Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Кафедра “Системы искусственного интеллекта”

УТВЕРЖДАЮ
Заведующий кафедрой

_____ _____
подпись инициалы, фамилия

« ____» _____ 2018 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

09.03.02.04 «Информационные системы и технологии в медиаиндустрии»

Численные методы расчета параметров надежности уникальных систем

Руководитель _____ Ст. преподаватель кафедры Т. Н. Сизова
подпись, дата

Студент _____ _____ _____ М. И. Малимонов
номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2018

Содержание

Введение

Теория надежности – наука, которая исследует методы обеспечения стабильности работоспособности объектов, изучает численные показатели надежности, а кроме того связь между эффективностью и надежностью. Уникальные системы представляют собой совокупность нескольких самостоятельных подсистем, предназначенных для достижения общей цели, и при отказе хотя бы одной из них может привести к катастрофе.

В данной дипломной работе рассматривается надежность судового электрооборудования.

Электрическая система на судне состоит из множества источников электроэнергии, различных регуляторов и стабилизаторов, электрических кабелей и устройств, потребляющих электрическую энергию.

Быстрое развитие прогресса позволило автоматизировать многие судовые системы. Именно это повлекло за собой повышение требований к качеству технических систем, к точности и быстродействию устройств. Технический прогресс зависим от надежности так же, как и надежность от технического прогресса. Успешные решения вопросов надежности позволяет прогрессу идти вперед. Тема надежности является далеко не новой, практические потребности человека выдвигали ее и находили более-менее разумные решения данной проблемы.

Общая проблема надежности охватывает широкий круг проблем и вопросов, направленных на поддержание и обеспечение надежности устройств, как отдельных элементов, так и системы в целом, совокупность оных.

1 Основные понятия теории надежности

В контексте данной работы мы будем использовать следующие понятия, термины и определения понятий в области надежности, установленные ГОСТ 27.002-2015. «Надежность в технике. Термины и определения»:

- Надежность – свойство объекта сохранять во времени способность выполнять требуемые функции в заданных режимах и условиях применения, технического обслуживания, хранения и транспортирования.
- Безотказность – свойство объекта непрерывно сохранять способность выполнять требуемые функции в течение некоторого времени или на работки в заданных режимах и условиях применения.
- Работоспособность – состояние объекта, при котором он способен выполнять заданные функции, сохраняя значение основных параметров.
- Система – объект, представляющий собой множество взаимосвязанных элементов, рассматриваемых в определенном контексте как единое целое и отделенных от окружающей среды.
- Восстанавливаемость – свойство объекта, заключающееся в его способности восстанавливаться после отказа без ремонта.
- Ремонтопригодность – свойство объекта, заключающееся в его приспособленности к поддержанию и восстановлению состояния, в котором объект способен выполнять требуемые функции, путем технического обслуживания и ремонта.
- Сохраняемость – свойство объекта непрерывно сохранять требуемые эксплуатационные показатели в течение (и после) срока хранения и транспортирования.
- Долговечность – свойство объекта сохранять работоспособность до наступления предельного состояния» [1, с. 2-3].

1.1 Классификация отказов

В настоящее время существуют различные схемы классификации отказов. Одна из схем представлена в таблице 1.

Таблица 1 – Классификация отказов

Классификационный признак	Вид отказа
Характер возникновения отказа	Внезапный Постепенный
Время существования отказа	Постоянный Временный Перемежающийся (временные отказы, следующие один за другим)
Характер проявления отказа	Явный Неявный
Зависимость отказов между собой	Зависимый Независимый
Причина возникновения отказа	Конструктивный Производственный Эксплуатационный Деградационный

Классификация отказов и его виды, установленные ГОСТ 27.002-2015. «Надежность в технике. Термины и определения»:

Отказ – Событие, заключающееся в нарушении работоспособного состояния объекта.

Дефект – каждое отдельное несоответствие объекта требованиям, установленным документацией.

Повреждение – событие, заключающееся в нарушении исправного состояния объекта при сохранении работоспособного состояния.

Вид отказа – единица классификации отказов, исходящей из установленных критериев: характера, причины, последствий отказа, функции, способность выполнения которой потеряна, или изменения состояния объекта.

Критерий отказа – признак или совокупность признаков нарушения работоспособного состояния объекта, установленные в документации.

Ресурсный отказ – отказ, в результате которого объект достигает предельного состояния.

Внезапный отказ – отказ, характеризующийся скачкообразным переходом объекта в неработоспособное состояние.

Постепенный отказ – отказ, возникающий в результате посте пенного изменения значений одного или нескольких параметров объекта.

Систематический отказ – отказ, однозначно вызванный определенной причиной, которая может быть устранена только модификацией проекта или производственного процесса, правил эксплуатации и документации.

Явный отказ – отказ, обнаруживаемый визуально или штатными методами и средствами контроля и диагностирования при подготовке объекта к применению или в процессе его применения.

Скрытый отказ – отказ, не обнаруживаемый визуально или штатными методами и средствами контроля и диагностирования, но выявляемый при проведении технического обслуживания или специальными методами диагностирования.

Конструктивный отказ – отказ, возникший по причине, связанной с несовершенством или нарушением установленных правил и (или) норм проектирования и конструирования.

Производственный отказ – отказ, возникший по причине, связанной с несовершенством или нарушением установленного процесса изготовления или ремонта, выполняемого на ремонтном предприятии.

Эксплуатационный отказ – отказ, возникший по причине, связанной с нарушением установленных правил и (или) условий эксплуатации

Деградационный отказ – отказ, обусловленный естественными процессами старения, износа, коррозии и усталости при соблюдении всех установленных правил и (или) норм проектирования, изготовления и эксплуатации».

1.2 Общая характеристика показателей надежности

Надёжность является комплексным свойством изделия. Для описания различных сторон этого свойства на практике пользуются показателями надёжности, представляющими собой количественные характеристики одного или нескольких свойств, определяющих надёжность изделия.
В таблице 2 можно увидеть основные показатели надежности.

Таблица 2 – Показатели надежности

Показатель надежности	Пояснение
Показатели безотказности	
$P(t)$	Вероятность безотказной работы за заданное время.
$Q(t)$	Вероятность отказа за заданное время.
$T_{\text{ср}}$	Средняя наработка до отказа. Если наработка выражается временем, то показатель называют средним временем безотказной работы.
$\lambda(t)$	Интенсивность отказов за заданное время. Численно равна числу отказавших устройств в единицу времени.
T_0	Средняя наработка на отказ, кратко – наработка на отказ
T_y	Гамма-процентная наработка до отказа.
Показатели ремонтопригодности	
$T_{\text{в}}$	Среднее время восстановления элемента. Представляет математическое ожидание времени восстановления.
$v(t)$	Вероятность восстановления элемента за заданное время.

1.3 Количественные показатели надежности

Вероятность безотказной работы. Под вероятностью безотказной работы изделия за время t понимают вероятность вида:

$$P(t) = \frac{N_0 - n(t)}{N_0} \quad (1)$$

где N_0 – общее количество рассматриваемых элементов;

$n(t)$ – число отказавших элементов за время t.

В случае экспоненциального распределения вероятность безотказной работу будет выглядеть так:

$$P(t) = e^{-\lambda t} \quad (2)$$

где λ – интенсивность отказов;

t – время работы.

Вероятность отказов вычисляется по формуле:

$$Q(t) = 1 - P(t) \quad (3)$$

где $P(t)$ – вероятность безотказной работы.

Еще одним важным показателям надёжности является показатель «интенсивность отказов», определяемый как «отношение числа отказавших объектов (образцов аппаратуры, изделий, деталей, механизмов, устройств, узлов и т. п.) в единицу времени к среднему числу объектов, исправно работающих в данный отрезок времени при условии, что отказавшие объекты не восстанавливаются и не заменяются исправными».

$$\lambda(t) = \frac{n(t)}{[N - n(t)]\Delta t} \quad (4)$$

где N – общее число рассматриваемых объектов;

$n(t)$ – число отказавших объектов в интервале Δt;

Δt – интервал времени.

Если же учитывать, что интенсивность отказов подчиняется экспоненциальному закону распределения, то логарифмируя формулу (2) получаем:

$$\lambda = -\ln P \ / t \quad (5)$$

Где P – вероятность безотказной работы всей системы;

t – Время работы.

Среднее время безотказной работы. Надёжность элементов с точки зрения продолжительности их работы до первого отказа характеризуют средним временем безотказной работы, под которым понимают математическое ожидание времени безотказной работы. В общем случае рассматриваемый показатель называют средней наработкой до отказа, так как он представляет собой математическое ожидание (среднее значение) случайной величины – наработки до отказа. Этот показатель может использоваться для любых изделий: восстанавливаемых и невосстанавливаемых.

$$T_0 = \frac{1}{\lambda} \quad (6)$$

где λ – интенсивность отказов.

Показатели T_0 и $T_{\text{ср}}$ по своей физической сущности различны, однако в случае экспоненциального распределения времени до отказа они совпадают по значению.

2 Методы расчета надежности

Существует несколько видов методов расчета надежности: методы прогнозирования, структурные методы и физические методы.

Методы прогнозирования основаны на использовании оценки ожидаемого уровня надежности объекта, на данных о достигнутых значениях. Методы прогнозирования делятся на несколько видов: эвристического прогнозирования – статическая обработка независимых оценок значений ожидаемых показателей надежности разрабатываемого объекта. Метод прогнозирования по статическим моделям – основана на зависимости, описывающих выявленные тенденции изменения показателей надежности объекта с учетом их технологических особенностей и других факторов информация о которых для разрабатываемого объекта известна или может быть получена в момент проведения оценки.

Структурные методы основаны на том что, нужно представить объект в виде логической структуры, описывающая зависимость состояний и переходов объекта с учетом их взаимодействия и выполняемых ими функций. Структурные методы являются основными методами расчета показателей безотказности, ремонтопригодности и комплексных показателей надежности в процессе проектирования объектов, поддающихся на элементы, характеристики надежности которых в момент проведения расчетов известны или могут быть определены другими методами (прогнозирования, физическими, по статистическим данным, собранным в процессе их применения в аналогичных условиях). Эти методы применяют также для расчета долговечности и сохраняемости объектов, критерии предельного состояния которых выражаются через параметры долговечности (сохраняемости) их элементов. Расчёт показателей надежности структурными методами в общем случае включает:- представление объекта в виде структурное схемы, описывающей логические соотношения между состояниями элементов и объекта в целом с учетом структурно-

функциональных связей и взаимодействия элементов, принятой стратегии обслуживания, видов и способов резервирования и других факторов;- описание построенной структурной схемы надежности объекта адекватной математической моделью, позволяющей в рамкам введенных предположений и допущений вычислить показатели надежности по данным о надежности его элементов в рассматриваемых условиях их применения.

В качестве структурных схем надежности могут применяться:

1. структурные блок-схемы надежности, представляющие объект в виде совокупности определенным образом соединенных элементов;

2. деревья отказов объекта, представляющие графическое отображение причинно-следственных связей, обуславливающих определенные виды его отказов;

3. графы состояний и переходов, описывающих возможные состояния объекта и его переходы из одного состояния в другое в виде совокупности состояний и переходов его элементов.

Математические модели, применяемые для описания соответствующей структурной схемы надежности, определяются видами и сложностью указанных структур, принятыми допущениями относительно видов законов распределения характеристик надежности элементов, точностью и достоверностью исходных данных для расчета и другими факторами.

Ниже рассмотрены наиболее употребительные математические методы расчета показателей надежности, что не исключает возможности разработки и применения других методов, более адекватных структуре и другим особенностям объекта.

Физические методы основаны на применении математических моделей, описывают их физические свойства и иные процессы, приводящие к отказам объекта.

Логико-вероятностные методы анализа надежности сложных технических систем используют математический аппарат бинарной алгебры логики и теорию вероятности. Методы теории массового обслуживания, к

которым относятся дифференциальный метод разложения на фазы, метод Кендалла, позволяют сводить немарковскую модель к марковской. Данные методы позволяют использовать лишь распределения Эрланга и приводят к значительному увеличению числа состояний, поэтому могут использоваться для расчета стационарных характеристик надежности и вероятности безотказной работы для систем кратковременного действия. Логико-вероятностный метод расчета надежности электрической сети с использованием дерева отказов применяется, когда число различных отказов системы относительно невелико (например, для анализа надежности автоматизированной системы диспетчерского управления электроснабжением). Этот метод широко распространился при исследованиях надежности технологических систем АЭС, включая схемы надежного питания установок собственных нужд.

Методы ступенчатой аппроксимации интенсивностей отказов и восстановлений элементов применяются для оценки надежности систем, имеющих незначительное число состояний и медленно изменяющиеся интенсивности.

Для прогнозирования надежности объектов применяют методы эвристического прогнозирования (экспертной оценки). Методы эвристического прогнозирования основаны на статистической обработке независимых оценок значений ожидаемых показателей надежности разрабатываемого объекта (индивидуальных прогнозов), даваемых группой квалифицированных специалистов (экспертов) на основе предоставленной им информации об объекте, условиях его эксплуатации, планируемой технологии изготовления и других данных, имеющихся в момент проведения оценки. Опрос экспертов и статистическую обработку индивидуальных прогнозов показателей надежности проводят общепринятыми при экспертной оценке любых показателей качества методами.

Сущность эвристического метода оценки надежности восстанавливаемых систем заключается в объединении групп элементов этой

системы в один эквивалентный элемент, который характеризуется альтернирующим процессом восстановления. Тем самым происходит уменьшение числа элементов в системе. Метод не позволяет установить погрешность вычислений и применяется исключительно для случая высоконадежных элементов и систем (например, для построения высоконадежных систем постоянного тока для объектов энергетики).

Метод декомпозиции сложных технических систем основан на построении математических моделей, позволяющих получать достаточно точные верхнюю и нижнюю границы оцениваемого показателя надежности. Метод эквивалентирования последовательных и дублированных цепей получил широкое распространение для расчета надежности систем с большим числом элементов при параллельном и последовательном их соединении.

Метод статистического моделирования применяется для исследования поведения вероятностных систем в условиях, когда неизвестны в полной мере внутренние взаимодействия в этих системах. Этот метод заключается в воспроизведении исследуемого физического процесса при помощи вероятностной математической модели и вычислении характеристик этого процесса. Одно такое воспроизведение функционирования системы называют реализацией (или испытанием). После каждого испытания регистрируют совокупность параметров, характеризующих случайный исход реализации. Метод основан на многократных испытаниях построенной модели с последующей статистической обработкой полученных данных с целью определения числовых характеристик рассматриваемого процесса в виде статистических оценок его параметров. Процесс моделирования функционирования технической системы сводится к машинной имитации изучаемого процесса, который копируется на ЭВМ со всеми сопровождающими его случайностями. Метод статистического моделирования является наиболее эффективным, а в ряде случаев - единственно возможным для оценки показателей надежности уникальных или малосерийных изделий, к которым относится оборудование атомных

энергетических установок. Статическая оценка законов распределения отказов применяется для различного оборудования электрических сетей, в том числе для воздушных и кабельных линий.

Методы имитационного моделирования в целом являются универсальными и допускают рассмотрение систем с большим количеством элементов. Однако их использование в качестве метода исследования задач надежности целесообразно лишь тогда, когда трудно или невозможно получить аналитическое решение. Основными этапами такого исследования являются: построение формальной модели, разработка программ имитации траекторий модели, проведение имитационных экспериментов.

При анализе высоконадежных систем с помощью имитационной модели возникают проблемы, связанные с очень большими затратами машинного времени, необходимого для вычислений с требуемой точностью. С увеличением надежности элементов эффективность моделирования уменьшается, и оно становится практически нереализуемым. Методы статистического и имитационного моделирования не позволяют в полном объеме определять надежность системы, если учесть большое количество сопутствующих факторов, влияющих на ее функционирование.

В теории надежности больших систем актуальной задачей является разработка математического аппарата для расчета, анализа и прогнозирования надежности функционирования, позволяющих анализировать технические системы, описываемые уравнениями больших размерностей. При разработке математической модели технической системы с большим числом состояний сталкиваются со следующими препятствиями, существенно затрудняющими анализ ее надежности: неоднозначность понятия отказа системы, взаимовлияние отказов элементов и частей системы, неопределенность исходных данных, многокритериальность, восстанавливаемость. Для оценки показателей надежности сложных технических систем с большим числом состояний используются методы имитационного моделирования, асимптотического анализа, случайных процессов и связанных с ними интегро-

дифференциальных уравнений. В теории надежности предполагается, что технические системы и их компоненты могут пребывать в двух возможных состояниях: работоспособном и отказываем. При этом отказы элементов независимы, и система попадает в состояние отказа при отказе определенного числа элементов. Для сложных систем эти допущения часто бывают неприемлемыми. Между характеристиками отдельных частей системы имеется тесная взаимосвязь, и отказы отдельных частей системы являются зависимыми событиями.

Сложная техническая система является, как правило, многофункциональной. При этом количество выполняемых системой функций может достигать нескольких десятков. В реализации одной функции может участвовать большое число компонентов. Один и тот же компонент может быть задействован в выполнении нескольких функций. Поэтому компоненты, образующие систему, имеют различную длительность эксплуатации. При изучении надежности систем, выполняющих несколько функций, как правило, применяется функциональный подход, при котором описание надежности производится по каждой функции в отдельности, поэтому надежность системы характеризуется вектором показателей надежности всех ее функций. Таким образом, сравнительная оценка различных систем одного и того же назначения часто является затруднительной. Сложные технические системы должны длительное время работать безотказно. Это требование диктуется необходимостью обеспечения высокой их эффективности, безопасности, живучести, готовности и других показателей качества. Сложные системы состоят из десятков и сотен тысяч элементов, а время их работы исчисляется тысячами часов. Надежность элементов непрерывно увеличивается. Появление материалов высокой прочности, защищенных от коррозии, твердых схем, не требующих большой энергии для их питания, существенно уменьшило интенсивность отказов элементов. Однако сложность технических систем и требования к показателям их надежности растут с такой же скоростью, как и надежность элементов.

Поэтому надежность многих сложных технических систем практически не растет. В этом основная проблема надежности техники.

Методы анализа надежности сложных систем должны учитывать: наличие последействия отказов энергетических систем и систем с восстановлением, два характера отказа электротехнических элементов, изменение основного параметра электрической схемы при отказе элементов структурно-резервированной системы, структуру сложной системы, неодновременность работы элементов. Математические модели функционирования сложных систем с точки зрения надежности, полученные без учета перечисленных факторов, не могут быть адекватными реальным системам.

Сравнительный анализ существующих методов показывает, что для оценки надежности и эффективности функционирования каждой сложной технической системы с большим числом состояний необходимо, основываясь на традиционных методах, разработать методику, учитывающую особенности функционирования и своеобразие конкретной системы, позволяющую оценить погрешности вычисления показателей надежности с требуемой точностью.

3 Надежность элементов

3.1 Основная характеристика безотказности элементов

Начиная какие-либо расчеты, необходимо располагать определенными справочными данными о показателях надежности. С уверенностью можно сказать, что одним из важных показателей является интенсивность отказов.

Именно интенсивность отказов указывается во всех справочниках и технической документации. Значение интенсивности отказов берется постоянным в течение определенной наработки. Также в наравне с интенсивностью отказов указываются и условия эксплуатации.

Таблица 3 – Интенсивность отказов основных элементов

Группа элементов	
Резисторы	
Композитные	0.044
Металлизированные	0.034
Проволочные	0.183
Непроволочные	0.179
Терморезисторы	0.007
Конденсаторы	
Слюдяные	0.04
Керамические	0.022
Металлобумажные	0.019
Полупроводниковые приборы	
Диоды выпрямительные	0.091
Диоды импульсные	0.025
Стабилитроны	0.0041
Транзисторы кремниевые	0.065
Транзисторы биполярные	0.044
Элементы соединения	
Переключатели	0.058
Тумблеры	0.1
Кнопки	0.16

Контакты замыкающие	0.0007
Контакты, переключающие	0.018

3.2 Коэффициенты

Таблица 4 – основные коэффициенты нагрузки

Параметр	Пояснение
Составляющие, входящие в модели для всех видов элементов	
$\lambda_Б$	Базовая интенсивность отказов элементов группы (или конкретного типа), отвечающая температуре окружающей среды +25 °С и номинальной электрической нагрузке, т. е. значению коэффициента электрической нагрузки $K_Н = 1$.
$K_Р$	Коэффициент режима работы, зависящий от электрической нагрузки (коэффициента $K_Н$) и температуры корпуса элемента.
K_t	Коэффициент, зависящий от температуры корпуса элемента (компонента).
$K_Э$	Коэффициент эксплуатации, зависящий от жёсткости условий эксплуатации.
$K_П$	Коэффициент приёмки, учитывающий степень жёсткости требований к контролю качества и правила приёмки элементов в условиях производства.

3.3 Методы повышения надежности

Все методы повышения и поддержания надежности делятся на три большие группы: методы, применяемые при проектировании, при изготовлении и во время эксплуатации. Методы повышения надежности, применяемые при проектировании. К таким методам относятся:

1) резервирование;

2) упрощение системы;

3) выбор наиболее надежных элементов;

4) создание схем с ограниченными последствиями отказов элементов;

5) контроль;

6) автоматизация проверок.

В соответствии с ГОСТ 13377-75 различает пять основных вида резервирования:

- структурное;
- информационное;
- временное;
- функциональное резервирование;
- нагрузочное резервирование.

Структурированная избыточность (или аппаратная) – поддержка надежности с помощью использование излишних элементов технической системы. Весь смысл этого способа заключается в том, что вводятся дополнительные или вспомогательные элементы, группы элементов или несколько идентичных систем, обеспечивающие повышение надежности системы.

Информационное резервирование - способ, который предусматривает использование дополнительной или избыточной информации. Реализация этого типа является множественная передача одного сообщения, повторение по каналу связи. Также в качестве примера можно выделить использование специальных кодов, которые используют саморедактирование. Они сами обнаруживают ошибки и исправляют их, если возникает сбой или отказ.

Временное избыточность подразумевает под собой использование чрезмерного времени. Предполагается, что техническая система приобретает возможность возобновления своего функционала, после прерывания своей работы в случае отказа. Происходит восстановление: ремонт или замена отказавшего элемента.

Эти типы резервирования могут быть использованы либо к группе элементов, либо к системе, а также отдельным элементам. Сочетание, можно даже сказать совокупность, видов резервирования в одном объекте, в его построении, называют смешанным. По способу включения резервных элементов различают постоянное, динамическое, резервирование замещением, скользящее и мажоритарное резервирование.

Постоянная избыточность – это резервирование, которые не затрагивает структуру объекта, если возникает отказ. Для постоянного резервирования характерно неиспользование дополнительных или внешних элементов, также такое резервирование говорит, что отсутствуют задержки в работе устройства (рисунок 1 и 2). Постоянное резервирование в простейшем случае представляет собой параллельное соединение элементов без коммутационных устройств.

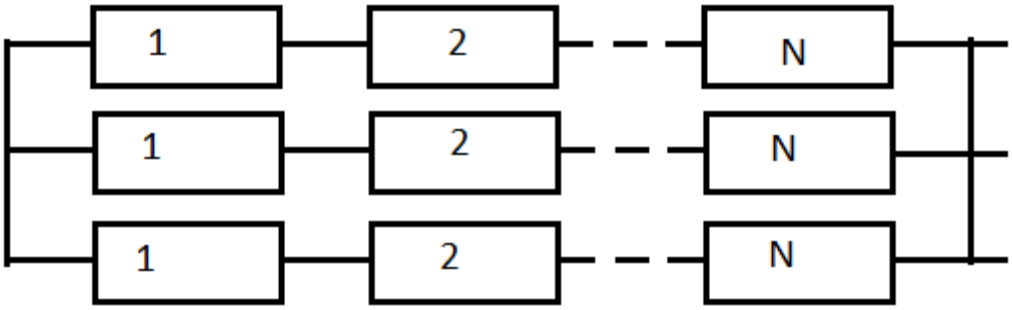


Рисунок 1 – Общее резервирование с постоянно включенным резервом

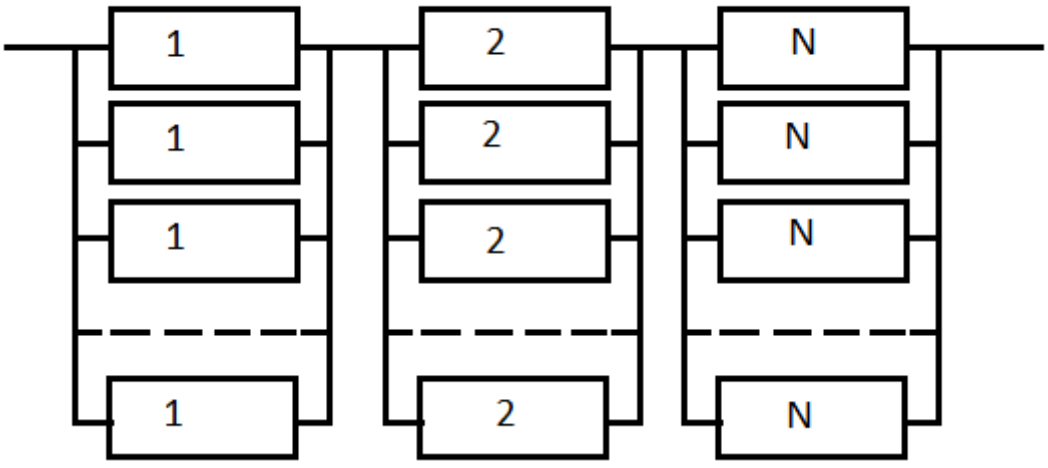


Рисунок 2 – Раздельное резервирование с постоянно включенным резервом

Динамическое резервирование — это резервирование с реструктуризацией схемы систем и узлов, если происходит отказ какого-либо элемента в этой системе. Происходит замена объекта, появляется задержка в работе системы.

Резервирование замещением — это такой тип динамического резервирования, что если происходит полный или частичный отказ, то функции главного элемента передаются резерву.

Включение резерва замещением (рисунок 3 и 4) обладает следующими преимуществами:

- не нарушает работу резервированной системы;

- сохраняет в большей степени надежность резервных элементов, так как при работе основных элементов они находятся в нерабочем состоянии;

- позволяет использовать резервный элемент на несколько основных элементов.

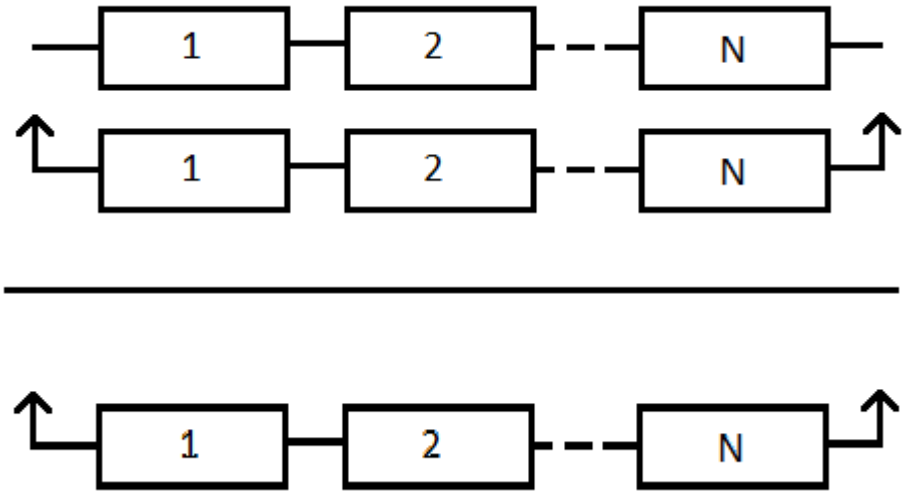


Рисунок 3 – Общее резервирование с включением резерва замещением

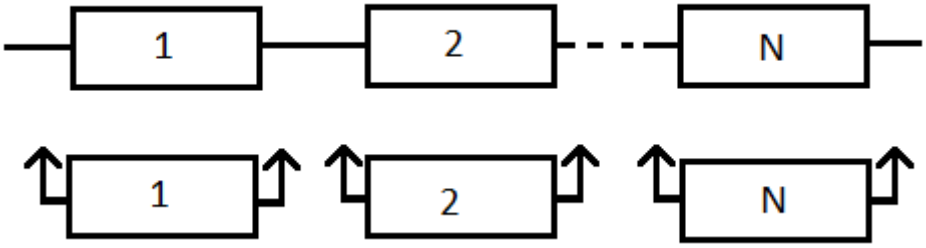


Рисунок 4 – Раздельное резервирование с включением резерва замещением

Резервирование имеет несколько режимов работы в зависимости от поставленных задач: резерв без нагрузки, легкий и нагруженный. Резерв без

нагрузки предполагает, что элементы такого типа резервирования не будут включены в работу, пока не выйдет из строя основной элемент системы. При таком резервировании дополнительные элементы считаются безотказными, так как не включены в работу. Легкий резерв – такой резерв, когда неосновные элементы нагружены меньше, чем основной. Естественно, что когда элементы, нагруженные меньше, то их показатели надежности выше, чем у основного элемента. Нагруженный резерв – это резерв, соединенный параллельно по отношению к основному элементу, и выполняет такую же роль, как и основной элемент.

Резервирование, когда параметры элемента ухудшаются или происходит отказ и такой элемент нельзя восстановить, то говорят, что такое резервирование называется резервирование без восстановления. Если же элемент подлежит ремонту или замене, то в таком случае имеет место резервирование с восстановлением. Чтобы обнаружить отказавший элемент необходимо проводить проверки, если отсутствует резервирование, то это особенно важно.

4. Практическая часть

В ходе выполнения выпускной квалификационной работы был создан программный продукт, позволяющий рассчитать надежность структурной электрической схемы.

После запуска программы появляется интерфейс, где в левом верхнем углу предлагается ввести размер рабочей области. Пример показан на рисунке 5.

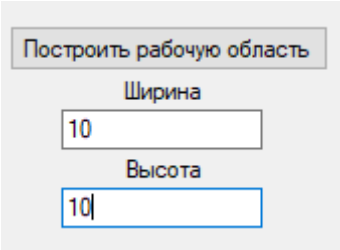


Рисунок 5 – интерфейс ввода размера сетки

При нажатии кнопки “Построить рабочую область” формируется сетка размером, указанном в полях. В данном случае отобразится сетка 10 на 10, как показано на рисунке 6.

Рисунок 6 – сформированная рабочая область

Все данные о элементах загружаются с базы данных Microsoft access. Это позволяет вносит свои элементы и менять существующие. На рисунке 7показана база данных.

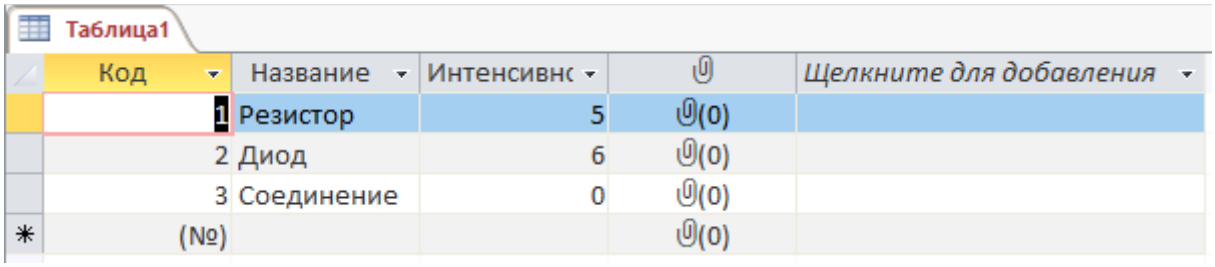


Таблица1

Код	Название	Интенсивнс		Щелкните для добавления
1	Резистор	5	(0)	
2	Диод	6	(0)	
3	Соединение	0	(0)	
(№)			(0)	

Рисунок 7 – База данных

Так база выглядит в программе, как показано на рисунке 8.

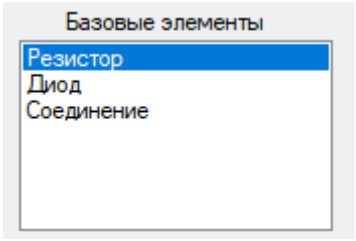


Рисунок 8 – База данных в прогармме

Теперь можно перейти к построению. Выбираем нужным элемент из списка кликом мышки и теперь нажав по рабочей области, выбранный элемент отобразится в ячейки сетки. Элемент “Соединение” имеет несколько состояний, меняя их в зависимости от условий. На рисунке 9 представлено начальное состояние: вокруг нет никаких элементов или все элементы соединены последовательно.

Рисунок 9 – пример соединения

Если же поместить слева и снизу по элементу, то соединение изменит свое начальное состояние и будет представлять собой угловой элемент, как показано на рисунке 10.

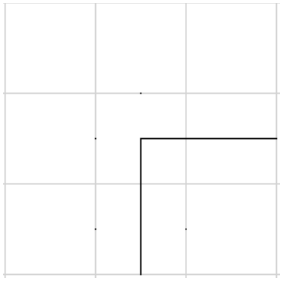


Рисунок 10 – угловое соединение

Алгоритм нахождения объекта соединения электрической цепи, заключается в следующем: при нахождении объекта отличного от нуля, оператор if проверяет соседние точки. Если и снизу, и справа есть объекты, то соединение меняет свое состояние. Пример кода одного из состояний указан в листинге 1.

Листинг 1

```
if (point0[i, j + 1] != 0 && point0[i + 1, j] != 0 && point0[i, j] != 0 && point0[i - 1,
j] == 0 && point0[i, j - 1] == 0)
                         {
                            point0[i, j] = 2;
                            povorotS[i, j] = 1;
                           g.FillRectangle(Brushes.White, i * 64 + 1, j * 64 + 1, 63, 63);
                        g.DrawLine(p, i * 64 + 32, j * 64 + 64, i * 64 + 32, j * 64 + 32);
                        g.DrawLine(p, i * 64 + 32, j * 64 + 32, i * 64 + 64, j * 64 + 32);
                         }
```

Несколько примеров соединений продемонстрированы на рисунке 11.

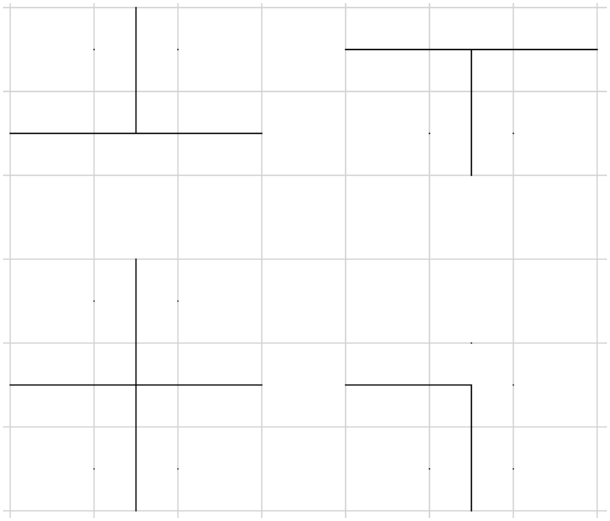


Рисунок 11 – примеры соединений

После того, как схема была простроена и была нажата кнопка “Произвести вычисления”, то в первую очередь запускается алгоритм нахождения связных объектов.

Для определения связности электрической схемы используется рекурсивный алгоритм выделения связных объектов (Листинг 2). Это позволяет установить, что на сетке отсутствует несколько схем. Если выбранная точка не приписана ни к какой области labels [x, y] == 0 и там есть объект, то приписываем эту точку к области и повторяем для всех соседних точек, не заходя за границы рабочей области.

Листинг 2

```
  void Fill(int[,] labels, int[,] count, int x1, int y1, int L1)
            {
                if ((labels[x1, y1] == 0) && (point0[x1, y1] == 1) || (labels[x1, y1] ==
0) && (point0[x1, y1] != 0))
                {
                    labels[x1, y1] = L1;
                    if (x1 > 0)
                        Fill(labels, count, x1 - 1, y1, L1);
                    if (x1 < 20 - 1)
                        Fill(labels, count, x1 + 1, y1, L1);
                    if (y1 > 0)
                        Fill(labels, count, x1, y1 - 1, L1);
                    if (y1 < 20 - 1)
                        Fill(labels, count, x1, y1 + 1, L1);
                }
            }
            int L = 1;
            int y = 0, x = 0;
```

```
            for (y = 0; y < 20; y++)
                for (x = 0; x < 20; x++)
                {
                    if ((labels[x, y] == 0) && (point0[x, y] == 1) || (labels[x, y] == 0)
&& (point0[x, y] != 0))
                    {
                        labels[x, y] = L;
                        if (x > 0)
                            Fill(labels, point0, x - 1, y, L);
                        if (x < 20 - 1)
                            Fill(labels, point0, x + 1, y, L);
                        if (y > 0)
                            Fill(labels, point0, x, y - 1, L);
                        if (y < 20 - 1)
                        {
                            Fill(labels, point0, x, y + 1, L++);
                            grap++;
                        }
                    }
}
```

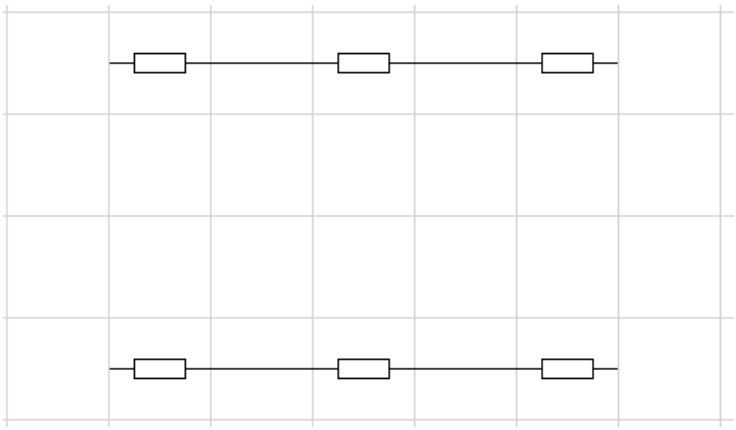
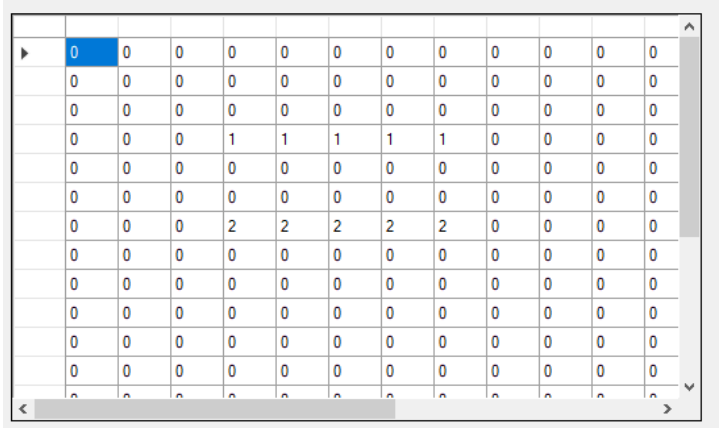


Рисунок 12 – Пример схемы

На рисунке 13 видно, как две схемы отображаются в программе.



0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	2	2	2	2	2	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

Рисунок 13 – Отображение схем в программе

После маркировки нужно определить, где находится последовательные и параллельные соединения. За поиск отвечает алгоритм, представленный в листинге 3. Если при работе алгоритма, находится угловое соединение povorotS[i, j] == 1 и оно не помечено metka[i, j] == 0, то идет поиск второго углового соединения и объекты, которые встречаются в момент поиска, маркируются metka[x1,y1] == 1. Так пока не будут найдены все четыре угловых соединения. Если это условие выполнено, то промаркированная конструкция считается параллельным соединением. Переменные bill и bill2 обозначают крайние элементы параллельной структуры. Если во время прохода найдется ответвление point0[x1, y1] == 3, то в массив leftbill[leftp] поочередно будут заносится вероятности безотказной работы по всем ответвлениям.

Листинг 3

```
for (i = 0; i < 20; i++)
                for (j = 0; j < 20; j++)
                {
                    //Первый поворот
                    if (povorotS[i, j] == 1 && metka[i, j] == 0)
                    {
                        bill = 1;
                        bill2 = 1;
                        Array.Resize(ref parallpointS, parallpointS.Length + 1);
                        x1 = i + 1;
                        y1 = j;
                     }
//
.
.
if (point0[x1, y1] == 3)
          {
            Array.Resize(ref leftbill, leftbill.Length + 1);
            o = x1 - 1;
            while (point0[o, y1] != 3)
               {
                  if (o < 1 || point0[o, y1] == 0)
                  break;
                  leftbill[leftp] = 1;
                 metka[o, y1] = 1;
  leftbill[leftp] *= Math.Pow(Math.E, -(lyambda[o, y1] * Convert.ToInt32(timein.Text)));
                                          o--;
                                      }
                                      leftp++;
                                  }
```

```
.
.
.
//
while (povorotS[x1, y1] == 0)
     {
       metka[x1, y1] = 1;
       y1--;
       if (povorotS[x1, y1] == 1)
       {
         metka[x1, y1] = 1;
         for (int i2 = 0; i2 < leftbill.Length; i2++)
           {
           bill3 *= (1 - leftbill[i2]);
           }
           parallpointS[parallpoint] = 1 - (1 - bill) * (1 - bill2) * bill3;
           parallpoint++;
        }
       }
```

На третьем этапе расчета необходимо выделить последовательные соединения, код представлен в листинге 4. Если объект промаркирован labels [i, j] == 1 и не принадлежит к параллельному соединению metka[i, j] == 0, то все такие соединения считаются последовательными и их вероятности перемножаются.

Листинг 4

```
//Последовательное
                    if (labels[i, j] == 1 && metka[i,j]==0)
                    {
                        consistently *= Math.Pow(Math.E, -(lyambda[i, j] *
Convert.ToInt32(timein.Text)));
                    }
```

Все вероятности безотказной работы параллельных соединений хранятся в массиве parallpointS[] и в конце перемножаются, так как идут последовательно друг за другом. Пример показан в листинге 5.

Листинг 5

```
for (i = 0; i < parallpoint; i++)
            {
                billparal *= parallpointS[i];
            }
```

В результате работы мы имеем отдельные данные о последовательных и параллельных структурах, которые перемножаются и заносятся в переменную ITOGO, и она выводится в соответствующее окно.

Листинг 6

```
ITOGO = billparal * consistently;
 textBox1.Text = ITOGO.ToString();
```

На рисунке 14 представлена структурная схема, созданная в программе.

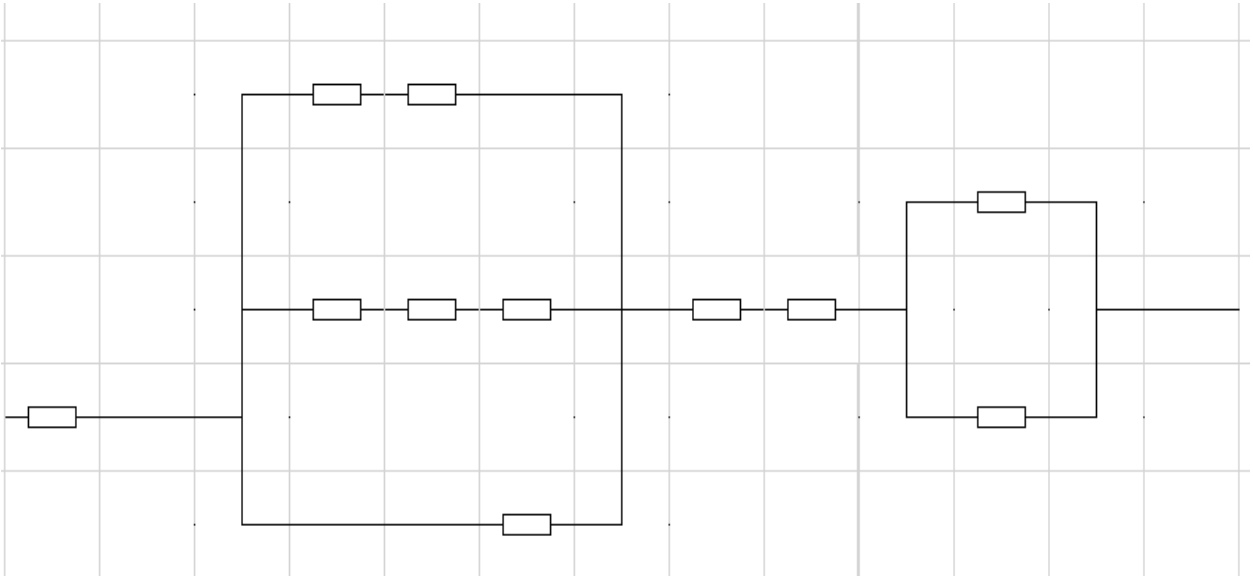


Рисунок 14 – структурная электрическая схема

Считая, что надежность этой схемы подчиняется экспоненциальному закону и не ремонтируется, то вычислим вероятность безотказной работы.

$$P1(65000) = e^{-0{,}000005*65000} = 0{,}72.$$

$$P2(65000) = 1 - (1 - (0{,}72 * 0{,}72) * (1 - 0{,}72 * 0{,}72 * {,}072) * (1 - 0{,}72) = 0{,}91.$$

$$P3(65000) = 0{,}72 * 0{,}72 = 0{,}51.$$

$$P4(65000) = 1 - (1 - 0{,}72)(1 - 0{,}72) = 0{,}92.$$

Таблица 5 – Вероятность безотказной работы

Номер элемента	Вероятность безотказной работы P (t)
1	0.72
2	0.91
3	0.51
4	0.92

Рассчитаем вероятность безотказной работы для всей системы:

$$P(65000) = 0{,}72 * 0{,}91 * 0{,}51 * 0{,}92 = 0{,}3074$$

Рассчитаем, зная время и вероятность безотказной работы, интенсивность отказов для всей системы:

$$\lambda = -\frac{\ln P}{t} = -\frac{\ln 0.3074}{65000} = 0.00001814, \text{час}^{-1}.$$

Средняя наработка до отказа будет равна:

$$T_0 = \frac{1}{\lambda} = \frac{1}{0.00001814} = 55126 \text{ часов}.$$

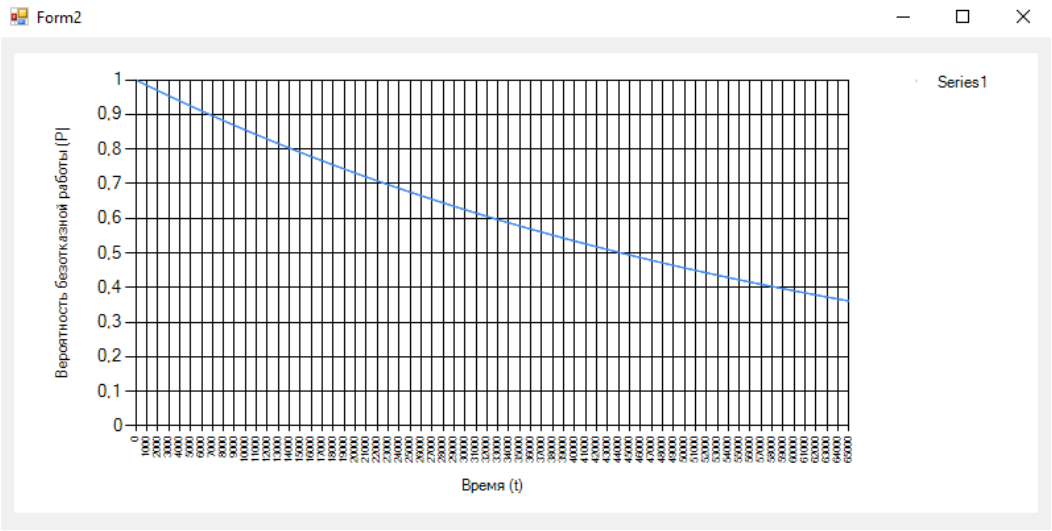


Рисунок 6 – График поведение системы

Заключение

В работе были рассмотрены понятия теории надежности, типы резервирования и виды отказов. На основы полученных знаний был сформирован программный продукт, позволяющий рассчитать надежность структурной схемы.

Список используемых источников

1. ГОСТ 27.002-2015. Надежность в технике. Термины и определения. – Взамен ГОСТ 27.002—89. Введ. 01.02.2017. – М.: Стандартинформ. 24 с.
2. Гук Ю. Б. Расчет надежности схем электроснабжения / Ю. Б. Гук, М. М. Синенко, В. А. Тремясов. – Л.: Энергоатомиздат, Ленингр. отд-ние, 1990. – 216 с.
3. Козлов В.А., Ушаков И.А. Справочник по расчету надежности аппаратуры радиоэлектроники и автоматики. – М.: Советское радио, 1985. – 462с.
4. Матвеевский В.Р. Надежность технических систем / Матвеевский В.Р. Надежность технических систем. Учебное пособие – Московский государственный институт электроники и математики. М., 2002. –113 с.
5. Матвеевский В.Р. Проектирование и надежность устройств автоматики и телемеханики: Учеб. пособие. / В.Р. Матвеевский. – М.: МИЭМ, 1990. – 96с.
6. Синьчугов Ф.И. Расчеты надежности схем электрических соединений / Ф.И. Синьчугов. Москва, Энергия, 1971. – 176 с.
7. СКИТ – система кабельного интерактивного телевидения (лицензии, статьи) [Электронный ресурс]: / – Режим доступа: http://tvskit.narod.ru/stati/stati21/stati21.html
8. Электротехнический портал – Аналитический метод расчёта надёжности электроустановок [Электронный ресурс]: / – Режим доступа: http://электротехнический-портал.рф/nadegnost-electroenergetich-sistem/
9. Викторова В.С. Модели и методы расчета надежности технических систем / В.С. Викторова, А. С. Степанянц Модели и методы расчета надежности технических систем. М., 2016. – 256с.
10. Боровиков С.М. «Теоретические основы конструирования, технологии и надёжности» Учебник для ВУЗов, Дизайн ПРО, 1998г.

11. Боровиков С.М. Погребняков А.В. «Теоретические основы конструирования, технологии и надёжности», Сборник задач, Учебное пособие для ВУЗов, – БГУИР, 2001.

ПРИЛОЖЕНИЕ А

Код программы

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        int leftp = 0, parallpoint = 0;
        int i = 0, j = 0, grap = 0, x1 = 0, y1 = 0, time = 65000, time2 = 0, o = 0;
        bool mostik = false;
        double mnozhitelb = Math.Pow(10, -6);
        Graphics g;
        double bill = 1, ITOGO = 0, consistently = 1;
        double bill2 = 1, bill3 = 1;
        double billparal = 1;
        double lyambdaS = 0;
        Pen p = new Pen(Color.Black, 1);
        Pen cells = new Pen(Color.LightGray, 1);
        //Позиции объекта
        int[,] point0 = new int[20, 20];
        int[,] labels = new int[20, 20];
        int[,] povorotS = new int[21, 21];
        int[,] metka = new int[20, 20];
        double[] leftbill = new double[0];
        double[] raschet;
        double[] parallpointS = new double[0];

        //Интенсивность отказов
        double[,] lyambda = new double[20, 20];
        double[] Znach;
        Form2 f2 = new Form2();
        Form3 f3 = new Form3();
        Image Rimg =
Image.FromFile("C:/Users/C16H16O2N2/source/repos/WindowsFormsApp1/WindowsFormsApp1/R.png"
);
        public Form1()
        {
            InitializeComponent();

            DataGridViewColumn column;
            for (int i = 0; i < 20; i++)
            {
                column = new DataGridViewTextBoxColumn();
                f2.dataGridView1.Columns.Add(column);
                f2.dataGridView1.Rows.Add();
            }
            //ФОН
            pictureBox1.BackColor = Color.White;
            f3.Owner = this;
            panel1.AutoScroll = true;
        }
        private void pictureBox1_MouseClick(object sender, MouseEventArgs e)
        {
            // РИСОВАНИЕ ОБЪЕКТОВ ПО КООРДИНАТАМ МЫШКИ
```

```
            int x = e.X / 64;
            int y = e.Y / 64;
            if (Convert.ToInt32(listBox1.SelectedValue) != 0)
            {
                g.DrawImage(Rimg, x * 64 + 1, y * 64 + 1);
                //   g.DrawEllipse(p, x * 64, y * 64, 64, 64);
                point0[x, y] = 1;
                lyambda[x, y] = Convert.ToDouble(listBox1.SelectedValue) * mnozhitelb;
            }

            if (Convert.ToInt32(listBox1.SelectedValue) == 0)
            {  //Соединения
                for (i = 1; i < 19; i++)
                    for (j = 1; j < 19; j++)
                    {
                        if (point0[i, j - 1] != 0 && point0[i - 1, j] != 0 && point0[i +
1, j] != 0 && point0[i, j] != 0) // ВВЕРХТрезубец
                        {
                            point0[i, j] = 2;
                            povorotS[i, j] = 0;
                            g.FillRectangle(Brushes.White, i * 64 + 1, j * 64 + 1, 63,
63);
                            g.DrawLine(p, i * 64, j * 64 + 32, i * 64 + 64, j * 64 + 32);
                            g.DrawLine(p, i * 64 + 32, j * 64 + 32, i * 64 + 32, j * 64);
                        }
                        if (point0[i, j + 1] != 0 && point0[i - 1, j] != 0 && point0[i +
1, j] != 0 && point0[i, j] != 0) // ВНИЗТрезубец
                        {
                            point0[i, j] = 2;
                            povorotS[i, j] = 0;
                            g.FillRectangle(Brushes.White, i * 64 + 1, j * 64 + 1, 63,
63);
                            g.DrawLine(p, i * 64, j * 64 + 32, i * 64 + 64, j * 64 + 32);
                            g.DrawLine(p, i * 64 + 32, j * 64 + 32, i * 64 + 32, j * 64 +
64);
                        }
                        if (point0[i, j - 1] != 0 && point0[i, j + 1] != 0 && point0[i +
1, j] != 0 && point0[i, j] != 0) //ВНИЗ ВПРАВО
                        {
                            point0[i, j] = 3;
                            povorotS[i, j] = 0;
                            g.FillRectangle(Brushes.White, i * 64 + 1, j * 64 + 1, 63,
63);
                            g.DrawLine(p, i * 64 + 32, j * 64, i * 64 + 32, j * 64 + 64);
                            g.DrawLine(p, i * 64 + 32, j * 64 + 32, i * 64 + 64, j * 64 +
32);
                        }
                        if (point0[i, j - 1] != 0 && point0[i, j + 1] != 0 && point0[i -
1, j] != 0 && point0[i, j] != 0) //ВНИЗ ВЛЕВО
                        {
                            point0[i, j] = 3;
                            povorotS[i, j] = 0;
                            g.FillRectangle(Brushes.White, i * 64 + 1, j * 64 + 1, 63,
63);
                            g.DrawLine(p, i * 64 + 32, j * 64, i * 64 + 32, j * 64 + 64);
                            g.DrawLine(p, i * 64 + 32, j * 64 + 32, i * 64, j * 64 + 32);
                        }
                        if (point0[i, j + 1] != 0 && point0[i + 1, j] != 0 && point0[i -
1, j] != 0 && point0[i, j - 1] != 0 && point0[i, j] != 0) //ЦЕНТР
                        {
                            point0[i, j] = 3;
                            povorotS[i, j] = 0;
                            g.FillRectangle(Brushes.White, i * 64 + 1, j * 64 + 1, 63,
63);
```

```
                            g.DrawLine(p, i * 64, j * 64 + 32, i * 64 + 64, j * 64 + 32);
                            g.DrawLine(p, i * 64 + 32, j * 64, i * 64 + 32, j * 64 + 64);

                        }
                        if (point0[x, y] == 0) // ПРЯМО
                        {
                            point0[x, y] = 2;
                            g.DrawLine(p, x * 64, y * 64 + 32, x * 64 + 64, y * 64 + 32);
                            //break;
                        }
                        if (point0[i - 1, j] != 0 && point0[i, j + 1] != 0 && point0[i,
j] != 0 && point0[i, j - 1] == 0 && point0[i + 1, j] == 0) // УГОЛ ВНИЗ СПРАВА
                        {
                            point0[i, j] = 2;
                            povorotS[i, j] = 1;
                            g.FillRectangle(Brushes.White, i * 64 + 1, j * 64 + 1, 63,
63);
                            g.DrawLine(p, i * 64, j * 64 + 32, i * 64 + 32, j * 64 + 32);
                            g.DrawLine(p, i * 64 + 32, j * 64 + 32, i * 64 + 32, j * 64 +
64);
                        }
                        if ((point0[i, j - 1] != 0 || point0[i, j + 1] != 0) && point0[i,
j] != 0 && point0[i - 1, j] == 0 && point0[i + 1, j] == 0) // ПРЯМО ВНИЗ
                        {
                            point0[i, j] = 2;
                            g.FillRectangle(Brushes.White, i * 64 + 1, j * 64 + 1, 63,
63);
                            g.DrawLine(p, i * 64 + 32, j * 64, i * 64 + 32, j * 64 + 64);
                        }
                        if (point0[i, j - 1] != 0 && point0[i - 1, j] != 0 && point0[i,
j] != 0 && point0[i + 1, j] == 0 && point0[i, j + 1] == 0) //УГОЛ ВВЕРХ СПРАВА
                        {
                            point0[i, j] = 2;
                            povorotS[i, j] = 1;
                            g.FillRectangle(Brushes.White, i * 64 + 1, j * 64 + 1, 63,
63);
                            g.DrawLine(p, i * 64 + 32, j * 64, i * 64 + 32, j * 64 + 32);
                            g.DrawLine(p, i * 64 + 32, j * 64 + 32, i * 64, j * 64 + 32);
                        }
                        if (point0[i, j - 1] != 0 && point0[i + 1, j] != 0 && point0[i,
j] != 0 && point0[i - 1, j] == 0 && point0[i, j + 1] == 0) //УГОЛ ВВЕРХ СЛЕВА
                        {
                            point0[i, j] = 2;
                            povorotS[i, j] = 1;
                            g.FillRectangle(Brushes.White, i * 64 + 1, j * 64 + 1, 63,
63);
                            g.DrawLine(p, i * 64 + 32, j * 64, i * 64 + 32, j * 64 + 32);
                            g.DrawLine(p, i * 64 + 32, j * 64 + 32, i * 64 + 64, j * 64 +
32);
                        }
                        if (point0[i, j + 1] != 0 && point0[i + 1, j] != 0 && point0[i,
j] != 0 && point0[i - 1, j] == 0 && point0[i, j - 1] == 0) //УГОЛ ВНИЗ СЛЕВА
                        {
                            point0[i, j] = 2;
                            povorotS[i, j] = 1;
                            g.FillRectangle(Brushes.White, i * 64 + 1, j * 64 + 1, 63,
63);
                            g.DrawLine(p, i * 64 + 32, j * 64 + 64, i * 64 + 32, j * 64 +
32);
                            g.DrawLine(p, i * 64 + 32, j * 64 + 32, i * 64 + 64, j * 64 +
32);
                        }

                    }
```

```
            }
            pictureBox1.Refresh();
            pictureBox1.Update();
        }

private void button2_Click(object sender, EventArgs e)
        {
            bill = 1;
            bill2 = 1;
            ITOGO = 0;
            consistently = 1;
           // Связнные объекта
           grap = 0;
            Array.Clear(labels, 0, labels.Length);
            void Fill(int[,] labels, int[,] count, int x1, int y1, int L1)
            {
                if ((labels[x1, y1] == 0) && (point0[x1, y1] == 1) || (labels[x1, y1] ==
0) && (point0[x1, y1] != 0))
                {
                    labels[x1, y1] = L1;
                    if (x1 > 0)
                        Fill(labels, count, x1 - 1, y1, L1);
                    if (x1 < 20 - 1)
                        Fill(labels, count, x1 + 1, y1, L1);
                    if (y1 > 0)
                        Fill(labels, count, x1, y1 - 1, L1);
                    if (y1 < 20 - 1)
                        Fill(labels, count, x1, y1 + 1, L1);
                }
            }
            int L = 1;
            int y = 0, x = 0;
            for (y = 0; y < 20; y++)
                for (x = 0; x < 20; x++)
                {
                    if ((labels[x, y] == 0) && (point0[x, y] == 1) || (labels[x, y] == 0)
&& (point0[x, y] != 0))
                    {
                        labels[x, y] = L;
                        if (x > 0)
                            Fill(labels, point0, x - 1, y, L);
                        if (x < 20 - 1)
                            Fill(labels, point0, x + 1, y, L);
                        if (y > 0)
                            Fill(labels, point0, x, y - 1, L);
                        if (y < 20 - 1)
                        {
                            Fill(labels, point0, x, y + 1, L++);
                            grap++;
                        }
                    }
                }
            f2.Show();
            raschet = new double[grap];
            //СЧЕТ
            for (int k = 0; k < grap; k++)
                raschet[k] = 1;
            for (i = 0; i < 20; i++)
                for (j = 0; j < 20; j++)
                {
                    //Первый поворот
                    if (povorotS[i, j] == 1 && metka[i, j] == 0)
                    {
                        bill = 1;
```

```
                        bill2 = 1;
                        Array.Resize(ref parallpointS, parallpointS.Length + 1);
                        x1 = i + 1;
                        y1 = j;
                        //Второй Поворот
                        while (povorotS[x1, y1] == 0)
                        {
                            if (point0[x1, y1] == 0)
                                break;
                            metka[x1, y1] = 1;
                            if (metka[x1, y1] == 1)
                            {
                                bill *= Math.Pow(Math.E, -(lyambda[x1, y1] *
Convert.ToInt32(timein.Text)));
                            }
                            x1++;
                            if (povorotS[x1, y1] == 1)
                            {
                                metka[x1, y1] = 1;
                                y1++;
                                //Третий поворот
                                while (povorotS[x1, y1] == 0)
                                {
                                    if (point0[x1, y1] == 0)
                                        break;
                                    if (point0[x1, y1] == 3)
                                    {
                                        Array.Resize(ref leftbill, leftbill.Length + 1);
                                        o = x1 - 1;
                                        while (point0[o, y1] != 3)
                                        {
                                            if (o < 1 || point0[o, y1] == 0)
                                                break;
                                            leftbill[leftp] = 1;
                                            metka[o, y1] = 1;
                                            leftbill[leftp] *= Math.Pow(Math.E, -
(lyambda[o, y1] * Convert.ToInt32(timein.Text)));
                                            o--;
                                        }
                                        leftp++;
                                    }
                                    metka[x1, y1] = 1;
                                    y1++;
                                    if (povorotS[x1, y1] == 1)
                                    {
                                        metka[x1, y1] = 1;
                                        //Четвертый поворот
                                        x1--;
                                        while (povorotS[x1, y1] == 0)
                                        {
                                            if (point0[x1, y1] == 0)
                                                break;
                                            //    labels[x1, y1] = 10;
                                            metka[x1, y1] = 1;
                                            if (point0[x1, y1] == 1 && metka[x1, y1] ==
1)
                                            {
                                                bill2 *= Math.Pow(Math.E, -(lyambda[x1,
y1] * Convert.ToInt32(timein.Text)));
                                            }
                                            x1--;
                                            if (povorotS[x1, y1] == 1)
                                            {
                                                metka[x1, y1] = 1;
```

```
                                                y1--;
                                                //Возврат к первому
                                                while (povorotS[x1, y1] == 0)
                                                {
                                                    // labels[x1, y1] = 10;
                                                    metka[x1, y1] = 1;
                                                    y1--;
                                                    if (povorotS[x1, y1] == 1)
                                                    {
                                                        metka[x1, y1] = 1;
                                                        for (int i2 = 0; i2 <
leftbill.Length; i2++)

                                                        {
                                                            bill3 *= (1 - leftbill[i2]);
                                                        }
                                                        parallpointS[parallpoint] = 1 -
(1 - bill) * (1 - bill2) * bill3;

                                                        parallpoint++;
                                                    }
                                                }
                                                break;
                                            }
                                        }
                                        break;
                                    }
                                }
                                break;
                            }
                        }
                    }
                    //Последовательное
                    if (labels[i, j] == 1 && metka[i,j]==0)
                    {
                        consistently *= Math.Pow(Math.E, -(lyambda[i, j] *
Convert.ToInt32(timein.Text)));
                    }
                }
            for (i = 0; i < parallpoint; i++)
            {
                billparal *= parallpointS[i];
            }
            ITOGO = billparal * consistently;
            textBox1.Text = ITOGO.ToString();
            lyambdaS = -Math.Log(ITOGO) / time;
            textBox2.Text = (1 / lyambdaS).ToString();
            textBox3.Text = lyambdaS.ToString();
            for (i = 0; i < 20; i++)
                for (j = 0; j < 20; j++)
                {
                    f2.dataGridView1.Rows[j].Cells[i].Value = metka[i, j];
                    f2.dataGridView1.Columns[i].Width = 40;
                }
            Znach = new double[Convert.ToInt32(time)];
            for (time2 = 0; time2 < time; time2++)
            {
                Znach[time2] = Math.Pow(Math.E, -lyambdaS * time2);
                f2.chart1.Series[0].Points.AddXY(time2, Znach[time2]);
            }
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            this.таблица1TableAdapter.Fill(this.dataDataSet.Таблица1);
        }
        private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
```

```
        {
            // textBox1.Text =
(Convert.ToDouble(listBox1.SelectedValue)*mnozhitelb).ToString();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            pictureBox1.Width = 64 * Convert.ToInt32(Widhtcells.Text);
            pictureBox1.Height = 64 * Convert.ToInt32(Heightcells.Text);
            Array.Clear(point0, 0, labels.Length);
            // РИСОВАНИЕ СЕТКИ
            pictureBox1.Image = new Bitmap(64 * Convert.ToInt32(Widhtcells.Text), 64 *
Convert.ToInt32(Heightcells.Text));
            g = Graphics.FromImage(pictureBox1.Image);
            for (i = 0; i < 64 * Convert.ToInt32(Widhtcells.Text); i += 64)
            {
                g.DrawLine(cells, i, 0, i, pictureBox1.Height);
            }
            for (j = 0; j < 64 * Convert.ToInt32(Heightcells.Text); j += 64)
            {
                g.DrawLine(cells, 0, j, pictureBox1.Width, j);
            }
            g.DrawLine(cells, 639, 0, 639, pictureBox1.Height);
            g.DrawLine(cells, 0, 639, pictureBox1.Width, 639);
        }
    }
}
```

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Кафедра “Системы искусственного интеллекта”

УТВЕРЖДАЮ
Заведующий кафедрой

_____ _____
подпись инициалы, фамилия

«_____» _____ 2018 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

09.03.02.04 «Информационные системы и технологии в медиаиндустрии»

Численные методы расчета параметров надежности уникальных систем

Руководитель _____ Ст. преподаватель кафедры Т. Н. Сизова
подпись, дата

Выпускник _____ М. И. Малимонов
подпись, дата инициалы, фамилия

Красноярск 2018