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 №4

Определение надежности систем при схеме

последовательного и параллельного

соединения элементов.

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 при последовательном и параллельном соединении элементов системы.

2. Вариант задания

№ 21:

Время работы = 7000.



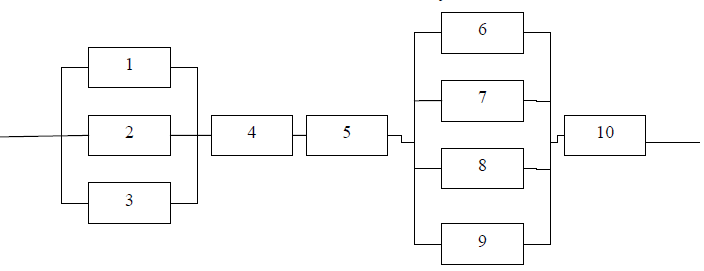


Рисунок 1 – Исследуемая схема.

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 <RIS.h>

#include <QDebug>

#include <QString>

struct Entity

{

Entity(): reliabilities(0), failureRate(0) {}

Entity(const double& \_reliabilities, const double& \_failureRate)

: reliabilities(\_reliabilities),

failureRate(\_failureRate) {}

double reliabilities;

double failureRate;

};

int main()

{

double workTime = 7000;

//sheme view

QVector<QVector<double>> sheme = {

{0.000125, 0.000135, 0.00014},

{0.000115},

{0.00015},

{0.00012, 0.00015, 0.000125, 0.000085},

{0.000175}

};

QVector<Entity> entities;

double fullReliabilities = 1.0;

double fullFailureRate;

for(auto& system : sheme)

{

double q = 1.0;

for(auto& entityFailureIntensity : system)

{

double realiabilities = RIS::reliabilities(entityFailureIntensity, workTime);

double failureRate = 1.0 - realiabilities;

entities << Entity(realiabilities, failureRate);

q \*= failureRate;

}

fullReliabilities \*= 1.0 - q;

}

fullFailureRate = 1.0 - fullReliabilities;

qDebug() << "Reliabilities / Failure rate:";

for(const Entity& entity : entities)

{

qDebug() << QString("%0 / %1").arg(entity.reliabilities, 0, 'f', 5).arg(entity.failureRate, 0, 'f', 5);

}

qDebug() << QString("Full reliability: %0").arg(fullReliabilities, 0, 'f', 5);

qDebug() << QString("Full failure rate: %0").arg(fullFailureRate, 0, 'f', 5);

return 0;

}

4. Расчетные данные.

На рисунке ниже изображены результаты выполнения программного кода.

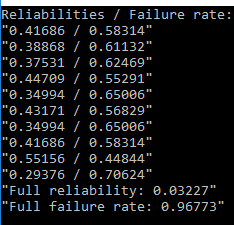


Рисунок 2 – Результаты выполнения программы.

Выводы

В ходе лабораторной работы были изучены методы расчета показателей надежности в последовательно и параллельно соединенной системе.