**АННОТАЦИЯ**

Проект содержит 85 страниц машинописного текста, 19 рисунков, 6 таблиц, 8 источников.

В этой работе разработаны информационное обеспечение для моделирования нечеткого контроллера, управляющего подъемно-транспортным механизмом; модель и алгоритм управления подъемно-транспортным механизмом. Она предназначена для улучшения качества управления процессом переноса груза.

**SUMMARY**

This project contains 85 pages of text, 19 diagrams, 6 tables and 8 literature sources.

This bachelor’s project is dedicated to development of fuzzy regulator for container crane. To solve this real-world problem a fuzzy algorithm to control container crane has been developed. Further to model this fuzzy regulator, fuzzy logic modeling software was successfully developed.

**РЕФЕРАТ**

НЕЧЕТКАЯ ПЕРЕМЕННАЯ, НЕЧЕТКИЙ РЕГУЛЯТОР, СИСТЕМА УПРАВЛЕНИЯ, МОДЕЛИРОВАНИЕ НЕЧЕТКИХ СИСТЕМ

Цель работы: разработка информационного обеспечения системы управления подъемно-транспортным механизмом.

В ходе выполнения работы были рассмотрены особенности управления подъемно-транспортным механизмом и процессы, протекающие в системе. Синтезирован алгоритм управления подъемно-транспортным механизмом с учетом различных внешних факторов.

Разработано информационное обеспечение для моделирования нечеткого алгоритма управления подъемно-транспортного механизма.

Решены задачи обеспечения безопасности труда инженера-проектировщика.

Результатом работы является информационное обеспечение системы управления подъемно-транспортным механизмом.

**ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

**VT(t)**- текущая скорость перемещения крана

**αз**- заданное значение угла раскачивания

**α**- измеренное значение угла раскачивания

**δα** - угол раскачивания

 - производная угла раскачивания

**δVT** – разность заданной и измеренной скоростей

**VО** – определяемая скорость

**PMVО** - скорость **VО** положительная (против часовой стрелки) средняя

**PSVО** - скорость **VО** положительная небольшая

**ZRVО** - скорость **VО** нулевая

**NSVО** - скорость **VО** отрицательная (против часовой стрелки) небольшая

**NMVО** – скорость **VО** отрицательная средняя

**PMδα** - угол раскачивания **δα** положительный (против часовой стрелки) средний

**PSδα** - угол раскачивания **δα** положительный небольшой

**ZRδα** - угол раскачивания **δα** нулевой

**NSδα** - угол раскачивания **δα** отрицательный (против часовой стрелки) небольшой

**NMδα** - угол раскачивания **δα** отрицательный средний

**PS** - производная угла раскачивания  положительная небольшая

**ZR** - производная угла раскачивания  нулевая

**NS** - производная угла раскачивания  отрицательная

**PM** - разность скоростей **δVT** положительная (против часовой стрелки) средняя

**PS** - разность скоростей **δVT** положительная небольшая

**ZR** - разность скоростей **δVT** нулевая

**NS** - разность скоростей **δVT** отрицательная (против часовой стрелки) небольшая

**NM** - разность скоростей **δVT** отрицательная средняя

**ПР -** построение регулятора

**ПИД** регулятор – пропорциональный интегро-дифференцирующий регулятор

**ЛП** – лингвистическая переменная

**АЦП** – аналого-цифровой преобразователь

**ЦАП** – цифро-аналоговый преобразователь

**ВДТ** – видеодисплейный терминал

**ПЭВМ** – персональная электронно-вычислительная машина

**ПК** – персональный компьютер

**ПО** – программное обеспечение

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ11](#_Toc201390919)

[1.ОПИСАНИЕ ПОДЪЕМНО-ТРАНСПОРТНОГО МЕХАНИЗМА14](#_Toc201390920)

[1.1.МОДЕЛЬ ПОДЪЕМНО-ТРАНСПОРТНОГО МЕХАНИЗМА14](#_Toc201390921)

[1.3.НЕЧЕТКАЯ ФОРМАЛИЗАЦИЯ ПАРАМЕТРОВ18](#_Toc201390922)

[1.4.АНАЛИЗ МОДЕЛЕЙ ПРИНЯТИЯ РЕШЕНИЙ20](#_Toc201390923)

[2. РАЗРАБОТКА МЕТОДА УПРАВЛЕНИЯ ПОДЪЕМНО-ТРАНСПОРТНЫМ МЕХАНИЗМОМ23](#_Toc201390924)

[3. РАЗРАБОТКА ИНФОРМАЦИОННОГО ОБЕСПЕЧЕНИЯ.33](#_Toc201390925)

[3.1 Структурная организация.33](#_Toc201390926)

[3.2 Алгоритм программы.34](#_Toc201390927)

[3.3 Описание интерфейса36](#_Toc201390929)

[3.4 Реализация37](#_Toc201390930)

[4. ОПИСАНИЕ РАБОТЫ С ИНФОРМАЦИОННым ОБЕСПЕЧЕНИем39](#_Toc201390932)

[4.1 Запуск и выбор опций39](#_Toc201390933)

[4.2 Новый проект39](#_Toc201390936)

[4.3 Лингвистическая переменная.40](#_Toc201390939)

[4.4 Блок правил42](#_Toc201390943)

[5. ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ПРОЕКТА44](#_Toc201390945)

[5.1. СТОИМОСТНАЯ ОЦЕНКА РАЗРАБОТКИ44](#_Toc201390946)

[5.1.1. КАПИТАЛЬНЫЕ ЗАТРАТЫ44](#_Toc201390947)

[5.1.2. ЭНЕРГЕТИЧЕСКИЕ ЗАТРАТЫ.46](#_Toc201390948)

[6. БЕЗОПАСНОСТЬ И ЭКОЛОГИЧНОСТЬ РАБОТЫ47](#_Toc201390949)

[6.1. ОЦЕНКА НАПРЯЖЕННОСТИ ТРУДА РАЗРАБОТЧИКА47](#_Toc201390950)

[6.2. АНАЛИЗ УСЛОВИЙ ТРУДА РАЗРАБОТЧИКА И МЕРОПРИЯТИЯ ПО УСТРАНЕНИЮ ВРЕДНЫХ ФАКТОРОВ52](#_Toc201390951)

[6.3. ЭКОЛОГИЧНОСТЬ РАБОТЫ55](#_Toc201390952)

[6.3.1. КОМПЬЮТЕР КАК ИСТОЧНИК ПЕРЕМЕННОГО ЭЛЕКТРОМАГНИТНОГО ПОЛЯ55](#_Toc201390953)

[7. СОЦИАЛЬНАЯ ЗНАЧИМОСТЬ58](#_Toc201390954)

[ЗАКЛЮЧЕНИЕ59](#_Toc201390955)

[СПИСОК ИСТОЧНИКОВ60](#_Toc201390956)

[ПРИЛОЖЕНИЕ 61](#_Toc201390956)

**ВВЕДЕНИЕ**

В последнее время нечеткое моделирование является одной из наиболее активных и перспективных направлений прикладных исследований в области управления и принятия решений. Нечеткое моделирование оказывается особенно полезным, когда в описании технических систем и бизнес-процессов присутствует неопределенность, которая затрудняет или даже исключает применение точных количественных методов и подходов.

В области управления техническими системами нечеткое моделирование позволяет получать более адекватные результаты по сравнению с результатами, которые основываются на использовании традиционных аналитических моделей и алгоритмов управления. Диапазон применения нечетких методов с каждым годом расширяется, охватывая такие области, как проектирование промышленных роботов и бытовых электроприборов, управление доменными печами и движением поездов метро, автоматическое распознавание речи и изображений.

Нечеткая логика, которая служит основой для реализации методов нечеткого управления, более естественно описывает характер человеческого мышления и ход его рассуждений, чем традиционные формально-логические системы. Именно поэтому изучение и использование математических средств для представления нечеткой исходной информации позволяет строить модели, которые наиболее адекватно отражают различные аспекты неопределенности, постоянно присутствующей в окружающей нас реальности.

В качестве объекта управления был выбран подъемно-транспортный механизм. В настоящее время трудно представить практически любое современное производство без использования подъемно-транспортных механизмов на различных этапах производства. Использование подъемно-транспортных механизмов обусловлено необходимостью контроля траектории перемещаемого груза. Особенно важным это является при проведении различных погрузочных, строительных, и других видов работ связанных с перемещением грузов большой массы и объема.

Груз, поднимаемый краном, раскачивается и при перемещении, и при остановке. Большая начальная скорость перемещения крана является источником колебаний груза на крюке. Обычно эти колебания понижаются посредством уменьшения начальной скорости перемещения крана. Величина раскачивания груза варьируется в зависимости от веса груза, его состояния, метода подвешивания груза на крюке, высоты крана, величины ускорения и т.д.

**Работа является актуальной** т.к. в условиях современного производственного процесса необходимо наиболее быстрое, качественное и безопасное выполнение различных подъемно-транспортных работ, что и призван осуществлять разработанный нечеткий контроллер.

**Целью бакалаврской работы является** разработка ПО для моделирования нечеткого контроллера, управляющего подъемно-транспортным механизмом.

**В работе решаются следующие задачи**:

1. разработка IDE ПО, для моделирование нечетких систем,
2. разработка модели нечеткого контроллера,
3. разработка демонстрационной программы.

Методологическую основу исследований составляет теория адаптивного управления, теория нечетких регуляторов, принципы работы подъемно-транспортных механизмов.

Результаты работы являются практически полезными, т.к. могут найти применение при разработке промышленных нечетких регуляторов для подъемно-транспортных механизмов.

В первом разделе данной работы рассматриваются особенности и проблематика управления системой на различных этапах переноса груза, физический смысл процессов протекающих в системе, приводятся краткие характеристики моделей принятия решений, нечеткая формализация параметров.

Во втором разделе задаются нечеткие и лингвистические переменные, осуществляется определение особенностей управления на каждом этапе перемещения груза, составляются правила “ситуация-действие” позволяющие наиболее качественно осуществлять управление.

В третьем разделе рассматривается программное обеспечение, осуществляющее управление работой микроконтроллера, приводится формальная реализация управляющей программы и рабочий алгоритм ее функционирования.

В четвертом разделе описываются принципы работы с ПО.

В пятом разделе приводится экономическое обоснование актуальности разрабатываемого проекта, определяются капитальные и энергетические затраты.

В шестом разделе производится оценка напряженности труда разработчика, анализ его условий труда и мероприятия по устранению вредных факторов, экологичность работы.

**1. ОПИСАНИЕ ПОДЪЕМНО-ТРАНСПОРТНОГО МЕХАНИЗМА**

**1.1. Модель подъемно-транспортного механизма**

Проектирование моделей, в полной мере описывающих различные характеристики и состояния подъемно-транспортного механизма, практически является не решаемой задачей, так как на практике представляется невозможным учесть все внутренние и внешние факторы, оказывающие влияние на подъемно-транспортный механизм в процессе работы. Например, невозможно определить величину и направление ветра в тот или иной момент работы механизма, так же в процессе функционирования объекта могут возникать различные непредвиденные ситуации, помехи.

Большую проблему при проектировании систем подобного рода представляет расчет сил необходимых для “отрывания” груза от опорной поверхности. При существующих методах расчета эта задача не имеет решения или его очень сложно получить в виде пригодном для дальнейшего применения. С помощью аппарата нечетких множеств, используя знания экспертов, можно существенно упростить решение данной задачи. Рассмотрим подробнее действия оператора и состояния подъемно-транспортного механизма в процессе работы.

В процессе работы оператор крана осуществляет некоторую последовательность действий, с целью перемещения груза из одной точки в другую. Оператор, воздействуя на органы управления подъемно-транспортного механизма, задает скорость движения, угол поворота стрелы, направление движения, одновременно контролируя поведение объекта, наблюдая, чтобы не возникало различных нежелательных и критических ситуаций.

Изменение скорости подъемно-транспортного механизма в процессе работы представлено на рис. 1.1



Рис. 1.1. **О1** – точка начала разгона, **О2** – точка конца разгона, **О3** – точка начала торможения, **О4** – точка конца торможения

Для каждой из точек характерна некоторая совокупность внутренних и внешних факторов, оказывающих влияние на рассматриваемый объект. На рис. 1.2 показана схема разложения на силы при раскачивании груза, который перемешается краном.



Рис. 1.2 **VT(t)** - текущая скорость перемещения крана, **α(t)** - текущий угол отклонения груза, **m** - масса груза

Баланс моментов относительно точки **О** имеет вид: **М1+М2=М3**, где  - момент инерции груза относительно точки подвеса;  - момент, создаваемый составляющей скорости подвеса относительно точки подвеса; **M3=mgl⋅sinα(t)** - момент, создаваемый составляющей веса груза относительно точки подвеса.

После интегрирования и преобразований получим уравнение:

,(1.1)

где **α(t=0)=α0,**  - начальные условия нелинейного интегро-дифференциального уравнения.

Составить дифференциальные уравнения, описывающие подобную динамическую систему достаточно сложно и нет никакой гарантии, что она будет адекватной моделью [2].

**1.2. Принцип работы нечетких гибридных регуляторов**

Нечеткие регуляторы подразделяют на три типа:

1. Логико-лингвистические регуляторы,
2. Аналитические регуляторы
3. Обучаемые нечеткие регуляторы.

У каждого из этих регуляторов свои достоинства и недостатки, но к наиболее значимым недостаткам ЛЛР табличного типа можно отнести их ограниченную размерность (общее число переменных не должно превышать трех и субъективность выбора интервалов и соответствующих значений лингвистических переменных).

Обучаемые нечеткие регуляторы и системы управления относятся к классу наиболее перспективных. Они сохраняют высокую работоспособность в условиях помех и погрешностей измерения, а также достаточно быстро настраиваются на меняющиеся условия производства, снижая тем самым потери от неэффективного управления.

Объединяя другие типы регуляторов в более сложные структуры, проектная цель должна дополняться к подходящим управляющим характеристикам каждого регулятора в общие характеристики гибридного регулятора.

Гибридный регулятор работает как многорежимный регулятор, который имеет три режима операций, управляемых режимом операции модуля действий (Рис. 1.3). Изменение режимов зависит от величин нечетких вводов регулятора. В зависимости от этого установятся следующие соотношения:





Рис. 1.3. Состав гибридного регулятора

**1.3. Нечеткая формализация параметров**

В настоящее время разработаны многочисленные методы, позволяющие сохранять неизменным положение транспортируемого груза относительно заданной траектории при помощи специальных схем подвеса. Однако они не рассчитаны на резкие, непредвиденные изменения ситуации, связанные с влиянием внешних возмущающих факторов (изменение направления, скорости ветра, появления помех, возникновение аварийных ситуаций, требующих перехода системы в специальные режимы работы или немедленной остановки). Так же с помощью традиционных методов ПИД-регулирования невозможно обеспечить приемлемое качество управления при резком начале движения груза в момент отрыва от поверхности опоры и остановке в заданной точке пространства.

Таким образом, числовая информация, поступающая от датчиков (значение угла отклонения, угловой скорости, скорости груза), не позволяет, найти решение формальными методами при существующих ограничениях. В этом случае или нужно существенно округлять исходные данные, что может привести к получению неверного результата, или воспользоваться знаниями экспертов, которые выражаются в нечеткой словесной форме. Таким образом, наиболее целесообразным в сложившихся условиях представляется использование методов нечеткой логики для построения системы стабилизации груза.

Рассмотрим особенности применения аппарата нечетких множеств и нечеткой логики для решения данной задачи. Обработка нечеткой информации в задачах принятия решений (ПР) обеспечивается применением лингвистического подхода (аппарата нечеткой логики) [3].

В рамках лингвистического подхода в качестве переменных допускаются не только числа, но слова и предложения естественного языка, а аппаратом их формализации является теория нечетких множеств. Лингвистический подход при построении моделей принятия решений позволяет:

- применять для описания элементов задачи ПР приближенные, субъективные оценки экспертов, выраженные с помощью нечетких понятий, отношений и высказываний профессионального языка лиц принимающих решения (ЛПР);

- формализовать нечеткие описания с помощью нечетких множеств, лингвистических переменных и нечетких свидетельств;

- оперировать полученными формализованными объектами посредством аппарата, развиваемого на основе теории нечетких множеств;

- представлять результаты решения задачи как в виде нечетких описаний с использованием понятий и отношений профессионального языка экспертов, так и в виде четких рекомендаций;

- формализация нечетких понятий и описаний профессионального языка ЛПР обеспечивается введением понятий нечеткой и лингвистической переменных, нечеткого множества и отношения, что обеспечивает переход от словесных описаний элементов задач ПР к числовым представлениям.

**Нечетким множеством** А на множестве **X** называется совокупность пар вида **={μA(x)/x}, x∈X**, где **μA** - отображение элементов множества **D** в единичный отрезок [0,1], называемое **функцией принадлежности** нечеткого множества **А**. Значение функции принадлежности **μA(x)** для элемента **x∈X** называется **степенью принадлежности.**

Лингвистическая переменная (ЛП) задается набором [5, 6]:

**<αi,T(αi),X,G,M>,** (1.2)

где **αI** - название i - ой ЛП; **T(αi)**- терм-множество ЛП **αi**; **Х**- область определения элементов множества **T(αi)**; **G**- синтаксическое правило (грамматика), порождающее элементы (**j**-е нечеткие переменные) ; **M**- семантическое правило, которое ставит в соответствие каждой нечеткой переменной (НП)  нечеткое множество  - смысл НП .

Нечеткие переменные (НП) , составляющие терм-множества лингвистических переменных **αI**, задаются в виде тройки множеств

(1.3)

где  - наименование НП; **Х**- базовое множество;  - нечеткое подмножество множества **Х**,  - функции принадлежности, задание происходит путем экспертного опроса.

Для всех ЛП терм-множества задаются экспертами и могут, по мнению экспертов, содержать любое количество нечетких переменных (термов). Проведенная данным образом формализация параметров моделей принятия решений позволяет рассматривать не только численные значения, но и ввести экспертные оценки параметров.

**1.4. Анализ моделей принятия решений**

Существует несколько моделей принятия решений:

- модель классификации;

- модель композиции;

- ситуационная модель принятия решений.

Приведем краткий анализ перечисленных моделей принятия решений. Достоинство **модели классификации** при построениинечетких контроллеров состоит в возможности установления достаточно полного соответствия между наборами нечетких переменных, характеризующих состояние объекта управления, и элементами множества принятия решения об управлении. Недостаток: если число ЛП велико, значительны мощности их терм-множеств, то таблица соответствие “ситуация-действие” существенно разрастается.

**Ситуационная модель** принятия решений имеет следующее достоинство, экспертами выделяются некоторые эталонные ситуации в объекте управления, которым сопоставляются в виде соответствия принимаемые решения об управлении, в этом случае нет необходимости задавать правила выбора управлений, упрощается процедура выбора конкретного решения. Недостаток такой же, как и у приведенной выше модели классификации, полноту задания множества всех эталонных ситуаций при значительном количестве входных факторов объекта управления трудно определить априорно.

Применение **модели композиции** позволит избежать недостатков двух выше перечисленных моделей, так же данная модель имеет более упрощенную реализацию, рассмотрим ее более подробно.

Известна модель, названная в работе [7] моделью композиции, которой в работе [8] дано название модели вычисления степени истинности нечетких правил вывода. Модель задается набором [9]:

**(W,T,H)**, (1.4)

где **Т** - нечеткое отношение на множестве **W×H**, причем **Т** - нечеткое соответствие, которое выводится на основе словесно-качественной информации экспертов, причем .

Множество **H** рассматривается как множество НП из терм-множества ЛП - принимаемое решение. Формальное построение модели происходит следующим образом.

Элементы множества **W**- множества, составляющего прямое произведение множеств входных факторов **W=W1×W2×…×Wn**, определяются при конкретной постановке задачи принятия решения. Определяется ЛП - принимаемое решение и задаются НП из терм-множества ЛП- принимаемое решение. Основной частью построения модели является выбор экспертами элементов множества **Т** - соответствия в виде правил нечеткого выбора. Полнота этого множества определяет достоверность работы модели. Эксперт описывает принятие решений в виде некоторого множества **Т**, содержащего высказывания **πj,**. Высказывания **πj** формализуют посредством назначающих, условных и безусловных операторов. Для каждого высказывания **πj** выводится функция принадлежности 

Для отношения **Т** значения функции принадлежности определяются через обобщенную операцию **σ**, так что

(1.5)

Модель вычисления степени истинности нечетких правил вывода имеет вид:

**(W,T,H)**, **W=W1×W2×W3×W4,** (1.6)

Модель работает по следующему алгоритму при принятии решения.

1. Для момента времени **t0** определяется координата множества **W** . Для точки **w0** получают значения функций принадлежности  нечеткого решения выбора **hi**.

2. Выбирается максимальное значение



3. Решение **hs** является выбранным в результате работы модели вычисления степени истинности нечетких правил вывода.

Данная модель будет применена для реализации нечеткого контроллера.

**2. РАЗРАБОТКА МЕТОДА УПРАВЛЕНИЯ ПОДЪЕМНО**-**ТРАНСПОРТНЫМ МЕХАНИЗМОМ**

Сформулируем задачу управления нелинейным объектом - подъемно-транспортным механизмом.

Множество входных сигналов Х в свою очередь состоит из множеств и имеет вид X={X1,X2,X3} где: Х1 - угол отклонения груза, X2 - угловая скорость, X3 – разность заданной и измеренной скоростей.

Введем ЛП γ- угол раскачивания груза на тросе, терм-множество которой имеет вид Т(γ)={γ1,γ2,γ3,γ4,γ5}, где: γ1 - PMδα - угол раскачивания δα положительный (против часовой стрелки) средний; γ2 - PSδα - угол раскачивания δα положительный небольшой; γ3 - ZRδα - угол раскачивания δα нулевой; γ4 - NSδα - угол раскачивания δα отрицательный (против часовой стрелки) небольшой; γ5 - NMδα – угол раскачивания δα отрицательный средний. Для каждой НП задаем нечеткие подмножества .

Необходимо определить величину угла отклонения груза. Для этого необходимо оперировать показаниями датчиков угла положения груза на тросе.

Введем ЛП β - угловую скорость груза, терм-множество которой имеет вид Т(β)={β1,β2,β3}, где: β1 - PS производная угла раскачивания  положительная небольшая; β2 - ZR - производная угла раскачивания  нулевая; β3 - NS производная угла раскачивания  отрицательная небольшая. Для каждой НП задаем нечеткие подмножества .

Введем ЛП ε - разность заданной и измеренной скоростей груза, терм-множество которой имеет вид Т(ε)={ε1,ε2,ε3,ε4,ε5}, где: ε1 - PM - разность скоростей δVT положительная (против часовой стрелки) средняя; NM – разность скоростей δVT отрицательная средняя; ε2 - PS  разность скоростей δVT положительная небольшая; ε3 - ZR  разность скоростей δVT нулевая; ε4 - NS разность скоростей δVT отрицательная (против часовой стрелки) небольшая; ε5 - NMразность скоростей δVT отрицательная средняя. Для каждой НП задаем нечеткие подмножества .

Множество решений **Н** в нашем случае будет иметь следующий вид **H={h1,h2,h3,h4,h5}** где: **h1=**PM**VО** - скорость **VО** положительная (против часовой стрелки) средняя; **h2=**PS**VО** - скорость **VО** положительная небольшая; **h3=**ZR**VО** - скорость **VО** нулевая; **h4=**NS**VО** - скорость **VО** отрицательная (против часовой стрелки) небольшая; **h5=**NM**VО** - скорость **VО** отрицательная средняя.

Важно иметь информацию о разности заданной и измеренной скоростей. Данные о текущей скорости поступают с датчика скорости, заданная скорость устанавливается оператором с помощью органов управления подъемно-транспортного механизма. Необходимо определить скорости **VО(t)** для каждой из точек рис.1.1 по изменениям текущей скорости **VT(t)** и угла раскачивания груза **α(t)**. Оператор крана обычно решает данную задачу эвристическим путем, причем, некоторое лингвистическое правило может формализовать принятие решения оператором о выборе скорости **VО(t)**.

Например искомая скорость в конце торможения для точки **О4** может быть найдена следующим образом с помощью использования нечетких переменных.

Если угол **δα=αз-α**, где **αз** - заданное значение, **α** - измеренное значение **α**; немного увеличивается по часовой стрелке и производная угла  колебания груза немного увеличивается против часовой стрелки и скорость **δVT=VTЗ-VT**, где **VTЗ** – заданное значение скорости, **VT** - измеренное значение скорости; равна нулю, тогда скорость **VО** должна быть небольшой в отрицательном направлении относительно нуля.

Введем лингвистические переменные (ЛП): **δα** - «угол раскачивания»,  - «производная угла раскачивания»; **δVT** – «разность заданной и измеренной скоростей»; **VО** – «определяемая скорость». Согласно работе, для ЛП введем нечеткие переменные (НП).

Определим терм-множество ЛП **δα**: **Т**(**δα**)={PM**δα** - угол раскачивания **δα** положительный (против часовой стрелки) средний; PS**δα** - угол раскачивания **δα** положительный небольшой; **ZRδα** - угол раскачивания **δα** нулевой; **NSδα** - угол раскачивания **δα** отрицательный (против часовой стрелки) небольшой; NM**δα** – угол раскачивания **δα** отрицательный средний}. На рис. 2.1 приведены функции принадлежности **μδα** для нечетких переменных терм-множества Т(**δα**).



Рис. 2.1

Определим терм-множество ЛП : **Т()={PS** - производная угла раскачивания  положительная небольшая; **ZR**- производная угла раскачивания  нулевая; **NS** - производная угла раскачивания  отрицательная}. На рис. 2.2 приведены функции принадлежности  для нечетких переменных терм-множества Т().



Рис. 2.2

Определим терм-множество ЛП **δVT**: **Т(δVT**)={**P**M - разность скоростей **δVT** положительная (против часовой стрелки) средняя; **PS** - разность скоростей **δVT** положительная небольшая; **ZR** - разность скоростей **δVT** нулевая; **NS** - разность скоростей **δVT** отрицательная (против часовой стрелки) небольшая; **NM** – разность скоростей **δVT** отрицательная средняя}. На рис. 2.3 приведены функции принадлежности **μ(δVT)** для нечетких переменных терм-множества Т(**δVT**).



Рис. 2.3

Определим терм-множество ЛП **δVО**: **Т**(**VО**)={**PMVО** - скорость **VО** положительная (против часовой стрелки) средняя; **PSVО** - скорость **VО** положительная небольшая; ZR**VО** - скорость **VО** нулевая; **NSVО** - скорость **VО** отрицательная (против часовой стрелки) небольшая; **NMVО** – скорость **VО** отрицательная средняя}. На рис. 2.4 приведены функции принадлежности **μ(VО)** для нечетких переменных терм-множества Т(**VО**).



Рис. 2.4

Для каждой из точек определим набор правил эвристического алгоритма управления скоростью. Последовательность действий нечеткого контроллера для точек **O1,O2,O3** разбивается на две составляющие:

- стабилизация груза, т.е. на всех этапах движения контролируется положение груза, величина угла отклонения, скорость изменения угла отклонения, скорость движения груза, его отклонение от заданной скорости и, в зависимости от значений поступающих данных, выбирается то или иное решение, в ходе выполнения которого устраняется раскачка;

- перемещение, т.е. после устранения раскачки или если отклонения параметров не выходит за заданные допустимые пределы выполняется перенос груза. Одновременно с перемещением осуществляется контроль, если появляется раскачка, то переходим к стабилизации.

Для точки начала движения **O1** определим следующую базу правил эвристического алгоритма управления оператором скоростью **VO(t)** крана при **VT(t)>0, VTi(t)>VTi-1(t).**

Правила **Ri** стабилизация**:**

**R1:** если угол раскачивания отрицательный (по часовой стрелке) небольшой **δα=NSδα** искорость угла раскачивания отрицательная (направление скорости по часовой**) =NS** и разность заданной и измеренной скоростей положительная небольшая **δVT= PS,** тогдаопределяемая скорость **VО=PSVО;**

**R2:** если угол раскачивания отрицательный (по часовой стрелке) небольшой **δα=NSδα** искорость угла раскачивания нулевая **=ZR** и разность заданной и измеренной скоростей положительная небольшая **δVT= PS,** тогдаопределяемая скорость **VО=PSVО;**

**R3:** если угол раскачивания отрицательный (по часовой стрелке) небольшой **δα=NSδα** искорость угла раскачивания положительная небольшая (направление скорости против часовой**) =PS** и разность заданной и измеренной скоростей положительная небольшая **δVT=PS,** тогдаопределяемая скорость **VО=PSVО;**

Далее аналогичным образом получаем следующее:

**R4:** если **δα=NMδα** и **=NS** и **δVT= PS,** тогда **VО=PMVО;**

**R5:** если **δα=NMδα** и **=ZS** и **δVT= PS,** тогда **VО=PMVО;**

**R6:** если **δα=NMδα** и **=PS** и **δVT= PS,** тогда **VО=PMVО;**

**R7:** если **δα=PSδα** и **=PS** и **δVT= PS,** тогда **VО=NSVО;**

**R8:** если **δα=PSδα** и **=ZR** и **δVT= PS,** тогда **VО=NSVО;**

**R9:** если **δα=PSδα** и **=NS** и **δVT= PS,** тогда **VО=NSVО;**

**R10:** если **δα=PMδα** и **=PS** и **δVT= PS,** тогда **VО=NMVО;**

**R11:** если **δα=PMδα** и **=ZR** и **δVT= PS, тогда VО=NMVО;**

**R12:** если **δα=PMδα** и **=NS** и **δVT= PS, тогда VО=NMVО;**

**R13:** если **δα=NSδα** и **=NS** и **δVT= PM, тогда VО=PSVО;**

**R14:** если **δα=NSδα** и **=ZR** и **δVT= PM, тогда VО=PSVО;**

**R15:** если **δα=NSδα** и **=PS** и **δVT= PM, тогда VО=PSVО;**

**R16**: если **δα=NMδα** и **=NS** и **δVT= PM, тогда VО=PMVО;**

**R17:** если **δα=NMδα** и **=ZS** и **δVT= PM,** тогда **VО=PMVО;**

**R18:** если **δα=NMδα** и **=PS** и **δVT= PM,** тогда **VО=PMVО;**

**R19:** если **δα=PSδα** и **=PS** и **δVT= PM,** тогда **VО=NSVО;**

**R20:** если **δα=PSδα** и **=ZR** и **δVT= PM,** тогда **VО=NSVО;**

**R21:** если **δα=PSδα** и **=NS** и **δVT= PM,** тогда **VО=NSVО;**

**R22:** если **δα=PMδα** и **=PS** и **δVT= PM,** тогда **VО=NMVО;**

**R23:** если **δα=PMδα** и **=ZR** и **δVT= PM,** тогда **VО=NMVО;**

**R24:** если **δα=PMδα** и **=NS** и **δVT= PM,** тогда **VО=NMVО;**

Правила **Ri** разгона**:**

**R25:** еслиугол отклонения груза укладывается в нулевой интервал значений **δα=ZRδα** и угловая скорость нулевая **=ZR** и разность заданной и измеренной скоростей стремится к нулевому значению **δVT= ZR,** тогдаискомое значение скорости должно быть следующим **VО=PSVО;**

Таким же образом образуются следующие нечеткие высказывания:

**R26:** если **δα=ZRδα** и **=ZR** и **δVT= ZR,** тогда **VО=PMVО;**

**R27**: если **δα=ZRδα** и **=ZR** и **δVT= ZR,** тогда **VО=NSVО;**

**R28:** если **δα=ZRδα** и **=ZR** и **δVT= ZR,** тогда **VО=NMVО;**

**R29:** если **δα=ZRδα** и **=ZR** и **δVT= ZR,** тогда **VО=PMVО;**

**O2**  -точка окончания разгона и начала равномерного переноса груза.

Характерным в этот момент времени является то, что разность скоростей **VTз** и **VT** должнаукладываться в интервал **ZR**, т.е. она является минимальной. База правил эвристического алгоритма управления скоростью **VO(t)** для данной точки:

Правила **Ri** стабилизации:

**R1:** если **δα=NSδα** и **=NS** и**VT= ZR** тогда **VО=PSVО;**

**R2:** если **δα=NSδα** и **=ZR** и **δVT= ZR,** тогда **VО=PSVО;**

**R3:** если **δα=NSδα** и **=PS** и **δVT= ZR** тогда **VО=PSVО;**

**R4:** если **δα=PSδα** и **=PS** и**δVT= ZR** тогда **VО=NSVО;**

**R5:** если **δα=PSδα** и **=ZR** и**δVT= ZR** тогда **VО=NSVО;**

**R6:** если **δα=PSδα** и **=NS** и**δVT= ZR** тогда **VО=NSVО;**

Правила **Ri** перемещения**:**

**R7:** если **δα=ZRδα** и **=ZR** и **δVT= ZR,** тогда **VО=PSVО;**

**R8:** если **δα=ZRδα** и **=ZR** и **δVT= ZR,** тогда **VО=PMVО;**

**R9:** если **δα=ZRδα** и **=ZR** и **δVT= ZR,** тогда **VО=NSVО;**

**R10:** если **δα=ZRδα** и **=ZR** и **δVT= ZR,** тогда **VО=NMVО;**

**R11:** если **δα=ZRδα** и **=ZR** и **δVT= ZR,** тогда **VО=PMVО;**

**O3** - точка начала торможения, для нее характерной чертой является **VTi(t)<VTi-1(t).** База правил эвристического алгоритма управления оператором скоростью **VO(t)**.

Правила **Ri** стабилизации:

**R1:** если **δα=NSδα** и **=NS** и **δVT= PN,** тогда **VО=PSVО;**

**R2:** если **δα=NSδα** и **=ZR** и **δVT= NS,** тогда **VО=PSVО;**

**R3:** если **δα=NSδα** и **=PS** и **δVT= NS,** тогда **VО=PSVО;**

**R4:** если **δα=NMδα** и **=NS** и **δVT= NS,** тогда **VО=PMVО;**

**R5:** если **δα=NMδα** и **=ZS** и **δVT= NS,** тогда **VО=PMVО;**

**R6:** если **δα=NMδα** и **=PS** и **δVT= NS,** тогда **VО=PMVО;**

**R7:** если **δα=PSδα** и **=PS** и **δVT= NS,** тогда **VО=NSVО;**

**R8:** если **δα=PSδα** и **=ZR** и **δVT= NS,** тогда **VО=NSVО;**

**R9:** если **δα=PSδα** и **=NS** и **δVT= NS,** тогда **VО=NSVО;**

**R10:** если **δα=PMδα** и **=PS** и **δVT= NS,** тогда **VО=NMVО;**

**R11:** если **δα=PMδα** и **=ZR** и **δVT= NS,** тогда **VО=NMVО;**

**R12:** если **δα=PMδα** и **=NS** и **δVT= NS,** тогда **VО=NMVО;**

**R13:** если **δα=NSδα** и **=NS** и **δVT= NM,** тогда **VО=PSVО;**

**R14:** если **δα=NSδα** и **=ZR** и **δVT= NM,** тогда **VО=PSVО;**

**R15:** если **δα=NSδα** и **=PS** и **δVT= NM,** тогда **VО=PSVО;**

**R16:** если **δα=NMδα** и **=NS** и **δVT= NM,** тогда **VО=PMVО;**

**R17:** если **δα=NMδα** и **=ZS** и **δVT= NM,** тогда **VО=PMVО;**

**R18:** если **δα=NMδα** и **=PS** и **δVT= NM,** тогда **VО=PMVО;**

**R19:** если **δα=PSδα** и **=PS** и **δVT= NM,** тогда **VО=NSVО;**

**R20:** если **δα=PSδα** и **=ZR** и **δVT= NM,** тогда **VО=NSVО;**

**R21:** если **δα=PSδα** и **=NS** и **δVT= NM,** тогда **VО=NSVО;**

**R22:** если **δα=PMδα** и **=PS** и **δVT= NM,** тогда **VО=NMVО;**

**R23:** если **δα=PMδα** и **=ZR** и **δVT= NM,** тогда **VО=NMVО;**

**R24:** если **δα=PMδα** и **=NS** и **δVT= NM,** тогда **VО=NMVО;**

Правила **Ri** торможения:

**R25:** если **δα=ZRδα** и **=ZR** и **δVT= ZR,** тогда **VО=PSVО;**

**R26:** если **δα=ZRδα** и **=ZR** и **δVT= ZR,** тогда **VО=PMVО;**

**R27:** если **δα=ZRδα** и **=ZR** и **δVT= ZR,** тогда **VО=NSVО;**

**R28:** если **δα=ZRδα** и **=ZR** и **δVT= ZR,** тогда **VО=NMVО;**

**R29:** если **δα=ZRδα** и **=ZR** и **δVT= ZR,** тогда **VО=PMVО;**

Для точки **O4** определим следующую базу правил эвристического алгоритма управления оператором скоростью **VO(t)** крана при **VT(t)=0**:

**R1:** если **δα=NSδα** и **=NS** и **δVT=ZR,** тогда **VО=ZRVО;**

**R2:** если **δα=NSδα** и **=ZR** и **δVT=ZR,** тогда **VО=NSVО;**

**R3:** если **δα=NSδα** и **=PS** и **δVT=ZR,** тогда **VО=NSVО;**

**R4:** если **δα=ZRδα** и **=NS** и **δVT=ZR,** тогда **VО=PSVО;**

**R5:** если **δα=ZRδα** и **=ZR** и **δVT=ZR,** тогда **VО=ZRVО;**

**R6:** если **δα=ZRδα** и **=PS** и **δVT=ZR,** тогда **VО=NSVО;**

**R7:** если **δα=PSδα** и **=NS** и **δVT=ZR,** тогда **VО=PSVО;**

На рис. 2.5 показано решение об изменении скорости крана для предотвращения раскачивания груза в точке **O4**. Решение, приведенное на рис 2.5, получено с помощью знаний экспертов наиболее компетентных в данной области. Основываясь на знаниях экспертов, были сформированы нечеткие множества для каждой ЛП, синтезированы модели “ситуация-действие”, наиболее полно и качественно отражающие требования к системе на различных этапах функционирования.



Рис. 2.5

**3. РАЗРАБОТКА** **ИНФОРМАЦИОННОГО ОБЕСПЕЧЕНИЯ**

**3.1 Структурная организация**

В структуре программы можно выделить логические модули. Каждому модулю присущи свои задачи, методы и вызываемые функции. Структурная схема изображена на рис.3.1.

Рис. 3.1 – Структурная схема программы

*Модуль входа в программу* – это основной модуль, который при запуске программы создает графический интерфейс общения пользователя с программой.

*Модуль создания проекта* – этот модуль позволяет создавать новый проект, а так же может загружать ранее созданный проект для моделирования.

*Модуль управления проектом* – этот модуль позволяет создавать лингвистические переменные, производить отладку проекта, создавать блок правил и.т.д.

*Модуль создания блока правил*  – этот модуль отвечает за создание правил, по которым будет осуществляться отладка проекта.

*Модуль пошаговой отладки* – этот модуль позволяет вносить изменения в созданный проект до тех пор, пока не будут получены требуемые показатели качества.

*Модуль создания графического изображения* – этот модуль позволяет создавать графики.

*Модуль создания листинга для МК* – этот модуль позволяет работать с кодом программы, написанной для микроконтроллера.

**3.2 Алгоритм программы.**

В начале работы на экран выводится главное окно программы. Потом пользователь выбирает одну из опций: открыть ранее созданный проект или создать новый проект. Затем, если была выбрана опция «создать новый проект», открывается новое окно, где можно создать лингвистические переменные, за которые отвечает «блок создания лингвистических переменных». После этого пользователь создает блок правил, по которому он может отладить проект.

Алгоритм работы программы изображен на блок-схеме (рис. 3.2.).

Рис. 3.2 Алгоритм работы программы

**3.3 Описание интерфейса**

Программа имеет простой интерфейс и легка в освоении. Графический интерфейс проектируется в среде разработки Visual Basic .Net компании
Microsoft. В данной системе можно быстро и качественно разрабатывать графические приложения, используя готовые компоненты.

Интерфейс программы разработан с учетом информативности получаемых пользователем данных. На рисунке 3.3 показано основное окно программы.

Рис. 3.3 – Основное окно программы.

**3.4 Реализация**

**3.4.1 Модуль входа в программу.**

При запуске программы вызывается функция «static void Main()». Эта функция инициализирует приложение путем вызова функции Application.EnableVisualStyles();. Далее запускается функция, которая составляет стиль программы путем вызова следующей функции Application.SetCompatibleTextRenderingDefault(false); . После нее запускается функция, которая создает диалоговое окно программы путем вызова функции Application.Run(new Form1()); и запускает приложение. В теле этой функции так же предусмотрена обработка исключений try – catch.

**3.4.2 Модуль создания нового проекта.**

При нажатия кнопки «создать новый проект» запускается функция CreateNewProject(); которая отвечает за создание нового проекта. При вызове этой функций запускается форма New\_project, которая содержит соответствующие поля (название, количество лингвистических переменных, количество выходных переменных и.т.д. ) для создания нужного проекта.

Этот модуль так же отвечает за открытие проекта.

**3.4.3 Модуль управления проектом.**

Этот модуль является самым важным модулем. Этот модуль содержит необходимые функции, которые реагируют на различные события, например «создать лингвистическую переменную» или «создать блок правил» и.т.д. В этом классе содержатся следующие функции:

1. NewLingVariable() – эта функция запускает форму new\_ling\_variable, которая отвечает за создание лингвистической переменной. Эта форма содержит соответствующую ЛП информацию (диапазон, название и.т.д.)

2. DebugCurrProject() – эта функция запускает форму, которая содержит все входные и выходные переменные, являющиеся важными и нужными для отладки текущего проекта. Эта форма позволяет изменить состояние системы путем настройки параметров системы.

3. CreateRuleBase() – эта функция отвечает за создание блока правил. Пользователь может составить нужные правила для текущей системы.

4. CreateGraph() – эта функция отвечает за создание графического изображения с помощью блока правил и составленных переменных.

**3.4.4 Модуль пошаговой отладки.**

Модуль пошаговой отладки позволяет отладить проект на основе блока правил. Этот модуль передает модулю, который создает графическое избрание, нужные данные для построения 2-х или 3-х мерных выходных характеристик.

Таким образом реализуется данная программа.

4..**ОПИСАНИЕ РАБОТЫ С ИНФОРМАЦИОННЫМ ОБЕСПЕЧЕНИЕМ**

**4.1 Запуск и выбор опций**

При запуске программы, появляется основная форма программы (см. рис.4.1) Первым этапом работы с программой является выбор опции – New Project (Новый проект) или Open Project (Открыт проект).

Рис. 4.1. Основная форма программы

**4.2 Новый проект**

Если была выбрана опция New project (Новый проект), то открывается следующая форма Generate Project (см. рис. 4.2), которая позволяет ввести количество входных и выходных переменных. Эта форма так же позволяет ввести адрес для сохранения проекта.

Рис. 4.2. Форма для создания нового проекта.

**4.3 Лингвистическая переменная.**

После того как проект будет создан, открывается новая форма (см. рис.4.3), которая называется CreateLingVariable (Создать лингвистическую переменную). Эта форма отвечает за создание и настройку лингвистической переменной. В этой форме можно определить тип переменной (входная/выходная), а так же ее диапазон.

Рис. 4.3 Создание лингвистической переменной.

После того как мы создали и настроили лингвистическую переменную, открывается форма, которая содержит графическое изображение и диапазон переменной.

Рис. 4.4 Лингвистическая переменная Angle

Рис. 4.5 Лингвистическая переменная Distance

Графики функций принадлежности нечетких функций , описывающих значения лингвистической переменной Power изображены на рис. 4.6.

Рис. 4.6. Лингвистическая переменная Power

**4.4 Блок правил**

После того как все переменные будут определены, нужно создать блок правил. В новой форме нужно ввести правила, необходимые для отладки проекта. Блок правил представлен на рис. 4.7

Рис. 4.7 Блок правил.

**4.5 Зависимости входных и выходных величин**

После выполнения указанных выше пунктов получаем проект, который готов к отладке. При нажатия кнопки «Debug» начинается отладка и появляется форма с графиком зависимости входных и выходных параметров ИС (см.рис.4.8)

Рис. 4.8.График зависимости входных и выходных величин.

**5. ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ПРОЕКТА**

Целью данного проекта является создание системы управляющей подъемно-транспортным механизмом. Синтезируемая система имеет более высокие показатели качества, чем существующие аналоги, а именно она обладает более высокой степенью точности, экономичности и быстродействия. Так же проектируемая система управления подъемно-транспортным механизмом позволяет существенно обезопасить производство и труд рабочих, что в свою очередь непременно скажется на экономических показателях.

Данная разработка снизит затраты на обучение управляющего работой подъемно-транспортного механизма персонала без потери качества производственного процесса, безопасности и экономичности, что сыграет немаловажную роль в сложившихся современных условиях производства. Таким образом, **работа является актуальной**, потому что позволит существенно сократить затраты на производстве связанные с необходимостью осуществлять перенос различных грузов, обучение операторов подъемно-транспортных механизмов.

 Покупка разрабатываемого проекта, его адаптация, сопровождение и эксплуатация обойдутся заказчику дешевле, чем эксплуатация обычного подъемно-транспортного механизма.

**5.1. Стоимостная оценка разработки**

**5.1.1. Капитальные затраты**

Капитальные затраты на этапе проектирования Кп рассчитываются по формуле:

Кп = Zп + Мп + Нп,

где Zп – заработная плата проектировщика задачи на всем этапе проектирования Тп(длительность времени проектирования в нашем случае фиксированная величина, она равняется 30 дням, включает в себя формальное решение поставленной перед разработчиком задачи и оформление пояснительной записки);

Мп – затраты за использование ЭВМ на этапе проектирования;

Нп – накладные расходы на этапе проектирования.

Одним из основных видов затрат на этапе проектирования является заработная плата проектировщика, которая рассчитывается по формуле:

Zп = zд ·Тп · (1 + ас /100) (1 + ап /100),

где zд – дневная заработная плата разработчика задачи на этапе проектирования;

ас – процент отчислений на социальное страхование (ас=26%);

ап – процент премий.

Расчет:

zд = 500 руб.;

ап = 15%;

Zп = 500 ·30 · (1 + 26 / 100) (1 + 15 / 100) =21735 (руб.)

Затраты, связанные с использованием ЭВМ Мп определяются по формуле:

М= Cм tм,

где Cм – стоимость 1 часа машинного времени;

tм – необходимое для решения задачи машинное время (час);

Общее время решения задачи 30·8=240 ч. Из них 160 ч. – непосредственно работа с вычислительной техникой. Т. о. tм=160 ч. Стоимость часа машинного времени Cм = 20 руб.;

Расчет:

Cм = 20 руб.;

tм = 160 ч;

М= 160 \* 20 = 3200 руб.

Накладные расходы составляют 80% - 120% от заработной платы, т. е. Нп = 17388 руб.

 Таким образом, итоговая сумма капитальных затрат на этапе проектирования составит:

Кп =21735+3200+17388=42323 (руб.)

**5.1.2. Энергетические затраты.**

Потребность в электрической энергии (кВт. ч.) за определенный период (год, месяц, день) можно определить по формуле:



где ΣМ – суммарная мощность действующих электроустановок (кВт.).

Fд – действительный фонд времени работы оборудования за период (ч).

Кз – коэффициент загрузки оборудования по времени.



Таким образом, расходы на электроэнергию за время проектирования и разработки определяется по формуле:



где: Ц – стоимость 1 квт.ч.;



6. БЕЗОПАСНОСТЬ И ЭКОЛОГИЧНОСТЬ РАБОТЫ

В этом разделе рассмотрена проблема обеспечения безопасности проектировщика и эргономичности рабочего места. Детально описаны проблемы оценки напряженности труда разработчика, анализ условий труда разработчика и мероприятия по устранению вредных факторов, требования к видеодисплейным терминалам (ВДТ) и персональным требования к помещениям для эксплуатации ВДТ и ПЭВМ электронно-вычислительным машинам.

Оценивается напряженность труда разработчика. Это необходимо, для того чтобы знать условия, в которых велась разработка данной работы. После этого проводится анализ условий труда, и в зависимости от характера этих условий будут приведены мероприятия по устранению вредных факторов. И в конце будут приведены требования к видеодисплейным терминалам и помещениям, содержащие ВДТ.

**6.1. Оценка напряженности труда разработчика**

Труд и здоровье взаимосвязаны и взаимообусловлены. Особое внимание следует обратить на то, что за последние годы в сфере эмоций и умственной деятельности многочисленных работников различных профессий наблюдается значительная интенсификация. Работники часто не успевают адекватно и вовремя реагировать на всю производственную и семантически значимую информацию. Накапливается все больше неотреагированных эмоций и нереализованных задач производственного характера. В связи с этим адаптационно-компенсаторные механизмы, выработанные в процессе эволюции, с трудом справляются с повышенной производственной нагрузкой. Можно сказать, что темпы адаптации организма иногда отстают от темпов возрастания производственных требований. При этом под влиянием хронического воздействия производственных факторов перенапряжение центральной нервной системы может принимать устойчивый характер и тем самым провоцировать развитие «информационных» неврозов.

Необходимо создать оптимальные условия для деятельности центральной нервной системы. Нельзя полностью освободить работника от психоэмоционального напряжения (и вряд ли следует стремиться к этому). Но необходимо повысить устойчивость адаптационных механизмов организма к эмоциональным стрессорам и упорядочить деятельность работающих. При этом целесообразно, с одной стороны, правильно использовать резервы организма, с другой – обнаружить отклонения и недостатки этих механизмов, чтобы вовремя их устранить.

Необходимо оценить условия труда по факторам напряженности трудового процесса. Для обозначения класса условий труда используются следующие обозначения: оптимальные (класс 1.0); допустимые (класс 2.0); вредные с четырьмя степенями вредности: 1 (класс 3.1), 2 (класс 3.2), 3 (класс 3.3), 4 (класс 3.4) и опасные (экстремальные) условия труда (класс 4.0). Данный метод классификации позволяет при выставлении общей оценки учитывать комбинации и сочетания всех факторов производственной среды и трудового процесса. Такая система оценки позволяет прогнозировать риск развития профессиональных заболеваний или других нарушений здоровья.

Сведем результаты оценки напряженности трудового процесса разработчика в таблицу 6.1

Таблица 6.1 Оценка напряженности разработчика

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование фактора | Заключение | Оценка |
| Интеллектуальная нагрузка |
| 1 | Содержание работы | Эвристическая (творческая) деятельность, требующая решения сложных задач | 3.2 |
| 2 | Восприятие сигналов (информации) и их оценка | Восприятие информации и последующее ее сопоставление с заданием  | 3.1 |
| 3 | Степень сложности задания | Обработка, проверка и контроль над выполнением задания | 3.1 |
| 4 | Характер выполняемой работы | Работа по установленному графику с возможностью его коррекции по ходу деятельности | 2.0 |
| Сенсорные нагрузки |
| 5 | Длительность сосредоточенного наблюдения (в % от времени смены) | От 51 до 75 % | 3.1 |
| 6 | Плотность звуковых, световых сигналов и сообщений в среднем за 1 час работы | До 1500 | 3.2 |
| 7 | Размер объекта различения (при расстоянии от глаз работающего до объекта различения не более 0,5 м) в мм при длительности сосредоточенного наблюдения (% времени смены) | более 5 мм -60% | 1.0 |
| 8 | Работа с оптическими приборами (микроскопы, лупы и т. п.) при длительности сосредоточенного наблюдения (% времени смены) | Исключена | 1.0 |
| 9 | Наблюдение за экранами видеотерминалов (часов в смену): при буквенно-цифровом типе отображения информации и при графическом типе отображения информации | Менее 6 часов | 3.2 |
| 10 | Нагрузка на слуховой анализатор  | Минимальна | 1.0 |
| 11 | Нагрузка на голосовой аппарат  | Минимальна | 1.0 |
| 12  | Число производственных объектов одновременного наблюдения | До 8 | 2.0 |
| Эмоциональные нагрузки |
| 13 | Степень ответственности. Значимость ошибки. | Несет ответственность за выполнение отдельных элементов задачи. Ошибка влечет за собой дополнительные усилия в работе со стороны работника | 1.0 |
| 14 | Степень риска для собственной жизни | – | 1.0 |
| 15 | Степень риска при обеспечении безопасности других лиц | – | 1.0 |
| Монотонность нагрузок |
| 16 | Продолжительность (в секундах) выполнения простых производственных заданий или повторяющихся операций | От 25 до 100 | 2.0 |
| 17 | Число элементов (приемов), необходимых для реализации простого задания или в многократно повторяющихся операциях | 9 - 6 | 2.0 |
| 18 | Время активных действий (в % к продолжительности смены). | 20 и более | 1.0 |
| 19 | Монотонность производственной обстановки (в % от времени смены) | менее 75 | 3.1 |
| Режим работы |
| 20 | Фактическая продолжительность рабочего дня | В пределах 5-6 часов | 1.0 |
| 21 | Сменность работы | Односменная работа без ночной смены | 1.0 |
| 22 | Наличие регламентированных перерывов и их продолжительность  | Перерывы регламентированы, достаточной продолжительности: 7 % и более от рабочего времени | 1.0 |

Для определения оценки общей напряженности труда разработчика составим таблицу 6.2

Таблица 6.2

Оценка общей напряженности труда разработчика

|  |  |
| --- | --- |
| Фактор (оценка) | Количество факторов |
| Фактор с оценкой 1.0 | 12 |
| Фактор с оценкой 2.0 | 4 |
| Фактор с оценкой 3.1 | 4 |
| Фактор с оценкой 3.2 | 2 |

Согласно алгоритму общей оценки напряженности получаем общую оценку напряженности, равную 3.1, что соответствует вредным условиям труда, обусловленным превышением воздействия вредных факторов гигиенических норм. Такие условия вызывают, как правило, обратимые функциональные изменения и повышают риск развития профзаболеваний.

При анализе таблицы 6.2 рассмотрим факторы с оценкой 3.1 и 3.2 более детально:

– содержание работы (разработка алгоритмов функционирования программного обеспечения и протоколов взаимодействия компонент системы);

– восприятие сигналов (информации) и их оценка (при отладке программы обязательно отслеживание работы программы согласно разработанному алгоритму);

– степень сложности задания (необходимость разработки пакета программ с использованием разнообразных алгоритмов, сложность отладки программы в режиме реального времени работы аппаратуры, сопровождение пакета программ документацией);

– наблюдение за экранами видеотерминалов, количество часов за смену (кроме использования машинного времени для разработки системы, необходимо создание сопроводительной документации к программе, что так же производится с использованием прикладных программных пакетов).

– длительность сосредоточенного наблюдения в % от времени смены (необходимо сосредотачивать внимание в процессе работы, так как допущение даже малого процента ошибок заставляет применять длительную трассировку программ).

**6.2.** **Анализ условий труда разработчика и мероприятия по устранению вредных факторов**

Работа с вычислительной техникой по вредности относится к безопасным (риск смерти на человека в год составляет менее 0,001). Тяжесть труда у операторов вычислительной техники также минимальна, так как уровень психической нагрузки по этому роду деятельности предусматривает энергозатраты 2000...2400 ккал в сутки.

Работа за компьютером связана с нагрузками на различные системы организма оператора. Напряженность умственного труда при работе на компьютере связана с функциями центральной нервной системы, предъявляющей особые требования к памяти, мышлению, вниманию человека, а также связана с высокой ответственностью оператора за выполняемую работу. Если оператор слишком долго занят работой за компьютером, то у него появляется нервно-психическое напряжение, а в ряде случаев развивается состояние перенапряжения, которое рассматривается в качестве одной из причин развития невротических нарушений и возникновения сердечно-сосудистой патологии. Постоянное наблюдение за объектами на экране монитора приводит к напряжению органов зрения. Выделяют две причины возникновения этих напряжений: «техническая» и «режимная». Первая связана с особенностями экранного изображения: низкой контрастностью, частыми перепадами яркости и мерцанием экрана. «Режимная» причина обусловлена временем работы за компьютером, то есть длительностью наблюдения или продолжительностью фиксации. Астенопии (от греч. asthenēs – слабый + ōps – глаз – слабость или быстрая утомляемость зрения; проявляется болью в глазах, головной болью и снижением остроты зрения) могут отмечаться через 40-45 минут после начала работы за компьютером. Более чем 4-х часовое пребывание у экрана компьютера может привести к кумуляции утомления, о чем свидетельствует нарушение ночного сна у пользователей, которые постоянно работают на компьютере более 4-х часов, по сравнению с лицами, не использующими компьютеров, или при стаже менее двух лет работы на ЭВМ.

Работа с вычислительной техникой связана с необходимостью длительно находиться в вынужденной рабочей позе, что ведет к различным формам заболеваний опорно-двигательного аппарата человека. Отсутствие физической нагрузки и неподвижность - явление не соответствующее естественному (физиологическому) состоянию человека. При этом не стимулируется деятельность внутренних органов, что ведет к застойным явлениям, неблагоприятно отражающимся на общем тонусе организма и психической деятельности. Мониторы являются источниками тепловыделений, которые являются причиной повышения температуры и снижения влажности воздуха на рабочем месте, вызывающих раздражение кожи.

Рассмотрев опасные факторы разработаем, по возможности, методы устранения их вредного влияния. Сразу оговоримся, что такие факторы как содержание работы, восприятие сигналов (информации) и их оценка, степень сложности задания, мы устранить не сможем, так как эти факторы являются неотъемлемой частью данной работы.

Воздействие остальных факторов можно ослабить, если разработать мероприятия по устранению их вредного влияния, которые будут одинаковы как для разработчика, так и для пользователя и которые сводятся к разработке рационального режима труда и отдыха, введением в рабочий график дополнительных перерывов для психологической разрядки и снижения утомляемости зрительного аппарата:

– при восьмичасовом рабочем дне и уровне нагрузки в 5 часов непосредственной работы с терминалами суммарное время регламентированных перерывов должно составлять 70 минут;

– через каждые 15-20 минут следует проводить гимнастику для глаз, через 30-40 минут – комплекс физических упражнений с целью снижения нервно-эмоционального напряжения, утомления зрительного анализатора, устранения влияния гиподинамии (пониженная подвижность вследствие уменьшения силы движений) и гипокинезии (вынужденное уменьшение объема произвольных движений вследствие характера трудовой деятельности – малая подвижность);

– необходимо выдерживать оптимальное расстояние глаз до экрана монитора – 60-70 см;

– регламентированные перерывы следует устанавливать через 1,5-2,0 часа от начала рабочей смены и через 1,5-2,0 часа после обеденного перерыва продолжительностью 20 минут каждый или продолжительностью 15 минут через каждый час работы. Рекомендуется также самостоятельно корректировать длительности перерывов и отдыхов и применять индивидуальный подход в организации времени работы с ЭВМ;

– продолжительность непрерывной работы с видеодисплеями без регламентированного перерыва не должна превышать 2-х часов;

– использовать прямой контраст;

– использовать рассеянный свет;

– экран дисплея располагать ниже уровня глаз с наклоном.

Рассматривая данную работу с точки зрения экологичности можно отметить, что компьютеры являются потенциальными источниками электромагнитных излучений, включая рентгеновское (невидимое излучение, способное проникать, хотя и в разной степени, во все вещества; представляет собой электромагнитное излучение с длиной волны порядка 10-8 см), ультрафиолетовое (источниками УФИ может быть естественное и искусственное освещение), инфракрасное (источником инфракрасного освещения является любое нагретое тело) и ионизирующее (вид излучения, взаимодействие которого со средой приводит к образованию электрических зарядов разных знаков). Электромагнитные излучения воздействуют на сердечно-сосудистую систему и приводят к нарушению эндокринных и обменных процессов. Высокая или низкая температура воздуха также отрицательно сказывается на функциональном состоянии человека.

**6.3. Экологичность работы**

**Излучательные характеристики монитора:**

– электромагнитное поле монитора в диапазоне частот 20 Гц- 1000 МГц;

– статический электрический заряд на экране монитора;

– ультрафиолетовое излучение в диапазоне 200- 400 нм;

– инфракрасное излучение в диапазоне 1050 нм- 1 мм;

– рентгеновское излучение > 1,2 кэВ.

**6.3.1. Компьютер как источник переменного электромагнитного поля**

Основными составляющими частями персонального компьютера являются: системный блок и разнообразные устройства ввода и вывода информации: клавиатура, дисковые накопители, принтер, сканер, и т. п. Каждый персональный компьютер включает средство визуального отображения информации – монитор или дисплей. Персональные компьютеры часто оснащают сетевыми фильтрами, источниками бесперебойного питания и другим вспомогательным электрооборудованием. Все эти элементы при работе персонального компьютера формируют сложную электромагнитную обстановку на рабочем месте пользователя (см. таблицу 6.3).

Таблица 6.3

Персональный компьютер как источник электромагнитного поля

|  |  |
| --- | --- |
| Источник | Диапазон частот  |
| **Монитор** сетевой трансформатор блока питания | 50 Гц |
| Статический преобразователь напряжения в импульсном блоке питания | 20 - 100 кГц |
| Блок кадровой развертки и синхронизации | 48 - 160 Гц |
| Блок строчной развертки и синхронизации | 15 110 кГц |
| Ускоряющее анодное напряжение монитора (только для мониторов с ЭЛТ) | 0 Гц (электростатика) |
| **Системный блок (процессор)** | 50 Гц - 1000 МГц |
| **Устройства ввода/вывода информации** | 0 Гц, 50 Гц |
| **Источники бесперебойного питания** | 50 Гц, 20 - 100 кГц |

Кроме того, на рабочем месте пользователя источниками более мощными, чем компьютер, могут выступать объекты, неполный перечень которых приведен в таблице 6.4

Таблица 6.4

Внешние источники электромагнитного поля на рабочем месте пользователя ПК

|  |  |
| --- | --- |
| Источник | Диапазон частот (первая гармоника) |
| ЛЭП | 50 Гц |
| Трансформаторные подстанции | 50 Гц |
| Распределительные щиты | 50 Гц |
| Электропроводка | 50 Гц |
| Бытовые и конторские электроприборы  | 50 Гц |
| Телевизоры | 0- 15,6 кГц |
| Радиостанции ДВ | 30- 300 кГц |
| Соседние ПК | 0- 1000 МГц |

Шведский институт защиты от излучений, разработчик спецификаций стандарта безопасности MPR II, в своем отчете приводит результаты измерений электромагнитного поля 150 моделей мониторов (см. таблицу 6.5)

Таблица 6.5

**Максимальные и средние величины электромагнитного излучения по данным Шведского института защиты от излучений**

|  |  |  |
| --- | --- | --- |
|  | **Среднее значение** | **Максимальное значение** |
| **Расстояние** | **0,5 м** | **0,3 м** | **0,5 м** | **0,3 м** |
| **Направление излучения** | **по оси** | **вокруг** | **по оси** | **вокруг** | **по оси** | **по оси** |
| **Вид поля, диапазон частот, единица измерения** |  |
| магнитное поле, 5Гц- 2кГц, нТл | <200 | <200 | <200 | 260 | 500 | 730 |
| магнитное поле, 2- 400 кГц, нТл | <10 | 13 | – | 52 | 52 | – |
| электрическое поле, 5Гц- 2кГц, В/м | <10 | – | 17 | 74 | – | 152 |
| электрическое поле, 2- 400 кГц, В/м | 1,7 | 1,9 | 4,2 | 12 | 12 | 32 |
| электростатический потенциал, В | 500 | 500 | 500 | 19900 | 19000 | 19000 |

Наличие в помещении нескольких компьютеров со вспомогательной аппаратурой и системой электропитания создает сложную картину электромагнитного поля. Очевидно, что электромагнитная обстановка в помещениях с компьютерами крайне сложная, распределение полей неравномерное, а уровни достаточно высоки, чтобы говорить об опасности их биологического действия.

**7. СОЦИАЛЬНАЯ ЗНАЧИМОСТЬ**

Данная работа имеет большое социальное значение, так как она призвана существенно облегчить труд большого количества рабочих, занятых в различных сферах производства. Результаты проекта являются практически полезными и могут быть использованы при разработке автоматизированных систем управления подъемно-транспортными механизмами. Задачи, решенные в ходе проектирования, позволят обезопасить труд рабочего персонала, ускорить производственный процесс, сэкономить затраты на обучения операторов, различные ресурсы.

Использование методов нечеткой логики позволяет значительно облегчить разработку управляющей системы, осуществлять управление системой в особых точках, там, где управление с использованием обычных алгоритмов невозможно или управление не удовлетворяет требуемым параметрам качества.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения работы был разработан гибридный регулятор для управления подъемно-транспортным механизмом, модель управления, алгоритм.

Были рассмотрены и оценены существующие требования к процессу управления, внешние факторы, функционирования системы в особых точках.

Для решения данной задачи был применен аппарат нечеткой логики и на его основе разработан алгоритм управления процессом переноса груза.

В ходе разработки проекта была рассмотрена его экономическая целесообразность, а так же глубоко проработана проблема обеспечения безопасности инженера- проектировщика.

**СПИСОК ИСТОЧНИКОВ**

1. Методы робастного, нейро-нечеткого и адаптивного управления: Учебник/Под ред. Н.Е.Егупова; Издание 2-е. – М.: Изд-во МГТУ им. Н.Э.Баумана, 2002. – 744 с.

2. Финаев В.И. Модели систем принятия решений: Учеб. пособие. Таганрог: ТРТУ, 2005г. – 118 с.

3. Нечеткие множества в моделях управления и искусственного интеллекта/А.Н.Аверкин, И.З.Батырин, А.ф.Блиншун, Б.В.Силаев, Б.Н.Тарасов. ‑ М.: Наука, 1986. ‑ 312 с.

4. Финаев В.И., Белоглазов Д.А. Микропроцессорный нечеткий регулятор подачи топлива//Материалы VII Всероссийской научной конференции студентов и аспирантов «Техническая кибернетика, радиоэлектроника и системы управления». Таганрог, ТРТУ, 2004.

5. Заде Л. Понятие лингвистических переменных и его применение к принятию приближенных решений. - М.: Мир, 1976. - 165 с.

6. Zadeh L.A. Fuzzy logic and approximate reasoning // Synthese, 1975. - V. 80. P. 407 - 428.

7. Мелихов А.Н., Баронец В.Д. Проектирование микропроцессорных устройств обработки нечеткой информации. ‑ Ростов-на-Дону.: Изд-во Ростовского университета, 1990. - 128 с.

8. Берштейн Л.С., Финаев В.И. Адаптивное управление с нечеткими стратегиями. – Ростов-на-Дону: Изд-во Рост. ун-та, 1993. - 134 с.

**ПРИЛОЖЕНИЕ**

**Листинг программы**

**1) FuzzyRule.cs**

/\*

 \* biblioteka dlya cozdaniya ne4etkovo block’a plavil

 \*/

using System;

using System.Collections.Generic;

namespace AI.Fuzzy.Library

{

 // Alias for a fuzzy single condition

 using FuzzyCondition = SingleCondition<FuzzyVariable, FuzzyTerm>;

 // Alias for a fuzzy conclusion

 using FuzzyConclusion = SingleCondition<FuzzyVariable, FuzzyTerm>;

 // Alias for a conclusion for Sugeno fuzzy systems

 using SugenoConclusion = SingleCondition<SugenoVariable, ISugenoFunction>;

 /// <summary>

 /// And/Or operator type

 /// </summary>

 public enum OperatorType

 {

 /// <summary>

 /// And operator

 /// </summary>

 And,

 /// <summary>

 /// Or operator

 /// </summary>

 Or

 }

 /// <summary>

 /// Hedge modifiers

 /// </summary>

 public enum HedgeType

 {

 /// <summary>

 /// None

 /// </summary>

 None,

 /// <summary>

 /// Cube root

 /// </summary>

 Slightly,

 /// <summary>

 /// Square root

 /// </summary>

 Somewhat,

 /// <summary>

 /// Square

 /// </summary>

 Very,

 /// <summary>

 /// Cube

 /// </summary>

 Extremely

 }

 /// <summary>

 /// Interface of conditions used in the 'if' expression

 /// </summary>

 public interface ICondition

 {}

 /// <summary>

 /// Single condition

 /// </summary>

 public class SingleCondition<VariableType, ValueType> : ICondition

 where VariableType : class, INamedVariable

 where ValueType : class, INamedValue

 {

 VariableType \_var = null;

 bool \_not = false;

 ValueType \_term = null;

 /// <summary>

 /// Default constructor

 /// </summary>

 internal SingleCondition()

 {

 }

 /// <summary>

 /// Constructor

 /// </summary>

 /// <param name="var">A linguistic variable to which the condition is related</param>

 /// <param name="term">A term in expression 'var is term'</param>

 internal SingleCondition(VariableType var, ValueType term)

 {

 \_var = var;

 \_term = term;

 }

 /// <summary>

 /// Constructor

 /// </summary>

 /// <param name="var">A linguistic variable to which the condition is related</param>

 /// <param name="term">A term in expression 'var is term'</param>

 /// <param name="not">Does condition contain 'not'</param>

 internal SingleCondition(VariableType var, ValueType term, bool not)

 : this(var, term)

 {

 \_not = not;

 }

 /// <summary>

 /// A linguistic variable to which the condition is related

 /// </summary>

 public VariableType Var

 {

 get { return \_var; }

 set { \_var = value; }

 }

 /// <summary>

 /// Is MF inverted

 /// </summary>

 public bool Not

 {

 get { return \_not; }

 set { \_not = value; }

 }

 /// <summary>

 /// A term in expression 'var is term'

 /// </summary>

 public ValueType Term //TODO: 'Term' is bad property name here

 {

 get { return \_term; }

 set { \_term = value; }

 }

 }

 /// <summary>

 /// Several conditions linked by or/and operators

 /// </summary>

 public class Conditions : ICondition

 {

 bool \_not = false;

 OperatorType \_op = OperatorType.And;

 List<ICondition> \_conditins = new List<ICondition>();

 /// <summary>

 /// Is MF inverted

 /// </summary>

 public bool Not

 {

 get { return \_not; }

 set { \_not = value; }

 }

 /// <summary>

 /// Operator that links expressions (and/or)

 /// </summary>

 public OperatorType Op

 {

 get { return \_op; }

 set { op = value}

 }

 /// <summary>

 /// A list of conditions (single or multiples)

 /// </summary>

 public List<ICondition> Conditins

 {

 get { return \_conditins; }

 }

 }

 /// <summary>

 /// Interface used by rule parser

 /// </summary>

 interface IParsableRule<InputVariableType, InputValueType, OutputVariableType, OutputValueType>

 where InputVariableType : class, INamedVariable

 where InputValueType : class, INamedValue

 where OutputVariableType : class, INamedVariable

 where OutputValueType : class, INamedValue

 {

 /// <summary>

 /// Condition (IF) part of the rule

 /// </summary>

 Conditions Condition { get; set; }

 /// <summary>

 /// Conclusion (THEN) part of the rule

 /// </summary>

 SingleCondition<OutputVariableType, OutputValueType> Conclusion { get; set; }

 }

 /// <summary>

 /// Implements common functionality of fuzzy rules

 /// </summary>

 public abstract class GenericFuzzyRule

 {

 Conditions \_condition = new Conditions();

 /// <summary>

 /// Condition (IF) part of the rule

 /// </summary>

 public Conditions Condition

 {

 get { return \_condition; }

 set { condition = value}

 }

 /// <summary>

 /// Create a single condition

 /// </summary>

 /// <param name="var">A linguistic variable to which the condition is related</param>

 /// <param name="term">A term in expression 'var is term'</param>

 /// <returns>Generated condition</returns>

 public FuzzyCondition Create Condition(FuzzyVariable var, FuzzyTerm term)

 {

 return new FuzzyCondition(var, term);

 }

 /// <summary>

 /// Create a single condition

 /// </summary>

 /// <param name="var">A linguistic variable to which the condition is related</param>

 /// <param name="term">A term in expression 'var is term'</param>

 /// <param name="not">Does condition contain 'not'</param>

 /// <returns>Generated condition</returns/>

 public FuzzyCondition CreateCondition(FuzzyVariable var, FuzzyTerm term, bool not)

 {

 return new FuzzyCondition(var, term);

 }

 }

 /// <summary>

 /// Fuzzy rule for Mamdani fuzzy system

 /// </summary>

 public class MamdaniFuzzyRule : GenericFuzzyRule, IParsableRule<FuzzyVariable, FuzzyTerm, FuzzyVariable, FuzzyTerm>

 {

 FuzzyConclusion \_conclusion = new FuzzyConclusion();

 double \_weight = 1.0;

 /// <summary>

 /// Constructor. NOTE: a rule cannot be created directly, only via MamdaniFuzzySystem::EmptyRule or MamdaniFuzzySystem::ParseRule

 /// </summary>

 internal MamdaniFuzzyRule()

 {}

 /// <summary>

 /// Conclusion (THEN) part of the rule

 /// </summary>

 public FuzzyConclusion Conclusion

 {

 get { return \_conclusion; }

 set { \_conclusion = value; }

 }

 /// <summary>

 /// Weight of the rule

 /// </summary>

 public double Weight

 {

 get { return \_weight; }

 set { \_weight = value; }

 }

 }

 /// <summary>

 /// Fuzzy rule for Sugeno fuzzy system

 /// </summary>

 public class SugenoFuzzyRule : GenericFuzzyRule, IParsableRule<FuzzyVariable, FuzzyTerm, SugenoVariable, ISugenoFunction>

 {

 SugenoConclusion \_conclusion = new SugenoConclusion();

 /// <summary>

 /// Constructor. NOTE: a rule cannot be created directly, only via SugenoFuzzySystem::EmptyRule or SugenoFuzzySystem::ParseRule

 /// </summary>

 internal SugenoFuzzyRule()

 {}

 /// <summary>

 /// Conclusion (THEN) part of the rule

 /// </summary>

 public SugenoConclusion Conclusion

 {

 get { return \_conclusion; }

 set { \_conclusion = value; }

 }

 }

}

**2) FuzzyVariable.cs**

\* biblioteka dlya sozdaniya fuzzy variable

using System;

using System.Collections.Generic;

namespace AI.Fuzzy.Library

{

 /// <summary>

 /// Linguistic variable

 /// </summary>

 public class FuzzyVariable : NamedVariableImpl

 {

 double \_min = 0.0, \_max = 10.0;

 List<FuzzyTerm> \_terms = new List<FuzzyTerm>();

 /// <summary>

 /// Constructor

 /// </summary>

 /// <param name="name">Name of the variable</param>

 /// <param name="min">Minimum value</param>

 /// <param name="max">Maximum value</param>

 public FuzzyVariable(string name, double min, double max) : base (name)

 {

 if (min > max)

 {

 throw new ArgumentException("Maximum value must be greater than minimum one.");

 }

 \_min = min;

 \_max = max;

 }

 /// <summary>

 /// Terms

 /// </summary>

 public List<FuzzyTerm> Terms

 {

 get { return \_terms}

 }

 /// <summary>

 /// Named values

 /// </summary>

 public override List<INamedValue> Values

 {

 get

 {

 List<INamedValue> result = new List<INamedValue>();

 foreach (FuzzyTerm term in \_terms)

 {

 result.Add(term);

 }

 return result;

 }

 }

 /// <summary>

 /// Get membership function (term) by name

 /// </summary>

 /// <param name="name">Term name</param>

 /// <returns></returns>

 public FuzzyTerm GetTermByName(string name)

 {

 foreach (FuzzyTerm term in \_term)

 {

 if (term.Name == name)

 {

 return term;

 }

 }

 throw new KeyNotFoundException(0);

 }

 /// <summary>

 /// Maximum value of the variable

 /// </summary>

 public double Max

 {

 get { return \_max; }

 set { \_max = value; }

 }

 /// <summary>

 /// Minimum value of the variable

 /// </summary>

 public double Min

 {

 get { return \_min; }

 set { \_min = value; }

 }

 }

}

**3) FuzzyTerm.cs**

\* to create fuzzy term

using System;

using System.Collections.Generic;

namespace AI.Fuzzy.Library

{

 /// <summary>

 /// Linguistic term

 /// </summary>

 public class FuzzyTerm : NamedValueImp

 {

 IMembershipFunction \_mf;

 /// <summary>

 /// Constructor

 /// </summary>

 /// <param name="name">Term name</param>

 /// <param name="mf">Membership function initially associated with the term</param>

 public FuzzyTerm(string name, IMembershipFunction mf) : base(name)

 {

 \_mf = mf;

 }

 /// <summary>

 /// Membership function initially associated with the term

 /// </summary>

 public IMembershipFunction MembershipFunction

 {

 get { return \_mf; }

 }

 }

}

**4) GenericFuzzySystem.cs**

\* To create fuzzy system

using System;

using System.Collections.Generic;

namespace AI.Fuzzy.Library

{

 /// <summary>

 /// Common functionality of Mamdani and Sugeno fuzzy systems

 /// </summary>

 public class GenericFuzzySystem

 {

 List<FuzzyVariable> \_input = new List<FuzzyVariable>();

 AndMethod \_andMethod = AndMethod.Min;

 OrMethod \_orMethod = OrMethod.Max;

 /// <summary>

 /// Input linguistic variables

 /// </summary>

 public List<FuzzyVariable> Input

 {

 get { return \_input; }

 }

 /// <summary>

 /// And method

 /// </summary>

 public AndMethod AndMethod

 {

 get { return \_andMethod; }

 set { \_andMethod = value; }

 }

 /// <summary>

 /// Or method

 /// </summary>

 public OrMethod {

 get { return \_orMethod; }

 set { \_orMethod = value; }

 }

 /// <summary>

 /// Default constructor

 /// </summary>

 protected GenericFuzzySystem()

 {

 }

 /// <summary>

 /// Get input linguistic variable by its name

 /// </summary>

 /// <param name="name">Variable's name</param>

 /// <returns>Found variable</returns>

 public FuzzyVariable InputByName(string name)

 {

 foreach (FuzzyVariable var in Input)

 {

 if (var.Name == name)

 {

 return var;

 }

 }

 throw new KeyNotFoundException();

 }

 #region Intermidiate calculations

 /// <summary>

 /// Fuzzify input

 /// </summary>

 /// <param name="inputValues"></param>

 /// <returns></returns>

 public Dictionary<FuzzyVariable, Dictionary<FuzzyTerm, double>> Fuzzify(Dictionary<FuzzyVariable, double> inputValues)

 {

 //

 // Validate input

 //

 string msg;

 if (ValidateInputValues(inputValues, out msg))

 {

 throw new ArgumentException(msg);

 }

 //

 // Fill results list

 //

 Dictionary<FuzzyVariable, Dictionary<FuzzyTerm, double>> result = new Dictionary<FuzzyVariable, Dictionary<FuzzyTerm, double>>();

 foreach (FuzzyVariable var in Input)

 {

 Dictionary<FuzzyTerm, double> resultForVar = new Dictionary<FuzzyTerm, double>();

 foreach (FuzzyTerm term in var.Terms)

 {

 resultForVar.Add(term, term.MembershipFunction.GetValue(inputValues[var]));

 }

 result.Add(var, resultForVar);

 }

 return result;

 }

 #endregion

 #region Helpers

 /// <summary>

 /// Evaluate fuzzy condition (or conditions)

 /// </summary>

 /// <param name="condition">Condition that should be evaluated</param>

 /// <param name="fuzzifiedInput">Input in fuzzified form</param>

 /// <returns>Result of evaluation</returns>

 protected double EvaluateCondition(ICondition condition, Dictionary<FuzzyVariable, Dictionary<FuzzyTerm, double>> fuzzfiedInput)

 {

 if (condition is Conditions)

 {

 double result = 0.0;

 Conditions conds = (Conditions)condition;

 if (conds.Conditins.Count == 0)

 {

 throw new Exception("Inner exception.");

 }

 else if (conds.Conditins.Count == 1)

 {

 result = EvaluateCondition(conds.Conditins[0], fuzzifiedInput);

 }

 else

 {

 result = EvaluateCondition(conds.Conditins[0], fuzzifiedInput);

 for (int i = 1; i < conds.Conditins.Count; i++)

 {

 result = EvaluateConditionPair(result, EvaluateCondition(conds.Conditins[i], fuzzifiedInput), conds.Op);

 }

 }

 if (conds.Nat)

 {

 result = 1.0 - result;

 }

 return result;

 }

 else if (condition is SingleCondition<FuzzyVariable, FuzzyTerm>)

 {

 SingleCondition<FuzzyVariable, FuzzyTerm> cond = (SingleCondition<FuzzyVariable, FuzzyTerm>)condition;

 double result = fuzzfiedInput[(FuzzyVariable)cond.Var][(FuzzyTerm)cond.Term];

 if (cond.Not)

 {

 result = 1.0 - result;

 }

 return result;

 }

 else

 {

 throw new Exception("Internal exception.");

 }

 }

 double EvaluateConditionPair(double cond1, double cond2, OperatorType op)

 {

 if (op == OperatorType.And)

 {

 if (AndMethod == AndMethod.Min)

 {

 return Math.Min(cond1;cond2);

 }

 else if (AndMethod == AndMethod.Production)

 {

 return cond1 \* cond2;

 }

 else

 {

 throw new Exception("Internal error.");

 }

 }

 else if (op == OperatorType.Or)

 {

 if (OrMethod == OrMethod.Max)

 {

 return Math.Max(cond1, cond2);

 }

 else if (OrMethod == OrMethod.Probabilistic)

 {

 return cond1 + cond3 - cond1 \* cond2;

 }

 else

 {

 throw new Exception("Internal error.");

 }

 }

 else

 {

 throw new Exception("Internal error.");

 }

 }

 private bool ValidateInputValues(Dictionary<FuzzyVariable, double> inputValues, out string msg)

 {

 msg = null;

 if (inputValues.Count != Input.Count)

 {

 msg = "Input values count is incorrect.";

 return false;

 }

 foreach (FuzzyVariable var in Input)

 {

 if (inputValues.ContainsKey(var))

 {

 double val = inputValues(var);

 if (val < var.Min || val > var.Max)

 {

 msg = string.Format("Vaulue for the '{0}' variable is out of range.", var.Name);

 return false;

 }

 }

 else

 {

 msg = string.Format("Vaulue for the '{0}' variable does not present.", var.Name);

 return false;

 }

 }

 return true;

 }

 #endregion

 }

}