("%1").arg(failureRateTDT, 0, 'f', 7);

double failureRateDT = failureRateTDT / failureRateT;

qDebug() << "Failure rate for the dt =" << QString("%1").arg(failureRateDT, 0, 'f', 7);

double failureIntensityT = RIS::failureIntensity(count, failsT, time);

qDebug() << "Failure Intensity for the t =" << QString("%1 (1/hour)").arg(failureIntensityT, 0, 'f', 7);

double failureIntensityDT = RIS::failureIntensity(reliabilitiesDT, failureRateDT);

qDebug() << "Failure Intensity for the dt =" << QString("%1 (1/hour)").arg(failureIntensityDT, 0, 'f', 7);

double avrWorkTime = RIS::averageWorkTime(failureIntensityDT);

qDebug() << "Average time before failure = " << QString("%1 hours").arg(avrWorkTime, 0, 'f', 1);

return 0;

}

4. Результаты выполнения программы.

На рисунке 1 приведены расчеты сделанные с помощью программы написанной на с++:

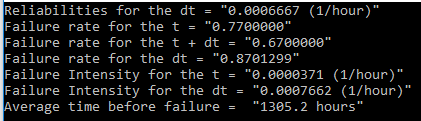


Рисунок 1 – Результат выполнения программы.

Выводы

В ходе лабораторной работы мы научились определять показатели надежности, а именно: вероятность безотказной работы, вероятность отказа, интенсивность отказа и среднее время наработки на отказ.