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 №2

Определение показателей надежности систем

по статистическим данным.

Выполнил:

ст. гр. ИС22-д

Серегин А.В.

Проверил:

Колесников

Севастополь

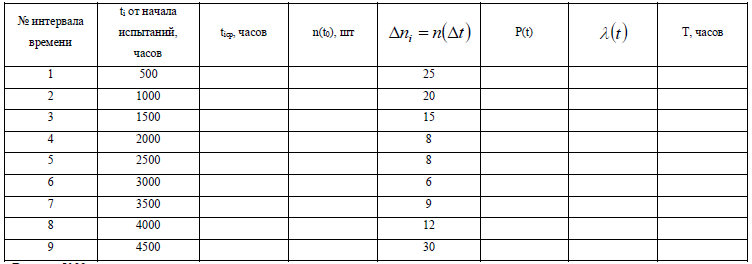
2016

1. Цель работы.

Произвести расчет показателей надежности систем по статистическим данным, полученных при их испытании.

2. Вариант задания.

№ 21:



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 <QDebug>

#include <QVector>

#include <RIS.h>

int main(int argc, char \*argv[])

{

int entityesCount = 200;

QVector<double> timePassed = {500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500};

QVector<uint64\_t> failureOnTimeInterval = {25, 20, 15, 8, 8, 6, 9, 12, 30};

QVector<double> timeInterval;

QVector<uint64\_t> entityesLeft;

QVector<double> reliabilities;

QVector<double> failureIntensity;

for(int i = 0; i < timePassed.size(); ++i)

{

if(i)

{

timeInterval << ((timePassed[i] + timePassed[i - 1]) / 2.0);

entityesLeft << (failureOnTimeInterval[i] <= entityesLeft[i - 1] ? entityesLeft[i - 1] - failureOnTimeInterval[i] : 0);

failureIntensity << RIS::failureIntensity(entityesLeft[i - 1], failureOnTimeInterval[i], timePassed[i] - timePassed[i - 1]);

}

else

{

timeInterval << (timePassed[i] / 2.0);

entityesLeft << (failureOnTimeInterval[i] <= entityesCount ? entityesCount - failureOnTimeInterval[i] : 0);

failureIntensity << RIS::failureIntensity(entityesCount, failureOnTimeInterval[i], timePassed[i]);

}

reliabilities << RIS::reliabilities(entityesCount, entityesLeft[i]);

}

qDebug() << "Time intervals: " << timeInterval;

qDebug() << "Entities left: " << entityesLeft;

qDebug() << "Failure rate: " << reliabilities;

qDebug() << "Failure intensivity: " << failureIntensity;

return 0;

}

4. Расчетные данные.

На рисунке ниже представлены расчетные данные, полученные в ходе выполнения программного кода.

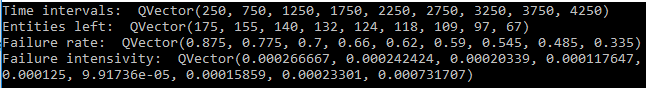


Рисунок 1 – Результат выполнения программы.

Рисунок 2 – График зависимости вероятности

безотказной работы от времени.

Рисунок 3 – График изменения интенсивности

отказов во времени.

Выводы

В ходе лабораторной работы были произведены расчеты по статистическим данным, выясненным в результате проведения испытаний. Стало ясно, что с течением времени растет интенсивность отказов и понижается вероятность безотказной работы.