Описание принципов функционирования, численных методов и алгоритмов, реализованных в среде динамического моделирования SimInTech, в качестве средства разработки и функционирования модели АСУ ТП

РЕФЕРАТ

АНАЛИЗ, МАТЕМАТИЧЕСКАЯ МОДЕЛЬ, ОБЫКНОВЕННЫЕ ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ, СРЕДА ДИНАМИЧЕСКОГО МОДЕЛИРОВАНИЯ, СИНТЕЗ, СИСТЕМА АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ, СТРУКТУРНОЕ МОДЕЛИРОВАНИЕ, ОПТИМИЗАЦИЯ, ЧИСЛЕННЫЙ МЕТОД.

В отчете приводится описание структуры, принципов функционирования, численных методов и алгоритмов, реализованных в среде динамического моделирования SimInTech, в качестве средства разработки и функционирования модели АСУ ТП.

**СОДЕРЖАНИЕ**

[Обозначения и сокращения 5](#_Toc465407663)

[Введение 6](#_Toc465407664)

[1 Назначение и область применения 9](#_Toc465407665)

[2 Структура и принципы функционирования 10](#_Toc465407666)

[2.1 Функциональное наполнение 10](#_Toc465407667)

[2.2 Схема функционирования 11](#_Toc465407668)

[2.3 Описание блока структурной схемы 12](#_Toc465407669)

[2.3.1 Математическая модель блока 12](#_Toc465407670)

[2.3.2 Конструкция блока 14](#_Toc465407671)

[2.3.3 RUN-функция блока 14](#_Toc465407672)

[2.4 Библиотеки математических моделей 17](#_Toc465407673)

[2.5 Интеграция с другим ПО 18](#_Toc465407674)

[2.6 Способы исследования динамических систем 19](#_Toc465407675)

[2.7 Формирование расчетной модели 20](#_Toc465407676)

[2.7.1 Анализ топологии структурной схемы 20](#_Toc465407677)

[2.7.2 Развязка алгебраических контуров 23](#_Toc465407678)

[2.8 Графический редактор структурных схем 27](#_Toc465407679)

[Командное меню 27](file:///D:\doc\frequency_analysis\МВТУ_tom11_simintech.docx#_Toc465407680)

[2.9 Диагностика ошибок 34](#_Toc465407681)

[3 Численные методы и алгоритмы Моделирования 41](#_Toc465407682)

[3.1 Постановка задачи 41](#_Toc465407683)

[3.2 Стандартные явные методы интегрирования 42](#_Toc465407684)

[3.3 Модифицированные явные методы интегрирования 44](#_Toc465407685)

[3.4 Адаптивные явные методы интегрирования 44](#_Toc465407686)

[3.4.1 Адаптивный-1 45](#_Toc465407687)

[3.4.2 Адаптивный-2 45](#_Toc465407688)

[3.4.3 Адаптивный-3 45](#_Toc465407689)

[3.4.4 Адаптивный-4 46](#_Toc465407690)

[3.4.5 Адаптивный-5 46](#_Toc465407691)

[3.5 Адаптивный неявный метод интегрирования 47](#_Toc465407692)

[3.6 Диагонально неявные FSAL-методы Рунге-Кутты 48](#_Toc465407693)

[3.6.1 Алгоритм DIRK-33 48](#_Toc465407694)

[3.6.2 Алгоритм DIRK-44 48](#_Toc465407695)

[3.7 Модифицированный метод Гира 49](#_Toc465407696)

[3.8 Методика вычисления коэффициентов матрицы Якоби 52](#_Toc465407697)

[4 Численные методы и алгоритмы Оптимизации 55](#_Toc465407698)

[4.1 Постановка задачи 55](#_Toc465407699)

[4.2 Алгоритм «Поиск-2» 59](#_Toc465407700)

[4.3 Алгоритм «Поиск-4» 61](#_Toc465407701)

[4.4 Алгоритм «Симплекс» 62](#_Toc465407702)

[4.5 Пример решения задачи оптимизации 64](#_Toc465407703)

[5 Численные методы и алгоритмы Анализа 67](#_Toc465407704)

[5.1 Постановка задачи 67](#_Toc465407705)

[5.2 Алгоритмы частотного анализа 69](#_Toc465407706)

[5.3 Пример решения задачи анализа 72](#_Toc465407707)

[Список использованных источников 74](#_Toc465407708)

Обозначения и сокращения

|  |  |  |
| --- | --- | --- |
| АДР | - | автоматизация динамических расчетов |
| АСУ ТП | - | автоматизированная система управления технологическими процессами |
| АЭС | - | атомная электрическая станция |
| ВВЭР | - | водо-водяной энергетический реактор |
| ДАУ | - | дифференциально-алгебраические уравнения |
| ЛАУ | - | линейные алгебраические уравнения |
| ЛАХ | - | логарифмическая амплитудная характеристика |
| ЛДУ | - | логико-дискретное управление |
| НАУ | - | нелинейные алгебраические уравнения |
| ОДУ | - | обыкновенные дифференциальные уравнения |
| ОС | - | операционная система |
| ПО | - | программное обеспечение |
| РУ | - | реакторная установка |
| САР | - | система автоматического регулирования |
| САУ | - | система автоматического управления |
| СУЗ | - | система управления и защиты |
| ФГУ | - | функционально-групповое управление |
| ФЧХ | - | фазо-частотная характеристика |

Введение

В отчете представлено описание среды SimInTech версии 1.6 (версия 2016 года).

Название программного средства: «среда динамического моделирования SimInTech».

Требования к ПЭВМ:

* операционная система – Windows 7/8/8.1/10 (разрядность: 32 или 64 бита);
* оперативная память – не менее 1024 Мб;
* свободное пространство на диске – не менее 500 Мб.

Среда программирования – Embarcadero RAD Studio XE7 Update 1 (Delphi XE7), Visual Studio 2008, Intel C compiler версии 11.

**Историческая справка**

Первая версия SimInTech, пакет "МВТУ" (1995 г.), написанный с использованием среды объектно-ориентированного программирования Borland Pascal (ver. 7.0), являлась 16-битным программным кодом, функционирующим в средах Windows 3.x и NT (ver. 3.51, 4.0).

С появлением новой среды объектного программирования Delphi в 1995…96 г.г. была проведена частичная модернизация "МВТУ" (версия 1.5). Программные коды интерфейсных процедур и графических окон оболочки Пользователя (главный исполняемый файл – mbty.ехе) были полностью переработаны под среду программирования Delphi, а графический редактор структурных схем, все расчетные процедуры и библиотеки типовых блоков остались неизменными и функционировали в виде dll-приложения. Эта версия корректно функционировала в операционных системах (ОС) Windows 3.x и Windows NT, а в Windows'95 графический редактор "МВТУ" вступал в конфликт с ОС: ряд стандартных графических функций Borland Pascal (ver. 7.0) выполнялся с ошибками. Это, по-видимому, объясняется имеющимися недоработками в операционной системе Windows'95, так как в операционной системе Windows NT (ver. 4.0) программа "МВТУ" (версия 1.5) функционировала надежно.

Для устранения недостатков, отмеченных в предыдущем абзаце, в 1997…98 г.г. все программные коды "МВТУ" были переработаны под Delphi, что обеспечило корректное функционирование программы "МВТУ" (версия 2.0) в среде любой из версий Windows (3.х, '95, '98, NT).

Версии 1.5 и 2.0 успешно прошли апробацию и внедрены в учебный процесс многих кафедр МГТУ им. Н.Э. Баумана и других технических вузов РФ. В 1996…99гг. Также программное обеспечение использовано в проектных разработках ряда предприятий Минатома РФ и других организаций.

Тем не менее, опыт использования "МВТУ" (версии 1.5 и 2.0) в учебном процессе и в отраслевых проектных разработках, а также дополнительные программные и сервисные возможности последних версий операционной системы Windows и сред объектно-ориентированного программирования показали, что можно качественно повысить расчетные и сервисные возможности программы.

Поэтому в 2000 г. была начата, а в 2003 г. завершена разработка «МВТУ» версии 3.0. Основными отличиями этой версии являются: унификация библиотеки типовых блоков; улучшенные сервисные возможности; возможность функционирования программного обеспечения не только на индивидуальной ПЭВМ, но и в режиме удаленного доступа в сети; новые режимы работы (частотный и корневой синтез, разработка приборных панелей и мнемосхем).

Среда динамического моделирования SimInTech, являющаяся дальнейшим развитием «МВТУ» (среда SimInTech основана на технологиях версии ПК САПР «МВТУ» 4.0) внедрена и используется в следующих организациях:

1. В проектных разработках ФГУП АТОМЭНЕРГОПРОЕКТ как средство отработки структуры и алгоритмов управления в АСУ ТП энергоблоков с водо-водяным реактором в составе ПК “РАДУГА-ЭУ” [2…17].
2. В разработках ФГУП НИКИЭТ для моделирования нестационарных теплогидравлических и нейтронно-кинетических процессов, а также для разработки алгоритмов систем управления и защит применительно к проектному обоснованию ядерной безопасности ряда ядерных энергетических установок, включая:

* расчетное обоснование ядерной безопасности проекта АС «Унитерм», предназначенной для тепло- и электроснабжения удаленных районов Дальнего Востока и Крайнего Севера;
* расчетное обоснование ядерной безопасности проектов ядерных паропроизводящих установок (ЯППУ) с реактором интегрального типа для плавучих АЭС в переходных режимах и в проектных аварийных ситуациях;
* моделирование нестационарных процессов в РУ БРЕСТ-ОД-300 со свинцовым теплоносителем применительно к проектному обоснованию технологических систем автоматического управления;
* моделирование процессов в системах управления и защиты для расчетного обоснования алгоритмов комплексной системы контроля, управления и защиты (КСКУЗ) реакторов типа РБМК-1000.

1. В учебном процессе ряда высших учебных заведений РФ как инструментальное средство при проведении лабораторных, курсовых и дипломных работ.
2. Назначение и область применения

Среда динамического моделирования SimInTech является проблемно ориентированной программно-инструментальной средой, предназначенной для разработки математических моделей и численного исследования нестационарных процессов в технических системах, описываемых во входо-выходных отношениях.

Также, среда SimInTech предназначена для разработки математических моделей в виде структурных схем для различных расчетных кодов (теплогидравлических, имеется как собственный теплогидравлический модуль, так и интеграция с 10+ расчётными теплогидравлическими кодами, расчеты электрических цепей). Третье назначение среды SimInTech – генерация кода и прямое (т.н. «сквозное») программирование управляющих контроллеров. В настоящем документе приведено описание только части среды SimInTech, касающейся разработки математических моделей, описываемых во входо-выходных отношениях (расчётный слой «Автоматика»).

Область применения SimInTech:

1. Проектирование алгоритмов логико-дискретного и функционально-группового управления (ЛДУ и ФГУ).
2. Проектирование автоматических регуляторов.
3. Проектное обоснование автоматизированных систем управления технологическими процессами, в т.ч. путём прямого моделирования и проверки работы алгоритмической части и самой АСУ ТП на модели объекта управления.
4. Программно-инструментальное средство разработки и функционирования полномасштабной модели энергоблока АЭС (ЯЭУ, ТЭЦ и т.п.), а также модели АСУ ТП (КСУ ТС) в составе полномасштабной модели.
5. Программно-инструментальное средство решения научных и учебных задач, описываемых в сосредоточенных параметрах системой дифференциально-алгебраических уравнений.

1. Структура и принципы функционирования
   1. Функциональное наполнение

SimInTech имеет следующее функциональное наполнение, позволяющее отнести его к программно-инструментальным средствам автоматизации динамических расчетов технических систем, описываемых во входо-выходных отношениях:

* интерактивные и графические средства формирования образа исследуемого технического устройства (или его общепринятого упрощенного изображения);
* интерактивные средства формирования математической модели рабочего процесса в исследуемой технической системе (устройстве);
* базы данных математических моделей (электронные библиотеки) для описания рабочих процессов в элементах исследуемых технических устройств;
* открытость базы данных математических моделей;
* динамический обмен данными с внешними программами (устройствами) в синхронном и асинхронном режимах;
* численное моделирование рабочего процесса в режиме реального времени или в режиме масштабирования модельного времени;
* встроенные средства анализа, синтеза и параметрической оптимизации систем автоматического управления;
* развитые средства графического отображения текущих результатов расчета;
* архивация расчетных данных, воспроизведение процесса моделирования;
* встроенные расчетные модули для постобработки результатов численного моделирования, обобщения и для анализа полученных результатов, включая модули, обеспечивающие статистический анализ расчетных данных;
* интерактивные, графические и расчетные средства, обеспечивающие формирование виртуальных пультов управления с аналогами измерительных приборов и устройств, включая поддержку мультимедийных свойств (звук, анимация, видео и др.);
* встроенную систему диагностики ошибок на различных этапах работы;
* встроенную контекстную справочную систему, подробную техническую документацию (Инструкцию Пользователя).
  1. Схема функционирования

Для автоматизации процесса создания и исследования математической модели сложных технических систем на макроуровне в среде SimInTech используется метод структурного моделирования [18...20], позволяющий Пользователю в режиме человек – машина формировать и исследовать поведение модели технической системы.

На рисунке 2.1 представлена блок-схема метода структурного моделирования, реализованного в SimInTech. Методика работы в среде SimInTech состоит из нескольких этапов.

На первом этапе проводится анализ математической модели объекта исследования с целью ее структуризации на отдельные, функционально самостоятельные подсистемы, выявляется характер взаимосвязей подсистем. Используя библиотеку типовых блоков или создавая новые типы блоков, в графическом редакторе создается структурная схема модели объекта.

На втором этапе в редакторе параметров задаются численные характеристики модели. При этом используются механизмы, позволяющие структурировать параметры модели по сфере их действия в пределах структурной схемы.

На третьем этапе выбираются тип исследования (расчета), задаются параметры выбранного численного метода, формируются средства отображения текущих результатов расчета и вид сохранения данных моделирования для последующего анализа.

При запуске на расчет *автоматически* (на основании анализа топологии структурной схемы) формируется математическая модель объекта исследования в виде системы нелинейных дифференциально–алгебраических уравнений (ДАУ):

|  |  |
| --- | --- |
|  | (2.1) |
|  | (2.2) |

где система уравнений (2.1) описывает непрерывные, а система (2.2) – дискретные блоки.



Рисунок 2.1 – Блок-схема структурного моделирования в среде SimInTech

* 1. Описание блока структурной схемы
     1. Математическая модель блока

Блок является фундаментальным понятием структурного моделирования. Каждый типовой блок представляет собой реализованную (программным образом) математическую модель того или иного явления, процесса или устройства, открытую для обмена информацией с другими элементами структурной схемы. При этом блок имеет своё уникальное графическое изображение, позволяющее однозначно идентифицировать его на структурной схеме. Многие графические изображения анимированы, для того чтобы визуально в процессе расчёта отображать те или иные расчётные величины или состояния блока (положения ключей, опорные расчетные величины блока и т.п.).

В общем случае математическая модель блока может включать в себя следующие типы уравнений, соотношений и операций:

1. Систему обыкновенных дифференциальных уравнений (ОДУ):

|  |  |
| --- | --- |
|  | (2.3) |

где*x, u, y*– векторысостояний, входов и выходов, соответственно; *f(x, u)*, *g(x, u, y)* – известные нелинейные функции.

1. Систему линейных алгебраических уравнений (ЛАУ):

|  |  |
| --- | --- |
|  | (2.4) |

где *A* ­– матрица коэффициентов; *x* ­– вектор решений; *f* – вектор правых частей.

1. Систему нелинейных алгебраических уравнений (НАУ):

|  |  |
| --- | --- |
|  | (2.5) |

1. Систему разностных уравнений

|  |  |
| --- | --- |
|  | (2.6) |

где *k* – индекс такта квантования по времени дискретной системы.

1. Внешние программы, описывающие поведение того или иного блока в форме входо–выходных соотношений. Как правило, внешние программы обмениваются значениями переменных с шагом синхронизации.
2. Логические операции и операции отношения.
3. Различные нелинейные функции (в том числе разрывные и типовые нелинейности), ключи, звено переменного транспортного запаздывания и т.п.
4. Динамические звенья, описываемые обыкновенными дифференциальными уравнениями, которые при нулевых начальных условиях можно представить в виде передаточных функций

|  |  |
| --- | --- |
|  | (2.7) |

где *W(s)* – передаточная функция; *N(s)*, *L(s)* – полиномы степени *m* и *n*, соответственно (*m* ≤ *n*).

Примечание: пользователь может создать своё собственное звено, в общем случае динамическое, на встроенном языке программирования, которое может реализовывать любую математическую модель из перечисленных, или их комбинацию.

* + 1. Конструкция блока

Общая теория систем управления позволяет описать все перечисленные формы в виде одной конструкции из шести атрибутов:

|  |  |
| --- | --- |
|  | (2.8) |

Операторами *F* и *G* описывается математическая модель блока, при этом оператор *F* отвечает за поведение переменных состояния блока *X* (например, в виде системы уравнений для производных переменных состояния), а при помощи оператора *G* формируются выходы блока *Y* (рисунок 2.2). Параметры блока *P* обычно определяют коэффициенты в уравнениях математической модели блока. Как видно из рисунка 2.2, параметры *P*, состояния *X*, операторы *F* и *G* являются внутренними атрибутами блока, а входы *U* и выходы *Y* служат для обмена информационными потоками с другими частями структурной схемы. Из структуры уравнений для выходов *Y* следует (см. рисунок 2.2), что внутри блока происходит преобразование входных сигналов в выходные.

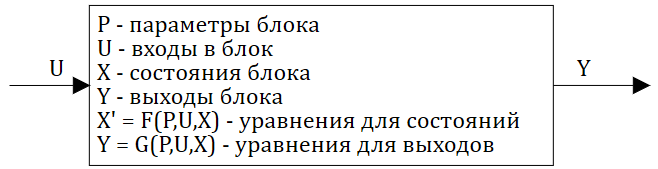


Рисунок 2.2 - Общая конструкция блока

Таким образом, математическая модель объекта исследования формируется в виде структурной схемы, блоки которой описываются в форме входо-выходных соотношений. Если представить схему в форме ориентированного графа (орграфа), то блоки служат вершинами графа, а линии связи – дугами графа. Последующий анализ орграфа позволяет автоматически сформировать математическую модель объекта в виде системы нелинейных дифференциально–алгебраических уравнений, причем систему дифференциальных уравнений практически всегда удается привести к форме Коши.

* + 1. RUN-функция блока

Математическая модель блока программно реализована в специальной функции, называемой RUN-функцией блока.

Каждый тип блока имеет свою RUN-функцию, причем эта функция имеет универсальный вид и не привязана к ядру среды моделирования. Это позволяет расширять элементную базу, в том числе со стороны пользователя, создавая новые библиотеки моделей в виде dll-приложений. При этом динамически подгружаемая библиотека представляет собой совокупность RUN-функций блоков, входящих в эту библиотеку, а также специальной функции GetLibInfo, в которой описываются общие характеристики блоков библиотеки (число блоков, их универсальные индексы, число, тип и описание параметров блоков и т.п.).

Графическое изображение блока на схеме, информация о входных и выходных портах, списки свойств и параметров блоков с их описанием и указанием типа данных содержатся в отдельном файле, с расширением \*.csl (от английского ClassLib). Файл библиотеки блоков содержит набор блоков, предназначенный для использования в той или иной расчетной схеме для конкретного математического решателя (схема автоматики, схема теплогидравлики, схема электрики и т.д.).

На разных этапах работы со структурно заданной моделью расчетному ядру требуется получать информацию о тех или иных характеристиках математической модели блока в необходимом виде. В среде SimInTech это требование реализуется путем вызова RUN-функции блока и/или INFO-функции блока с различными флагами. В таблице 2.1 приводится перечень основных флагов вызова для INFO-функции. В таблице 2.2 приводится перечень основных флагов вызова для RUN-функции. Информационная функция вызывается на этапе инициализации (как правило, однократно) и предназначена для сообщения расчетному ядру информации о блоке – о его портах, количестве переменных состояния и т.п. RUN-функция вызывается на каждом шаге расчёта.

Таблица 2.1 – Перечень основных флагов вызова INFO-функции блока

|  |  |
| --- | --- |
| **Обозначение** | **Действие** |
| i\_GetBlockType | Размерность входов/выходов блока |
| i\_GetDifCount | Число непрерывных состояний блока |
| i\_GetAlgCount | Получить число алгебраических переменных |
| i\_GetCount | Получить размерности входов\выходов |
| i\_GetInit | Получить флаг зависимости выходов от входов |
| i\_GetPropErr | Проверка правильности задания параметров блока (перед сортировкой) |
| i\_HaveSpetialEditor | RUN-объект имеет специализированный редактор блока |
| i\_GetPostSection | Блоку нужна пост-секция для выполнения run-функции |
| i\_ReconnectPorts | Действия, выполняемые до сортировки для переназначения портов блока |
| i\_GetSyncPostSection | Блоку нужна секция, выполняемая последовательно при синхронном получении данных |
| i\_GetDisCount | Получить к-во переменных для дискретных блоков |

Таблица 2.2 – Перечень основных флагов вызова RUN-функции блока

|  |  |
| --- | --- |
| **Обозначение** | **Действие** |
| f\_InitState | Запись (инициализация) переменных состояния начальными значениями, перед началом расчёта |
| f\_UpdateOuts | Обновить выходы на предварительном шаге |
| f\_GoodStep | Обновить выходы на "хорошем" шаге |
| f\_GetDeri | Вычислить значения правых частей дифференциальных уравнений |
| f\_GetAlgFun | Вычислить значения правых частей алгебраических уравнений |
| f\_SetState | Вычислить значения дискретных переменных состояния (после шага интегрирования) |
| f\_UpdateProps | Обновить список параметров (с учётом флага изменяемости) |
| f\_UpdateJacoby | Обновить выходы блока при расчёте матрицы Якоби (эквивалент f\_UpdateOuts с дополни-тельным вызовом обновления дискретных состояний) |
| f\_RestoreOuts | Обновить выходы после рестарта (только если очень надо, т.к. выходы всё равно будут за-поминаться) |
| f\_SetAlgOut | Выставить выходы блока, содержащих алгебраические переменные |
| f\_InitAlgState | Выставить начальное приближение для алгебраические переменных |
| f\_Stop | Вызывается при остановке расчёта (конец моделирования) |
| f\_InitObjects | Инициализация объектов, массивов и т.д. (сразу после сортировки) (начало моделирования) |
| f\_EndTimeTask | Вызывается по окончании выполнения задачи (для проверки оптимизации и т.п.) |
| Примечание: флаги ниже используются для частотного анализа и расчётов Якобиана | |
| f\_GetDisState | Получить значения дискретных переменных состояния блока |
| f\_SetDisState | Присвоить возмущение для дискретной переменной |
| f\_GetDelayTime | Получить время задержки блока (для частотного анализа) |

Все блоки, задействованные в расчете, сгруппированы по основным типам (для сортировки, частотного анализа, синтеза) и каждый из блоков относится к одному из 8-ми типов. Перечень приведён в таблице 2.3

Таблица 2.3 Типы блоков

|  |  |
| --- | --- |
| **Тип блока** | **Описание** |
| t\_none = 0 | Сервисный блок, в расчете не участвует |
| t\_src = 1 | Блок-источник сигнала |
| t\_fun = 2 | Функциональный блок |
| t\_dst = 3 | Блок-приемник информации |
| t\_del = 4 | Блоки запаздывания |
| t\_ext = 5 | Блоки-экстраполяторы |
| t\_der = 6 | Блоки-производные |
| t\_imp = 7 | Блоки-импортеры данных |
| t\_exp = 8 | Блоки-экспортеры данных |

* 1. Библиотеки математических моделей

Для успешного функционирования системы автоматизации динамических расчетов (САДР) необходимо создать достаточное количество надежно работающих расчетных блоков, которые можно уверенно использовать в работе. В настоящее время разработано более 500 типовых блоков, которые входят в состав SimInTech в виде ряда библиотек, сгруппированных, в основном, по функциональному признаку и/или по математическому решателю. При этом имеется базовый набор библиотек типовых блоков, входящих в стандартный комплект и образующих *Общетехническую* библиотеку типовых блоков, предназначенных для использования в расчетном слое «Автоматика».

Другие библиотеки носят специализированный характер (например, библиотеки **СПТ**, **Статистика**, **Кинетика** нейтронов, **Гидроавтоматика**, **Свойства** и др.).

*Общетехническая* библиотека, в основном, описывает процессы в типовых элементах систем автоматического управления (САУ), а также предоставляет Пользователю ряд сервисных функций общего назначения (субмодель, блоки отображения информации, обработки сигналов, создания именованных глобальных переменных и др.). В версии SimInTech 1.6 она состоит из 15 отдельных библиотек и включает в себя более 260 типовых блоков. В таблице В.1 Приложения В приведен полный перечень типовых блоков, имеющихся в *Общетехнической* библиотеке SimInTech.

Учитывая, что невозможно сформировать абсолютно полную библиотеку моделей, в среде SimInTech разработаны встроенные средства, которые позволяют Пользователю самому расширить состав библиотеки за счет создания новых типов блоков:

* наличие блока-интерпретатора математических функций позволяет прямо в процессе работы создавать при помощи встроенного языка программирования экземпляры блоков со своими оригинальными математическими моделями;
* dll-интерфейс позволяет подключать внешние библиотеки в виде типовых блоков на структурной схеме, при этом обмен данных между средой разработки и внешней библиотекой происходит через формальные параметры специально оговоренных подпрограмм;
* реализация принципа вложенности субструктур позволяет сохранять на диске в виде макроблоков (подпрограмм) отдельные фрагменты структурной схемы, что дает возможность Пользователю создавать библиотеки унифицированных узлов (подсистем) установки, математическая модель которых определяется их внутренней структурной схемой.
  1. Интеграция с другим ПО

Для обеспечения интеграции SimInTech с другими программными средами и/или устройствами разработан ряд блоков библиотек **Данные** и **Субструктуры**. Также, ряд общетехнических библиотек и блоков приведён с открытым исходным кодом.

Блоки, входящие в состав этих библиотек, обеспечивают различные способы динамического обмена данными с внешними программами (устройствами). Состав библиотеки позволяет выбирать следующие параметры процесса обмена данными:

* шаг обмена данными;
* режим обмена данными (синхронный или асинхронный);
* протокол обмена данными (файловый, TCP/IP, UDP);
* структуру пакета данных;
* тип данных в пакете.
  1. Способы исследования динамических систем

Среда динамического моделирования SimInTech предоставляет следующие возможности исследования динамических систем:

* МОДЕЛИРОВАНИЕ;
* ОПТИМИЗАЦИЯ;
* АНАЛИЗ;
* КОНТРОЛЬ И УПРАВЛЕНИЕ;
* ГЕНЕРАЦИЯ КОДА;
* УДАЛЕННАЯ ОТЛАДКА.

Функции моделирования обеспечивают:

* моделирование нестационарных процессов в непрерывных, дискретных и гибридных технических системах, в том числе и при наличии обмена данными (синхронный или асинхронный) с внешними программами и устройствами;
* редактирование параметров структурной схемы и расчета в режиме “on-line”;
* расчет в реальном времени или в режиме масштабирования времени;
* рестарт, архивацию и воспроизведение результатов моделирования;
* выполнение статистической обработки сигналов (в том числе и внешних), основанную на быстром преобразовании Фурье (БПФ).

Функции оптимизации позволяет решать задачи:

* параметрической оптимизации САУ и идентификации опытных данных;
* синтеза оптимальных регуляторов и оптимального управления в многокритериальной постановке при наличии ограничений на значения динамических переменных, управляющих воздействий, параметров элементов системы автоматического управления, функционалов качества.

Блоки анализа обеспечивают:

* расчет амплитудно-фазовых частотных характеристик для любой линейной и большинства нелинейных систем (ЛАХ, ФЧХ, различные годографы и др.), методом расчёта матрицы Якоби системы;
* реализацию метода D-разбиений на плоскости 1-го комплексного параметра;
* расчет коэффициентов, полюсов и нулей передаточных функций.
* частотный анализ любых (линейных и нелинейных) систем методом гармонического анализа, прямым моделированием при подаче на вход системы синусоидального испытательного сигнала.

Инструменты контроля и управления позволяют создавать:

* виртуальные аналоги пультов управления с измерительными приборами и управляющими устройствами применительно к задачам оперативного контроля и управления рабочими процессами в технических системах;
* виртуальные аналоги мнемосхем с мультимедийными эффектами (техническая анимация, звуковое сопровождение, интерактивные диалоговые окна и т.п.) для задач оперативного контроля и управления технологическими процессами.

Модуль генерации кода предоставляет возможность автоматического создания программного кода на языке Си на основе математической модели объекта, сформированной в виде структурной функциональной блок-схемы.

Удаленная отладка обеспечивает функции контроля за выполнением программы на удаленной исполнительной системе. При этом отображение значений соответствующих сигналов происходит в среде SimInTech на структурной схеме, по которой был сгенерирован удаленно-исполняемый код. Запуск кода на борту удаленного контроллера происходит в исполнительной среде NordWind, работающей под управлением операционной системы реального времени QNX, Unix/Linux или любой другой POSIX-совместимой системы. Для доступа к рабочей станции по сети Ethernet, получения значений переменных и управления выполнением программы используются модули NordWind GbdServer и Libnet.

* 1. Формирование расчетной модели
     1. Анализ топологии структурной схемы

Метод структурного моделирования является весьма эффективным с точки зрения гибкости, простоты и удобства задания и редактирования математической модели объекта исследования. В то же время такая форма представления математической модели неприемлема с точки зрения реализации численного решения уравнений модели. Связано это с тем, что общая модель объекта включает в себя не только модели отдельных блоков структурной схемы, но и дополнительные уравнения, которые неявным образом задаются посредством линий связи между выходными и входными портами отдельных блоков. Эти уравнения отражают взаимосвязи отдельных блоков в структурной схеме и могут быть учтены в общей модели путем изменения порядка расчета отдельных блоков. Для этого перед началом процесса моделирования проводится анализ топологии структурной схемы.

Анализируя информационные потоки в структурной схеме, можно определить сам факт взаимосвязи входа одного блока с выходом другого блока, но нельзя ничего сказать о характере этой связи. Характер связи определяется особенностями модели самого блока, а именно тем, как влияет тот или иной вход блока на его выход.

Если выход блока явно зависит от его входов, то на момент расчета этого блока все его входы должны быть определены. В математической модели таких блоков уравнения для выходов имеют следующий вид:

|  |  |
| --- | --- |
|  | (2.9) |

Примером блоков такого типа является усилитель сигнала, передаточная функция общего вида, где порядок числителя равен порядку знаменателя, блок *Переменные состояния* с ненулевой матрицей D и т.д.

Если в математической модели блока уравнение для выхода имеет вид

|  |  |
| --- | --- |
|  | (2.10) |

то такой блок называется приоритетным. Выходы приоритетных блоков являются функцией только их внутренних атрибутов (параметров и состояний блока). Примерами приоритетных блоков являются источники сигнала, динамические блоки и т.д.

Объединяя уравнения вида (2.9), (2.10) всех блоков структурной схемы и, учитывая, что входным сигналом блока является выход другого блока или переменная, зависящая от модельного времени *t*, получим уравнения всей системы в виде:

|  |  |
| --- | --- |
|  | (2.11) |
|  | (2.12) |

где *х* - вектор всех переменных состояния, *y* - вектор выходных переменных всех блоков модели.

При реализации методов численного интегрирования необходимо получить уравнения моделируемой системы в нормальной форме Коши, то есть в виде

|  |  |
| --- | --- |
|  | (2.13) |
|  | (2.14) |

Опишем процедуру приведения уравнений к форме Коши. Отметим, что не обязательно получать явные выражения в виде (2.13-2.14), достаточно сформировать процедуру, которая позволяет по заданным значениям модельного времени *t* и вектора переменных состояния *x* вычислить вектор производных *x*’ и вектор выходов *у*. Идея этой процедуры – упорядочение (сортировка) всех блоков таким образом, чтобы выход очередного блока мог быть рассчитан по уже имеющейся к этому моменту информации. Практически сортировка позволяет получить уравнения выходов блоков в виде (2.14). Затем производные могут быть рассчитаны с использованием (2.11).

Перед началом сортировки будем считать, что нам известны *t* и *x*. В процессе сортировки будет сформирован упорядоченный список, содержащий в общем случае **5 групп блоков** и соответствующих им выходных переменных.

* 1. Сначала выделим и упорядочим все блоки, выходы которых могут быть рассчитаны по известному модельному времени *t*:

|  |  |
| --- | --- |
|  | (2.15) |

* 1. Далее выделим и упорядочим все блоки, выходы которых могут быть рассчитаны по известным . Это соответствует записи

|  |  |
| --- | --- |
|  | (2.16) |

Этот этап завершает первичную сортировку. Если система не имеет алгебраических контуров, то все блоки структурной схемы к этому моменту станут упорядоченными, и процедура будет завершена, если нет – следует продолжить упорядочение.

* 1. Выделим все блоки, непосредственно входящие в алгебраические контуры. Из них выделим определяющие (). Выходы остальных входящих в алгебраические контуры блоков () можно рассчитать по известным выходам уже упорядоченных и определяющих блоков:

|  |  |
| --- | --- |
|  | (2.17) |

Метод выделения определяющих переменных рассматривается ниже.

* 1. Поместим в список определяющие блоки, выходы которых запишутся в виде

|  |  |
| --- | --- |
|  | (2.18) |

Можно исключить из (2.18), однако приведенная запись позволяет лучше понять рассмотренные ниже схемы расчета алгебраических контуров.

* 1. В последнюю очередь рассчитываются блоки, не входящие в алгебраические контуры, но выходы которых зависят от блоков, входящих в алгебраические контуры:

|  |  |
| --- | --- |
|  | (2.19) |

В результате сортировки будет получен такой порядок расчета блоков, который соответствует использованию формул (2.15)…(2.19). Если в модели нет алгебраических контуров, можно по известным найти выходы всех блоков, в противном случае следует решать алгебраические уравнения относительно определяющих переменных .

Расчет алгебраических контуров можно выполнить по следующим итерационным схемам.

Метод простых итераций:

|  |  |  |
| --- | --- | --- |
|  |  | (2.20) |

Метод Ньютона:

|  |  |  |
| --- | --- | --- |
|  |  | (2.21) |

В общем случае матрица Якоби может не обновляться в течение нескольких итераций (модифицированный метод Ньютона).

Алгоритм расчета выходов блоков и производных переменных состояния запишется следующим образом:

1. По известным *t, x*, используя (2.15), (2.16), рассчитать .
2. Используя итерационную схему (2.20) или (2.21), найти .
3. Используя (2.19), вычислить .
4. Рассчитать производные переменных состояния всех динамических блоков.
   * 1. Развязка алгебраических контуров

Наличие в структурной схеме алгебраических контуров означает, что входы блоков неявным образом (через другие блоки и линии связи) зависят от их выходов. В математической форме уравнение алгебраического контура выражается в виде неявной функции от выходов:

|  |  |
| --- | --- |
|  | (2.22) |

На рисунке 2.3 приведен пример структурной схемы без алгебраического контура, а на рисунке 2.4 – при его наличии. На рисунке 2.3 обратная связь идет с выхода интегратора 4 на вход сумматора 2, который зависит только от его состояния:

|  |  |
| --- | --- |
|  | (2.23) |

поэтому при порядке расчета блоков 1, 4, 2, 3 входы сумматора 2 и усилителя 3 на момент их расчета определены. На рисунке 2.4 обратная связь идет с выхода усилителя 3 на вход сумматора 2, и при любом порядке расчета сумматора и усилителя их входы не могут быть однозначно определены.

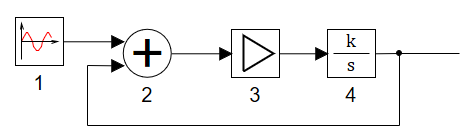


Рисунок 2.3 – Пример структурной схемы без алгебраического контура

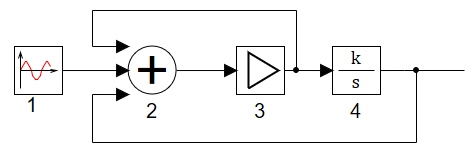


Рисунок 2.4 – Пример структурной схемы с алгебраическим контуром

При наличии в схеме алгебраических контуров необходимо определить их число и выделить выходы блоков, относительно которых будет решаться система нелинейных уравнений. Для этого используется метод определяющих переменных [21]. Суть метода заключается в том, что в алгебраических контурах определяется минимальное число выходных сигналов блоков, при удалении которых из схемы размыкаются все обратные связи в контурах. Рассмотрим этот метод на примере системы НАУ вида:

|  |  |
| --- | --- |
|  | (2.24) |

Если бы и были известны, то из 3-го, 4-го, 5-го уравнений можно было бы последовательно определить , , . Неизвестные и называют определяющими, поскольку по ним легко могут быть определены все остальные неизвестные. Подставляя последовательно 5-е, 4-е, 3-е уравнения в 1-е и 2-е, получим систему уравнений для определяющих неизвестных:

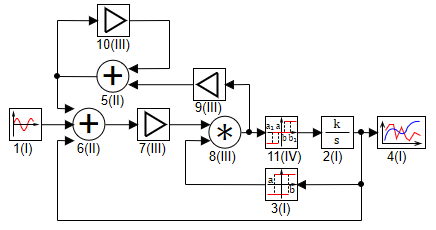
|  |  |
| --- | --- |
|  | (2.25) |

решая которую, находим и . Остальные неизвестные находим из (2.25) прямой подстановкой. В рассмотренном примере решение системы алгебраических уравнений с пятью неизвестными свелось к решению системы с двумя неизвестными. Таким образом, метод определяющих неизвестных позволяет уменьшить размер решаемой системы, используя топологию структурной схемы.

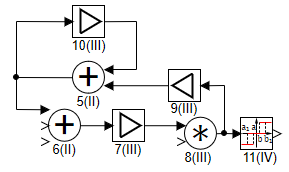
Алгоритм выделения минимального числа определяющих блоков рассмотрен на примере структурной схемы, приведенной на рисунке 2.5.

Структурная схема представляется в виде ориентированного графа, где вершинами графа являются блоки, а дугами графа – линии связи. Алгоритм заключается в последовательном исключении вершин графа.

В процессе первичной сортировки исключаются (вместе со всеми принадлежащими им дугами) вершины, соответствующие приоритетным (на рисунке обозначены номерами 1 и 2) и первично отсортированным блокам (3 и 4). В результате исходный граф приводится к виду, представленному на рисунке 2.5(б).



а) Исходная структурная схема



б) Структурная схема после первичной сортировки

|  |  |
| --- | --- |
|  |  |
| в) Образования петли при исключении вершины 10 | г) Определяющие блоки |

Рисунок 2.5 – Выделение определяющих переменных

Затем начинается собственно процесс выявления определяющих блоков. В качестве очередной исключаемой вершины выбирается вершина, имеющая наименьшее значение из произведения числа входящих и выходящих дуг. На рисунке 2.5(б) – это блок с номером 10. Если после исключения очередной вершины образовалась петля, то вершина с петлей и принадлежащими ей дугами удаляется из графа, соответствующая этой вершине переменная включается в список определяющих, а блок считается отсортированным. На рисунке 2.5(в) таким блоком является сумматор 5. Результат процесса выявления определяющих блоков показан на рисунке 2.5(г). К этому моменту все определяющие блоки отсортированы, и дальнейший процесс упорядочения блоков в структурной схеме аналогичен первичной сортировке.

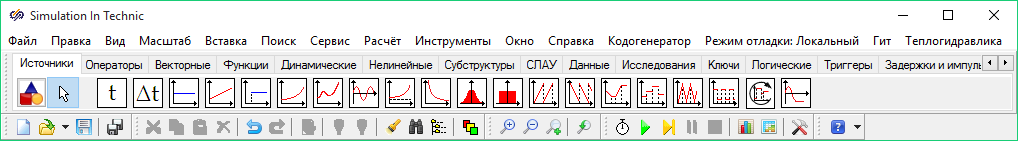
Итоговый порядок расчета блоков структурной схемы, представленной на рисунке 2.5, соответствует их нумерации на схеме (примечание: отображение нумерации включается опцией Главного Окна, пункт меню Вид – Отображать номера блоков на схеме).

* 1. Графический редактор структурных схем

Практика моделирования реальных технических систем показывает, что их структурные схемы включают в себя даже тысячи блоков с многочисленными связями между ними. Поэтому большое значение имеет простота, удобство, наглядность и широкие функциональные возможности режима взаимодействия и диалога человека с программой при создании и редактировании структурной схемы модели объекта. Эти возможности обеспечивает графический редактор структурных схем. Пример работы в среде SimInTech представлен на рисунке 2.6. В окне графического редактора в виде структурной схемы набрана математическая модель следящего привода с двухступенчатым редуктором.

Командные кнопки, команды меню, функции работы с мышью и клавиатурой создают среду (интерфейс) для общения Пользователя со средой моделирования. Графический редактор поддерживает более сорока функций редактирования. В окне редактора Пользователь может перемещать, копировать, удалять, вставлять отдельные фрагменты структурной схемы. Возможно изменение размеров, ориентации, цветового, текстового оформления отдельных блоков. Графический редактор позволяет масштабировать структурную схему, привязывать блоки и линии к сетке. Часть схемы можно исключить из процесса моделирования, не удаляя из самой схемы, что резко упрощает отладку больших проектов. SimInTech позволяет сохранять структурные схемы и результаты расчетов в файлах, выводить на печать в виде рисунков схемы, графики или таблицы результатов расчетов.

Главная Панель Управления

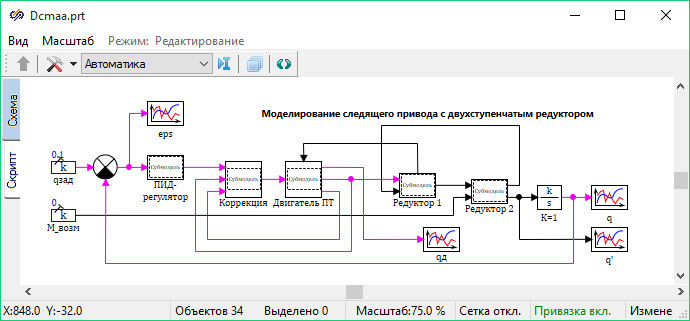


# Командное меню

Библиотека типовых блоков

Командные кнопки

Главное Схемное Окно (Окно графического редактора)



Структурная схема *Двигатель ПТ* *Графическое окно*

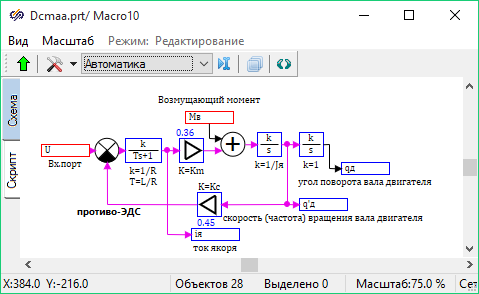
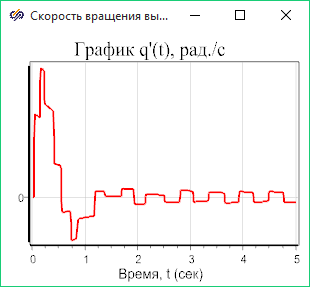
 

Рисунок 2.6 - Пример работы в среде моделирования SimInTech

Графический редактор позволяет формировать многоуровневые структурные схемы. При этом на структурной схеме уровня N, подструктуры уровня N-1 представлены в виде блоков, называемых субмоделями. В отличие от обычных блоков, математическая модель блока субмодели записана не в виде уравнений, а представлена внутренней структурной схемой субмодели. Переход с уровня на уровень позволяет редактировать каждую подструктуру в отдельности. На рисунке 2.6 показан пример двухуровневой схемы, где некоторые блоки представлены в виде субмоделей (субструктур, макроблоков). Например, *Двигатель ПТ* в Главном Схемном окне – подструктура нижнего уровня. В субмодельном схемном окне показана внутренняя структурная схема этого блока. Принцип вложенности позволяет представить структурную схему сложной системы в удобном, легко читаемом виде, при этом уровень вложенности субструктур неограничен. В Главном меню (Поиск – Структура) доступен инструмент, позволяющий в древовидном виде осуществлять навигацию по структурной схеме как текущего проекта, так и любого другого проекта, открытого в текущем экземпляре программы.

Создание математической модели объекта предполагает задание значений внутренних свойств *P* и начальных значений переменных состояния каждого из блоков структурной схемы. Для сложных технических объектов общее число этих параметров исчисляется тысячами, при этом очень часто параметры функционально зависят друг от друга, от некоторых глобальных для модели параметров или являются переменными во времени.

Для решения этой проблемы все свойства (параметры, сигналы, переменные) модели делятся по области действия на *локальные* и *глобальные* свойства и сигналы. *Локальные* свойства являются характеристиками элементарного блока. Сфера их действия ограничена математической моделью блока и, возможно, его панелью управления.

*Глобальные* сигналы и параметры являются параметрами всего проекта (задачи) или конкретной субмодели (макроблока), причем они разделяются на константы и переменные (так же и свойства блока могут быть неизменными величинами, а могут меняться в процессе расчета). Константы задаются или вычисляются один раз перед началом моделирования и не изменяют своих значений при моделировании. Переменные вычисляются в процессе моделирования, и их значения в общем случае зависят от независимой переменной – модельного времени. Константы задаются своими численными значениями (первичные константы) или в виде выражений от определенных выше констант (вторичные константы) и используются для задания локальных параметров элементарных блоков модели. Переменные определяются как функции времени или как произвольные интерпретируемые выражения, в том числе зависящие от других свойств и параметров данного блока, либо от других сигналов, определённых на уровнях выше.

Каждое свойство субмодели может быть однозначно идентифицировано с помощью уникального (в пределах данной субмодели) имени. Область действия именованного свойства строго определена – свойство «видно» только в блоках, находящихся на внутренних (по отношению к субмодели, где задано данное глобальное свойство) уровнях структурной схемы. Если свойство переопределено во внутренней субмодели, он заслоняется этим новым свойством. На рисунке 2.7 приведен пример использования глобальных свойств субмодели.

Необходимость использования глобальных свойств субмодели диктуется следующими соображениями:

1. При выполнении вариантных расчетов необходимо часто изменять некоторые параметры модели. Это удобнее делать, когда все такие параметры поименованы и их описания сосредоточены в одном месте.
2. Один и тот же параметр (например, период квантования дискретной системы) может присутствовать в нескольких блоках модели. В этом случае изменение этого параметра потребовало бы внесения изменений во все эти блоки. Если задать этот параметр глобальным, то все изменения сводятся к изменению одного параметра.
3. При создании модели (субмодели) какого-либо устройства можно разделить все параметры модели на изменяемые, зависящие от конкретной модификации данного устройства, и неизменяемые, одинаковые для всех устройств такого типа. В этом случае целесообразно задать изменяемые параметры как глобальные.
4. С помощью глобальных сигналов удобно задавать оптимизируемые параметры в задачах параметрической оптимизации и оптимального управления.



Рисунок 2.7 - Состав, структура и параметры субмодели

Наряду с глобальными параметрами модели в среде моделирования SimInTech введено понятие *глобальных переменных* модели. Глобальные переменные формируются путем использования типового блока *В память*, входной сигнал которого Пользователь может сделать именованным. По сути блок *В память* задаёт имя для подключенной к нему линии связи.

Кроме того, глобальными переменными проекта являются элементы списка сигналов проекта и/или элементы подключенной к проекту базы данных сигналов (например, базы сигналов SDB), запись и чтение значений которых производится с помощью блоков *Чтение из списка сигналов*, *Запись в список сигналов*. Переменные, записанные с помощью блока *В память*, и глобальные сигналы проекта не пересекаются друг с другом и представляют из себя два разных списка, потому что предназначены они для разных задач, а именно: глобальные переменные, записываемые с помощью блока *В память*, передаются без запаздывания на один расчетный шаг, которое возникает при записи и чтении сигналов проекта. Однако использование сигналов проекта более предпочтительно в некоторых случаях, так как позволяет получать доступ к сигналам из любого места проекта (из скрипта, графического контейнера и т. д.), либо из любого места комплексной модели, в случае использования пакета проектов или сетевого расчета. В обоих случаях, конструкция блока записи глобальной переменной описывается в виде:

*K* = {*U*, *PU*}; *PU* ≡ *U*,

где *PU* - именованный глобальный параметр или сигнал модели, являющийся внутренним параметром типового блока *В память* или блока *Запись в список сигналов.*

*Глобальные переменные* и сигналы модели задаются, а затем используются в любой части структурной схемы проекта. Сигналы базы сигналов могут быть общими между несколькими проектами. Типовой блок *Из памяти* позволяет просмотреть список всех имеющихся в структурной схеме глобальных переменных. Перечень доступных глобальных сигналов можно увидеть в перечне сигналов проекта (пункт Главного меню Сервис – Сигналы), а также в подключенной базе сигналов при наличии такого подключения.

Выбранную Пользователем глобальную переменную либо сигнал проекта можно назначить выходом блока *Из памяти* или *Чтение из списка сигналов* соответственно. Конструкция данных блоков похожа и имеет описание:

*K* = {*Y*, *PY*}; *PY* ≡ *PU*; *Y* ≡ *PY*,

где *PY* – внутренний параметр блоков *Из памяти* или *Чтение из списка сигналов,* обозначающий имя считываемой переменной.

Именованные глобальные переменные широко используются в различных режимах работы. Такой механизм является удобным средством для проведения «невидимых» линий связи между любыми уровнями структурной схемы в любом направлении (с верхних уровней на нижние и наоборот), что позволяет улучшить читаемость сложных многосвязных моделей. Также механизм передачи сигналов и переменных используется при реализации функций КОНТРОЛЯ и УПРАВЛЕНИЯ при помощи построения динамических видеокадров и щитов управления.

Каждый блок имеет свой уникальный графический образ (векторный рисунок и пиктограмму в палитре блоков), позволяющий однозначно идентифицировать его на структурной схеме, а также набор стандартных диалоговых форм для взаимодействия со свойствами и параметрами блока.

На рисунке 2.8 представлена экранная копия закладки ***Свойства*** типового диалогового окна редактирования свойств, на примере блока *Инерционное звено 1-го порядка*, которое на рисунке 2.6 в субмодели *Двигатель ПТ* использовано для описания динамики основной составляющей тока якоря. Свойства математической модели, реализуемой типовым блоком, могут быть заданы как в численном виде, так и с использованием интерпретируемого выражения. Так, на рисунке 2.8 свойство *Начальные условия* задано в численном виде (**[0]**), а свойства *Коэффициенты усиления* и *Постоянные времени* введены с использованием глобальных сигналов (**k\_i** и **t\_i**, соответственно).

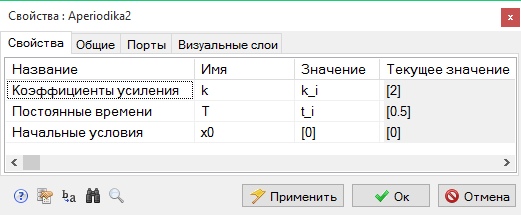


Рисунок 2.8 - Копия диалогового окна (закладка ***Свойства***)

Примечание: запись вектора единичной длины **[0]** в данном случае идентична записи действительного числа **0**. Но, если бы глобальные переменные **k\_i** и **t\_i** были бы векторами из 2-х и более элементов (например, векторами с кол-вом элементов, равным 4), то и начальные условия необходимо было бы сформировать вектором (из 4-х элементов, и записать его как **[0,0,0,0]** или **4#0**, при нулевых начальных условиях). Таким образом, в колонке «значение» в действительности всегда записывается строковое начальное интерпретируемое выражение, которое в момент инициализации блока вычисляется, а вычисленное значение присваивается текущему значению свойства.

Закладка ***Общие*** (рисунок 2.9) того же окна позволяет производить редактирование свойств, являющихся общими для всех блоков, например:

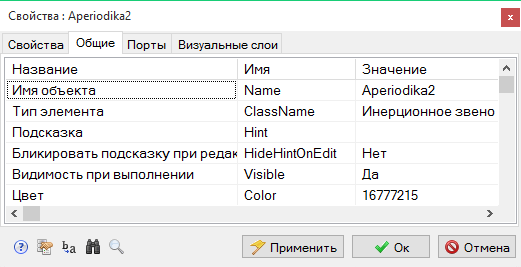


Рисунок 2.9 – Копия диалогового окна (закладка ***Общие***)

* изменять имя блока (уникальный идентификатор в пределах проекта);
* добавлять скрипты инициализации и исполнения объекта;
* изменять поясняющую подпись, ее положение и шрифт;
* изменять цвет фона блока, его прозрачность и положение;
* редактировать изображение блока и др.

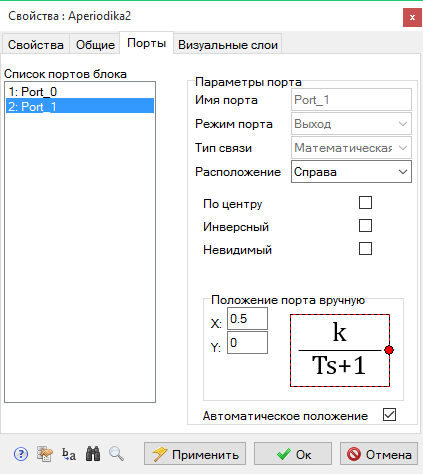


Рисунок 2.10 - Копия диалогового окна (закладка ***Входы***)

Закладка ***Порты*** – содержит инструменты управления расположением входных и выходных портов (рисунок 2.10).

Закладка ***Визуальные слои*** позволяет размещать блок на различных визуальных слоях. Отображением и активностью визуальных слоев расчетной схемы можно управлять, включая или выключая видимость соответствующих наборов блоков. Расчет математической модели блока происходит одинаково, независимо от того, на каком визуальном слое расположен блок. Управление видимостью и активностью слоёв происходит в панели инструментов проекта. Там же возможно переименование слоёв.

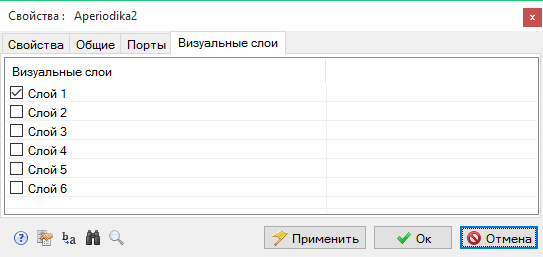
******

Рисунок 2.11 – Копия диалогового окна (закладка ***Визуальные слои***)

* 1. Диагностика ошибок

Среда SimInTech включает в себя систему диагностики ошибок. В таблице 2.4 приведен перечень основных типовых ошибок, определяемых системой диагностики на различных этапах работы среды SimInTech (применительно к расчетному слою «Автоматика» и общетехническим блокам).

Таблица 2.4 – Основные ошибки, формируемые системой диагностики SimInTech

|  |  |
| --- | --- |
| **Сообщение системы диагностики** | **Этап возникновения ошибки** |
| T и k должны быть больше нуля | Проектирование |
| Аргумент арксинуса выходит за границы интервала [-1,1] | Проектирование |
| Аргумент выходит за пределы применимости | Инициализация |
| Аргумент гиперболического котангенса не может быть равен 1 | Проектирование |
| Аргумент логарифма c защитой нуля должен быть больше или равен нулю | Расчет |
| Аргумент логарифма должен быть больше нуля | Расчет |
| Блок должен иметь 2 входа | Инициализация |
| Блок должен иметь два входа | Инициализация |
| Блок должен иметь не менее двух входов | Инициализация |
| Блок должен иметь не менее одного входа | Инициализация |
| Блок должен иметь один входной порт | Инициализация |
| Блок должен иметь хотя бы один вход | Инициализация |
| Блок задания свойства с выхода другого блока не соединён | Инициализация |
| Блок задания уровня свободных портов не подсоединён | Инициализация |
| Блок имеет повторяющийся UID, возможны проблемы генерации кода. Переименуйте его! | Генерация кода |
| Блок подключен к служебному блоку или блоку другого слоя | Инициализация |
| В конце выражения стоит неверный символ | Проектирование |
| Векторная переменная задана неверно | Проектирование |
| Возможно не присвоен выход | Проектирование |
| Возможно не присвоена производная | Инициализация |
| Время запаздывания должно быть БОЛЬШЕ НУЛЯ, т.к. скорость не бесконечна | Инициализация |
| Время запаздывания не может быть отрицательным | Инициализация |
| Вход блока не подсоединён+ | Инициализация |
| Вход блока связан с выключенным из расчета блоком | Инициализация |
| Вход блока соединён с блоком для которого код не может быть сгенерирован | Генерация кода |
| Входная матрица не является квадратной | Инициализация |
| Входная переменная не найдена | Инициализация |
| Выражение задано не полностью | Проектирование |
| Деление на ноль | Расчет |
| Делитель должен иметь два входа | Инициализация |
| Для данного блока программа не может быть сгенерирована | Генерация кода |
| Доступ к элементу невозможен | Проектирование |
| Задан решатель системы другого типа | Инициализация |
| Заданная точность не достигается! | Расчет |
| Запись задана неверно | Проектирование |
| К-во столбцов в файле меньше заданного в параметрах блока | Проектирование |
| К-во функций в файле меньше заданного в параметрах блока | Проектирование |
| Ключевое слово задано неверно | Проектирование |
| Количество аргументов не совпадает | Проектирование |
| Количество шагов должно быть ненулевым | Инициализация |
| Комплексное число не может быть так задано | Проектирование |
| Константу нельзя присваивать | Проектирование |
| Коэффициент в знаменателе равен нулю | Проектирование |
| Массив не может быть создан с данными операндами | Инициализация |
| Массив не существует | Проектирование |
| Матрица вырождена или линейно зависима | Инициализация |
| Матрица должна быть квадратной | Инициализация |
| Матричная переменная задана неверно | Проектирование |
| Метка не найдена | Проектирование |
| Модуль генерации кода не загружен | Генерация кода |
| Модуль генерации кода не инициализирован | Генерация кода |
| Найден неподключенный входной порт | Инициализация |
| Найдена алгебраическая петля | Инициализация |
| Не загружена run-функция | Инициализация |
| Не задано имя линейной системы | Инициализация |
| Не инициализирована run-функция блока | Инициализация |
| Не создать файл для записи данных | Инициализация |
| Не удалось открыть файл обмена | Инициализация |
| Не удалось открыть файл с данными | Проектирование |
| Не удалось получить данные из таблицы | Проектирование |
| Не удалось прочитать строку таблицы | Проектирование |
| Неверный разделитель | Проектирование |
| Невозможно вычислить производные | Расчет |
| Невозможно получить доступ к полю переменной | Проектирование |
| Невозможно привести типы выходов за заданное число итераций | Расчет |
| Недопустимое имя переменной | Проектирование |
| Неизвестная ошибка выполнения | Инициализация |
| Несовпадение с исполняемой системой, свойство: | Удаленная отладка |
| Несоответствие размерностей входов/выходов | Инициализация |
| Номер решателя блока задан неверно! | Инициализация |
| Операнд не найден или не может быть присвоен | Проектирование |
| Оператор не может быть создан | Проектирование |
| Операция не применима к данному операнду | Проектирование |
| Отрицательный аргумент в действительном корне | Расчет |
| Ошибка в файле | Проектирование |
| Ошибка времени выполнения | Инициализация |
| Ошибка выделения памяти под переменную | Инициализация |
| Ошибка выполнения библиотечной функции | Инициализация |
| Ошибка выполнения функции | Инициализация |
| Ошибка вычисления LU-декомпозиции матрицы | Расчет |
| Ошибка генерации кода | Генерация кода |
| Ошибка генерации кода в секции присвоения переменных состояния | Генерация кода |
| Ошибка доступа к массиву или матрице | Инициализация |
| Ошибка доступа к переменной по ссылке | Инициализация |
| Ошибка доступа к файлу | Инициализация |
| Ошибка доступа к файлу обмена | Инициализация |
| Ошибка запроса списка имён генерируемого текста | Инициализация |
| Ошибка именование выходов блоков | Инициализация |
| Ошибка именования выходных переменных блока | Инициализация |
| Ошибка нахождения решения СЛАУ | Расчет |
| Ошибка при возведении числа в заданную степень | Расчет |
| Ошибка при вызове флага f\_InitObjects | Расчет |
| Ошибка при вызове флага f\_SetAlgCount | Расчет |
| Ошибка при вызове флага f\_SetState | Расчет |
| Ошибка при вызове флага f\_Stop | Расчет |
| Ошибка при итерации алгебраической петли | Расчет |
| Ошибка при расчёте алгебраических функций | Расчет |
| Ошибка при расчёте возмущений по переменным | Расчет |
| Ошибка при расчёте правых частей ДАУ | Расчет |
| Ошибка при расчёте производных блоков | Расчет |
| Ошибка при расчёте якобианов для блоков | Расчет |
| Ошибка присвоения типов данных для выходов блока | Инициализация |
| Ошибка удаления памяти по ссылке | Инициализация |
| Ошибка форматирования текста | Проектирование |
| Переменная не найдена | Проектирование |
| Переменные состояния для блока не могут быть декларированы | Инициализация |
| Перемножитель должен иметь два входа | Инициализация |
| Перемножитель должен иметь хотя бы один вход | Инициализация |
| Переполнение в правых частях системы | Расчет |
| Период сигнала должен быть больше нуля | Инициализация |
| Полином задан неверно | Инициализация |
| Порядок знаменателя меньше 2 | Проектирование |
| Порядок знаменателя меньше чем числителя | Проектирование |
| Порядок числителя меньше 2 | Проектирование |
| Постоянная времени T2 блока равна или меньше нуля | Проектирование |
| Постоянная времени блока равна или меньше нуля | Проектирование |
| Превышено ограничение по количеству блоков | Инициализация |
| Превышено ограничение по количеству динамических переменных | Инициализация |
| Произошло деление на ноль | Расчет |
| Произошло деление на ноль - введите ненулевое значение eps | Инициализация |
| Работа модуля генерации кода завершена некорректно | Генерация кода |
| Развязка петли рекомендуется для этого блока | Инициализация |
| Размер вектора должен быть степенью 2 | Инициализация |
| Размерности массивов tau\_on и tau\_of должны быть одинаковыми | Инициализация |
| Размерности массивов параметров не совпадают | Инициализация |
| Размерности матриц не подходят | Инициализация |
| Размерность tau меньше чем y0 | Инициализация |
| Размерность вектора задана неверно | Проектирование |
| Размерность векторов времён и значений не совпадают | Инициализация |
| Размерность массива a меньше чем у массива x0 | Инициализация |
| Размерность массива a1 или массива a2 меньше чем у массива a0 | Проектирование |
| Размерность массива b или массива c меньше чем у массива a | Проектирование |
| Размерность массива b меньше чем у массива a | Проектирование |
| Размерность массива b меньше чем у массива a | Инициализация |
| Размерность массива d или qt меньше чем у массива m | Инициализация |
| Размерность массива eps меньше чем у массива k | Проектирование |
| Размерность массива k или массива T меньше чем у массива x0 | Проектирование |
| Размерность массива k или массива T меньше чем у массива y0 | Проектирование |
| Размерность массива k или массивов T1, T2 меньше чем у массива x0 | Проектирование |
| Размерность массива k меньше чем у массива x0 | Проектирование |
| Размерность массива k меньше чем у массива y0 | Проектирование |
| Размерность массива t или массива dy меньше чем у массива y | Инициализация |
| Размерность массива w или массива f меньше чем у массива a | Инициализация |
| Размерность массива w или массива f меньше чем у массива a | Инициализация |
| Размерность массива xmax или qt меньше чем у массива xmin | Инициализация |
| Размерность массива y0 или массива yk меньше чем у массива t | Инициализация |
| Размерность массива ymin меньше чем у массива ymax | Проектирование |
| Размерность матриц не соответствует указанным количествам переменных | Проектирование |
| Размерность одного из массивов меньше чем у массива x0 | Проектирование |
| Размерность одного из массивов меньше чем у массива y01 | Проектирование |
| Размерность одного из массивов меньше чем у массива y1 | Инициализация |
| Размножитель должен иметь один вход | Инициализация |
| Результат выражения не может быть найден | Проектирование |
| Решение ДАУ не сходится | Расчет |
| Символ не может быть использован | Проектирование |
| Символ не может быть использован в выражении | Проектирование |
| Скобка не закрыта или содержит недопустимое выражение | Проектирование |
| Списка имён данных или списка сигналов не найдено | Инициализация |
| Сумматор должен иметь два входа | Инициализация |
| Сумматор должен иметь хотя бы один вход | Инициализация |
| Табличная функция задана неоднозначно | Инициализация |
| Тип данных и тип ссылки несовместимы | Инициализация |
| Тип переменной задан неверно | Проектирование |
| Типы данных несовместимы | Проектирование |
| У данного блока должно быть чётное число входов | Инициализация |
| У этого блока должно быть не менее двух портов | Инициализация |
| Уровень свободного порты задан дважды | Инициализация |
| Файл не найден | Проектирование |
| Файл обмена не создан | Инициализация |
| Функция задана неверно | Инициализация |
| Функция не может быть переопределена | Проектирование |
| Функция не найдена | Проектирование |
| Элемент массива задан неверно | Проектирование |

1. Численные методы и алгоритмы Моделирования
   1. Постановка задачи

В результате анализа графа структурной схемы автоматически формируется математическая модель объекта исследования в виде системы дифференциально–алгебраических уравнений (ДАУ) вида

|  |  |
| --- | --- |
|  | (3.1) |
|  | (3.2) |

где система уравнений (3.1) описывает непрерывные, а система уравнений (3.2) – дискретные блоки. Здесь *t* – время; *x* – переменные состояния; *y* – выходы блоков; *u* – входы в блоки; *k* – индекс такта квантования по времени. Вторые уравнения в системах (3.1) и (3.2) появляются при наличии в схеме алгебраических контуров.

Первое векторное уравнение системы (3.1) является системой обыкновенных дифференциальных уравнений (ОДУ). Остальные уравнения систем (3.1) и (3.2) образуют в общем случае систему нелинейных алгебраических уравнений (НАУ). Таким образом, основу численных методов и алгоритмов моделирования динамических систем составляют методы интегрирования системы ОДУ и методы решения системы НАУ, а также алгоритмы, обеспечивающие их функционирование (алгоритм численного расчета матрицы Якоби, алгоритмы решения системы линейных алгебраических уравнений (ЛАУ) и др.).

Эффективность численного решения системы ОДУ определяется соответствием между выбранным алгоритмом интегрирования и свойствами решаемой системы уравнений. К определяющим свойствам системы уравнений относятся ее размерность, жесткость и наличие неоднозначных и разрывных характеристик. Формирование структурной схемы в виде входо–выходных отношений позволяет использовать как явные, так и неявные методы интегрирования (в отличие, например, от узлового метода, где использование явных методов невозможно).

Недостатком известных явных методов является необходимость выбирать шаг интегрирования, в несколько раз меньше наименьшей постоянной времени, даже если искомое решение изменяется достаточно медленно. Этого недостатка лишены неявные методы, в которых шаг зависит от скорости изменения решения и может превышать наименьшую постоянную времени.

Явные методы просты в реализации, а неявные требуют решения на каждом шаге интегрирования системы НАУ. Кроме того, неявные методы весьма чувствительны к наличию в структурной схеме таких блоков, как ключи, реле, логические блоки и т.д.

Поэтому для численного интегрирования систем ОДУ в среде SimInTech предусмотрено несколько явных и неявных алгоритмов. При описании алгоритмов используются следующие обозначения:

*x* – вектор переменных состояния;

*t* – текущее модельное время;

*f* – вектор производных по времени;

*H* – шаг интегрирования в методах с постоянным шагом;

*h* – шаг интегрирования в методах с автоматическим выбором шага;

*eps* – оценка локальной погрешности.

* 1. Стандартные явные методы интегрирования

Численный метод Эйлера реализован в классической постановке [22]:

Метод рекомендуется применять в задачах, где шаг интегрирования ограничен не условием устойчивости метода, а другими факторами (например, шаг обмена данными с внешними расчетными программами для структурной схемы, моделирующей логическую часть системы управления реакторной установкой; шаг вывода информации для структуры, не содержащей дифференциальные переменные и т.д.).

Численный метод Рунге-Кутта 4-го порядка с фиксированным шагом интегрирования также реализован в классической постановке [22]:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Метод Рунге-Кутта 4-го порядка с автоматическим выбором шага реализован в сочетании с экстраполяцией по Ричардсону. Полученный в результате метод задается формулой

где и – численные решения, полученные, соответственно, путем выполнения одного шага величиной *h* и двух шагов величиной *h /2* классического метода Рунге-Кутта, расчетные формулы которого имеют вид

|  |  |  |
| --- | --- | --- |
|  |  |  |

Численный метод Кутта-Мерсона 4-го порядка с автоматическим выбором шага интегрирования заимствован из работы [23] и его алгоритм описывается в следующем виде:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Классические методы Рунге-Кутта и Кутта-Мерсона рекомендуется использовать для интегрирования нежестких систем с высокой точностью.

* 1. Модифицированные явные методы интегрирования

Модифицированный метод Рунге-Кутта отличается от классического способом вычисления коэффициента :

где вектор вычисляется покомпонентно, в зависимости от соответствующих компонент вектора

Для нежестких компонент () используется классическая формула

а для жестких () принимается:

Модифицированный метод Кутта-Мерсона отличается от классического способом вычисления :

где вектор вычисляется покомпонентно в зависимости от соответствующих компонент вектора

где деление также выполняется покомпонентно. Для нежестких компонент () используется классическая формула

а для жестких () принимается

Модификация методов Рунге-Кутта и Кутта-Мерсона позволила увеличить их эффективность при решении жестких задач. Методы рекомендуется использовать для интегрирования нежестких и слабожестких систем с высокой или средней точностью.

* 1. Адаптивные явные методы интегрирования

Здесь предлагаются явные одношаговые адаптивные методы, использующие оценки наибольших по модулю собственных значений матрицы Якоби для повышения эффективности численного интегрирования жестких систем ОДУ [24].

* + 1. Адаптивный-1

Этот метод - явный одношаговый трехстадийный (на каждом шаге производится три обращения к процедуре вычисления правых частей). Стадии выполняются по формулам



где β=1, а значение α вычисляются на основе информации предыдущего шага. Принимаем



и покомпонентно вычисляем вектор оценок наибольшего по модулю собственного значения матрицы *h***J**, где **J** - якобиан в текущей точке решения. Формула шага интегрирования имеет следующий вид:



Для оценивания ошибки используется двухшаговая формула типа Адамса. Метод имеет второй порядок. Его рекомендуется использовать для решения нежестких и жестких задач с вещественным жестким спектром при низких требованиях к точности ().

* + 1. Адаптивный-2

Расчетные формулы этого метода практически такие же, как у метода Адаптивный-1. Отличие заключается в том, что ошибка решения оценивается по правилу Рунге, т.е. используя один шаг размером *h* и два шага размером *h*/2. Метод имеет третий порядок. Его рекомендуется использовать для решения жестких задач с вещественным жестким спектром при низких требованиях к точности ().

* + 1. Адаптивный-3

Явный четырехстадийный адаптивный метод реализован в соответствии с формулами (все арифметические операции с векторами выполняются покомпонентно):







Для оценивания погрешности используется формула

,

где коэффициенты *b*1,...,*b*4 рассчитываются в зависимости от значения α. Метод имеет 4-й порядок для линейных и 3-й для нелинейных систем. Его рекомендуется использовать для интегрирования жестких систем с вещественным жестким спектром при средних требованиях к точности ()

* + 1. Адаптивный-4

Этот метод является многошаговым и реализуется в 3 этапа:

* прогноз по явной формуле Адамса;
* покомпонентное оценивание наибольшего собственного значения;
* покомпонентная коррекция по неявной многошаговой формуле.

Формула коррекции позволяет стабилизировать расчетную схему в полученных точках жесткого спектра. Метод имеет переменный порядок (от 2-го до 6-го). Для оценивания погрешности используется многошаговая формула более низкого порядка. Данный метод рекомендуется использовать для решения нежестких и жестких задач с вещественным жестким спектром при любых требованиях к точности.

* + 1. Адаптивный-5

Стадии явного пятистадийного одношагового метода выполняются по формулам



На основе полученной информации вычисляются покомпонентные оценки двух наибольших по модулю собственных значений (которые могут быть комплексно-сопряженными). Шаг интегрирования выполняется по формуле

,

где коэффициенты  вычисляются с использованием полученных оценок наибольшего собственного значения. Оценка ошибки решения производится по двухшаговой формуле.

Метод имеет 4-й порядок для линейных и 3-й для нелинейных систем. Его рекомендуется использовать для решения умеренно жестких задач с комплексным жестким спектром, а также осциллирующих задач с собственными значениями якобиана вблизи мнимой оси.

* 1. Адаптивный неявный метод интегрирования

Этот метод построен на основе неявного метода трапеций, расчетная формула которого имеет вид

x(t + h) = x + h/2⋅( x’(t) + x’(t + h) ).

Выполняя один шаг величиной h, получим x1; выполняя два шага величиной h/2, получим x2. Заключительная расчетная формула имеет вид:



Погрешность оценивается по формуле:

eps = x2 – x1

Порядок точности метода изменяется от 2-го до 4-го в зависимости от жесткости системы. Метод весьма эффективен при интегрировании систем, жесткость которых существенно изменяется в процессе интегрирования или от одной компоненты к другой. Рекомендуется использовать этот метод при моделировании таких систем со средней или высокой точностью.

* 1. Диагонально неявные FSAL-методы Рунге-Кутты
     1. Алгоритм DIRK-33

Диагонально неявный метод Рунге-Кутты (DIRK - Diagonally Implicit Runge-Kutta) реализуется с использованием трех неявных стадий и имеет третий порядок. Первая стадия является явной и совпадает с последней стадией предыдущего шага, поэтому методы такого типа обычно называют FSAL-методами [25] (FSAL - First Same As Last). Один шаг метода выполняется в соответствии с формулами

 (3.3)

Здесь *s*=3 - число неявных стадий, а коэффициенты метода равны



Детали реализации изложены в [25].

Метод рекомендуется использовать для решения жестких и дифференциально-алгебраических систем при низких и средних требованиях к точности ().

* + 1. Алгоритм DIRK-44

Диагонально неявный FSAL-метод Рунге-Кутты реализуется с использованием четырех неявных стадий и имеет четвертый порядок. Один шаг метода выполняется в соответствии с формулами (3.3) при *s*=4. Коэффициенты метода равны:



Метод рекомендуется использовать для решения жестких и дифференциально-алгебраических систем при средних и высоких требованиях к точности ().

* 1. Модифицированный метод Гира

В качестве одного из неявных жестко устойчивых методов интегрирования реализован модифицированный метод Гира. Метод относится к семейству многошаговых методов полиномиальной аппроксимации. Общий вид многошаговой формулы численного интегрирования имеет следующую форму [26]:

 (3.4)

где k – порядок метода; n – временной слой; h – шаг интегрирования; ai, bi – коэффициенты метода; индекс m относится к m-ому уравнению исходной системы ОДУ, представленной в нормальной форме Коши: x'(t) = f(x, t); M – число уравнений в системе.

Метод Гира может быть представлен как метод прогноза-коррекции, канонический вид которого, выраженный посредством векторного представления Нордсика, имеет следующую форму представления:

прогноз : ; (3.5)

коррекция:  (3.6)

где N – число итераций; m = 1, 2…, M;  – вектор Нордсика; Z – треугольная матрица Паскаля, элемент  которой определяется выражением ;  – ”корректирующая” функция на j-ой итерации; cz – {cz1 , cz2 , cz3 ,... czk+1} – вектор коэффициентов, зависящий от текущего порядка метода.

Для определения вектора “корректирующих” функций gj используется модифицированный итерационный метод Ньютона-Рафсона. При этом система линейных уравнений для определения gj имеет следующий вид:

, (3.7)

где b-1 – коэффициент при неявном члене в уравнении метода;  – вектор производных по времени на j-ой итерации; – вектора, сформированные соответственно из первого и второго элементов векторов Нордсика, полученных на j-ой итерации; I – единичная матрица; J – матрица Якоби.

При переходе с n-го на n+1-ый временной слой последовательность действий выглядит следующим образом:

1. Подставить запомненный вектор Нордсика  в уравнение прогноза (3.5) и получить предсказываемый вектор Нордсика .
2. В векторе Нордсика выделить компоненты , , используя их, сформировать правые части в системе ЛАУ (3.7), решив эту систему, определить вектор корректирующих функций gj.
3. C помощью уравнения (3.6) вычислить вектор Нордсика  на (j+1)-ой итерации, при этом достаточно получить два первых элемента этого вектора.
4. Повторить шаги 2 и 3, пока итерационный процесс не сойдется с точностью до заданных пределов.
5. Оценить локальную методическую ошибку метода ε; если ошибка меньше допустимой, перейти к шагу 6, иначе изменить шаг и/или порядок метода и перейти к шагу 1 без перехода на следующий временной слой.
6. Проверить, нужно ли изменить порядок метода и величину шага перед вычислением x(t) на следующем временном слое; если нужно, изменить шаг и/или порядок и перейти к шагу 1.

Представление метода Гира с помощью вектора Нордсика позволяет разработать эффективный алгоритм автоматического изменения порядка и величины шага. Для этого записывается выражение для локальной методической ошибки метода Гира для трех разных порядков через разностные вектора:

 (3.8)

где ∇p[] – разностный вектор порядка p;

– последняя компонента вектора Нордсика ;

Ck – константа, зависящая от порядка метода.

При изменении шага от h до α⋅h локальная ошибка при t = tn для методов разных порядков имеет вид:

 (3.9)

Поскольку максимально допустимая ошибка **ε**max задана, из уравнений (3.9) можно получить значения α для разных порядков. Максимальное из трех значений α дает максимально допустимую величину шага α⋅h и соответствующий порядок метода – оптимальный для вычисления xn+1.

Для системы M уравнений разностные векторы в уравнении (3.8) нормируются следующим образом:

 (3.10)

Векторная разность первого порядка может быть оценена путем суммирования скалярной корректирующей функции:

 (3.11)

где N – число итераций, а векторные разности нулевого и второго порядка определяются из соотношений:

∇0 []= , ∇2 []= ∇1 []– ∇1 [] (3.12)

Алгоритм автоматического изменения порядка метода и величины шага применяется при следующих обстоятельствах:

* если точность удовлетворяется для k+1 последовательных шагов после последнего изменения порядка или величины шага;
* если при данной величине шага не удовлетворяется заданная точность;
* если итерационный процесс нахождения скалярной корректирующей функции расходится или превышено максимальное число итераций и не достигнута заданная точность (в этом случае, кроме изменения величины шага, заново рассчитывается матрица Якоби).
  1. Методика вычисления коэффициентов матрицы Якоби

Отдельной проблемой при интегрировании системы ДАУ является вычисление коэффициентов матрицы Якоби

 (3.13)

Для явных методов интегрирования матрицу Якоби требуется вычислять, если в схеме есть алгебраические контуры, а в качестве итерационной схемы используется метод Ньютона. В этом случае в уравнении (3.13) x – вектор определяющих переменных, f – нелинейная векторная функция.

При реализации неявных методов к определяющим переменным в матрице Якоби добавляются дифференциальные переменные модели, для которых в уравнении (3.12) f – производные переменных состояния.

Расчет матрицы Якоби проводится численно. Используя конечные разности 1-го порядка, получим j-й столбец матрицы Якоби в виде

 (3.14)

где Δxi ­– величина приращения по i-ой координате, ei – единичный вектор, направленный вдоль i-ой координаты.

При этом, если не использовать знание информационных потоков в структурной схеме, необходимо N раз рассчитать все блоки схемы, где N – размер системы ДАУ. В то же время для сложных технических систем большой размерности характерна большая разреженность матрицы Якоби. Для иллюстрации сказанного на рисунке 3.1 приведена структурная схема динамической системы второго порядка. Для простоты в качестве динамических блоков выбраны интеграторы, выход которых равен значению переменных состояния x1 и x2. Из рисунка 3.1 видно, что для расчета коэффициента матрицы Якоби ∂x2/∂x1 нет необходимости рассчитывать алгебраические блоки f1, f2, f3 и источниковый блок sin(t), надо просто рассчитать реакцию блока x2 на изменение переменной состояния x1.

Для алгебраических контуров метод определяющих переменных является одним из способов учета разреженности уравнений, сформированных по структурной схеме. Поэтому число определяющих блоков обычно намного меньше общего числа блоков, входящих в алгебраические контуры, благодаря чему снижается общий порядок системы ДАУ.

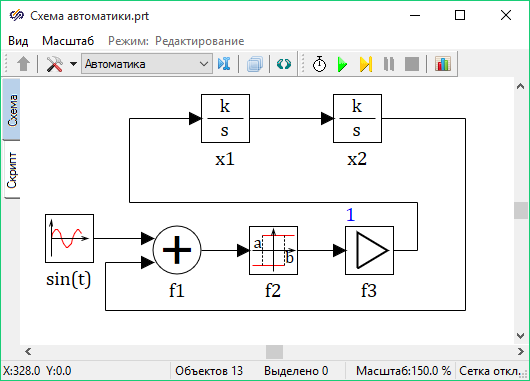


Рисунок 3.1 - Алгоритм вычисления коэффициентов матрицы Якоби

Кроме того, разработан алгоритм расчета якобиана, позволяющий в ходе определения столбца коэффициентов матрицы Якоби рассчитывать только необходимое число блоков структурной схемы. Алгоритм выявляет для каждой координаты xi количество и порядок расчета блоков структурной схемы, входящих в “сферу действия” этой координаты. Если провести аналогию с методом разреженных матриц, данный алгоритм позволяет работать только с ненулевыми элементами матрицы Якоби.

Итерационный метод Ньютона для алгебраических контуров, неявные методы интегрирования, алгоритм определения статического состояния системы требуют решения системы линейных алгебраических уравнений (ЛАУ). Во многих случаях матрица коэффициентов этой системы –разреженная. Большинство методов решения таких систем ЛАУ ориентировано на специальный вид матрицы коэффициентов [27…29]. В данном случае структура матрицы определяется топологией структурной схемы системы, на которую не накладывается никаких ограничений. Поэтому разработан численный алгоритм, ориентированный на матрицы коэффициентов общего вида. Основные характеристики алгоритма [29]:

* хранятся только ненулевые элементы матрицы коэффициентов;
* производятся операции только над ненулевыми элементами матрицы коэффициентов;
* используется прямой метод LU-разложения с хранением получающихся нижней и верхней треугольной матрицы;
* для выбора главного элемента используется улучшенная обобщенная стратегия Марковица, позволяющая при сохранении устойчивости процесса исключения снизить величину заполнения;
* предусмотрена возможность введения барьера (при этом возникающие в ходе LU-разложения новые элементы, модуль которых меньше барьера, принимаются равными 0), что в ряде случаев позволяет значительно снизить заполнение и уменьшить временную стоимость LU-разложения; в этом случае для повышения точности предусмотрено итерационное уточнение полученного прямым методом решения.

1. Численные методы и алгоритмы Оптимизации
   1. Постановка задачи

Одной из основных задач, возникающих при проектировании сложных технических систем, в том числе и САР, является определение оптимальных значений настраиваемых параметров системы исходя из заданных технических требований. Термин “оптимальный” предполагает наличие критериев качества, каждый из которых характеризует точность, быстродействие, энергетические затраты, стоимость и т.д.

Сложность задачи оптимизации определяется прежде всего трудностью ее формализации. Формальный подход (например, путем свертывания нескольких критериев в один) часто приводит к неприемлемому решению. Поэтому процедура параметрической оптимизации должна быть организована таким образом, чтобы в нее в любой момент мог вмешаться Проектировщик и скорректировать задаваемые технические требования. В то же время эта процедура должна быть основана на использовании известных методов параметрической оптимизации.

Исходя из вышесказанного, следует организовать решение задачи параметрической оптимизации в виде диалоговой итерационной процедуры. Очередной этап этой процедуры начинается с ввода (на первом этапе) или корректировки задания на проектирование, после чего запускается программа оптимизации. В процессе оптимизации промежуточная информация о значениях критериев и оптимизируемых параметров должна в наглядной форме выводиться на экран. Пользователь может дождаться окончания очередного этапа либо, при необходимости, прервать его досрочно. Поэтому диалоговую процедуру параметрической оптимизации можно свести к поочередному выполнению двух операций:

* корректировка технического задания на оптимальное проектирование;
* решение задачи параметрической оптимизации по заданным техническим требованиям.

Обозначим x1, x2, ... , xn - параметры проектируемой системы, которые можно изменять в процессе оптимизации для обеспечения заданного качества. На каждый параметр могут накладываться ограничения вида

xi min ≤ xi ≤ xi max (4.1)

Пусть также Q1(x), Q2(x), ... , Qm(x) - показатели (критерии) качества, вычисляемые в результате моделирования при заданных значениях параметров. Требование к каждому из критериев будем задавать в виде ограничений

Qi min ≤ Qi (**x**)≤ Qi max. (4.2)

Задачу параметрической оптимизации сформулируем следующим образом: найти вектор параметров **x**\* = [ x1\*, . . . , xn\* ]T, удовлетворяющий ограничениям (4.1), при котором показатели качества удовлетворяют ограничениям (4.2). Подчеркнем еще раз, что задачу многокритериальной параметрической оптимизации трудно формализовать, поэтому мы формализовали только один этап решения этой задачи. Если после выполнения заданного количества расчетных итераций, критерии оптимизации не достигли заданных значений, то оптимизация завершается. В этом случае Пользователю, на основании полученной информации, следует принять решение об окончании оптимизации либо о корректировке задания на оптимизацию. Корректировка может заключаться в изменении набора оптимизируемых параметров и критериев, а также в изменении граничных значений в ограничениях (4.1), (4.2).

Эффективность оптимизации определяется прежде всего обусловленностью задачи. Плохо обусловленные задачи - это задачи овражные, негладкие, многоэкстремальные. Обусловленность задачи сильно зависит также от выбранного масштаба параметров и критериев. Действительно, параметры и критерии могут быть величинами, имеющими самую различную физическую природу, и диапазоны их изменения могут различаться весьма существенно. Поэтому только изменение масштаба и переход к нормированным величинам часто приводит к значительному повышению эффективности оптимизации.

Задание диапазонов изменения параметров и критериев с учетом их физического смысла позволяет произвести их естественное масштабирование. Например, начальные шаги по параметрам задаются в виде

Δxi = 0,1 (ximax -ximin ). (4.3)

Эффективные алгоритмы обычно сами осуществляют перестройку системы координат в процессе оптимизации, тем не менее удачный выбор масштаба существенно повышает эффективность на начальных этапах поиска.

Аналогичным образом следует произвести масштабирование критериев качества. Сформируем нормированные частные критерии в виде

 (4.4)

В этом случае нормированные критерии (4.4) принимают нулевые значения, если соответствующие ограничения (4.2) выполняются и положительные значения в противном случае.

Для решения задачи многокритериальной оптимизации обычно переходят от нескольких частных критериев q1, ... , qm к одному общему критерию, который формируется в виде функции частных критериев. Такую процедуру называют свертыванием критериев. В результате получаем общий критерий (целевую функцию)

f(**x**) = ϕ (q1(**x**), ... , qm(**x**) ) (4.5)

в виде функции от оптимизируемых параметров. Решение задачи многокритериальной оптимизации сводится к минимизации этого критерия. Существует множество способов свертывания частных критериев [30]. Приведем один из них, обобщающий многие известные способы. Общий критерий можно сформировать в виде среднего степенного критерия оптимальности [30].

 (4.6)

При p = 1 получим аддитивный критерий

 (4.7)

При p = 2 получим квадратичный критерий

 (4.8)

При p, стремящемся к бесконечности, общий критерий (4.6) сводится к наибольшему из нормированных частных критериев (минимаксный критерий)

ϕ∞(q1 ,..., qm) = max (q1 ,..., qm). (4.9)

При p = 0, логарифмируя выражение (4.6) и переходя к пределу по p, стремящемуся к нулю, после применения правила Лопиталя получаем средний геометрический (мультипликативный) критерий оптимальности:

 (4.10)

Можно показать, что для среднего степенного критерия (4.6) при любом p оптимальное решение **x**\* задачи параметрической оптимизации

ϕp(q1(**x**) ,..., qm(**x**) ) → min (4.11)

является эффективной точкой и принадлежит множеству Парето исходной задачи многокритериальной оптимизации.

Методы оптимизации в зависимости от порядка используемых производных подразделяются на методы нулевого, первого и второго порядка.

Методы нулевого порядка (прямые методы или методы поиска) используют только значения функции. Среди эффективных методов прямого поиска можно назвать методы Розенброка и Пауэлла [21], метод конфигураций Хука-Дживса, симплекс-метод Нелдера-Мида, комплекс-метод Бокса [31]. Все они эвристические и особенно эффективны в тех случаях, когда другие методы с высокой скоростью сходимости неспособны решить задачу, например, если минимизируемая функция не является гладкой.

Методы первого порядка (градиентные) используют, кроме значений функции, значения ее производных. Наиболее эффективными среди градиентных являются методы переменной метрики или квазиньютоновские [21], основанные на аппроксимации матрицы вторых производных или обратной к ней. Градиентные методы целесообразно применять в тех случаях, когда есть возможность вычислять производные минимизируемой функции.

Методы второго порядка (ньютоновские) дополнительно требуют вычисления вторых производных, обладают квадратичной сходимостью и быстро сходятся, если функция выпуклая и достаточно гладкая. Однако эти методы могут быть малоэффективными или даже расходиться, если функция невыпуклая. Методы второго порядка следует использовать лишь в тех случаях, когда достаточно легко получить матрицу вторых производных минимизируемой функции.

Использование методов первого и второго порядка требует расчета производных общего критерия качества. Если критерий вычисляется по результатам моделирования, то его производные можно рассчитывать либо численно, давая приращения параметрам, либо по уравнениям чувствительности. Численный расчет производных требует большого объема вычислений и часто приводит к значительным ошибкам. Использование уравнений чувствительности вызывает увеличение порядка исследуемой системы и требует дополнительной памяти. Определенные трудности встречает моделирование чувствительности систем, содержащих неоднозначные и разрывные характеристики (люфт, сухое трение, реле). Поэтому при оптимизации по результатам моделирования целесообразно использовать методы нулевого порядка.

Дополнительные ограничения накладываются на методы поиска при использовании описанной выше процедуры диалоговой оптимизации по векторному критерию. В этом случае нельзя применять методы, использующие квадратичную интерполяцию, поскольку они требуют вычисления значений скалярной целевой функции. Плохо подходят также методы, в которых для определения новой текущей точки поиска необходимо сравнивать более двух вариантов, поскольку в этом случае значительно возрастает нагрузка на Проектировщика. Поэтому наиболее эффективные из существующих методов поиска, такие как методы Пауэлла и Нелдера-Мида, мало подходят для диалоговой оптимизации.

Ниже описываются алгоритмы оптимизации, в которых решение о переходе в новую точку поиска принимается на основании сравнения значений критерия в двух точках.

* 1. Алгоритм «Поиск-2»

Реализуется алгоритм деления шага пополам при одном оптимизируемом параметре (n = 1) и алгоритм преобразований матрицы направлений при n >1 [21]. Далее рассматривается алгоритм многомерного поиска.

Направления поиска на k-том этапе задаются матрицей Sk. На очередном этапе производится серия спусков в направлениях векторов s1,...,sn, представляющих собой столбцы матрицы Sk. Векторы перемещений на каждом из спусков равны соответственно γ1s1, ..., γnsn. После выполнения спусков матрица направлений преобразуется по формуле

Sk+1 = Sk Λk Pk , (4.12)

где Λk****- диагональная матрица, элементы которой равны λk = γi, если γi ≠ 0, и λk = 0.5, если γi = 0; Pk - ортогональная матрица.

Умножение на ортогональную матрицу необходимо для изменения набора направлений поиска. Если на всех этапах Pk = I , то направления поиска не изменяются от этапа к этапу и мы имеем алгоритм покоординатного спуска. Очевидно, что выбор матриц Pk существенно влияет на эффективность поиска.

Было испытано несколько различных способов выбора ортогональных матриц Pk , в том числе и случайный выбор. Лучшим оказался способ, при котором все матрицы Pk равны между собой и определяются в виде

 (4.13)

Рассмотрим этапы алгоритма в многомерном случае.

1. Начальная матрица направлений задается диагональной с элементами на главной диагонали, равными начальным приращениям по параметрам.
2. Выполнить цикл для i = 1 , . . . , n :
   1. Выполнить пробный шаг в направлении **s**i :

y = x + si .

Если этот шаг удачный ( f(y) < f(x) ), перейти к пункту 2.3.

1. Выполнить пробный шаг в противоположном направлении:

y = x - si .

Если оба пробных шага оказались неудачными, принять λ= 0.5 и перейти к пункту 2.4.

1. Выполнить спуск в выбранном направлении, в результате получим новую точку поиска

x = x + γsi , принять λ = ⏐γ⏐.

1. Принять si = λsi. Перейти к следующему значению счетчика цикла либо выйти из цикла (если i = n).
2. Умножить матрицу направлений S на ортогональную матрицу P, задаваемую (4.13).
3. При выполнении условия окончания поиска завершить работу алгоритма, в противном - перейти к п. 2 с новыми значениями вектора x и матрицы S.

Поиск прекращается при выполнении одного из следующих условий:

* + целевая функция достигла минимума (все требования выполняются);
  + превышено заданное число вычислений целевой функции;
  + приращения по каждому из параметров стали меньше заданного значения;
  + принудительный останов.
  1. Алгоритм «Поиск-4»

Реализуется алгоритм квадратичной интерполяции при одном оптимизируемом параметре (n = 1) и алгоритм преобразований вращения и растяжения-сжатия (n >1) [21].

Рассмотрим алгоритм при n > 1. Он основан на выполнении преобразований растяжения - сжатия и преобразований вращения для такого преобразования системы координат, при котором матрица вторых производных (матрица Гессе) приближается к единичной, а направления поиска становятся сопряженными. Этот алгоритм использует квадратичную интерполяцию.

Пусть H- симметричная положительно определенная матрица. Будем строить последовательность матриц

H0 = H, H1 ,..., Hk ,

каждая из которых получается из предыдущей путем выполнения следующего преобразования

Hk = PkT ΛkT Hk-1 Λk Pk , (4.14)

где Λk - диагональная матрица с элементами λi = hii-1/2 (*hii* - диагональные элементы Hk 1); Pk - ортогональная матрица. После умножения матрицы Hk-1 слева и справа на Λk получаем матрицу с единичными диагональными элементами. Можно надеяться, что при подходящем выборе ортогональных матриц Pk матрица Hk будет стремиться к единичной. На этом, в частности, основан метод вращений для расчета собственных значений симметричных матриц.

Рассмотрим задачу поиска минимума функции нескольких переменных. На k-м этапе поиска поочередно минимизируется функция в направлениях векторов s1,...,sn, представляющих собой столбцы матрицы Sk. Для нахождения точки минимума в направлении si используется квадратичная интерполяция по трем равноотстоящим точкам

z = x - si , x , y=x + si . (4.15)

Одновременно для каждого направления вычисляется

λi = (f(y) + f(z)-2f(x)) - 1/2 . (4.16)

После выполнения серии спусков матрица S преобразуется по формуле

Sk+1 = Sk Λk Pk , (4.17)

где Λk - диагональная матрица, элементы которой определяются по (4.16); Pk - некоторая ортогональная матрица. Для квадратичной целевой функции матрица SkT H Sk , где H - матрица Гессе, совпадает с матрицей Hk, полученной при выполнении преобразований (4.12). Таким образом, при надлежащем выборе матриц Pk для квадратичной функции получаем SkT H Sk → Iи направления поиска приближаются к сопряженным. В рассматриваемом алгоритме матрицы Pk одинаковы на всех этапах и определяются по формуле (4.13).

Этапы работы алгоритма Поиск - 4 аналогичны рассмотренным выше этапам алгоритма Поиск - 2.

* 1. Алгоритм «Симплекс»

Используется метод “деформируемого многогранника” Нелдера и Мида [31].

В методе Нелдера-Мида минимизируется функция n независимых переменных с использованием n+1 вершин деформируемого многогранника. Каждая вершина может быть идентифицирована вектором **x**. Вершина (точка), в которой значение f(**x**) максимально, проектируется через центр тяжести (центроид) оставшихся вершин. Улучшенные (меньшие) значения целевой функции находятся последовательной заменой точки с максимальным значением f(**x**) на более “хорошие” точки, пока не будет найден минимум f(**x**).

Далее кратко излагается суть алгоритма.

Пусть **x**i(k) = [xi1(k),..., xij(k),..., xin(k)]T, i = 1,..., n+1, является i-й вершиной (точкой) на k-том этапе поиска, k = 0, 1,..., и пусть значение целевой функции в **x**i(k) равно f(**x**i(k)). Также отметим векторы многогранника, которые дают максимальное и минимальное значения.

Определим

f(**x**h(k)) = max{f(**x**1(k)),...,f(**x**n+1(k))}, (4.18)

где **x**h(k) = **x**i(k) , и

f(**x**l(k)) = min{f(**x**1(k)),...,f(**x**n+1(k)), (4.19)

где **x**l(k) = **x**i(k) .

Поскольку многогранник в En состоит из (n+1) вершин **x**1,...,**x**n+1, пусть **x**n+2 будет центром тяжести всех вершин, исключая **x**h.

Тогда координаты этого центра определяются формулой

**x**n+2,j(k) = (1/n)[(**x**ij(k))-**x**hj(k) ], i = 1,..., n+1; j =1,..., n; (4.20)

где индекс j обозначает координатное направление.

Начальный симплекс обычно (не всегда) выбирается в виде регулярного симплекса, причем начало координат можно поместить в центр тяжести. Процедура отыскания вершины в En , в которой f(**x**) имеет лучшее значение, состоит из следующих операций.

*Отражение* - проектирование **x**h(k) через центр тяжести в соответствии с выражением

**x**n+3(k) = **x**n+2(k) +**x**n+2(k)-**x**h(k)), (4.21)

где  является коэффициентом отражения; **x**n+2(k) - центр тяжести, вычисляемый по формуле (4.20); **x**h(k) - вершина, в которой функция f(**x**) принимает наибольшее из n+1 ее значений на k- том этапе.

*Растяжение*. Эта операция состоит в следующем: если f(**x**n+3(k)) <= f(**x**l(k)), то вектор(**x**n+3(k)-**x**n+2(k)) растягивается в соответствии с соотношением

**x**n+4(k)= **x**n+2(k) +**x**n+3(k)-**x**n+2(k)), (4.22)

где >1 представляет собой коэффициент растяжения. Если f(**x**n+4(k)) <f(**x**l(k)) , то **x**h(k) заменяется на **x**n+4(k) и процедура продолжается снова с операции 1 при k = k+1. В противном случае **x**h(k) заменяется на **x**n+3(k) и также осуществляется переход к операции 1 при k = k+1.

*Сжатие*. Если f(**x**n+3(k)) > f(**x**i(k)) для всех i < > h , то вектор (**x**h(k)-**x**n+2(k)) сжимается в соответствии с формулой

**x**n+5(k)= **x**n+2(k) +**x**h(k)-**x**n+2(k)), (4.23)

где 0 < <1 представляет собой коэффициент сжатия. Затем **x**h(k) заменяем на **x**n+5(k) и возвращаемся к операции 1 для продолжения поиска на (k+1) шаге.

*Редукция*. Если f(**x**n+5(k)) > f(**x**h(k)), все векторы (**x**i(k)-**x**l(k)), i = 1, ..., n +1, уменьшаются в 2 раза с отсчетом от **x**l(k) в соответствии с формулой

**x**i(k) = **x**l(k) +0.5**x**i(k)-**x**l(k)), i = 1, ..., n+1. (4.24)

Затем возвращаемся к операции 1 для продолжения поиска на (k + 1) шаге.

Критерий окончания поиска - проверка условия

{[1/(n+1)] [f(**x**i(k))-f(**x**n+2(k))]2}1/2 <=  , (4.25)

где  - произвольное малое число, а f(**x**n+2(k)) - значение целевой функции в центре тяжести **x**n+2(k).

На процесс оптимизации оказывают влияние коэффициенты отражения , растяжения  и сжатия . Коэффициент отражения  используется для проектирования вершины с наибольшим значением f(**x**) через центр тяжести деформируемого многогранника. Коэффициент  вводится для растяжения вектора поиска в случае, если отражение дает вершину со значением f(**x**) меньшим, чем наименьшее значение f(**x**), полученное до отражения. Коэффициент сжатия  используется для уменьшения вектора поиска, если операция отражения не привела к вершине со значением f(**x**), меньшим, чем второе по величине (после наибольшего) значение f(**x**), полученное до отражения. Таким образом, с помощью операций растяжения или сжатия размеры и форма деформируемого многогранника масштабируются так, чтобы они удовлетворяли топологии решаемой задачи.

После того, как деформируемый многогранник подходящим образом масштабируется, его размеры должны поддерживаться неизменными, пока изменения в топологии задачи не потребуют применения многогранника другой формы. Анализ, проведенный Нелдером и Мидом /22/, показал, что компромиссное значение = 1. Ими также рекомендованы значения = 0,5 = 2. Более поздние исследования показали, что рекомендуются диапазоны 0,4<= <= 0,6; 2,8 <=  <= 3,0; причем при 0<  < 0,4 существует вероятность того, что из-за уплощения многогранника будет иметь место преждевременное окончание процесса, а при  > 0,6 может потребоваться большее число шагов для достижения окончательного решения.

* 1. Пример решения задачи оптимизации

В качестве иллюстрации возможностей среды SimInTech при решении задач оптимизации рассмотрим следующую задачу из области синтеза оптимальных регуляторов. Объект управления задан передаточной функцией



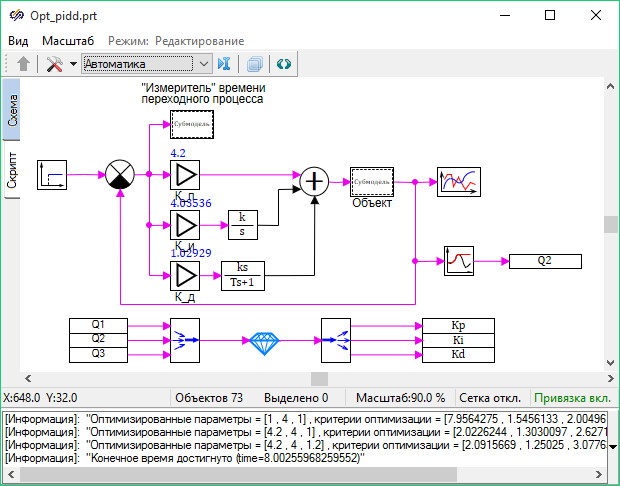
Задача: Найти параметры последовательного корректирующего устройства, обеспечивающего 2-й порядок астатизма, добротность по ускорению 4…6 с-2, величина перерегулирования не более 36 % , время переходного процесса по уровню 5 % не более 1,8 с, ограничение по модулю скорости изменения выходного сигнала – не более 3 c -1. Корректирующий ПИД - регулятор реализуется в виде 3-х параллельно соединенных звеньев (идеального усилительного, идеального интегрирующего и инерционно-дифференцирующего) с передаточной функцией



Для обеспечения заданной добротности скоростная эффективность (параметр K\_и) должна удовлетворять ограничениям 4 ≤ K\_и ≤ 6. Ограничения на остальные параметры: 0 ≤ K\_п ≤ 20 и 0 ≤ K\_д ≤ 10. Пусть начальные значения параметров равны: K\_п = 0,5; K\_и = 5; K\_д = 2.

На рисунке 4.1 представлены экранные копии Главного схемного окна с отчетом по результатам процесса оптимизации, одного из субмодельных схемных окон нижнего уровня вложенности (“***Измеритель времени переходного процесса****”*), а также окно графика переходного процесса. Формирование 2-го критерии качества – **Q2** (максимальное перерегулирование выходной переменной y(t)) осуществлено в Главном схемном окне с использованием типового блока *Запоминание максимума*, выход которого являлся фактически данным критерием качества. Время переходного процесса (критерий **Q1**) формируется в субструктуре “***Измеритель времени переходного процесса***”. Ключ из типовой библиотеки ***Ключи*** (блок “***Величина “трубки***”) формирует на выходе сигнал, равный модельному времени, если абсолютная величина рассогласования больше 0,05. Если модуль рассогласования (ошибки) меньше 0,05, то на выход блока “***Величина “трубки***” поступает сигнал обратной связи. Блок с названием “***Время переходного процесса***” на структурной схеме - дискретный типовой блок *Задержка на шаг интегрирования*, выход которого фактически является 1-м критерием качества. При входе переходного процесса в “трубку” на выходе этого блока сохраняется значение времени переходного процесса (Т\_пп). Ограничения на критерии качества: 1 ≤ y\_max ≤ 1,36, 0 ≤ Т\_пп ≤ 1,8 с. Третий критерий качества (**Q3**) формируется в “***Объекте управления***” и имеет ограничение: 0 ≤ abs(dy/dt) ≤ 3 с –1.

В окне графиков пунктирной линией отображены результаты расчета переходного процесса при начальных значениях параметров, которые иллюстрируют факт неустойчивости САР при этих значениях параметров. Полученные с использованием алгоритма оптимизации Поиск - 2 при квадратичной оценки общего критерия качества, результаты оптимизационных расчетов представлены в графическом окне сплошной линией. В результате имеем: (K\_п)опт = 4,2, (K\_и)опт = 4, (K\_д)опт = 1,17; величина перерегулирования – 25 %, время переходного процесса Т\_пп = 2,09 с, а (dy/dt)max = 3,078.



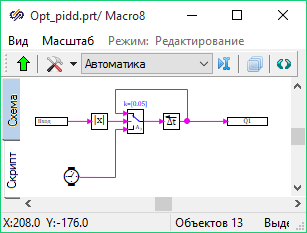
 

Рисунок 4.1 - Копии схемных окон и результатов оптимизационных расчетов

1. Численные методы и алгоритмы Анализа
   1. Постановка задачи

Частотные характеристики, наряду с переходными, широко используются при анализе и синтезе различных динамических систем. Расчет частотных характеристик может быть выполнен двумя основными способами: экспериментальным и алгебраическим. Экспериментальный расчет сводится к моделированию системы при гармоническом входном воздействии и последующему вычислению амплитуды и фазы выходного сигнала как первых коэффициентов разложения в ряд Фурье. Достоинство этого способа - возможность получать характеристики систем, содержащих сложные нелинейные элементы, недостатки - большой объем вычислительной работы и невозможность получить характеристики неустойчивых объектов. Алгебраический способ основан на формальной замене операторов дифференцирования, дискретизации и запаздывания соответствующими комплексными коэффициентами, зависящими от частоты, и решении полученной системы линейных алгебраических уравнений с комплексными коэффициентами. Этот способ требует меньших вычислительных затрат и позволяет получить характеристики неустойчивых объектов, но он применим только к линейным или предварительно линеаризованным моделям.

В зависимости о того, в каком виде формируется математическая модель, при реализации алгебраического способа может быть использован метод передаточных функций либо метод переменных состояния. Метод передаточных функций удобно использовать в тех случаях, когда система полностью непрерывная и не содержит блоков чистого запаздывания либо полностью дискретная. В более сложных случаях этот метод встречает почти непреодолимые трудности, связанные с необходимостью формирования передаточной функции всей системы по передаточным функциям составляющей ее блоков, описывающих непрерывные элементы, запаздывания разной величины, а также дискретные элементы с различными периодами дискретизации. Один из способов преодоления этих трудностей - использование дробно-рациональной аппроксимации (аппроксимации Паде) экспоненциальной функции. Однако такой подход не всегда корректен, поскольку полученная аппроксимация может оказаться недостаточно точной.

Метод переменных состояния может потребовать большей вычислительной работы при расчете характеристик непрерывных либо дискретных систем. Однако он более универсален и позволяет без труда рассчитывать частотные характеристики непрерывно-дискретных систем, содержащих блоки запаздывания и дискретные блоки с разными периодами дискретизации. Поэтому для реализации в среде SimInTech был выбран именно этот метод.

При расчете систем, содержащих дискретные элементы, необходимо иметь в виду следующее. Строго говоря, экстраполятор является нелинейным элементом. Однако если рассматривать все функции времени как решетчатые функции, определенные только в моменты дискретизации, то экстраполятор становится линейным элементом. Если при этом моменты дискретизации всех дискретных элементов совпадают, то непрерывно-дискретную систему, не содержащую нелинейных функциональных зависимостей, можно рассматривать как линейную дискретную систему (необходимо только произвести дискретизацию непрерывной части системы, что может быть сведено к расчету матричной экспоненты). Это является обоснованием применения преобразования Фурье для расчета систем, содержащих непрерывную и дискретную части. Однако такой подход неприменим в общем случае, когда система содержит дискретные элементы с различными периодами дискретизации.

Мы будем использовать другой подход, основанный на линеаризации экстраполятора. В непрерывном времени гармонически линеаризованная модель экстраполятора может быть представлена в виде передаточной функции

*W*(*s*) = (1- *e -sT* ) */* (*sT*). (5.1)

Поскольку при таком подходе передаточная функция экстраполятора не равна единице, то его последовательное включение приводит к изменению частотных характеристик. Поэтому следует четко определить, в каких случаях и в каких местах в схему должен быть включен экстраполятор. При моделировании функцию экстраполятора выполняет каждый дискретный блок, и его включение не обязательно. Включение экстраполятора с тем же периодом до или после дискретного блока никак не скажется на результатах моделирования. При частотном анализе экстраполятор с соответствующим периодом должен разделять непрерывную и дискретную части системы. При этом следует учитывать, что включение лишнего экстраполятора приведет к изменению результатов, поэтому следует ставить экстраполятор только на входе либо только на выходе дискретной части. В соответствии с этим правилом, если система полностью дискретная, то и нет разделения на дискретную и непрерывную части, поэтому включать экстраполятор в этом случае не следует.

* 1. Алгоритмы частотного анализа

При использовании метода переменных состояния расчет частотных характеристик сводится к выполнению следующих действий:

* 1. Сформировать математическую модель в виде совместной системы дифференциальных, алгебраических и разностных уравнений.
  2. Линеаризовать полученные уравнения. Будем использовать линеаризацию в окрестности заданной рабочей точки, тогда она сводится к численному либо аналитическому дифференцированию.
  3. Выполнить замену операторов дифференцирования, сдвига по времени и запаздывания на соответствующие комплексные коэффициенты усиления. В результате будет сформирована система линейных алгебраических уравнений с комплексными коэффициентами.
  4. Решив полученную систему, найти значения действительной *R*(*ω*) и мнимой *I*(*ω*) частей комплексной частотной характеристики.
  5. По значениям *R*(*ω*), *I*(*ω*) определить значения интересующих нас характеристик.

Действия 3…5 выполняются для каждого из значений частоты, на которых рассчитываются частотные характеристики.

Рассмотрим каждый из этапов расчета более подробно. Пусть нужно рассчитать характеристики от входа *u* к выходу *y*. Введем следующие обозначения:

*xdif* - вектор переменных состояния дифференциальных уравнений;

*xalg* - вектор определяющих алгебраических переменных (т.е. переменных, по которым однозначно определяются все переменные алгебраических контуров;

*xdis* - вектор дискретных переменных состояния;

*xdel* - вектор выходов блоков запаздывания;

*xext* - вектор выходов экстраполяторов.

Тогда уравнения динамической системы запишутся в виде

*xdif* ‘ = *f*1 ( *xdif* , *xalg* , *xdis* , *xdel* , *xext* , *u* ) ,

*xalg* = *f*2 ( *xdif* , *xalg* , *xdis* , *xdel* , *xext* , *u* ) ,

*xdis* = DIS(*Tdis*) *f*3 ( *xdif* , *xalg* , *xdis* , *xdel* , *xext* , *u* ) , (5.2)

*xdel* = DEL(*Tdel*) *f*4 ( *xdif* , *xalg* , *xdis* , *xdel* , *xext* , *u* ) ,

*xext* = EXT(*Text*) *f*5 ( *xdif* , *xalg* , *xdis* , *xdel* , *xext* , *u* ) ,

*y* = *f*6 ( *xdif* , *xalg* , *xdis* , *xdel* , *xext* , *u* ) .

Здесь DIS(*Tdis*) - оператор векторного покомпонентного дискретного запаздывания, *Tdis* - вектор соответствующих периодов дискретизации. Аналогично, DEL(*Tdel*), EXT(*Text*) - операторы покомпонентного запаздывания и экстраполяции; *Tdel* , *Text* - векторы соответствующих постоянных времени.

В среде SimImTech принято представление математической модели в виде структурной схемы, заданной уравнениями отдельных блоков и связями блоков между собой. В этом случае формирование уравнений в виде (5.2) обеспечивается процедурой сортировки блоков, осуществляемой в режиме моделирования.

Линеаризация полученных уравнений сводится к определению матриц частных производных

*А = ∂ f /∂ x, B = ∂ f /∂ u, C = ∂ g /∂ x, D = ∂ g /∂ u ,*

где *f* = [*f*1, *f*2, *f*3, *f*4, *f*5] T , *x* = [*xdif* , *xalg* , *xdis* , *xdel* , *xext* ] T . Расчет этих матриц будем производить численно, заменяя дифференциалы конечными приращениями. Используя операторы языка Паскаль, запишем алгоритм расчета матрицы *A* и вектор-строки *C* :

{ *x*0 - заданная рабочая точка }

*f* 0 := *f* (*x*0, *u*);

*g*0 := *g* (*x*0, *u*);

**for** *i* := 1 **to** *N* **do**

**begin**

{ relinc - относительное приращение, absinc - абсолютное приращение }

*dxi* := *relinc*\*abs(*x*[*i*]) + *absink* ;

*xsave* := *x*0[*i*];

*x*0[i] := *xsave* + *dxi* ;

*f* 1 := *f* (*x*0, *u*);

*g*1 := *g* (*x*0, *u*);

**for** *j* := 1 **to** *N* **do** *A*[*j*, *i*] := ( *f* 1[*j*] - *f* 0 [*j*] ) */ dxi* ;

*C*[*j*] := (*g*1[*j*] - *g*0[*j*] ) */ dxi* ;

*x*0[*i*] := *xsave*

**end**;

Аналогично, задавая приращение переменной *u*, определим *B* и *D*.

Далее следует произвести замену оператора дифференцирования на коэффициент *jω*, а операторов дискретного и непрерывного запаздывания - на exp(-*jωT*), учитывая при этом, что передаточная функция экстраполятора имеет вид (5.1). В результате получим систему алгебраических уравнений вида

*E x* = *A x* + *B u*,

(5.3)

*y* = *C x* + *D u*,

где *E* = diag [*Edif* , *Ealg* , *Edis*, *Edel*, *Eext*] - комплексная диагональная матрица, которая состоит из пяти диагональных матриц, ненулевые элементы которых равны

*Edif* [*i*, *i*] *= jω ,*

*Ealg* [*i*, *i*] *=* 1,

*Edis* [*i*, *i*] *=* cos (*ωTdis\_i* ) + *j* sin (*ωTdis\_i* ),

*Edel* [*i*, *i*] *=* cos (*ωTdel\_i* ) + *j* sin (*ωTdel\_i* ),

*Eext* [*i*, *i*] *= ωText\_i /* (2 tg (*ωText\_i /*2) ) + *j ωText\_i /* 2.

Принимая *u* = 1 и решая систему (5.3), найдем значения вещественной и мнимой частотных характеристик на заданной частоте ω:

*y* = *R*(*ω*) + *j I*(*ω*).

Решение алгебраической системы с комплексными коэффициентами можно свести к решению системы с вещественными коэффициентами, которая имеет вид

,

где *EI*, *ER* - действительная и мнимая части матрицы *E*. Тогда значения вещественной и мнимой характеристик определяются по формулам

*R*(*ω*) = *C* *xR* +*D*,

*I*(*ω*) = *C xI*.

Значения амплитудной *A*(*ω*), логарифмической амплитудной *L*(*ω*) и фазовой *ϕ*(*ω*) частотных характеристик можно найти в виде



Здесь arg - аргумент комплексного числа, который вычисляется по формуле



где k - целое число, задаваемое так, чтобы исключить разрывы фазовой характеристики.

* 1. Пример решения задачи анализа

Для расчета частотных характеристик прежде всего необходимо определить точки, между которыми рассчитываются эти характеристики. В среде SimInTech для этой цели используется типовой блок *Построение частотных характеристик*, на входы которого подаются сигналы входа и выхода исследуемой системы. Последовательность действий рассмотрим на конкретном примере. На рисунке 5.1 представлен вид окна со схемой, подготовленной для расчета характеристик. После набора схемы нужно установить желаемые опции частотного анализа в свойствах соответствующего блока. Соответствующее окно представлено на рисунке 5.2. После старта расчета получим графики логарифмической амплитудной и фазовой характеристик, изображенные на рисунке 5.3.

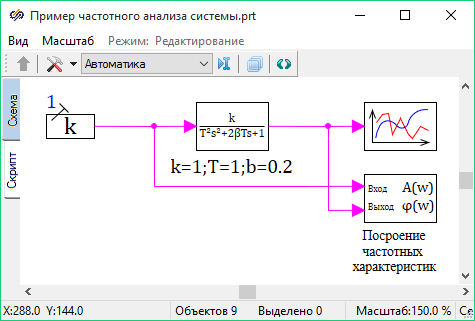


Рисунок 5.1 – Экранная копия схемного окна

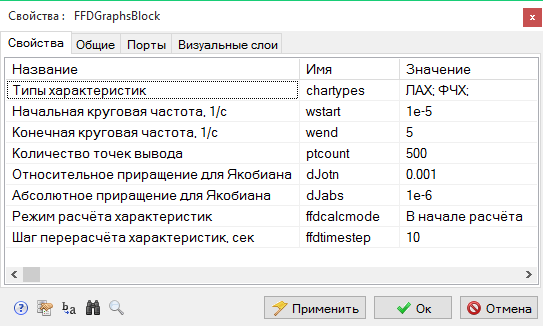


Рисунок 5.2 – Копия окна свойств блока ***Построение частотных характеристик***

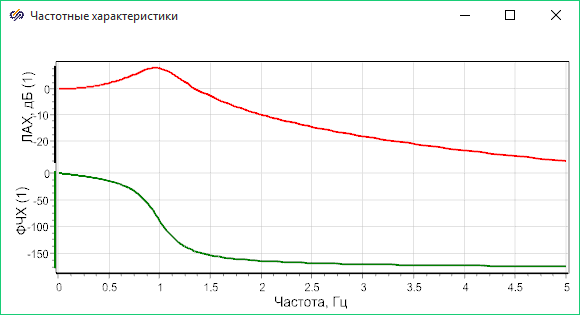


Рисунок 5.3 – Копия графического окна

Список использованных источников

1. “МВТУ” / Свидетельство РФ № 970053, зарегистрированное в Реестре программ для ЭВМ, М., 10 февраля 1997 г.
2. Отчет о научно-исследовательской работе. Представительный комплекс АСУ ТП АЭС «Бушер». Компьютерная модель динамики энергоблока АЭС «Бушер» на базе ПТК «Радуга-ЭУ». Часть 4. Модель системы аварийных защит и блокировок для системы аварийного и планового расхолаживания первого контура и охлаждения бассейна выдержки. Описание применения. Арх.№ 92/НИР, Атомэнергопроект, М, 2001 г.
3. Отчет о научно-исследовательской работе. Представительный комплекс АСУ ТП АЭС «Бушер». Компьютерная модель динамики энергоблока АЭС «Бушер» на базе ПТК «Радуга-ЭУ». Часть 5. Модель системы аварийных защит и блокировок для системы главных циркуляционных насосных агрегатов РУ. Описание применения. Арх.№ 88.5/НИР, Атомэнергопроект, М, 2001 г.
4. Отчет о научно-исследовательской работе. Представительный комплекс АСУ ТП АЭС «Бушер». Компьютерная модель динамики энергоблока АЭС «Бушер» на базе ПТК «Радуга-ЭУ». Комплекс математических моделей КЭ СУЗ. Описание применения. Руководство пользователя. Арх.№ 96/НИР, Атомэнергопроект, М, 2001 г.
5. Отчет о научно-исследовательской работе. Представительный комплекс АСУ ТП АЭС «Бушер». Компьютерная модель динамики энергоблока АЭС «Бушер» на базе ПТК «Радуга-ЭУ». Комплекс математических моделей АСУЗ-УСБИ. Описание применения. Руководство пользователя. Арх.№ 95/НИР, Атомэнергопроект, М, 2001г.
6. Отчет о научно-исследовательской работе. Представительный комплекс АСУ ТП АЭС «Бушер». Компьютерная модель динамики энергоблока АЭС «Бушер» на базе ПТК «Радуга-ЭУ». Локальная математическая модель датчиков СВРК. Описание применения. Руководство пользователя. Арх.№ 97/НИР, Атомэнергопроект, М, 2001г.
7. Отчет о научно-исследовательской работе. Представительный комплекс АСУ ТП АЭС «Бушер». Компьютерная модель динамики энергоблока АЭС «Бушер» на базе ПТК «Радуга-ЭУ». Локальные математические модели АЛОС АЗ-ПЗ, АЛОС УСБИ. Отчет о верификации. Арх.№ 107/НИР, Атомэнергопроект, М, 2001г.
8. Отчет о научно-исследовательской работе. Представительный комплекс АСУ ТП АЭС «Бушер». Компьютерная модель динамики энергоблока АЭС «Бушер» на базе ПТК «Радуга-ЭУ». Комплекс математических моделей КЭ СУЗ. Отчет о верификации. Арх.№ 109/НИР, Атомэнергопроект, М, 2001 г.
9. Отчет о научно-исследовательской работе. Представительный комплекс АСУ ТП АЭС «Бушер». Компьютерная модель динамики энергоблока АЭС «Бушер» на базе ПТК «Радуга-ЭУ». Комплекс математических моделей КЭ СУЗ. Программа верификации. Арх.№ 110/НИР, Атомэнергопроект, М, 2001 г.
10. Отчет о научно-исследовательской работе. Разработка математической модели основных технологических процессов в энергоблоке для проведения по ней расчетов, необходимых для проекта АСУ ТП и PSAR. Разработка модели турбинного отделения для АЭС «Бушер». Книга 2. Разработка модели АСУ ТП турбинного отделения энергоблока АЭС «Бушер». Арх.№ 177.2/НИР, Атомэнергопроект, М, 2002 г.
11. Отчет о научно-исследовательской работе. Проверка устойчивости основных контуров регулирования 1-го блока АЭС «Бушер». Книга 1. Методика проверки основных контуров регулирования энергоблока АЭС «Бушер»Арх.№ 178.1/НИР, Атомэнергопроект, М, 2002 г.
12. Отчет о научно-исследовательской работе. Проверка устойчивости основных контуров регулирования 1-го блока АЭС «Бушер». Книга 2. Проверка работы контуров регулирования системы ТА и системы YP энергоблока АЭС «Бушер». Арх.№ 178.2/НИР, Атомэнергопроект, М, 2002 г.
13. Отчет о научно-исследовательской работе. Проверка устойчивости основных контуров регулирования 1-го блока АЭС «Бушер». Проверка работы контуров регулирования системы ТН энергоблока АЭС «Бушер». Арх.№ 180/НИР, Атомэнергопроект, М, 2002 г.
14. Отчет о научно-исследовательской работе. Проверка устойчивости основных контуров регулирования 1-го блока АЭС «Бушер». Проверка работы автоматического регулятора мощности реактора Арх.№ 191/НИР, Атомэнергопроект, М, 2002 г.
15. Отчет о научно-исследовательской работе. Проверка устойчивости основных контуров регулирования 1-го блока АЭС «Бушер». Проверка работы БРУ-К энергоблока АЭС «Бушер». Арх.№ 189/НИР, Атомэнергопроект, М, 2002 г.
16. Отчет о научно-исследовательской работе. Проверка устойчивости основных контуров регулирования 1-го блока АЭС «Бушер». Проверка работы контуров регулирования системы питания парогенераторов энергоблока АЭС «Бушер». Арх.№ 190/НИР, Атомэнергопроект, М, 2002 г.
17. Калининская АЭС. Блок 3. Автоматизированная система управления технологическими процессами. Анализ устойчивости контуров регулирования. Арх.№ 373 – 156 с., М, 2003 г.
18. Автоматизированное проектирование систем управления / Под ред. М.Джамшиди; Пер. с англ. −М.: Машиностроение, 1989. – 344 с.
19. Бусленко Н.П. Моделирование сложных систем. –М.: Наука, 1978. – 355 с.
20. Дорри М.Х. Некоторые тенденции развития автоматизированных систем управления технологическими процессами и их влияние на системы управления АЭС // Вопросы атомной науки и техники. –М., 1996.–C. 3-17. (Сер. Физика ядерных реакторов; Вып. 3).
21. Крутько П.Д., Максимов А.И., Скворцов Л.М. Алгоритмы и программы проектирования автоматических систем / Под ред. П.Д.Крутько. –М.: Радио и связь, 1988. –306 c.
22. Flannery B.P., Teukolsky S.A., Vetterling W.T. Numerical Recipes. – Cambridge: Cambridge University Press, 1986. – 302 p.
23. Форсайт G.V. Машинные методы математических вычислений. –М.: Мир, 1980. –194 c.
24. Скворцов Л. М. Явные адаптивные методы численного решения жестких систем // Математическое моделирование. -2000. -Т. 12, № 12. -С. 97‑107.
25. Скворцов Л. М. Диагонально неявные FSAL-методы Рунге-Кутты для жестких и дифференциально-алгебраических систем // Математическое моделирование. -2002. -Т. 14, № 2. -С. 3‑17.
26. Чуа Л.О., Лин Пен-Мин Машинный анализ электронных схем: Алгоритмы и вычислительные методы. –М.: Энергия, 1980. –640 с.
27. Тьюарсон Р. Разреженные матрицы. –М.: Мир, 1977. –189 с.
28. Брамеллер А., Аллан Р., Хэмэм Я. Слабозаполненные матрицы: Анализ электроэнергетических систем: Пер. с англ. –М.: Энергия, 1979. –192 с.
29. Эстербю О., Златев З. Прямые методы для разреженных матриц: Пер. с англ. –М.: Мир, 1987. –120с.
30. Батищев Д.И. Методы оптимального проектирования. –М.: Радио и связь, 1984. –248 с.
31. Химмельблау Д. Прикладное нелинейное программирование. –М.: Мир, 1975. –534 c.
32. Проектирование следящих систем. Физические и методические основы / Под ред. Н.А. Лакоты. -М.: Машиностроение, 1992. -352с.
33. *Скворцов Л.М.* Интерполяционный частотный метод синтеза систем управления // Изв. РАН. Техн. кибернетика. -1994. -№ 6. -С. 16-24.
34. *Скворцов Л.М.* Интерполяционные методы синтеза систем управления // Проблемы управления и информатики. -1998. -№ 6. -С. 25-30.
35. *Скворцов Л.М.* Интерполяционный метод автоматической настройки регуляторов // Изв. РАН. Теория и системы управления. -1998. -№ 6. -С.100-103.