Министерство науки и высшего образования
Российской федерации
ФГБОУ ВО «Российский химико-технологический университет
имени Д.И. Менделеева»

Новомосковский институт (филиал)

Автоматизированные системы управления химико-технологическими процессами и производствами

Утверждено Редакционно–издательским советом
института в качестве учебного пособия

Новомосковск 2021

УДК 681.5
ББК 32.965
Т 338

Рецензенты:
кандидат технических наук, доцент Соболев А.В.
НИ (филиал) ФГБОУ ВО РХТУ им. Д.И. Менделеева

кандидат технических наук, старший мастер ЦЦР ТО КИПиА Лопатин К.Г.
АО МХК Еврохим

Составители: Предместьин В.Р., Лопатин А.Г., Брыков Б.А.
Т 338 «Автоматизированные системы управления химико-технологическими процессами и производствами» / ФГБОУ ВО Российский химико-технологический университет им. Д.И. Менделеева, Новомосковский институт (филиал), Сост.: Предместьин В.Р., Лопатин А.Г., Брыков Б.А. 2021. - 110 с.

В пособии приведен теоретический базис для освоения курса «Автоматизированные системы управления химико-технологическими процессами и производствами» Изложены методические указания для выполнения лабораторных работ с применением специальной установки. Пособие предназначено для студентов профиля 15.03.04 «Автоматизация технологических процессов и производств» при изучении дисциплины «Автоматизированные системы управления химико-технологическими процессами и производствами», а также будет интересно инженерам и специалистам в сфере управления процессами.

Табл. 5 Ил. 26. Библиогр.: 3 назв.

УДК 681.5

Содержание

Введение

Целью изучения дисциплины «Автоматизированные системы управления химико-технологическими процессами и производствами» является обеспечение базовой подготовки студентов в области разработки, моделирования и синтеза автоматизированных систем управления химико-технологическими процессами и производствами (АСУТП).

Задачи преподавания дисциплины:

- изучение структуры современной АСУТП процессами и разновидностей автоматизированных систем управления (АСУ) в зависимости от решаемых ею задач;

-идентификация технологического процесса с использованием различных видов математических моделей;

- приобретение знаний о классификации объектов и систем автоматического управления;

- формирование и развитие умений описывать происходящие в системах динамические процессы;

- приобретение и формирование навыков проведения синтеза автоматизированных систем управления, их испытания и эксплуатацию;

В этом пособии представлены основные сведения по вопросам разработки, моделирования и синтеза АСУТП, а также предлагается цикл лабораторных работ для закрепления полученных знаний и получения необходимых практических навыков.

1 Основные понятия и определения курса

В современной технике используется огромное число автоматических устройств и систем, предназначенных, однако, для решения лишь нескольких основных задач автоматизации: сигнализации, контроля, блокировки и защиты, пуска и останова, управления. В соответствии с этими задачами подразделяются и системы автоматизации.

Системы автоматической сигнализации предназначены для извещения обслуживающего персонала о состоянии технологической установки или протекающего в ней технологического процесса.

Системы автоматического контроля осуществляют без участия человека контроль (т.е. измерение и сравнение с нормативными показателями) различных величин, характеризующих работу технологического агрегата или протекающий в нем технологический процесс.

В промышленном производстве часто используют системы централизованного контроля, в которых вся технологическая информация собирается и обрабатывается на центральном пульте управления.

Системы автоматической блокировки и защиты служат для предотвращения возникновения аварийных ситуаций в агрегатах и устройствах.

Системы автоматического пуска и останова обеспечивают включение, переключение и отключение различных приводов и механизмов агрегата или технологической установки по заранее заданной программе.

Системы автоматического управления предназначены для управления работой тех или иных технических устройств и агрегатов или протекающими в них технологическими процессами.

Важнейшими и наиболее сложными из перечисленных систем являются системы автоматического управления.

Управлением в широком смысле слова называется организация какого-либо процесса, обеспечивающего достижение поставленной цели.

Основной задачей любого процесса управления является выработка и реализация таких решений, которые при данных условиях обеспечивают наиболее эффективное достижение цели управления.

Целями управления технологическими процессами и агрегатами могут быть:

- поддержание постоянного значения некоторой физической величины с заданной точностью;
- изменение величины по определенной, заранее заданной программе;
- получение оптимального значения величины или некоторого обобщающего комплекса величин (максимальная производительность

агрегата, минимальная стоимость продукта, минимальное время перехода объекта из одного состояния в другое) и т.д.

Если управление осуществляется непосредственно человеком, то такое управление называют **ручным**. Если же управление осуществляется без непосредственного участия человека, то такое управление называют **автоматическим**. Автоматическое управление производится с помощью автоматически действующих управляющих устройств. Объект управления (ОУ) и управляющее устройство составляют систему автоматического управления (САУ).

При наиболее простых целях управления (поддержание постоянного значения величины, изменение величины по заданной программе и др.) процесс управления называют регулированием. Объекты управления - объектами регулирования, управляющие устройства - автоматическими регуляторами, а системы автоматического управления - системами автоматического регулирования (САР).

Автоматическое регулирование – это одна из важнейших функций автоматического управления, без осуществления которой невозможна работа большинства систем управления. В сложных системах управления, особенно с использованием ЭВМ, управлением называют процесс выработки необходимого решения, а регулированием - его реализацию на объекте.

2 Автоматизированные системы управления технологическими процессами

Со второй половины шестидесятых годов в связи с развитием ЭВМ и появлением достаточно дешёвых, надёжных и быстродействующих ЭВМ в мире появились первые автоматизированные системы управления (АСУ).

Это особенно стало необходимым в связи с появлением и развитием высокопроизводительных агрегатов большой единичной мощности и быстродействующих технологических процессов. В металлургии были созданы 350-ти тонные кислородные конвертеры, прокатные станы производительностью более 5 млн. тонн проката в год и др., поэтому существенно возросли требования к качеству продукции и экономичности производства.

АСУ построены на базе управляющих вычислительных комплексов (УВК), представляющих собой специализированную промышленную ЭВМ, предназначенную для вычислений и реализации функций автоматизированных систем управления. Именно разнообразие этих функций позволило поднять автоматизацию на качественно новый уровень. Автоматизированные системы управления развиваются в двух основных направлениях: автоматизированные системы управления технологическими процессами (АСУТП) и автоматизированные системы управления производственными процессами (АСУП).

До АСУТП имели место локальные САР, в которых за функционирование отдельно взятого контура регулирования определённого технологического параметра отвечал свой автоматический регулятор. Согласованная работа контуров, число которых в технологическом процессе может быть большим, проводилась оперативным персоналом.

В АСУТП насчитываются десятки – тысячи отдельных локальных контуров регулирования, согласование которых также проводит оперативный персонал, но при использовании управляющего вычислительного комплекса. Таким образом, локальные САР входят в АСУТП, как составная часть.

Автоматизированные системы управления производственными процессами выполняет функции: маркетинга, календарного планирования, поставок сырья, сбыта готовой продукции, финансирования и т.д.

Объектом управления для АСУП является трудовой процесс непосредственного производства товарной продукции и вся административно-хозяйственная деятельность предприятия, неизбежно сопровождающая основной процесс производства продукции.

В настоящее время созданы принципиально новые системы управления – интеллектуальные АСУ, использующие принципы и методы искусственного интеллекта.

Развитие промышленного производства включает в себя три основные составляющие:

- наука;
- проектирование;
- производство (внедрение).

Автоматизация используется не только в промышленном производстве в виде АСУТП и АСУП. В науке создаются автоматизированные системы научных исследований (АСНИ), которые позволяют на порядок увеличить производительность труда ученых.

В промышленности созданы системы автоматизированного проектирования (САПР), которые позволяют увеличить скорость проектирования, значительно уменьшая число ошибок в проекте.

Технический прогресс, осуществляемый на основе автоматизации, включает в себя три основные составляющие: АСНИ – САПР - АСУТП, что позволяет значительно повысить эффективность, как научных разработок, так и конечных производственных результатов.

Эффективное управление производственным предприятием – сложный процесс, требующий системного подхода и комплексных решений.

В современном производственном предприятии большее значение приобретает возможность оперативного доступа к достоверной и точной информации из любой точки управления производством, поскольку это определяющим образом влияет на эффективность работы предприятия, включая производительность труда, качество и конкурентоспособность выпускаемой продукции.

В рамках производственного предприятия можно выделить следующие уровни управления:

- стратегический
- тактический
- оперативный
- технологический

Каждому из представленных уровней управления свойственны специфические задачи:

- стратегического планирования
- управления ресурсами
- управления производственными процессами
- управления технологическими процессами

С целью повышения эффективности процессов управления многие современные производственные предприятия внедряют различные автоматизированные системы.

Задача управления химико-технологическим процессом решается путем создания интегрированной многоуровневой распределенной автоматизированной системы управления (РАСУ). Интегрированная система автоматизации предприятия может быть представлена в виде 5-уровневой пирамиды (рисунок 2.1):

Рисунок 2.1 – Уровни интегрированной системы автоматизации производства

Первый уровень, уровень ввода-вывода **(*Input/Output*)**, включает набор датчиков и исполнительных устройств, встраиваемых в конструктивные узлы технологического оборудования и предназначенных для сбора первичной информации и реализации управляющих воздействий.

Современные интеллектуальные датчики помимо процесса измерения, преобразования измеряемых сигналов в унифицированные аналоговые или цифровые значения, выполняют также самодиагностику своей работы, дистанционную настройку диапазона измерения, первичную обработку измерительной информации, иногда еще ряд достаточно простых, типовых алгоритмов контроля и управления.

Они имеют интерфейсы к стандартным/типовым полевым цифровым сетям, что делает их совместимыми с практически любыми современными средствами автоматизации, и позволяет информационно общаться с этими средствами и получать питание от блоков питания этих средств.

Второй уровень (*Control*), непосредственное управление, служит для непосредственного автоматического управления технологическими процессами с помощью с помощью различных устройств связи с объектом (УСО) и программируемых логических контроллеров (ПЛК), и характеризуется следующими показателями:

1. предельно высокой реактивностью режимов реального времени;

2. предельной надежностью (на уровне надежности основного оборудования);

3. возможностью встраивания в основное оборудование;

4. функциональной полнотой модулей УСО;

5. возможностью автономной работы при отказах комплексов управления верхних уровней;

6. возможностью функционирования в цеховых условиях.

В промышленные контроллеры загружаются программы и данные из ЭВМ третьего уровня, уставки, обеспечивающие координацию и управление агрегатом по критериям оптимальности управления технологическим процессом в целом, выполняется вывод на третий уровень управления служебной, диагностической и оперативной информации, т.е. данных о состоянии агрегата, технологического процесса.

Устройство связи с объектом – устройство для объединения аналоговых и цифровых параметров реального технологического объекта. Предназначено для ввода сигналов с объекта управления в ПЛК и вывода управляющих воздействий на объект управления.

Третий уровень, (SCADA - Supervisory Control and Data Acquisition - сбор данных и диспетчерское управление), предназначен для отображения (или визуализации) данных в производственном процессе и оперативного комплексного управления различными агрегатами, в том числе и с участием диспетчерского персонала.

Этот уровень управления должен обеспечивать:

1. диспетчерское наблюдение за технологическим процессом по его графическому отображению на экране в реальном масштабе времени;

2. выбор законов управления, уставок расчет настроек регуляторов, соответствующих заданным показателям качества управления и текущим (или прогнозным) параметрам объекта управления;

3. оперативное сопровождение моделей объектов управления типа «агрегат», «технологический процесс», корректировку моделей по результатам обработки информации от второго уровня;

4. синхронизацию и устойчивую работу систем типа «агрегат» для группового управления технологическим оборудованием;

5. ведение единой базы данных технологического процесса;

6. связь с четвертым уровнем.

Отвечая этим требованиям, ЭВМ на третьем уровне управления должны иметь достаточно высокую производительность как при решении задач в реальном масштабе времени, так и при обработке графической информации, обеспечивая

работу в реальном времени с базами данных среднего объема и с расширенным набором интеллектуальных видеотерминалов.

Третий уровень управления реализуется на базе специализированных промышленных компьютеров, или в ряде случаев на базе персонального компьютера. Диспетчерский интерфейс реализуется SCADA-системами.

Машины третьего уровня должны объединяться в однородную локальную сеть предприятия (типа Ethernet) с выходом на четвертый уровень управления.

Четвертый уровень MES (Manufacturing Execution System – производственная исполнительная система управления процессами на уровне цеха, реализующая оперативное планирование и диспетчеризацию производства), уровень управления производством характеризуется необходимостью решения задач оперативной упорядоченной обработки первичной информации из цеха и передачи этой информации на верхний уровень планирования ресурсов предприятия. Решение этих задач на данном уровне управления обеспечивает оптимизацию управления ресурсами цеха как единого организационно-технологического объекта по заданиям, поступающим с верхнего уровня, и при оперативном учете текущих параметров, определяющих состояние объекта управления. Решение этих задач возлагается обычно на серверы в локальных сетях предприятия.

Пятый уровень, ***MRP (Manufacturing Resource Planning – Планирование производственных ресурсов)*** и ***ERP (Enterprise Resource Planning – Планирование ресурсов предприятия*** **–** интегрированный набор бизнес-приложений, инструменты которого предназначены для автоматизации сквозных процессов, например, в области финансов, управления персоналом, распределения, производства, обслуживания и цепочки поставок.

Задачи, решаемые на этом уровне, в аспекте требований, предъявляемых к ЭВМ, отличаются главным образом повышенными требованиями к ресурсам (например, для ведения единой интегрированной - централизованной или распределенной, однородной или неоднородной - базы данных, планирования и диспетчирования на уровне предприятия в целом, автоматизации обработки информации в основных и вспомогательных административно-хозяйственных подразделениях предприятия: бухгалтерский учет, материально-техническое снабжение и т.п.). Обычно для решения задач данного уровня выбирают универсальные ЭВМ, а также многопроцессорные системы повышенной производительности.

Основная цель управления производством: своевременное выполнение производственной программы. Для достижения указанной цели требуется:

1. учитывать потребность в производстве с учетом имеющихся заказов и запасов готовой продукции;

2. обеспечить достаточный уровень ресурсов (материальные запасы, производственные мощности, персонал);

3. обеспечить слаженную работу производственных подразделений на межцеховом уровне;

4. организовать эффективное управление процессами на уровне производственного цеха.

Ключевыми целями систем управления производством являются:

1. высокое качество обслуживания клиентов:

- быстрое определение возможного срока изготовления продукции по запросу клиента;
- своевременное выполнение обязательств перед клиентом по срокам и ассортименту;
- мониторинг хода исполнения заказов;

2. гибкая система оперативного управления:

- управление приоритетами выполнения заказов;
- формирование согласованного по доступным мощностям и ресурсам графика производства;
- оперативная реакция на отклонения в выполнении графика и изменение заказов, включая перепланирование;

3. эффективное использование производственных ресурсов и снижение себестоимости:

- исключение работ, не востребованных внешним и внутренним спросом;
- контроль выполнения нормативов и использования замен, аналогов;
- мотивация персонала.

Эту модель комплексной автоматизации предприятия можно упрощать, объединяя любые соседние уровни.

Исторически сложилось так, что верхний уровень, то есть АСУП и нижние уровни, т.е. АСУТП развивались независимо друг от друга и фактически отсутствовал достаточно интеллектуальный интерфейс, который бы их объединял. Это обстоятельство на современном уровне развития промышленности стало тормозящим фактором. Для эффективной работы производственного предприятия и для принятия на верхнем уровне как стратегических, так и тактических решений требуется интеграция всех систем управления производством.

Возможности систем управления производством во многом определяются составом и функциями комплекса инструментальных программных средств, предназначенного для построения АСУТП и для интеграции их как с системами управления производством верхнего уровня, так и со средствами управления нижнего уровня (датчики, исполнительные устройства и др.). Использование

такого инструментария обеспечивает возможность создания интегрированных сквозных систем управления производством в реальном масштабе времени.

Важной причиной появления на рынке инструментальных систем для решения задач комплексной автоматизации является низкая эффективность традиционного и необходимость структурированного подхода к построению интегрированных систем управления производством.

Недостатки традиционного построения АСУТП:

1. множество интерфейсов, сложность и запутанность связей между объектами;
2. несовместимость форматов данных и структуры сообщений;
3. сложность внесения изменений, что может вызвать переработку большого объема программ.

Преимущества структурированного подхода:

1. нормализация данных;
2. стандартные формы сообщений;
3. гибкие средства интеграции приложений, включая АСУП.

Такой модульный структурированный подход к построению АСУТП обеспечивает возможность эффективной модернизации системы, облегчает внесение в нее изменений, что в совокупности гарантирует защиту ранее вложенных инвестиций и уменьшение стоимости управления.

3 Элементная база систем автоматического регулирования и управления

Элементы автоматики чрезвычайно разнообразны по выполняемым функциям, конструкции, принципу действия, характеристикам, физической природе преобразуемых сигналов и т.д.

1) В зависимости от того, как элементы получают энергию, необходимую для преобразования входных сигналов, они делятся на пассивные и активные.

Пассивные элементы автоматики – это элементы, у которых входное воздействие (сигнал $x_{вх}$) преобразуется в выходное воздействие (сигнал $x_{вых}$) за счёт энергии входного сигнала (например, редуктор).

Активные элементы автоматики для преобразования входного сигнала используют энергию от вспомогательного источника (например, двигатель, усилитель).

2) В зависимости от энергии на входе и выходе элементы автоматики подразделяются на:

- электрические;
- гидравлические;
- пневматические;
- механические;
- комбинированные.

3) По выполняемым функциям в системах регулирования и управления элементы автоматики подразделяются на:

- датчики;
- усилители;
- исполнительные устройства;
- реле;
- вычислительные элементы;
- согласующие элементы;
- вспомогательные элементы и т.д.

Датчики воспринимают поступающую на их вход информацию об управляемой величине объекта управления и преобразуют её в форму, удобную для дальнейшего использования в устройстве автоматического управления.

Большинство датчиков преобразует входной неэлектрический сигнал $x_{вх}$ в выходной электрический сигнал $x_{вых}$. В зависимости от вида входного неэлектрического сигнала $x_{вх}$ выделяют:

- датчики механических величин (датчики перемещения, датчики скорости, датчики ускорения и т.д.);
- датчики тепловых величин (датчики температуры);
- датчики оптических величин (датчики излучения) и т.д.

Часто применяются датчики с двойным преобразованием сигнала, например, входной неэлектрический сигнал $x_{вх}$ сначала преобразуется в перемещение, а затем перемещение преобразуется в выходной электрический сигнал $x_{вых}$.

Так, например, термопара преобразует температуру θ в термоЭДС e, затем электрическая цепь преобразует термоЭДС e в силу тока I, магнитоэлектрический измерительный механизм преобразует силу тока I во вращающий момент M, далее упругая подвижная система преобразует момент M в угловое перемещение φ, являющееся выходным сигналом прибора.

Усилители - это элементы автоматики, которые осуществляют количественное преобразование, усиление мощности входного сигнала $x_{вх.}$ В некоторых случаях одновременно с количественным преобразованием, усилители осуществляют и качественное преобразование (например, преобразование постоянного тока в переменный, в пневматических и гидравлических усилителях осуществляется преобразование перемещения в изменение давления).

В зависимости от вида энергии, получаемой усилителем, последние делятся на:

- электрические;
- гидравлические;
- пневматические;
- электрогидравлические;
- электропневматические.

Наибольшее распространение получили электрические усилители, имеющие высокую чувствительность, большой коэффициент усиления и удобные в эксплуатации.

Исполнительные устройства относятся к элементам автоматики, создающим управляющие воздействия на объект управления. Они изменяют состояние или положение регулирующего органа объекта таким образом, чтобы регулируемый параметр соответствовал заданному значению. К исполнительным устройствам, создающим управляющее воздействие в виде силы или вращающего момента, относятся силовые электромагниты, электромагнитные муфты, двигатели.

Двигатели в зависимости от вида применяемой для работы энергии могут быть:

- электрическими;
- гидравлическими;
- пневматическими.

В качестве исполнительных устройств, изменяющих состояние регулирующего органа, могут использоваться усилители или реле.

Реле – это элементы автоматики, у которых изменение выходного сигнала ($x_{вых}$) происходит дискретно (т.е. скачкообразно) при достижении входным сигналом ($x_{вх}$) определённого значения, вызывающего срабатывание реле.

Это значение входного сигнала называется уровнем срабатывания реле.

Мощность входного сигнала ($x_{вх}$), вызывающего срабатывание реле, значительно меньше мощности, которой реле может управлять. Поэтому реле используется и как усилительный, и как исполнительный элемент.

Реле часто используются и как автоматически управляемые коммутаторы сигналов в многоканальных системах сбора и передачи данных, в которых обрабатывается информация от десятков, сотен и даже тысяч датчиков. Они применяются также в системах контроля, сигнализации, блокировки и защиты.

Вычислительные элементы в устройствах автоматического управления осуществляют математические преобразования с поступающими на их вход сигналами. Эти операции осуществляются с целью обеспечения заданного алгоритма работы системы.

В простейшем случае вычислительные элементы выполняют отдельные математические операции, такие как алгебраическое суммирование, дифференцирование, интегрирование, логическое сложение, логическое умножение и т.д.

В замкнутых САУ необходимо осуществлять суммирование сигнала датчика и сигнала обратной связи. В корректирующих устройствах используется дифференцирование и интегрирование сигналов. Для выполнения этих операций главным образом используются вычислительные элементы аналогового типа.

В более сложных случаях в качестве вычислительного элемента может использоваться микропроцессор, специализированные и унифицированные ЭВМ цифрового и аналогового типов или комплекс этих машин. Такие задачи автоматического управления, как оптимизация, создание адаптивных (приспосабливающихся) САУ, использование алгоритмов управления, основанных на вероятностных и статистических методах обработки сигналов, невозможно осуществить без применения ЭВМ.

Согласующие и вспомогательные элементы включаются в устройство автоматического управления для улучшения его параметров, расширения функциональных возможностей основных элементов и т.д.

В качестве согласующих элементов часто используют трансформаторы, редукторы, позволяющие согласовать параметры исполнительного элемента с параметрами объекта управления.

В системах автоматического управления, в которых качестве вычислительного элемента используется микропроцессор или ЭВМ, часто возникает необходимость согласования ЭВМ с датчиками информации и исполнительными элементами аналогового типа, широко применяемыми в автоматике. Для этой цели на входе ЭВМ устанавливаются аналого-цифровые преобразователи (АЦП). Аналого-цифровые преобразователи преобразуют механический сигнал (перемещения, скорости и т.д.) или электрический сигнал (напряжения, силы тока, сопротивления

и т.д.), получаемый от аналоговых датчиков, в дискретный кодовый сигнал, способный восприниматься ЭВМ.

Управляющее воздействие в таких системах получают в дискретной форме как результат обработки в ЭВМ поступившей информации.

Если в устройстве автоматического управления в качестве исполнительного элемента используются электродвигатели постоянного или переменного тока, электромагнитные муфты, усилители мощности постоянного или переменного тока и т.д., то возникает потребность обратного преобразования дискретного сигнала ЭВМ в аналоговый сигнал, воспринимаемый исполнительным элементом.

Эта задача решается с помощью цифро-аналоговых преобразователей (ЦАП).

Они преобразуют кодовый сигнал, полученный от ЭВМ, в перемещение, напряжение, ток, частоту и т.д.

Вспомогательные элементы автоматики – это стабилизаторы напряжения или тока, коммутаторы и распределители, генераторы напряжения специальной формы («пила»), формирователи импульсов, индикаторные и регистрирующие приборы, сигнальные и защитные устройства.

Эти элементы автоматики, не являясь принципиально необходимыми для работы устройства автоматического управления, в то же время позволяют увеличить точность и стабильность его работы, облегчают наладку и эксплуатацию, расширяют возможности использования этого устройства при создании САУ.

4 Функциональные схемы автоматизации

Функциональные схемы автоматизации (ФСА) являются основными чертежами, определяющими построение системы автоматического управления технологической установкой. Системы автоматизации на этих схемах представляют в виде блоков автоматического контроля управления и регулирования, дающих полное представление об оснащении объекта приборами и средствами автоматизации, включая средства телемеханики и вычислительной техники.

На схеме автоматизации упрощенно изображают технологический агрегат и располагают приборы и средства автоматизации в условных изображениях с указанием связей между ними.

Основные условные изображения приборов и средств автоматизации (ГОСТ 21.208–2013), приведены в таблице 4.1. Для обозначения измеряемых и регулируемых величин и функциональных признаков приборов приняты прописные буквы латинского алфавита – таблица 4.2.

Таблица 4.1 – Условные обозначения приборов и средств автоматизации

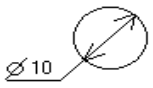
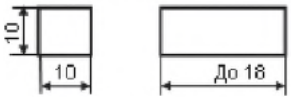
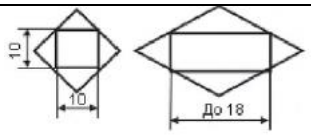
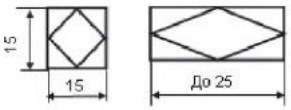
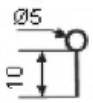
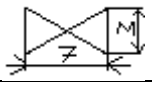
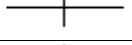
Наименование	**Обозначение**
Первичный измерительный преобразователь (датчик), прибор (контролирующий, регулирующий) – базовое обозначение:	
Первичный измерительный преобразователь (датчик), прибор (контролирующий, регулирующий) – допускаемое обозначение:	
Прибор (устройство), входящее в контур ПАЗ а) основное обозначение:	
б) допускаемое обозначение:	
Исполнительный механизм:	
Регулирующий орган:	
Линия связи:	
Пересечение линий связи без соединения друг с другом	
Пересечение линий связи с соединением между собой	

Таблица 4.2 – Буквенные условные обозначения приборов и средств автоматизации

Обозначение латинской буквы	**Измеряемая величина**		**Функциональный признак прибора**		
	Основное значение первой буквы	Дополнительное значение, уточняющее значение первой буквы	Отображение информации	Формирование выходного сигнала	Дополн. значение
1	2	3	4	5	6
A	Анализ Величина, характериз. качество: состав, концентрация, детектор дыма	—	Сигнализац.	—	—
B	Пламя, горение	—	—	—	—
C	По выбору пользователя	—	—	Регулирование управление	—
D	По выбору пользователя	Разность, Перепад	—	—	Величина отклонен. от зад. измер. величины
E	Напряжение	—	—	Чувств. элемент	-
F	Расход	Соотношение, доля, дробь	—	—	—
G	По выбору пользователя	—	Первичный показывающ. прибор	—	—
H	Ручное воздействие	—	—	—	Верхний предел изм.вел.
I	Ток	—	Вторичный показыв. прибор	—	—
J	Мощность	Автоматическое переключение, обегание	—	—	—
K	Время, временная программа	—	—	Станция Управл.	-
L	Уровень	—	—	—	Нижний предел Изм. велич.
M	По выбору пользователя	—	—	—	Величина или сред. положен. между H и L

Продолжение таблицы 4.2

1	2	3	4	5	6
N, O	По выбору пользователя	—	—	—	—
P	Давление, вакуум	—	—	—	—
Q	Количество	Интегрирование, суммирование по времени	—	—	—
R	Радиоактивность	—	Регистр.	—	—
S	Скорость, частота	Самосрабатыв. устройство безопасности	—	Включение, переключ., отключен., блокировка	—
T	Температура	—	—	Преобраз.	-
U	Несколько разнородных измеряемых величин	—	—	—	—
V	Вибрация	—	—	—	—
W	Вес, сила, масса	—	-	—	—
X	Нерекомендуемая резервная буква	-	Вспомог. компьют. устройст.	-	-
Y	Событие, состояние	-	-	Вспомог. вычислит. устройст.	—
Z	Размер, положение, перемещение	Система инструментал. безопасности, ПАЗ			

В верхней части окружности, обозначающей прибор, проставляют буквенное обозначение измеряемой величины и функционального признака прибора, в нижней – позиционное обозначение, служащее для нумерации.

Порядок расположения буквенных обозначений следующий:

1) обозначение основной измеряемой величины;

2) обозначение, уточняющее (если это необходимо), основную измеряемую величину;

3) обозначение функционального признака прибора: если их несколько, то порядок обозначений следующий: IRCSA.

4) В нижней части окружности показывают позиционное обозначение, состоящее из арабской цифры и русской буквы.

Порядок нумерации, обозначающей позицию и контур:

1) 100-199 – температура;
2) 200-299 – давление;
3) 300-399 – расход;
4) 400-499 – уровень.

5 Классификация математических моделей объектов управления

Математические модели объектов управления технологических процессов являются частью математического обеспечения АСУТП и представляют собой описание объекта на формальном математическом языке (алгебраические, дифференциальные, интегральные уравнения с соответствующими ограничениями - начальными и граничными условиями), позволяющие выносить количественное суждение о параметрах процесса. При автоматизации технологических процессов математические модели дают возможность рассчитывать изменение выходных величин объекта при различных входных воздействиях, а также соответствующие управляющие воздействия.

Математическая модель сложного технологического процесса даёт упрощенное, приближённое описание этого процесса, однако, при использовании современных ЭВМ математическую модель можно усложнить практически до любого уровня точности. Вместе с тем, потребная точность и, следовательно, сложность модели зависят от её назначения, и при разработке модели следует исходить из поставленных задач с тем, чтобы не усложнять модели тогда, когда это не требуется.

Примерная классификация математических моделей (рисунок 5.1) позволяет выяснить их основные особенности.

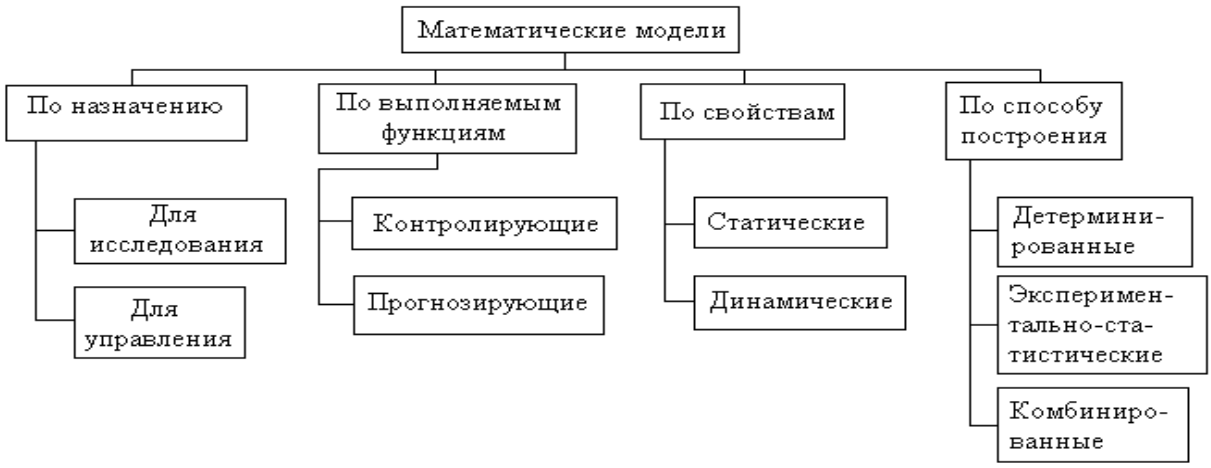


Рисунок 5.1 – Классификация математических моделей

Так, по назначению математические модели могут разрабатываться **для технологических исследований и для управления.** Первые должны давать возможность исследовать технологический процесс без экспериментов с целью его совершенствования, а в некоторых случаях разрабатывать новые технологические процессы.

Эти модели могут быть весьма сложными, поскольку они, с одной стороны, должны давать наиболее точное описание процесса, позволяющее исследовать "тонкости" технологии, а с другой стороны, они не ограничены временем расчёта на ЭВМ (ограничения могут быть только о точки зрения стоимости и расчётов).

Вторые должны давать информацию для управления или рассчитывать управляющие воздействия и во многих случаях могут быть проще моделей для исследования, тем более сложность модели для управления ограничена временем расчёта по ним, поскольку они должны рассчитывать управление быстрее или в темпе с процессом.

Математические модели могут быть предназначены для расчёта тех выходных величин процесса, которые невозможно определить непосредственными измерениями на объекте (отсутствуют датчики), или для расчёта некоторых комплексных величин, лучше характеризующих процесс, чем измеряемые величины. Такие модели называются **контролирующими.**

Прогнозирующие модели позволяют рассчитывать изменение выходных величин или их значения в какой-то будущий момент времени, т.е. прогнозировать ход процесса. Эти модели преимущественно используются при оптимальном управлении, поскольку предварительный расчёт и последующая реализация оптимальных управляющих воздействий возможны только тогда, когда мы можем прогнозировать ход процесса при этих воздействиях еще до получения окончательного результата.

По свойствам во времени модели подразделяются на статические и динамические. **Статические** модели позволяют рассчитывать параметры процесса без учёта времени. Это могут быть модели для расчёта некоторых комплексных параметров на, основе измерения ряда величин в данный момент времени. Например, расчёт теплоусвоения ванны мартеновской печи по так называемым мгновенным обратным тепловым балансам.

К статическим относятся также модели для расчета (прогнозирования) конечных значений управляемых величин (например, конечного содержания углерода в металле в конвертере) без привязки их по времени и расчёта интегральных (не распределенных во времени) управляющих воздействии (например, расчет общего количества кислорода на плавку в конвертере). Статические модели состоят ив алгебраических уравнений; очень часто таковыми являются уравнения балансов энергии или вещества.

Динамические модели дают возможность рассчитывать значение выходных величин и управляющих воздействий во времени и отроятся на основе дифференциальных уравнений, хотя обычно в них присутствуют и алгебраические соотношения.

Модели, построенные на основе теоретических представлений о процессе и использующие физические и химические закономерности, называют **детерминированными** (теоретическими). Строго детерминированные модели можно создать только для довольно простых процессов. Обычно в таких моделях присутствует некоторое число экспериментальных соотношений, но за моделями

сохраняют название детерминированных, если роль экспериментальных соотношений невелика.

Экспериментально-статистические модели строятся в том случае, когда нет чёткого представления о физике процесса или когда описываются очень сложные процессы. Для их построения используются экспериментальные данные или результаты длительной эксплуатации агрегата, подвергаемые затем статистической обработке (регрессивный и корреляционный анализы). В результате получаются вероятностные соотношения, называемые стохастическими моделями.

Наиболее часто применяются **комбинированные модели**, в которых используется и детерминированный, и статистический принципы составления моделей. В таких моделях основные уравнения получены на основе теоретических представлений, но входящие в них коэффициенты определяются статистическим путём.

6 Классификация систем автоматического управления

По методу управления САР и САУ делятся на:

а) системы, не приспосабливающиеся к изменяющимся режимам работы объекта регулирования;

б) приспосабливающиеся системы, т.е. адаптивные.

Не приспосабливающиеся системы - это наиболее простые системы, которые не изменяют своей структуры и параметров в процессе управления. Для этих систем на основе информации существующей до начала их работы (т.е. априорной) выбирают структуру и рассчитывают параметры, обеспечивающие заданные свойства системе для типовых и наиболее вероятных условий ее работы. Этот класс систем включает в себя три типа:

1) Стабилизирующие системы – обеспечивают поддержание регулируемой величины на постоянном заданном значении. Например: система автоматического регулирования, поддерживающая заданное значение расхода воздуха на дутье доменной печи.

2) Программные системы – обеспечивают изменение регулируемой величины во времени по заранее заданной программе. Например: система автоматического регулирования, обеспечивающая изменение расхода воздуха по ходу продувки в конвертере.

3) Следящие системы – обеспечивающие изменение регулируемой величины в заданном соотношении с управляющим воздействием, которое изменяется произвольным образом, не зависящим от данной системы. Например: система автоматического регулирования соотношения топливо – воздух при управлении сжиганием топлива в мартеновской печи.

Большинство систем, действующих в настоящее время, относится к не приспосабливающимся системам. Их структура и настройка определяется при проектировании и наладке и в дальнейшем автоматически не изменяется. При необходимости перенастройка системы может осуществляться вручную.

Приспосабливающиеся (адаптивные) системы – это такие системы, в которых параметры управляющих устройств или алгоритмы управления автоматически и целенаправленно изменяются для осуществления управления объектом, причем характеристики объекта или внешнее воздействие на него могут изменяться непредвиденным образом. Адаптивная система способна изменить свою структуру, параметры или программу действий в процессе управления. Особенный случай адаптивной системы – это **экстремальные системы**, которые автоматически ищут экстремум регулируемой величины, а так как его положение изменяется в процессе работы объекта, система автоматически изменяет направление поиска, скорость поиска и т. д.

Примером экстремальной системы является САР температуры в печи, отапливаемой газом. Требуется обеспечить максимальную температуру в печи, т.е. экстремум регулируемой величины при любых расходах топлива и изменении других параметров работы печи. Значение температуры печи – выходной величины объекта управления при данном топливе будут зависеть от расхода воздуха – входной величины объекта.

Если воздуха мало, то топливо сгорает не полностью и температура в печи меньше заданной. Если воздуха в печи много, то топливо сгорает полностью, но требуется тепло на нагрев избыточного воздуха, ненужного на сжигание топлива, и температура в печи также меньше необходимой. Если расход воздуха близок к теоретически необходимому для сжигания топлива ($V_в^o$), достигается максимальная температура в печи t_n^{max}.

Адаптивные системы реализуются с использованием ЭВМ и, следовательно, по существу их следует отнести к АСУТП, работающим в супервизорном режиме или режиме непосредственного цифрового управления. Исключение составляют некоторые экстремальные системы.

Для работы ЭВМ необходимо наличие аналитического описания объекта, т.е. его математической модели и алгоритмов адаптации и управления.

По характеру использования информации САР и САУ делятся на:

а) замкнутые системы;

б) разомкнутые системы;

в) комбинированные системы.

Замкнутые системы для своей работы используют текущую рабочую информацию о выходных величинах, определяют отклонение регулируемой величины от заданного значения и принимают меры для устранения этого отклонения.

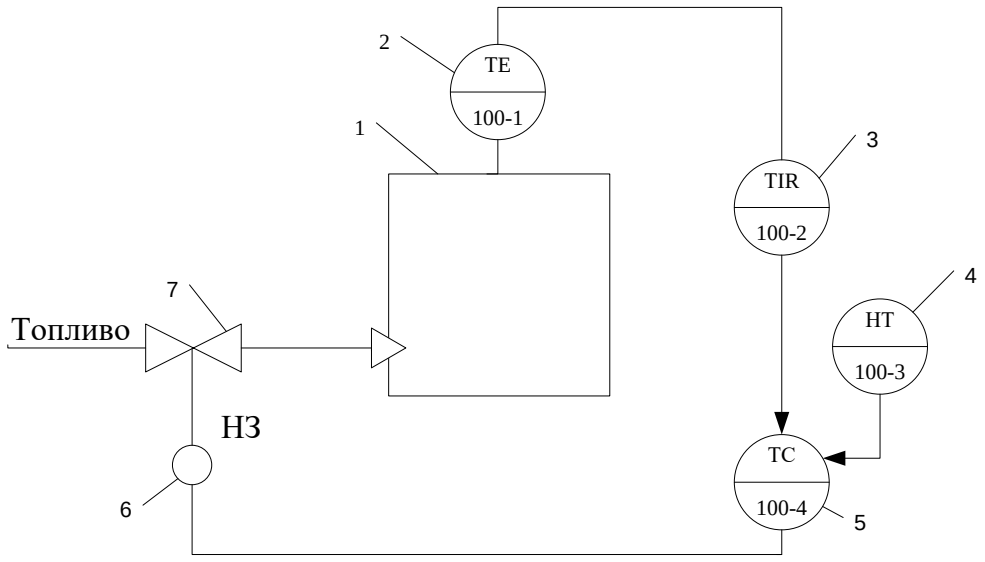


Рисунок 6.1 – Функциональная схема замкнутой САР температуры в печи

На рисунке 6.1 представлена структура САР температуры $t_п$ в печи 1. Чувствительным элементом – датчиком температуры служит термопара 2 (поз. 100-1) Информация о значении температуры в печи поступает на показывающий и регистрирующий прибор 3 (поз. 100-2), а с него в регулятор 5 (поз. 100-4).

В регулятор с задатчика 4 (поз. 100-3) поступает сигнал о заданном значении температуры $t_{по}$, в состав которого входит сравнивающий элемент. Сравнивающий элемент вырабатывает отклонение $\varepsilon = t_{по} - t_п$, и в соответствии с алгоритмом управления, регулятор формирует управляющее воздействие. Это воздействие в виде управляющего сигнала передаётся на исполнительный механизм 6, обеспечивающий перемещение регулирующего органа 7.

В качестве регулирующего органа используется поворотная заслонка в трубопроводе. Если температура в печи меньше заданной, то расход топлива увеличивается, а если больше - то уменьшается.

В рассмотренном примере имеется замкнутый контур регулирования, в котором информация о результатах работы объекта, т.е. о значениях регулируемой выходной величины поступает на его вход в преобразованном виде. Такая подача сигнала называется обратной связью. А элементом обратной связи является регулятор, обеспечивающий отрицательную обратную связь т.к. его действие направлено на уменьшение и устранение отклонения регулируемой величины от заданного значения.

Замкнутые системы работают при возмущениях, действующих по любым каналам, т.к. регуляторы в таких системах вступают в действие при наличии ошибки регулирования ε независимо от того, чем вызвано появление этого отклонения. Замкнутые системы не могут обеспечить соответствие заданного и реального значений регулируемой величины во всем диапазоне управления ($X=Xo$). Это равенство может установиться лишь в конце переходного процесса в положении равновесия.

Разомкнутые системы не используют рабочую информацию о регулируемых величинах т.к. отсутствует обратная связь. Работа таких систем основана на информации о входных величинах. Разомкнутые системы делятся на:

1) системы с жесткой программой.

2) системы с регулированием или управлением по возмущению.

Примером системы с жесткой программой служит система автоматического пуска и останова комплекса механизмов, в котором должна выдерживаться определенная последовательность работы отдельных механизмов. На вход регулятора поступает определенная программа действий. Регулятор, являющийся устройством, реализующим заданную программу, вырабатывает регулирующее воздействие, обеспечивающее необходимое изменение выходной величины.

В промышленности примерами разомкнутых систем являются: система автоматического управления загрузкой доменной печи, система автоматической перекидки клапанов мартеновской печи.

Разомкнутые системы с регулированием по возмущению используют информацию о входных величинах – возмущениях и принимают меры, чтобы указанные возмущения не оказывали влияние на выходную величину, т.е. как бы компенсируют их. Поэтому их называются инвариантными или системами с компенсацией возмущений. Рассмотрим структуру разомкнутой системы автоматического регулирования температуры в печи (рисунок 6.2), по своим задачам аналогичную замкнутой САР, рассмотренной выше (рисунок 6.1).

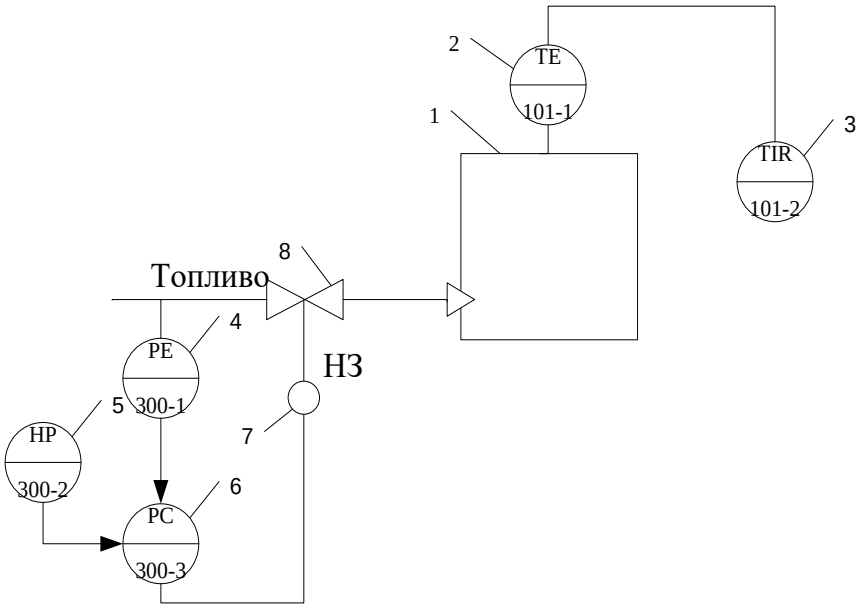


Рисунок 6.2 – Функциональная схема разомкнутой САР температуры в печи с регулированием по возмущению

Регулируемой величиной является температура t_n в печи 1. Основным возмущением является изменение давления газа в газопроводе, которое вызывает изменение расхода топлива и изменение температуры в печи, т.е. изменение регулируемой величины. Для компенсации влияния возмущения на значение выходной величины применяют регулятор 6 (поз. 300-3), называемый компенсатором возмущений. Регулятор получает информацию о значении давления газа от датчика давления 4 (поз. 300-1) и заданном значении давления от ручного задатчика 5 (поз. 300-2). Затем по заранее заданной программе с помощью исполнительного механизма 7 регулятор изменяет положение регулирующего органа 8. Давление перед горелкой при правильно выбранной структуре и законе действия компенсатора не будет зависеть от давления в газопроводе и, следовательно, не будет сказываться на расходе топлива и значении температуры в печи. В этом заключается принцип компенсации возмущений.

В рассмотренном примере регулируемая величина – температура в печи измеряется термопарой 2 (поз. 101-1) и регистрируется прибором 3 (поз. 101-2). Но эта текущая информация не используется системой регулирования, т.е. отсутствует обратная связь по результатам работы системы. Контур компенсации возмущения разомкнут, т.е. выходная величина контура не оказывает влияния на входную величину – изменение давления в газопроводе.

Приведенный пример показывает, что возможна компенсация определенного контролируемого возмущения. Если таких возмущений несколько, то для компенсации каждого из них необходим свой контур.

Но в системе всегда имеются возмущения, в том числе случайные и не контролируемые, которые могут вызвать отклонение регулируемой величины от заданного значения, поэтому на практике часто используют **комбинированные системы автоматического регулирования.** Они сочетают в себе оба принципа регулирования: по отклонению и по возмущению. В системе используется один регулятор для регулирования по отклонению, а для компенсации возмущений используется разомкнутые контуры с устройствами ввода возмущения (УВВ), которые изменяют задание регулятору в зависимости от величины возмущений.

По результатам работы в установившемся состоянии САР и САУ делятся на:

а) астатические системы;

б) статические системы.

В **астатических системах** регулируемая величина после окончания переходного процесса точно равна заданному значению. Практически она может отличаться на некоторую малую величину, обусловленную нечувствительностью системы.

В **статической системе** после окончания переходного процесса возникает разность между заданным и установившимся значениями регулируемой величины. Эта разность называется статической ошибкой. Она зависит от величины возмущения, в том числе задания и от параметров настройки регуляторов, но принципиально неизбежна в статических системах.

По числу регулируемых величин САР и САУ делятся на:

а) одномерные;

б) многомерные.

К **одномерным системам** относятся простейшие системы с одной регулируемой величиной, например в электрической нагревательной печи с неконтролируемой системой имеется одна регулируемая величина – температура.

Большинство систем относится к **многомерным**, т.к. они имеют множество регулируемых величин. В некоторых многомерных системах можно выделить несколько каналов регулирования, в которых каждая регулируемая величина определяется своим регулирующим воздействием и канал имеет свой

регулирующий орган. Положение его практически не оказывает влияния на другие регулируемые величины, в этом случае объект как бы распадается на несколько одномерных объектов со своими одномерными системами регулирования, такие системы являются автономными по задающим и регулирующим воздействиям. Вместе с тем многомерные системы характеризуются наличием связей между регулируемыми величинами, такие системы называются многосвязными.

Связи между регулируемыми величинами могут быть двух родов:

1) Внутренние - обусловленные физическими свойствами объектов (если, например, в печи регулируется температура свода, содержания кислорода в продуктах сгорания и давление в рабочем пространстве, то изменение расхода топлива, предназначенного для управления температурой свода, будет оказывать влияние и на содержание кислорода в продуктах сгорания и на давление в рабочем пространстве).

2) Внешние - т.е. накладываемые на систему по условиям ее функционирования или на основе требований технологического процесса, например, при автоматическом составлении шихты агломерационного процесса, задание регулятором количества отдельных компонентов устанавливается в зависимости от потребного суммарного количества шихты.

По характеру изменения регулирующих воздействий во времени САР и САУ делятся на**:**

а) непрерывные системы;

б) дискретные системы.

В **непрерывных системах** информация об их работе и регулирующие воздействия являются непрерывными функциями времени, т.е. в каждом элементе системы при наличии непрерывного изменения входной величины также непрерывными являются и выходные величины.

В **дискретных системах** информация и регулирующие воздействия появляются только в определенные моменты времени. Дискретные системы делятся на три класса:

1) Релейные системы;

2) Импульсные системы;

3) Цифровые системы.

В релейных системах один из элементов (обычно регулятор) имеет релейную характеристику.

В релейной системе выходная величина изменяется скачкообразно при определенном значении входной величины. В релейных системах происходит квантование выходной величины по уровню.

В импульсных системах существует хотя бы один элемент с импульсной характеристикой: при непрерывном изменении входной величины, выходная величина появляется только в определенные, дискретные моменты времени.

Импульсные системы осуществляют квантование выходной величины по времени. Обычно импульсным элементом является регулятор. Регулятор состоит из импульсного элемента (ИЭ) и исполнительного механизма (ИМ), формирующего управляющие воздействия $Y(\tau)$, в определённые моменты времени. На выходе импульсного элемента формируются импульсы $X_{вых}(\tau)$, параметры которых зависят от входной величины $\varepsilon(\tau)$, причем импульсные элементы могут осуществлять амплитудную и широтную модуляцию.

К дискретным системам относятся и **цифровые системы**, использующие в своем составе различные цифровые устройства: ЭВМ, цифровые измерительные приборы, микропроцессорные регуляторы.

В цифровых системах осуществляется квантование величин и по уровню и по времени, т.е. они являются релейно-импульсными. Цифровые системы обладают высоким быстродействием, имеют малый интервал квантования по времени и по результатам своей работы близки к непрерывным системам.

По виду энергии, применяемой для работы, САР и САУ делятся на**:**

а) прямого действия;

б) косвенного действия.

В **системах прямого действия** для перемещения регулирующего органа применяется внутренняя энергия системы, например, энергия чувствительного элемента.

В **системах косвенного действия** для работы используется внешняя энергия.

В зависимости от вида используемой внешней энергии, системы косвенного действия делятся на электрические, пневматические, гидравлические, комбинированные.

7 Статические и динамические характеристики объектов управления

Объект управления является определяющим компонентом любой САР, ведь вся структура САР выбирается, исходя из статических и динамических свойств ОУ.

Статическая характеристика ОУ – это зависимость регулируемой величины *Y* (температуры, давления, расхода и т.д.) от управляющего воздействия *U* (мощность нагревателя, процент открытия клапана и т.д.) в **установившемся** режиме.

В идеальном случае статическая характеристика ОУ может быть линейной (рисунок 7.1*а*), в таком случае коэффициент усиления (передачи) ОУ будет иметь одинаковое значение во всем рабочем диапазоне (K = const).

Обычно статическая характеристика имеет нелинейный вид в рабочем диапазоне (рисунок 7.1*б*), вследствие чего коэффициент усиления (передачи) ОУ нестационарен, т.е. зависит от величины управляющего воздействия *U*.

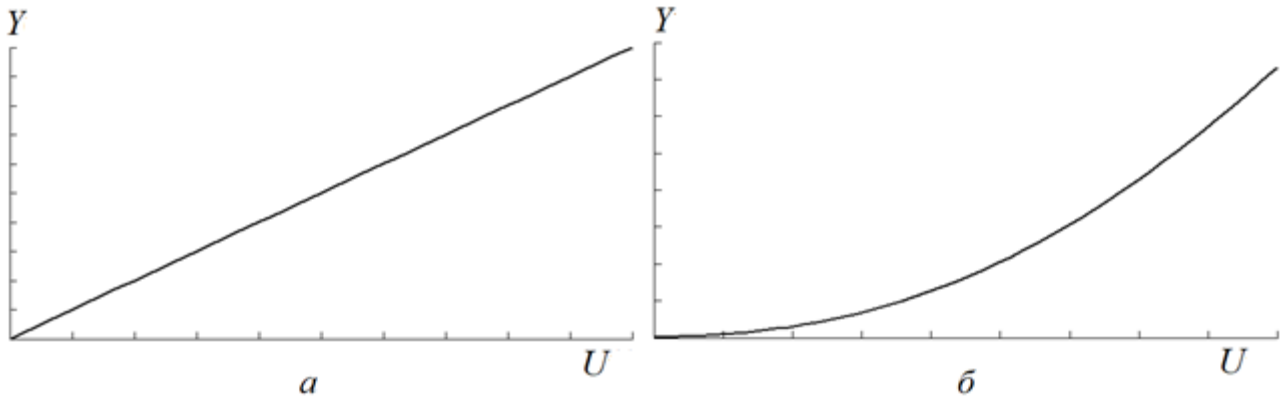


Рисунок 7.1 –Статическая характеристика ОУ
а – линейная
б – нелинейная

Статическая характеристика ОУ – это то, к чему стремится регулируемая величина со временем, т.к. в промышленности и, в частности, в химической технологии, на ОУ постоянно воздействует совокупность возмущающих воздействий (например, перепад температур зимой и летом), из-за действия которых ОУ не может перейти в установившийся режим и, таким образом, постоянно находится в переходном режиме.

Важно помнить, что понятие статической характеристики применимо только для ОУ с самовыравниванием.

Самовыравнивание – свойство ОУ после возмущения вновь прийти в состояние равновесия без внешнего воздействия.

ОУ без самовыравнивания характеризуются тем, что при нарушении равновесия за счет отклонения регулируемой величины равновесие не восстанавливается, такие ОУ называют соответственно **астатическими**.

Самовыравнивание может быть положительным и отрицательным. ОУ с положительным самовыравниванием характеризуется тем, что при изменении регулируемой величины нарушенное равновесие восстанавливается без участия регулятора, а с отрицательным самовыравниванием нарушенное равновесие может быть восстановлено с помощью регулятора.

На рисунке 7.1 приведены кривые разгона ОУ с самовыравниванием и без самовыравнивания.

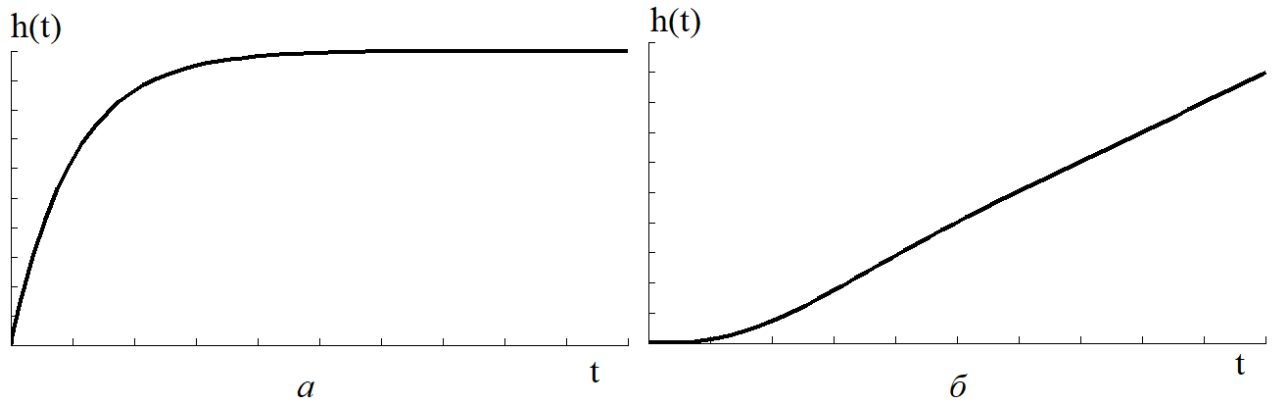


Рисунок 7.2 – Кривые разгона ОУ
а – с самовыравниванием
б – без самовыравнивания

Способность объекта регулирования к самовыравниванию характеризуется коэффициентом самовыравнивания ρ (7.1):

$$\rho = \frac{\Delta X}{\Delta Y}, \qquad (7.1)$$

где ΔX – величина возмущающего воздействия; ΔY – отклонение регулируемой величины в ОУ.

Коэффициент самовыравнивания численно равен отношению величины возмущающего воздействия к отклонению регулируемой величины, вызванному этим воздействием.

Самовыравнивание способствует устойчивости регулируемого объекта; чем больше коэффициент самовыравнивания, тем быстрей объект самостоятельно восстанавливает заданное значение регулируемой величины и более устойчивым будет процесс регулирования. Коэффициент самовыравнивания не является постоянной величиной, т.к. зависит от нагрузки на объект; чем меньше нагрузка,

тем меньше коэффициент и тем труднее обеспечить устойчивое и качественное регулирование.

Для описания ОУ в переходном режиме используются динамические характеристики, позволяющие количественно оценить инерционность и запаздывание реального ОУ.

Динамическая характеристика – это реакция ОУ на возмущающее воздействие определенного типа. Таким возмущающим воздействием может быть гармоническое, импульсное, а также единичное ступенчатое (в виде функции Хевисайда $x(t) = 1, t \geqslant 0$).

Наиболее часто при анализе рассматривают реакцию ОУ на единичное ступенчатое воздействие x(t), которую также называют переходной характеристикой (кривой разгона).

Большинство ОУ в химической технологии обладают т.н. "S - образной" кривой разгона, которую обычно можно аппроксимировать кривой, соответствующей передаточной функции апериодического звена 1 порядка с запаздыванием (7.2):

$$W(s) = \frac{K}{T \cdot s + 1} e^{-\tau \cdot s}, \qquad (7.2)$$

где K – коэффициент усиления (передачи) ОУ; T – постоянная времени; τ – время запаздывания.

Коэффициент усиления (передачи) ОУ K – это отношение приращения выходного сигнала ΔY к приращению входного ΔX в окрестности рабочей точки, это величина, обратная коэффициенту самовыравнивания. Коэффициент усиления ОУ определяется в соответствии с выражением (7.3):

$$K = \frac{\Delta Y}{\Delta X} = \frac{1}{\rho}. \qquad (7.3)$$

Отличие коэффициента усиления от коэффициента передачи заключается в том, что коэффициент усиления – безразмерная величина, в то время как коэффициент передачи имеет размерность $\left[\frac{\text{ед. изм. регулируемой величины}}{\%\text{ хода ИУ}}\right]$.

Постоянная времени (время разгона) ОУ. Временем разгона объекта является время, в течение которого выходная величина достигла бы нового установившегося значения, если бы изменялась со скоростью, равной скорости в момент подачи входного ступенчатого рассогласования. Чем больше емкость

объекта регулирования, тем больше время разгона. Время разгона определяется уравнением (7.4):

$$T = \frac{\Delta X \cdot dt}{\Delta Y}, \tag{7.4}$$

где ΔX - относительная величина возмущения; dt - изменение времени; ΔY - относительное значение регулируемой величины.

Величина, обратная времени разгона, называется скоростью разгона. Чем больше возмущающее воздействие, тем быстрее изменяется регулируемая величина, т.е. тем больше будет скорость этого изменения.

Время запаздывания. Изменение регулируемой величины с момента возмущения происходит не сразу, а через некоторое время, которое называется запаздыванием процесса в объекте. Запаздывание присуще не только регулируемому объекту, но и регулятору и зависит от инерционности чувствительного элемента, кинематики привода регулирующего органа и системы передачи командного сигнала. К свойствам ОУ относится запаздывание передаточное и переходное.

Передаточное запаздывание – это время, в течение которого регулируемая величина, несмотря на возмущение, не изменяется. Передаточное запаздывание зависит от нагрузки объекта. Чем больше нагрузка, тем меньше передаточное запаздывание, т.к. при большой нагрузке регулируемая среда движется быстрее и чувствительный элемент будет реагировать на это возмущение раньше. Передаточное запаздывание зависит от емкости объекта: чем больше емкость, тем больше время передаточного запаздывания. Для уменьшения времени передаточного запаздывания регулирующий орган необходимо располагать ближе к объекту, чтобы емкость между ними была наименьшей.

Переходным запаздыванием процесса регулирования называется запаздывание, зависящее от тепловых, гидравлических и других сопротивлений между емкостями объекта, и определяется как промежуток времени от момента подачи возмущения до начала изменения регулируемой величины. Сумма времени передаточного и переходного запаздываний называется временем полного запаздывания (7.5):

$$\tau = \tau_0 + \tau_n, \tag{7.5}$$

где τ_0 - время передаточного запаздывания; τ_n - время переходного запаздывания.

Запаздывание существенно влияет на величину динамической ошибки: чем больше запаздывание, тем больше ошибка.

8 Методы определения динамических характеристик объектов управления

Динамические характеристики могут определяться аналитическим или экспериментальным путем. Определение динамических свойств ОУ аналитическим путем сводится к составлению дифференциальных уравнений динамики поведения ОУ на основании использования основных законов физики (сохранения энергии, массы и т.д.). Однако для сложных объектов эти уравнения имеют высокий порядок, их решение является трудоемкой задачей. В связи с этим в практических приложениях динамические свойства ОУ весьма часто определяют экспериментальным путем.

Экспериментальный способ требует минимальных сведений об устройстве ОУ, о протекающих в нем процессах (в этом случае объект выступает как «черный ящик»), и может быть применен к работающим объектам в реальных, рабочих условиях эксплуатации, обеспечивая приемлемую точность определения передаточной функции.

Для экспериментального определения переходной характеристики (кривой разгона) объект приводят в равновесное состояние, близкое к номинальному. После этого изменяют управляющее воздействие на ΔX (% от диапазона изменения управляющего воздействия) – т.е. вносят скачкообразное возмущение и через определенные промежутки времени регистрируют изменяющиеся значения выходной величины Y (регулируемого параметра) до прихода ее к новому установившемуся значению. На основании полученных данных строят график переходной характеристики (кривой разгона) объекта, т.е. график изменения выходной величины Y при скачкообразном изменении входной величины X.

Рассмотрим графический и аналитический метод определения передаточной функции ОУ в виде апериодического звена 1 порядка с запаздыванием (7.2).

Графический способ предполагает нахождение значений постоянной времени T и времени запаздывания τ с помощью дополнительных графических построений на полученной кривой разгона.

Сущность **графического метода** заключается в следующем (рисунок 8.1):

1) На графике экспериментальной переходной характеристики находят точку максимальной скорости изменения выходной величины, являющуюся точкой перегиба (смены знака кривизны) кривой, точку А.

Однако, ввиду того, что переходные функции ОУ в химической промышленности, как правило, не имеют явно выраженной точки перегиба, то определение ее координат осуществляют так: в средней, наиболее быстро изменяющейся, части графика кривой разгона берется 5-7 ординат $Y_g(t_g)$ через равные интервалы времени Δt, и вычисляются расстояния между ними

$\Delta Y_{g,g-1} = Y_g - Y_{g-1}$. Далее находится максимальное из $\Delta Y_{g,g-1}$, определяется соответствующее ей $t_\omega = t_g - \Delta t$, и на графике находится точка $A = Y_\omega(t_\omega)$.

2) Через точку А проводят касательную ВС к кривой переходной характеристики до пересечения её с линиями начального значения y(0) и установившегося значения y(∞) выходной величины.

Точки пересечения В и С сносят на ось времени и отмечают два отрезка времени: один от момента скачка входной величины (t = 0) до момента, определяемого точкой В, другой от момента, определяемого точкой В, до момента, определяемого точкой С, первый отрезок времени является временем запаздывания τ, второй – постоянной времени *T* объекта.

Таким образом, постоянная времени ОУ *T* – это проекция касательной ВС на ось абсцисс.

3) Коэффициент передачи ОУ *K* определяется отношением приращений выходной величины параметра объекта Δ*Y* к величине воздействия на объект Δ*X* по формуле (7.3).

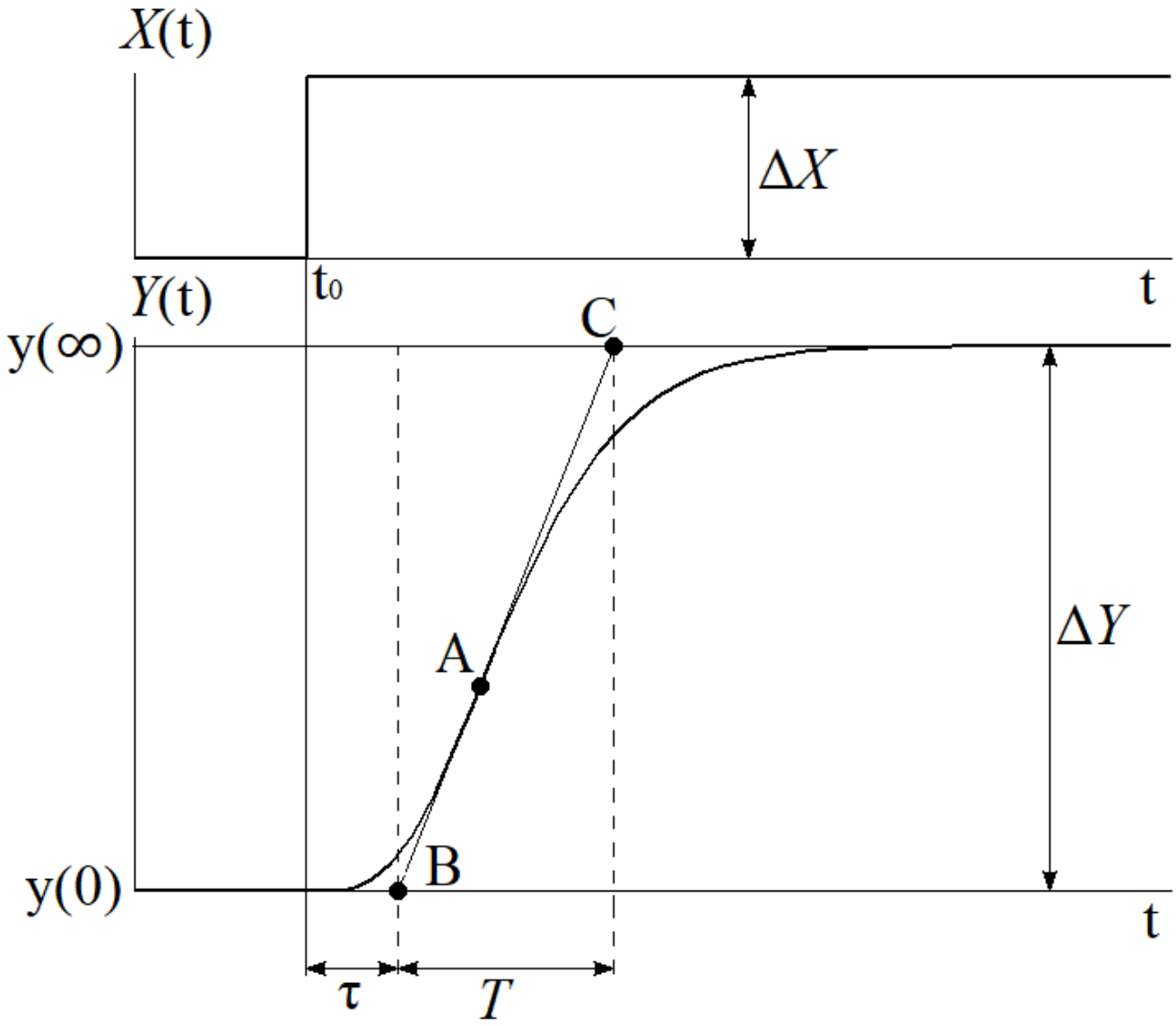


Рисунок 8.1 – Графический метод определения передаточной функции ОУ

Сущность **интерполяционного метода** определения постоянной времени *T* и времени запаздывания τ, базирующегося на методике Ормана и предложенной Кругом и Мининой, заключается в следующем (рисунок 8.2):

1) По кривой разгона находят значения y1 (8.1) и y2 (8.2):

$$y1 = 0.33(y(\infty) - y(0)) + y(0) \quad (8.1)$$

$$y2 = 0.7(y(\infty) - y(0)) + y(0) \quad (8.2)$$

2) Проведя перпендикуляры к кривой разгона, определяют значения t1' и t2'.

3) Определяют значения t1 и t2 по формулам (8.3) и (8.4) соответственно:

$$t1 = t1' - t0 \quad (8.3)$$

$$t2 = t2' - t0 \quad (8.4)$$

4) Определяют значения времени запаздывания τ и постоянной времени *T* по формулам (8.5) и (8.6) соответственно:

$$\tau = 0.5(3t1 - t2) \quad (8.5)$$

$$T = \frac{t2 - \tau}{1.2} \cong 1.25(t2 - t1) \quad (8.6)$$

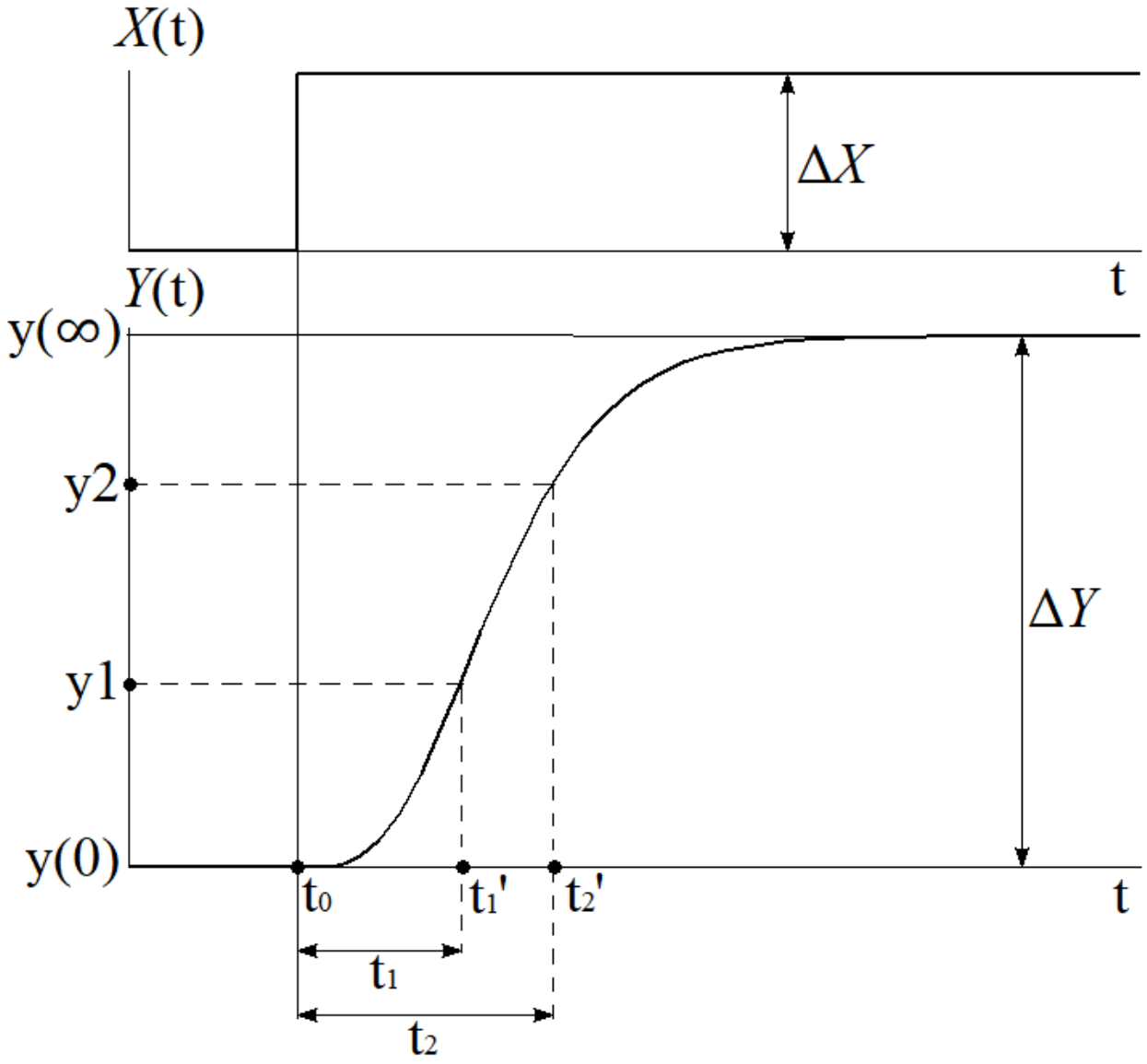


Рисунок 8.2 – Интерполяционный метод Ормана

5) Коэффициент передачи ОУ K определяется отношением приращений выходной величины параметра объекта ΔY к величине воздействия на объект ΔX по формуле (7.3).

Далее рассмотрим случай аппроксимации кривой разгона астатического объекта управления, передаточная функция которого в общем случае будет иметь вид (8.7):

$$W(s) = \frac{1}{\overline{T} \cdot s} e^{-\tau \cdot s} = \frac{\overline{K}}{s} e^{-\tau \cdot s}, \qquad (8.7)$$

где $\overline{T}$ – условная постоянная времени объекта, $\overline{K}$ – условный коэффициент усиления ($\overline{K} = 1/\overline{T}$); τ – время запаздывания.

Для аппроксимации кривой разгона такого объекта можно использовать как аналитический, так и графический методы.

Графический метод (рисунок 8.3) предполагает, что для аппроксимации кривой разгона необходимо провести прямую, касательную к точке перегиба А кривой разгона (точка, после которой кривая не меняет свой угол наклона).

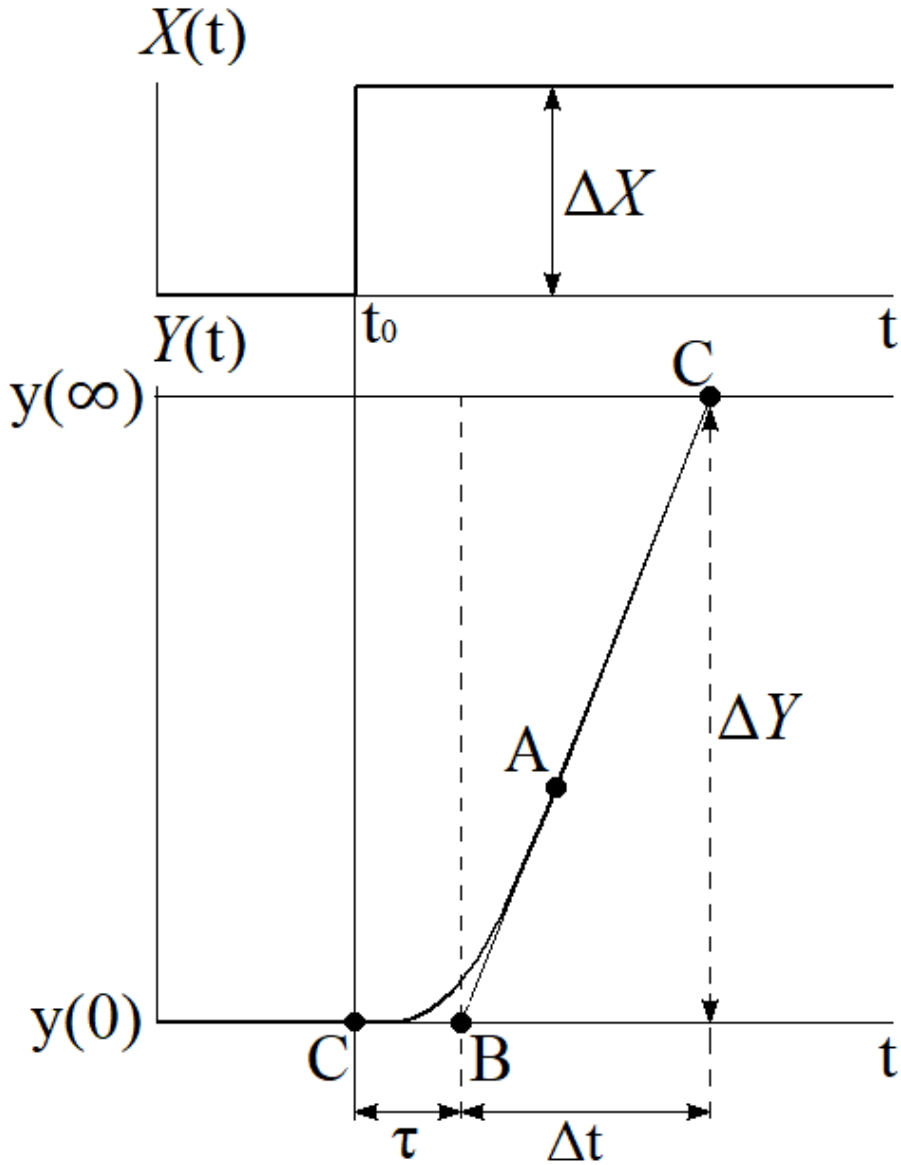


Рисунок 8.3 – Аппроксимация кривой разгона астатического объекта

Расстояние от точки С, соответствующей времени подачи ступенчатого воздействия до точки пересечения этой прямой с осью абсцисс В – это время полного запаздывания τ.

Тангенс угла наклона построенной прямой к оси абсцисс α определяется по формуле (8.8):

$$tg\alpha = \frac{\Delta Y}{\Delta t}, \tag{8.8}$$

где *t* соответствует значению по оси абсцисс, *Y* – значению по оси ординат.

Условный коэффициент усиления объекта определяется по формуле (8.9):

$$\overline{K} = \frac{\Delta Y}{\Delta X \Delta t} = \frac{tg\alpha}{\Delta x} = \frac{1}{\overline{T}}, \tag{8.9}$$

где *X* соответствует величине поданного на объект ступенчатого воздействия.

Аналитический метод аппроксимации кривой разгона астатического ОУ приведен на рисунке 8.4.

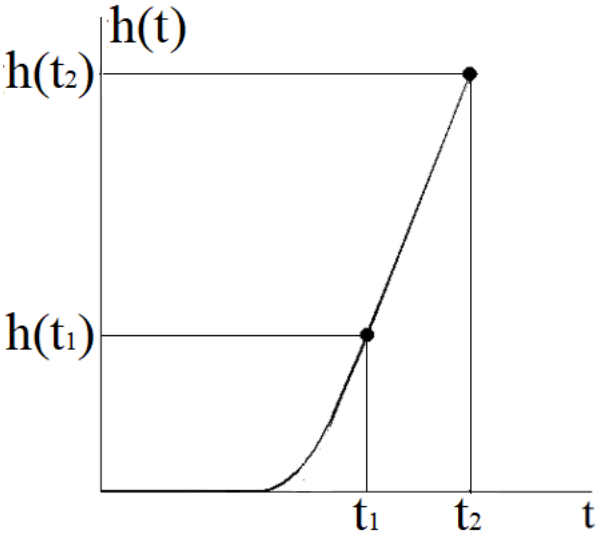


Рисунок 8.4 – Аналитический метод аппроксимации кривой астатического объекта

Здесь также предполагается передаточная функция вида (8.7), где:

$$\tau = \frac{h_2(t_2)\cdot t_1 - h_1(t_1)\cdot t_2}{h_2(t_2) - h_1(t_1)},$$

$$\overline{K} = \frac{h_2(t_2)}{t_2 - \tau}, \tag{8.10}$$

$$\overline{T} = \frac{1}{\overline{K}} = \frac{t_2 - \tau}{h_2(t_2)}.$$

Значения t_1 и t_2, а также соответствующие им ординаты $h_1(t_1)$ и $h_2(t_2)$ определяются по графику кривой разгона, как показано на рисунке выше.

9 Законы управления аналоговых САР

Для аналоговых САР можно выделить несколько типовых законов управления: пропорциональный (П-регулятор), интегральный (И-регулятор), пропорционально-интегральный (ПИ-регулятор), пропорционально-дифференциальный (ПД-регулятор), а также пропорционально-интегрально-дифференциальный (ПИД-регулятор).

Пропорциональный закон управления выдает управляющее воздействие на ОУ пропорционально величине ошибки регулирования. П-регулятор имеет передаточную функцию вида (9.1):

$$W_{reg}(s) = K_p \tag{9.1}$$

где K_p - коэффициент усиления регулятора.

В случаях, когда ОУ обладает самовыравниванием, применение П-регуляторов приводит к наличию статической ошибки – т.е. регулируемая величина не достигнет заданного значения. Чем больше коэффициент усиления, тем меньше статическая ошибка, однако, при слишком большом коэффициенте усиления САР может войти в колебательный режим, и при дальнейшем увеличении коэффициента САР и вовсе выйдет из устойчивого состояния. Неизбежное наличие статической ошибки при использовании П-регуляторов может быть неприемлемо для некоторых объектов в химической технологии. На рисунке 9.1 приведена переходная характеристика П-регулятора при $K_p = 2.1$.

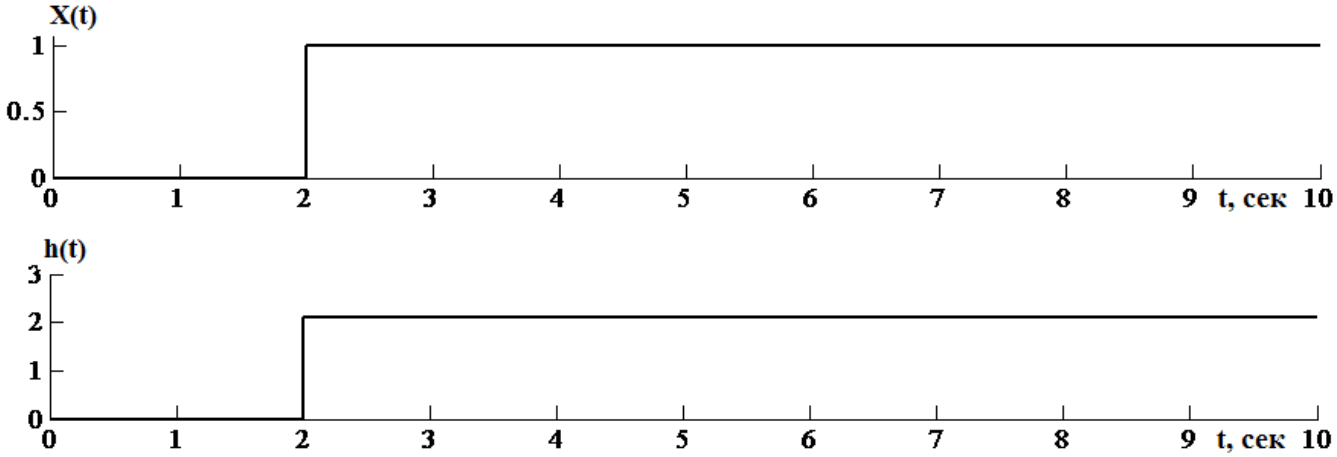


Рисунок 9.1 – Переходная характеристика П-регулятора

Интегральный закон управления. Величина управляющего воздействия зависит от накопленной суммы (интеграла) ошибки регулирования.

Передаточная функция И-регулятора имеет вид (9.2):

$$W_{reg}(s) = \frac{1}{T_i \cdot s} \qquad (9.2)$$

где T_i - постоянная времени интегрирования, сек – это время, в течение которого выходная величина достигнет значения входного ступенчатого рассогласования.

Применение И-регуляторов позволит со временем свести ошибку к нулю, однако такие регуляторы очень инерционны, т.е. переходный процесс будет продолжаться значительное время, что также не подходит ОУ в химической технологии.

Переходная характеристика И-регулятора с $T_i = 1.7$ сек приведена на рисунке 9.2.

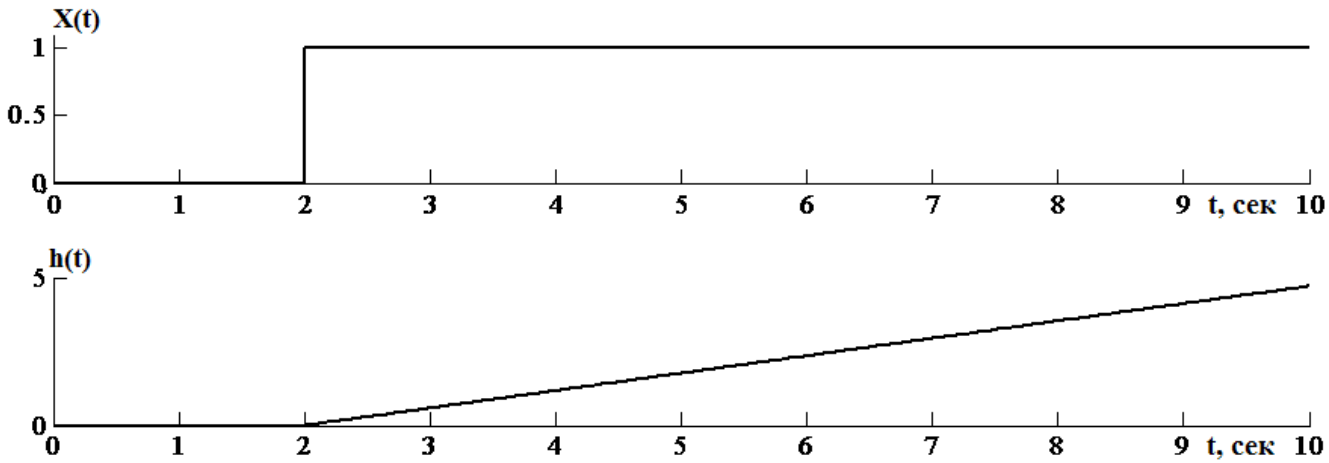


Рисунок 9.2 – Переходная характеристика И-регулятора

Перечисленные выше П- и И-регуляторы не всегда могут обеспечить требуемых показателей качества регулирования, поэтому на практике обычно применяют ПИ-, ПД-, или ПИД-регуляторы.

Пропорционально-интегральный закон управления, как видно из названия, объединяет в себе пропорциональный и интегральный регулятор, его передаточная функция имеет вид (9.3), а переходная характеристика с настройками: $K_p = 2.1$, $T_i = 1.7$ сек показана на рисунке 9.3.

$$W_{reg}(s) = K_p + \frac{1}{T_i \cdot s} = K_p\left(1 + \frac{1}{T_{iz} \cdot s}\right) \qquad (9.3)$$

где T_{iz} - время изодрома, $T_{iz} = T_i \cdot K_p$ - это время, в течение которого выходная величина достигает значения удвоенного произведения $K_p \cdot x(t)$.

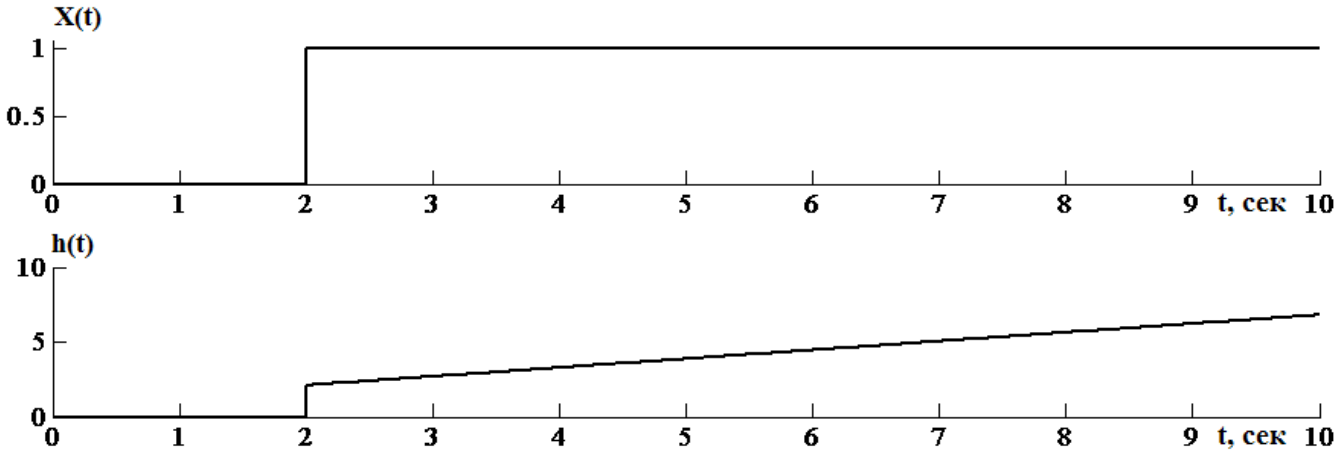


Рисунок 9.3 – Переходная характеристика ПИ- регулятора

Такие регуляторы находят широкое применение в химической технологии именно за счет лучшего быстродействия, по сравнению с И-регулятором и за счет отсутствия статической ошибки, благодаря П-части.

Пропорционально-дифференциальный закон является суммой пропорциональной и дифференциальной составляющих.

Дифференциальная составляющая выдает управляющий сигнал пропорционально скорости изменения отклонения регулируемой величины. Передаточная функция ПД-регулятора имеет вид (9.4):

$$W_{reg}(s) = K_p + T_d \cdot s = K_p\left(1 + T_{pr} \cdot s\right) \tag{9.4}$$

где T_d - постоянная времени дифференцирования, сек; $T_{pr} = T_d / K_p$ - время предварения – время, в течение которого входной сигнал, изменяющийся с постоянной скоростью, достигает значения выходного сигнала, изменяющегося с той же скоростью в момент начала отсчета.

На рисунке 9.4 приведена переходная характеристика реального регулятора с настройками: $K_p = 2.1$, $T_d = 4.6$ сек.

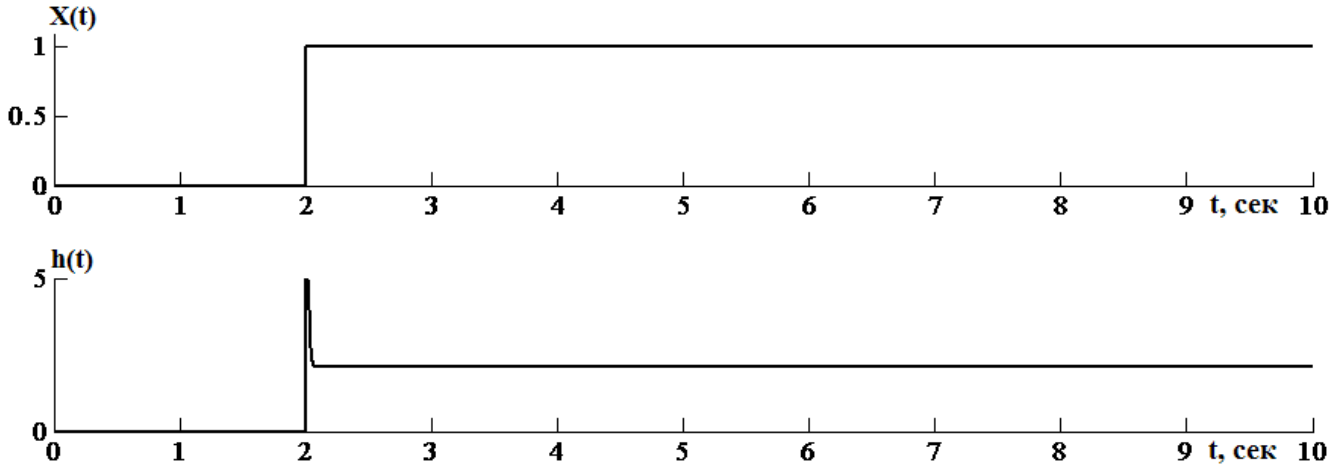


Рисунок 9.4 – Переходная характеристика реального ПД-регулятора

Главным недостатком ПД-регуляторов, очевидно, является отсутствие интегральной составляющей, которая сводила бы к нулю статическую ошибку, а также высокая чувствительность к помехам.

Пропорционально – интегрально – дифференциальный закон управления. Как следует из названия закона управления, ПИД-регулятор выдает управляющее воздействие в виде суммы П, И и Д частей. Таким образом, ПИД-регулятор сочетает в себе все достоинства перечисленных выше регуляторов и компенсирует их недостатки. Передаточная функция ПИД-регулятора имеет вид (9.5):

$$W_{reg}(s) = K_p + \frac{1}{T_i \cdot s} + T_d \cdot s = K_p\left(1 + \frac{1}{T_{iz} \cdot s} + T_{pr} \cdot s\right) \qquad (9.5)$$

Переходная характеристика ПИД-регулятора с настройками: $K_p = 2.1$, $T_i = 1.7$ сек, $T_d = 4.6$ сек показана на рисунке 9.5.

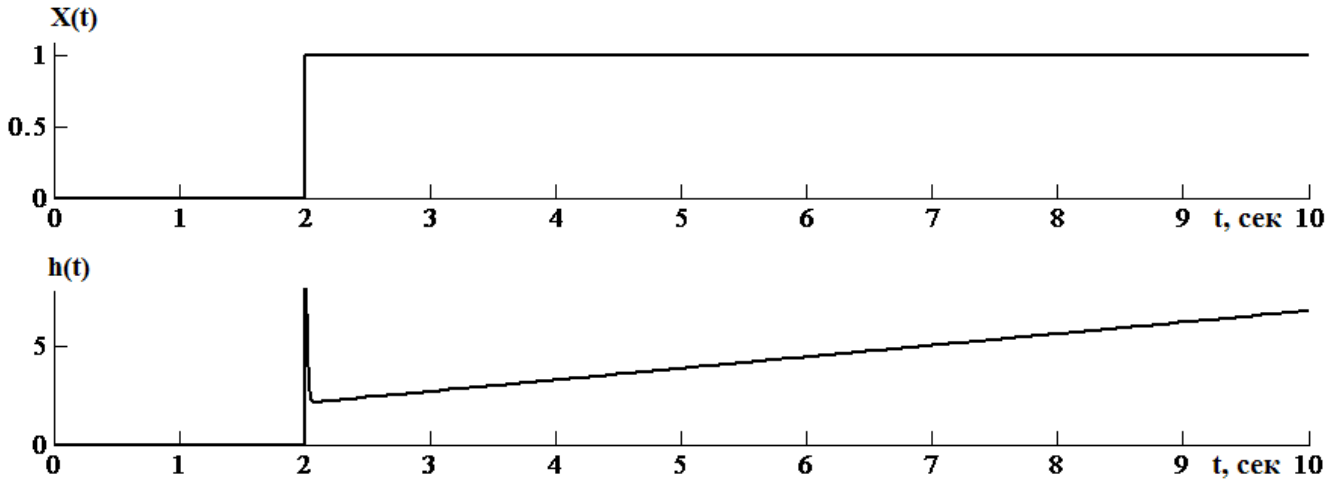


Рисунок 9.5 – Переходная характеристика ПИД-регулятора

Именно ПИД-регуляторы наиболее часто применяются на объектах управления в промышленности благодаря качеству работы и своей универсальности – ПИД-регулятор всегда можно превратить в ПИ-, ПД- или П-регулятор, приравняв к нулю постоянную времени дифференцирования (предварения), либо увеличивая время интегрирования (изодрома) до бесконечности соответственно.

10 Настройка аналоговых регуляторов

Существует множество методов настройки регуляторов – как аналитических, так и экспериментальных. Рассмотрим некоторые из них более подробно, принимая во внимание, что передаточные функции ПИ и ПИД регулятора имеют вид (10.1) и (10.2), а передаточная функция объекта управления (10.3):

$$R(s) = K_p\left(1 + \frac{1}{T_{iz} \cdot s}\right) \tag{10.1}$$

$$R(s) = K_p\left(1 + \frac{1}{T_{iz} \cdot s} + T_{pr} \cdot s\right) \tag{10.2}$$

$$W(s) = \frac{K}{T \cdot s + 1} e^{-\tau \cdot s} \tag{10.3}$$

10.1 Метод Копеловича

При синтезе параметров регулятора исходят из того, какое качество регулирования хотят получить. Данный метод настройки регуляторов позволяет получить три типовых процесса регулирования:

1) апериодический с минимальным временем регулирования;

2) с 20% перерегулированием;

3) с минимальной квадратичной площадью отклонения.

Приближённые значения настроек для регуляторов будут определяться по следующим формулам:

- для ПИ регулятора:

- апериодический процесс с минимальным временем регулирования (10.4):

$$\begin{cases} K_p = \dfrac{0.6}{K \cdot (\tau / T)}, \\ T_{iz} = 0.8\tau + 0.5T. \end{cases} \tag{10.4}$$

- 20%-ое перерегулирование (10.5):

$$\begin{cases} K_p = \dfrac{0.7}{K \cdot (\tau / T)}, \\ T_{iz} = \tau + 0.3T. \end{cases} \tag{10.5}$$

- минимальной квадратичной площадью отклонения (10.6):

$$\begin{cases} K_p = \dfrac{1.7}{K \cdot (\tau / T)}, \\ T_{iz} = \tau + 0.35T. \end{cases} \tag{10.6}$$

- для ПИД регулятора:
- апериодический процесс с минимальным временем регулирования (10.7):

$$\begin{cases} K_p = \dfrac{0.95}{K \cdot (\tau / T)}, \\ T_{iz} = 2.4\tau, \\ T_{pr} = 0.4\tau \end{cases} \tag{10.7}$$

- 20%-ое перерегулирование (10.8):

$$\begin{cases} K_p = \dfrac{1.2}{K \cdot (\tau / T)}, \\ T_{iz} = 2\tau, \\ T_{pr} = 0.4\tau \end{cases} \tag{10.8}$$

- минимальной квадратичной площадью отклонения (10.9):

$$\begin{cases} K_p = \dfrac{1.4}{K \cdot (\tau / T)}, \\ T_{iz} = 1.3\tau, \\ T_{pr} = 0.5\tau \end{cases} \tag{10.9}$$

Приведенные выше формулы были получены советским учёным Копеловичем А.П. ещё в 1960 г. опытным путём. Для их получения проводились многочисленные эксперименты на электронной моделирующей установке и на промышленных объектах. После многочисленных опытов были составлены номограммы, на основании которых и были выведены приближённые формулы для определения оптимальных настроек регуляторов.

10.2 Метод Зиглера-Никольса

Метод настройки параметров типовых регуляторов Зиглера-Никольса один из самых старых. Зиглер и Никольс предложили свой метод настройки регуляторов ещё в 1942 году. Этот метод, также как и метод Копеловича, относится к эмпирическим и основан на использовании данных, полученных экспериментально на реальном объекте. Существует два варианта метода. Первый вариант основан на использовании запасов устойчивости. Во втором варианте используется реакция объекта регулирования на ступенчатое воздействие. Рассмотрим оба варианта.

Преимуществом **первого варианта** метода заключается в том, что все операции проводятся с реальным объектом управления, т.е. нет необходимости аппроксимировать ОУ каким-либо типовым звеном, как в случае применения метода Копеловича.

Суть первого варианта заключается в поиске критического значения коэффициента усиления П - регулятора, при котором система окажется на границе устойчивости, после чего либо определяется период колебаний T, установившихся в системе, либо критическая частота $\omega_{кр}$ по графику фазо-частотной характеристики ОУ. (Искать частоту по графику удобнее, чем период колебаний, поэтому сначала найдем частоту, а затем выразим через неё период колебаний T).

Частотные характеристики ОУ можно определить в Mathcad. Расчетные формулы для Mathcad приведены ниже:

Ампитудно – частотная характеристика (АЧХ) $A(\omega) := |W(j\omega)|$

Фазовая частотная характеристика (ФЧХ) $\varphi(\omega) := \arg(W(j\omega))$

Вещественная частотная характеристика $\mathrm{Re}(\omega) := \mathrm{Re}(W(j\omega))$

Мнимая частотная характеристика $\mathrm{Im}(\omega) := \mathrm{Im}(W(j\omega))$

Из графика ФЧХ легко определить критическую частоту ωkr (это точка пересечения графика ФЧХ с прямой -π).

Чтобы график ФЧХ приобрел правильную форму, в Mathcad надо задать:

$$\varphi(\omega) := \begin{cases} \arg(W(j\omega)) \ if \ \omega < \omega kr \\ (\arg(W(j\omega)) - 2\pi) \ otherwise \end{cases}$$

Критическую настройку П - регулятора определим как Kkr:=1/A(ωkr).

Период колебаний T:=2π/ωkr.

Расчетные формулы для определения настроек ПИ – регулятора (10.10):

$$\begin{cases} K_p = 0.45 \cdot Kkr, \\ T_{iz} = T / 1.2. \end{cases} \quad (10.10)$$

Расчетные формулы для определения настроек ПИД – регулятора (10.11):

$$\begin{cases} K_p = 0.6 \cdot Kkr, \\ T_{iz} = T / 2, \\ T_{pr} = T / 8. \end{cases} \quad (10.11)$$

Второй вариант метода использует параметры объекта регулирования, также как и метод Копеловича. Параметры регулятора во втором варианте метода определялись Зиглером и Никольсом исходя из требования к декременту затухания, равному 4. Второй вариант метода Зиглера-Никольса никак не учитывает требования к запасу устойчивости, что относят к недостатку. Метод даёт удовлетворительные результаты, если отношение $0.15 < \tau/T < 0.6$.

Формулы для расчёта настроек ПИ и ПИД-регуляторов в виде, подходящем для принятых передаточных функций регуляторов (10.1) и (10.2):

- для ПИ-регулятора (10.12):

$$\begin{cases} k_p = \dfrac{0{,}9T}{k_{\text{об}}\tau}, \\ T_{\text{из}} = 3\tau \end{cases} \quad (10.12)$$

- для ПИД-регулятора (10.13):

$$\begin{cases} k_p = \dfrac{1{,}2T}{k_{\text{об}}\tau}, \\ T_{\text{из}} = 2\tau, \\ T_{\text{пр}} = \dfrac{\tau}{2}. \end{cases} \quad (10.13)$$

Регуляторы, параметры которых рассчитаны по методу Зиглера-Никольса, не всегда обеспечивают требуемое качество процесса регулирования. Как правило, требуется дополнительная подстройка их параметров. Несмотря на это, метод Зиглера-Никольса и некоторые его модификации весьма популярны, и многие производители регуляторов рекомендуют их для настройки регуляторов.

10.3 Метод Чина-Хронеса-Резвика

В отличие от Зиглера и Никольса, которые использовали в качестве критерия качества настройки декремент затухания, равный 4, Чин, Хронес и Резвик использовали критерий максимальной скорости нарастания при отсутствии перерегулирования или при наличии не более чем 20%-го перерегулирования. Такой критерий позволяет получить больший запас устойчивости, чем в методе Зиглера-Никольса.

Метод Чина-Хронеса-Резвика даёт две разные системы параметров регулятора. Одна из них получена при наблюдении отклика на изменения задания, вторая – при наблюдении отклика на внешние возмущения (по нагрузке). Какую систему параметров выбирать – зависит от того, что важнее для конкретного регулятора: качество регулирования при изменении задающего воздействия, или ослабление внешних возмущений. Если же важно и то, и другое, то необходимо использовать регуляторы с двумя степенями свободы.

Метод Чина-Хронеса-Резвика использует аппроксимацию объекта апериодическим звеном первого порядка с запаздыванием. Однако идентификация модели объекта регулирования по методу Чина-Хронеса-Резвика имеет свои особенности по сравнению с методом Ормана и использует для расчёта параметров настройки регулятора несколько другие параметры. Особенности аппроксимации кривой разгона приведены в [4]. Приведем итоговый вариант формул, адаптированных к заданной в работе структуре регуляторов и методам аппроксимации кривой разгона, показанным ранее.

Регулирование по задающему воздействию:

Приближённые значения настроек для регуляторов по задающему воздействию будут определяться по формулам:

- для ПИ регулятора:

- без перерегулирования (10.14):

$$\begin{cases} k_p = \dfrac{0{,}35T}{\tau\, k_{\text{об}}}, \\ T_{\text{из}} = 1{,}2\ \tau. \end{cases} \tag{10.14}$$

- 20% -ное перерегулирование (10.15):

$$\begin{cases} k_p = \dfrac{0{,}6\ T}{\tau\, k_{\text{об}}}, \\ T_{\text{из}} = \tau. \end{cases} \tag{10.15}$$

- для ПИД регулятора:

- без перерегулирования (10.16):

$$\begin{cases} k_p = \dfrac{0{,}6\,T}{\tau\,k_{\text{об}}}, \\ T_{\text{из}} = \tau, \\ T_{\text{п}} = \dfrac{\tau}{2} \end{cases} \tag{10.16}$$

- 20% -ное перерегулирование (10.17):

$$\begin{gathered} k_p = \frac{0{,}95\,T}{\tau\,k_{\text{об}}}, \\ T_{\text{из}} = 1{,}4\tau, \\ T_{\text{п}} = 0{,}47\tau \end{gathered} \tag{10.17}$$

Формулы, приведенные выше, подходят для передаточных функций регуляторов (10.1) и (10.2), а также к модели объекта в виде апериодического звена 1 порядка с запаздыванием.

Регулирование по отклику на внешние возмущения

Приближённые значения настроек для регуляторов по отклику на внешние возмущения будут определяться по следующим формулам:

- для ПИ регулятора:

- без перерегулирования (10.18):

$$\begin{cases} k_p = \dfrac{0{,}6T}{\tau\,k_{\text{об}}}, \\ T_{\text{из}} = 4\tau. \end{cases} \tag{10.18}$$

20% -ное перерегулирование (10.19):

$$\begin{cases} k_p = \dfrac{0{,}7\,T}{\tau\,k_{\text{об}}}, \\ T_{\text{из}} = 2{,}3\tau. \end{cases} \tag{10.19}$$

- для ПИД регулятора:

- без перерегулирования (10.20):

$$\begin{cases} k_p = \dfrac{0{,}95\,T}{\tau\, k_{\text{об}}}, \\ T_{\text{из}} = 2{,}4\tau, \\ T_{\text{пр}} = 0{,}42\tau \end{cases} \tag{10.20}$$

20% -ное перерегулирование:

$$\begin{cases} k_p = \dfrac{1{,}2\,T}{\tau\, k_{\text{об}}}, \\ T_{\text{из}} = 2\tau, \\ T_{\text{п}} = 0{,}42\tau \end{cases} \tag{10.21}$$

Формулы, приведенные выше, подходят для передаточных функций регуляторов (10.1) и (10.2), а также к модели объекта в виде апериодического звена 1 порядка с запаздыванием.

10.4 Метод Коэна-Куна

Коэн и Кун, выполняя исследования метода настройки Зиглера - Никольса, обратили внимание на чрезвычайную чувствительность переходного процесса на выходе системы y(t) от отношения времени запаздывания к постоянной времени объекта регулирования τ/T.

Чтобы обойти это ограничение, Коэн и Кун выполнили дополнительные исследования, чтобы найти настройки регулятора для такой же модели, но такие, чтобы они давали меньшую зависимость от отношения постоянной запаздывания к постоянной времени объекта регулирования.

Также как и в предыдущих методах, этот метод базируется на анализе реакции системы на ступенчатое воздействие. В методе Коэна и Куна объект регулирования также аппроксимируется апериодическим звеном 1-ого порядка и звеном чистого (транспортного) запаздывания. Однако определяются эти параметры для модели аппроксимации по методике отличной той, которая изложена в начале работы. Описание метода аппроксимации модели по методу Коэна-Куна можно найти на сайте Мичиганского технологического университета. Далее будут приведены адаптированные к методу Ормана формулы определения настроек регулятора.

Отношение времени запаздывания к постоянной времени $r = \dfrac{\tau}{T}$.

- для ПИ-регулятора (10.22):

$$\begin{cases} K_p = \dfrac{1}{K \cdot r}\left(0.9 + \dfrac{r}{12}\right), \\ T_{iz} = \tau\left(\dfrac{30 + 3r}{9 + 20r}\right). \end{cases} \quad (10.22)$$

- для ПИД-регулятора (10.23):

$$\begin{cases} K_p = \dfrac{1}{K \cdot r}\left(\dfrac{4}{3} + \dfrac{r}{4}\right), \\ T_{iz} = \tau\left(\dfrac{32 + 6r}{13 + 8r}\right), \\ T_{pr} = \tau\left(\dfrac{4}{11 + 2r}\right). \end{cases} \quad (10.23)$$

10.5 Метод ВТИ

В СССР наиболее длительную проверку временем выдержали формулы настройки типовых регуляторов предложенные Всесоюзным (в настоящее время Всероссийским) теплотехническим институтом (ВТИ).

Формулы ВТИ как и формулы предыдущих методов являются приближёнными и согласно разработчикам метода должны обеспечивать минимум квадратичного критерия качества со степенью затухания $\psi = 0{,}75$. Формулы получены для различных соотношений времени запаздывания объекта регулирования к постоянной времени объекта τ/T.

- для ПИ-регулятора:

В случае, когда $0 < \tau/T \leq 0{,}2$, расчетные формулы (10.24):

$$\begin{cases} k_{\text{р}} = \dfrac{0{,}6T}{k\tau} \\ T_{\text{из}} = 3{,}3\tau \end{cases} \quad (10.24)$$

В случае, когда $0{,}2 < \tau/T \leq 1{,}5$, расчетные формулы (10.25):

$$\begin{cases} k_{\text{р}} = \dfrac{0{,}38(\tau + 0{,}6T)}{k(\tau - 0{,}08T)} \\ T_{\text{из}} = 0{,}8T \end{cases} \quad (10.25)$$

В случае, когда τ/T>1,5, расчетные формулы (10.26):

$$\begin{cases} k_{\text{р}} = \dfrac{1}{2\ k_{\text{об}}} \\ T_{\text{из}} = 0{,}6\tau \end{cases} \tag{10.26}$$

- для ПИД-регулятора:

В случае, когда $0 < \tau/T \leq 0{,}2$, расчетные формулы (10.27):

$$\begin{cases} k_{\text{р}} = \dfrac{T}{k\tau} \\ T_{\text{из}} = 2{,}5\tau \\ T_{\text{пр}} = 0{,}2T_{\text{из}} \end{cases} \tag{10.27}$$

В случае, когда $0{,}2 < \tau/T \leq 1{,}5$, расчетные формулы (10.28):

$$\begin{cases} k_{\text{р}} = \dfrac{0{,}22(\tau + 1{,}5T)}{k(\tau - 0{,}13T)} \\ T_{\text{из}} = 0{,}45T \\ T_{\text{пр}} = 0{,}2T_{\text{из}} \end{cases} \tag{10.28}$$

В случае, когда τ/T>1,5, расчетные формулы (10.29):

$$\begin{cases} k_{\text{р}} = \dfrac{1}{1{,}7\ k_{\text{об}}} \\ T_{\text{из}} = 0{,}7\tau \\ T_{\text{пр}} = 0{,}2T_{\text{из}} \end{cases} \tag{10.29}$$

В формулах ВТИ используется аппроксимация объекта регулирования апериодическим звеном 1-го порядка и звеном чистого (транспортного) запаздывания.

10.6 Метод Копеловича для астатических объектов управления

Для случая, при котором передаточная функция объекта управления имеет вид (10.30):

$$W(s) = \frac{1}{\overline{T} \cdot s} e^{-\tau \cdot s} = \frac{\overline{K}}{s} e^{-\tau \cdot s}, \tag{10.30}$$

предусмотрен особый метод настройки регуляторов, расчетные формулы для которого приведены в таблице 10.1.

Таблица 10.1 – Расчетные формулы для астатических объектов

	Апериодический процесс (**σ = 0**)	Процесс с **σ = 20%**	Процесс с минимальной квадратичной площадью отклонения
П - регулятор	$K_p = \frac{0.4}{\tau / T}$	$K_p = \frac{0.7}{\tau / T}$	-
ПИ - регулятор	$K_p = \frac{0.4}{\tau / T};$ $T_{iz} = 6\tau$	$K_p = \frac{0.7}{\tau / T};$ $T_{iz} = 3\tau$	$K_p = \frac{1}{\tau / T};$ $T_{iz} = 4\tau$
ПИД - регулятор	$K_p = \frac{0.6}{\tau / T};$ $T_{iz} = 5\tau$ $T_{pr} = 0.2\tau$	$K_p = \frac{1.1}{\tau / T};$ $T_{iz} = 2\tau$ $T_{pr} = 0.4\tau$	$K_p = \frac{1.4}{\tau / T};$ $T_{iz} = 1.6\tau$ $T_{pr} = 0.5\tau$

Как видно из вышеприведенной таблицы, в данном методе предусмотрен расчет для 3 типовых переходных процессов, а также форма регуляторов с зависимыми настройками.

11 Показатели качества переходных процессов в аналоговых САР

Качество САР определяется совокупностью свойств, обеспечивающих эффективное функционирование как самого объекта управления, так и управляющего устройства, т.е. всей системы управления в целом. Свойства, составляющие эту совокупность и имеющие количественные измерители, называются показателями качества системы управления.

При определении показателей качества в переходном режиме в качестве типового воздействия используется ступенчатое возмущение $g = k \times 1(t)$. Характер переходного процесса не зависит от величины k. Реакция системы $y(t)$ на входное воздействие g пропорциональна переходной функции $h(t)$, являющейся реакцией системы на единичное ступенчатое воздействие $1(t)$: $y(t) = k \times h(t)$.

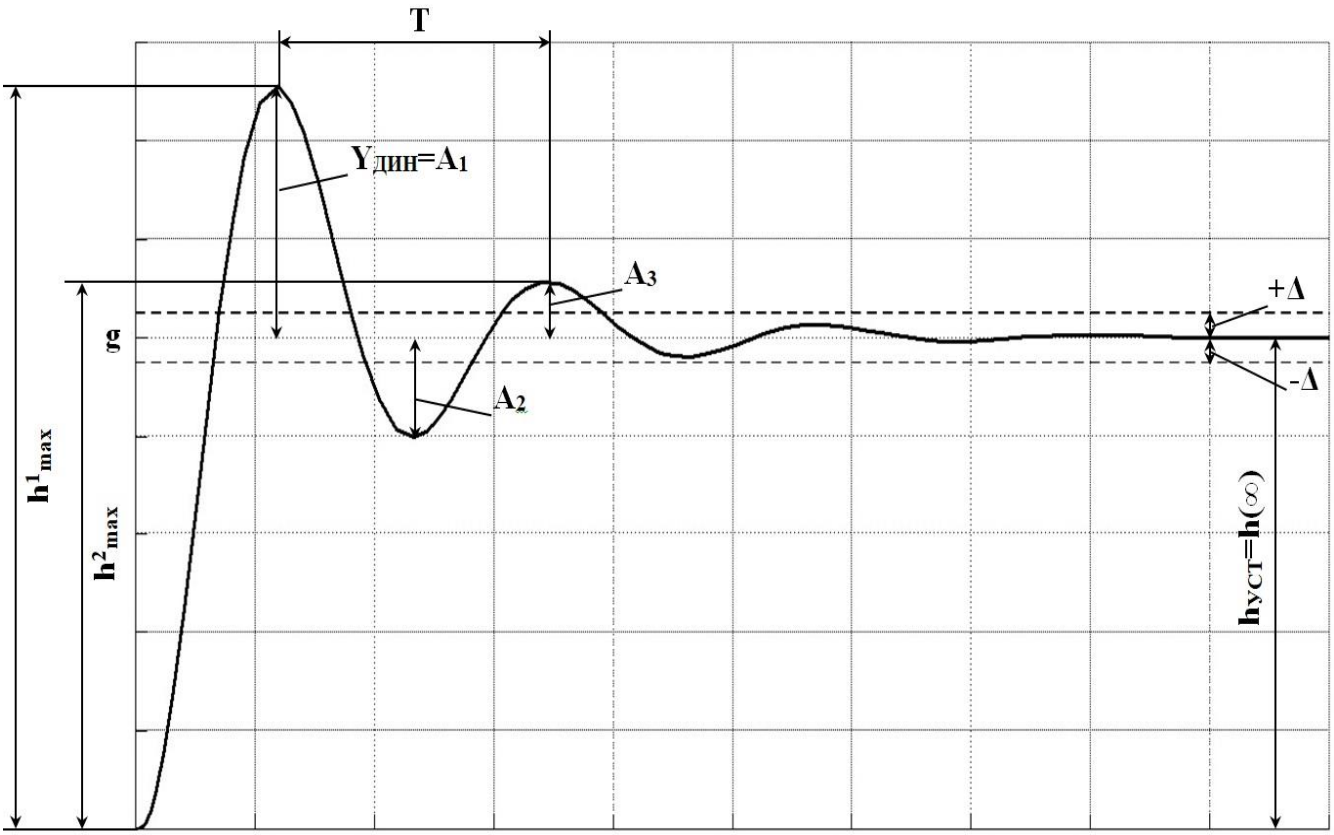


Рисунок 11.1 – График переходного процесса при возмущении по заданию

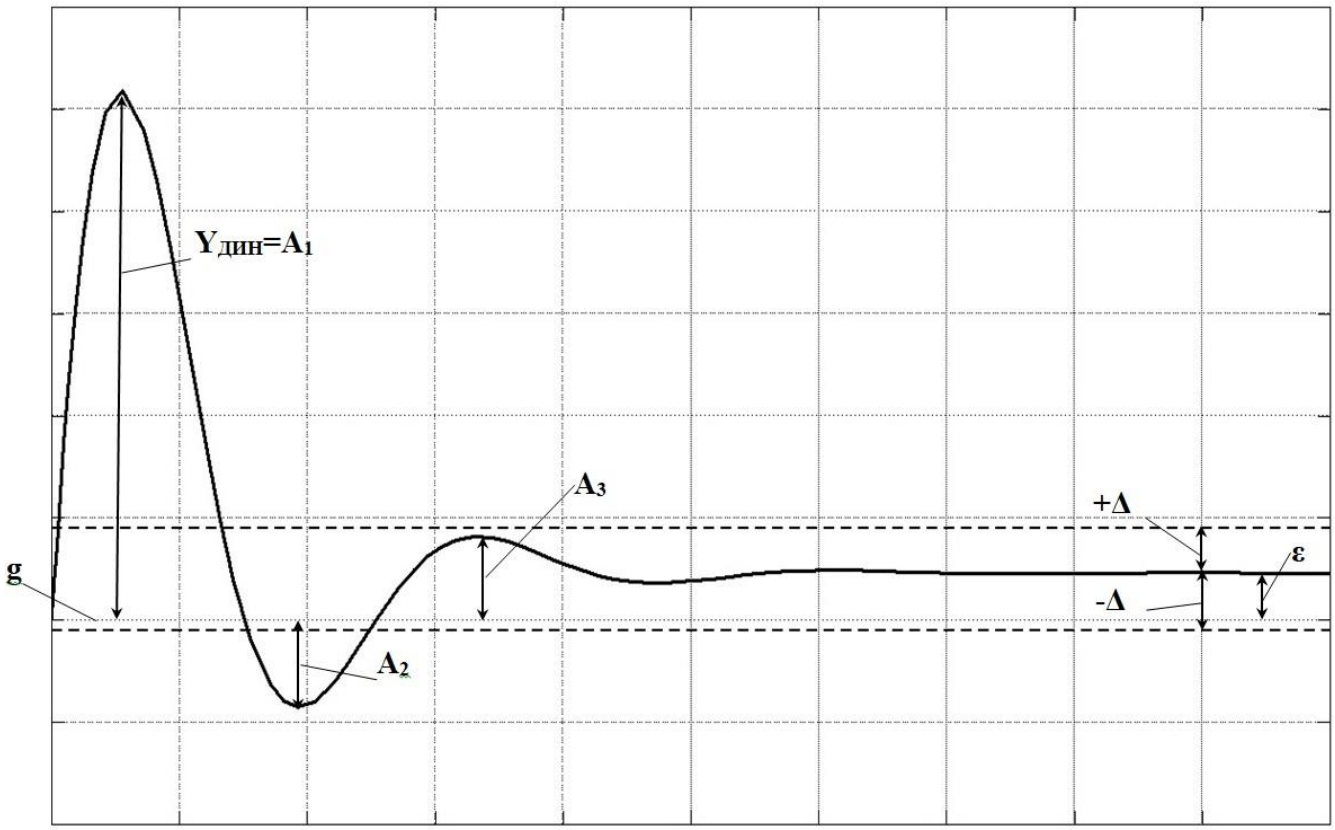


Рисунок 11.2 – График переходного процесса при действии внешнего возмущения

Прямыми показателями качества называются показатели, которые получаются непосредственно по переходной характеристике. Из прямых показателей качества наиболее часто используют:

1. статическая ошибка регулирования;
2. максимальная динамическая ошибка регулирования;
3. время регулирования;
4. перерегулирование;
5. декремент затухания;
6. степень затухания;
7. частота колебаний;
8. интегральный квадратичный показатель качества переходного процесса.

Статическая ошибка регулирования – разница между заданным и реальным значением выходного сигнала при времени, стремящемся к бесконечности. В астатических системах статическая ошибка равна нулю.

$$\varepsilon(\infty) = g - h(\infty) \tag{11.1}$$

Максимальная динамическая ошибка регулирования равна первому максимальному отклонению регулируемой переменной от заданного значения.

$$Y_{\text{ДИН}} = max(|g - h(t)|) \tag{11.2}$$

Время регулирования (время переходного процесса) характеризует быстродействие системы и определяется как минимальное время от момента приложения ступенчатого воздействия до момента, когда отклонение выходной величины $h(t)$ от ее нового установившегося значения $h(\infty)$ становится меньше некоторой заданной величины Δ.

$$t_P = min(|h(t) - h(\infty)| \leq \Delta) \tag{11.3}$$

где Δ – заданная малая постоянная величина, представляющая собой допустимую ошибку, составляющую 1-5% значения входного ступенчатого воздействия.

Для определения времени регулирования по переходной характеристики нужно провести по обе стороны от прямой $h(t) = h(\infty)$ на расстоянии Δ параллельные ей прямые. И время регулирования определяется как время t_P, когда переходная характеристика в последний раз пересекает любую из проведенных прямых.

Склонность системы к колебаниям, а следовательно, и запас устойчивости могут быть охарактеризованы **перерегулированием**. Перерегулированием называют относительную величину максимального отклонения управляемой величины h_{max} от установившегося значения $h(\infty)$ в переходном процессе, выраженное в процентах:

$$\sigma = \frac{h^1_{max} - h(\infty)}{h(\infty)} \times 100\% = \frac{Y_{\text{ДИН}}}{h(\infty)} \times 100\% = \frac{A_1}{h(\infty)} \times 100\% \tag{11.4}$$

Для переходных процессов, вызванных возмущающим воздействием на входе объекта регулирования, перерегулирование определяется как отношение второго (отрицательного) максимального отклонения A_2 к первому максимальному отклонению A_1, выраженное в процентах:

$$\sigma = \frac{A_2}{A_1} \times 100\% \tag{11.5}$$

В большинстве случаев считается, что запас устойчивости является достаточным, если величина перерегулирования не превышает 10–30%. Однако в некоторых случаях требуется, чтобы переходный процесс протекал вообще без перерегулирования, т.е. был монотонным; в ряде других случаев может допускаться перерегулирование 50–70%.

Декремент затухания равен отношению двух смежных перерегулирований:

$$\varkappa = \frac{h_{max}^1 - h(\infty)}{h_{max}^2 - h(\infty)} = \frac{A_1}{A_3} \tag{11.6}$$

Для систем автоматического регулирования второго порядка характеризует быстроту затухания переходного процесса.

Степень затухания переходного процесса определяется из соотношения:

$$\psi = \frac{A_1 - A_3}{A_1} = 1 - \frac{A_3}{A_1} \tag{11.7}$$

Значения ψ изменяются от 0 до 1. Интенсивность затухания колебаний в системе считается удовлетворительной, если $\psi = 0{,}75 \div 0{,}95$.

Частота колебаний переходного процесса определяется по выражению:

$$\omega = \frac{2\pi}{T} \tag{11.8}$$

где T – период колебаний для колебательных переходных процессов

При анализе и синтезе систем регулирования с колебательными свойствами наиболее широко используется **интегральный квадратичный показатель качества** переходного процесса, который равен площади под кривой $\varepsilon^2(t)$.

$$J = \int_0^\infty \varepsilon^2(t)dt \tag{11.9}$$

Квадратичная оценка учитывает величину и длительность отклонений. Однако из-за возведения ошибки регулирования $\varepsilon(t)$ в квадрат первые (большие) отклонения приобретают в конечном значении интеграла существенно больший вес, чем последующие (малые) отклонения. Поэтому минимальные значения оценки J всегда соответствуют колебательным процессам с малым затуханием.

12 Описание лабораторной установки и её составных частей

Для проведения лабораторных работ используется экспериментальная установка, схема которой представлена на рисунке 12.1. Установка состоит из вертикально расположенной трубы, по которой при помощи вентилятора пропускается воздух, нагреваемый ТЭНом. Температура воздуха на выходе трубы измеряется при помощи термометра сопротивления с диапазоном измерения -50…200 °С. Управление осуществляется регулированием мощности ТЭНа, а изменение нагрузки – при помощи заслонки, регулирующей количество подаваемого воздуха.

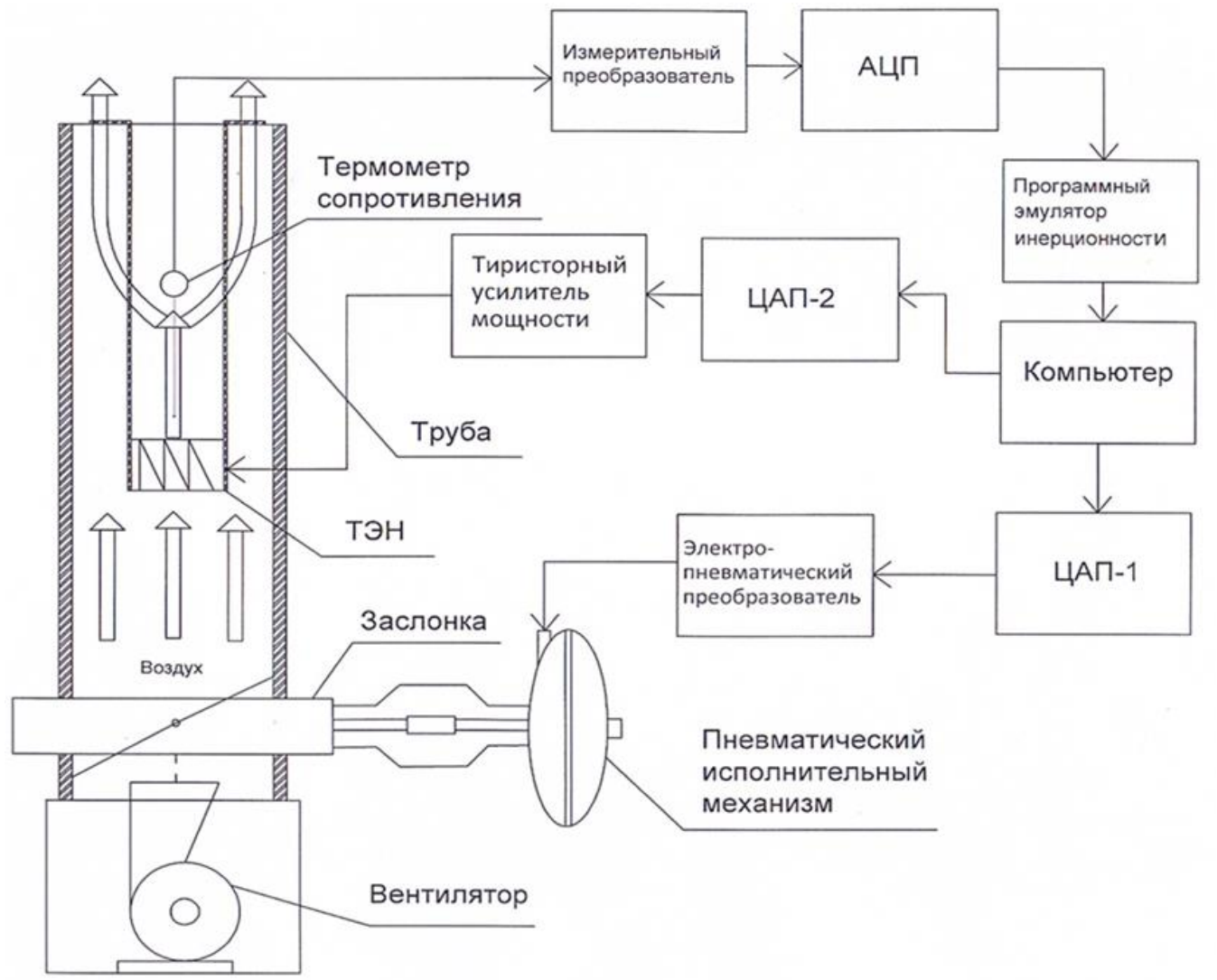


Рисунок 12.1 – Схема установки

Рассмотрим более подробно работу лабораторной установки. Сигнал от платинового термометра сопротивления поступает на измерительный преобразователь ИПМ 0104, который преобразовывает его в унифицированный сигнал постоянного тока 4...20 мА.

Сигнал с измерительного преобразователя поступает на аналого-цифровой преобразователь (АЦП), который преобразует входной аналоговый в цифровой сигнал. Сигнал с АЦП поступает на программный эмулятор инерционности, который служит для имитации изменения инерционности объекта. Далее сигнал поступает на компьютер. В зависимости от температуры с компьютера подается сигнал на цифро-аналоговый преобразователь (АЦП) – 2, который преобразует цифровой сигнал в аналоговый. Аналоговый сигнал с ЦАП-2 поступает на тиристорный усилитель мощности, который регулирует мощность ТЭНа.

Помимо управления в данной установке предусмотрена **возможность изменения нагрузки.** Изменение нагрузки осуществляется путём подачи с компьютера цифрового сигнала на ЦАП-1. Аналоговый сигнал с ЦАП-1 поступает на электропневматический преобразователь, преобразовывающий унифицированный непрерывный сигнал постоянного тока в унифицированный пропорциональный пневматический непрерывный сигнал (величиной 0,2-1кгс/см^2.), который поступает на исполнительный механизм.

В качестве **нагревательного элемента** в установке использовался ТЭН, состоящий из керамической трубы диаметром 60 мм и длиной 200мм, в которой через просверленные отверстия продета спираль. Мощность данной спирали составляет 800 Вт. Крепление нагревательного элемента в центре трубы осуществляется шпильками, которые в свою очередь являются токоведущими частями для подачи напряжения на спираль. Нагревательный элемент в форме трубки позволяет воздуху более эффективно его обтекать.

Регулирующая заслонка представляет собой цилиндрическую конструкцию, которая состоит из верхнего и нижнего фланцев, одной подвижной решетки и двух неподвижных. В нижнем фланце проделано квадратное отверстие для подачи воздуха от центробежного вентилятора. В верхнем сделано отверстие, в которое вставляется труба с уплотнителем.

Элемент, перекрывающий доступ воздуха в верхнюю часть, выполнен в виде подвижной решетки, которая двигается в каркасе сверху и снизу которого расположены неподвижные конструкции решетчатого типа. Подобная конструкция заслонки обеспечивает возможность линейно изменять подачу воздуха. Рабочий ход заслонки от положения «закрыто» до положения «открыто» составляет 12 мм.

В установке используется **промышленный центробежный вентилятор** на базе микромотора УАД 72ф со следующими характеристиками: напряжение питания – 220 В; Производительность $500 м^3/ч$; частота вращения – 2750/2700 $мин^{-1}$; номинальная мощность – 50.0/70.0 Вт.

Корпус вентилятора выполнен из пластмассы. Основание железное, в котором предусмотрены отверстия для крепления. В них установлены резиновые амортизаторы. Всасывающее отверстие закрыто набором из пластмассовых сеток, что позволяет предотвратить попадание в вентилятор посторонних предметов.

Термометр сопротивления — датчик, предназначенный для измерения температуры, сопротивление чувствительного элемента которого зависит от температуры. Один из самых распространенных типов термометров сопротивления — платиновые термометры. Это объясняется тем, что платина имеет высокую стойкость к окислению и значительный температурный коэффициент сопротивления.

Преимущества термометров сопротивления: высокая точность измерений, практически линейная характеристика, возможность исключения влияния изменения сопротивления линий связи на результат измерения при использовании 3-х или 4-х проводной схемы измерений.

Недостатки термометров сопротивления: малый диапазон измерений (по сравнению с термопарами), не могут измерять высокую температуру (по сравнению с термопарами).

При работе с установкой использовался, представленный на рисунке 12.2 термометр сопротивления Pt-100, поскольку не требуется большой диапазон температур и термометр сопротивления дает более высокую точность измерения, чем термопара.

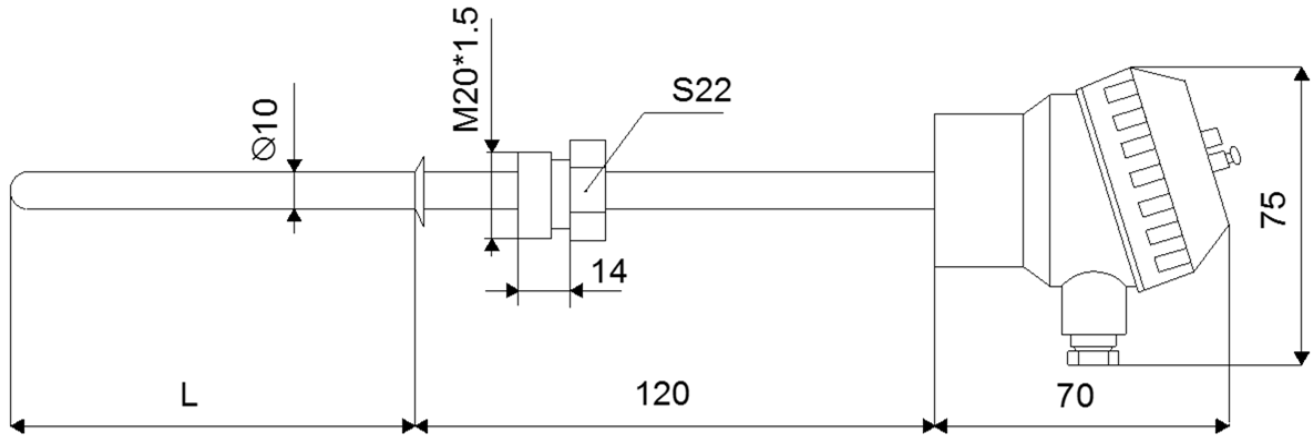


Рисунок 12.2 – Термометр сопротивления Pt100

Измерительный преобразователь — техническое средство с нормируемыми метрологическими характеристиками, которое служит для преобразования измеряемой величины в другую величину или измерительный сигнал, который крайне удобен для обработки, хранения, дальнейших преобразований, индикации и передачи, однако непосредственно не воспринимаемый оператором.

Классификация измерительных преобразователей по характеру преобразования:

1) Аналоговый измерительный преобразователь — измерительный преобразователь, преобразующий одну аналоговую величину (аналоговый измерительный сигнал) в другую аналоговую величину (измерительный сигнал);

2) Аналого-цифровой измерительный преобразователь — измерительный преобразователь, предназначенный для преобразования аналогового измерительного сигнала в цифровой код;

3) Цифро-аналоговый измерительный преобразователь — измерительный преобразователь, предназначенный для преобразования числового кода в аналоговую величину.

Классификация измерительных преобразователей по месту в измерительной цепи:

1) Первичный измерительный преобразователь — измерительный преобразователь, на который непосредственно воздействует измеряемая физическая величина;

2) Датчик — конструктивно обособленный первичный измерительный преобразователь;

3) Детектор — датчик в области измерений ионизирующих излучений;

4) Промежуточный измерительный преобразователь — измерительный преобразователь, занимающий место в измерительной цепи после первичного преобразователя.

По принципу действия измерительные преобразователи делятся на генераторные и параметрические.

В представленной лабораторной установке используется **измерительный модульный преобразователь** модели ИПМ 0104 со следующими характеристиками: один входной канал, один (4...20 мА) или два (0...5 и 4...20 мА) выходных канала, напряжение питания 24 или 36 В.

Одноканальные блоки питания БП60 выполнены по схеме однотактного обратноходового преобразователя напряжения и являются импульсными по принципу действия. Блоки питания так же имеют фильтр радиопомех на входе и гальваническую развязку между входом и выходом. Выходное напряжение стабилизируется с помощью отрицательной обратной связи.

Особенности: преобразование переменного (постоянного) напряжения в постоянное стабилизированное напряжение, ограничение пускового тока, защита от перенапряжения и импульсных помех на входе, защита от короткого замыкания и перегрева, защита от перегрузки, регулировка выходного напряжения с помощью внутреннего подстроечного резистора в диапазоне ± 8 % от номинального выходного напряжения с сохранением мощности, индикация о наличии напряжения на выходе.

Технические характеристики блока питания БП60: входное напряжение переменного тока от 90 до 264 В, входное напряжение постоянного тока от 110 до 370 В, частота входного переменного напряжения от 47 до 63 Гц, максимальная выходная мощность - 60 Вт, рабочий диапазон температур от –20 до +50 ºС, номинальное выходное напряжение – 24 В.

Электрические предохранители (Автоматы) необходимы для автоматического отключения нагрузки от электрической сети при перегрузках в сети или короткого замыкания.

Электрические предохранители снабжены расцепителем – специальным исполнительным механизмом, который непосредственно осуществляет размыкание электрической цепи. Большинство подобных современных устройств имеют электромагнитный и тепловой расцепитель и могут одновременно защитить и от перегрузки сети, и от короткого замыкания.

Пускатель электромагнитный (магнитный пускатель) — это электромагнитное низковольтное устройство, которое предназначено для пуска и разгона электродвигателя до номинальной скорости, а так же обеспечения его непрерывной работы, отключения питания и защиты электродвигателя от рабочих перегрузок.

Магнитные пускатели имеют магнитную систему, которая состоит из заключенных в пластмассовый корпус якоря и сердечника. Якорь магнитной системы и мостики главных и блокировочных контактов с пружинами собраны траверсе, которая скользит по направляющим верхней части пускателя.

Принцип работы пускателя заключается в следующем: при подаче напряжения на катушку якорь притягивается к сердечнику, нормально-открытые контакты замыкаются, нормально-закрытые размыкаются. При отключении пускателя всё происходит наоборот: под действием возвратных пружин подвижные части возвращаются в исходное положение, при этом главные контакты и нормально-открытые блокконтакты размыкаются, нормально-закрытые блокконтакты замыкаются.

Тиристорный усилитель мощности – это прибор, предназначенный для управления мощностью в электронагревателях и других устройствах. В лабораторной установки применяется тиристорный усилитель мощности типа У13Н со следующими характеристиками: напряжение питания, 220 или 380 В., частота тока -50+1; 60+2 Гц., потребляемая мощность не более - 15 В·А., диапазон изменения напряжения питания, -15...+10 В.

Электропневматический преобразователь – устройство, предназначенное для линейно-пропорционального преобразования электрического сигнала в пневматический.

Принцип действия электропневматического преобразователя основан на методе силовой компенсации, при котором момент, развиваемый катушкой, которая расположена в поле постоянного магнита, пропорциональный входному сигналу, будет компенсироваться моментом силы, развиваемым сильфоном обратной связи. Принципиальная схема приведена на рисунке 12.3.

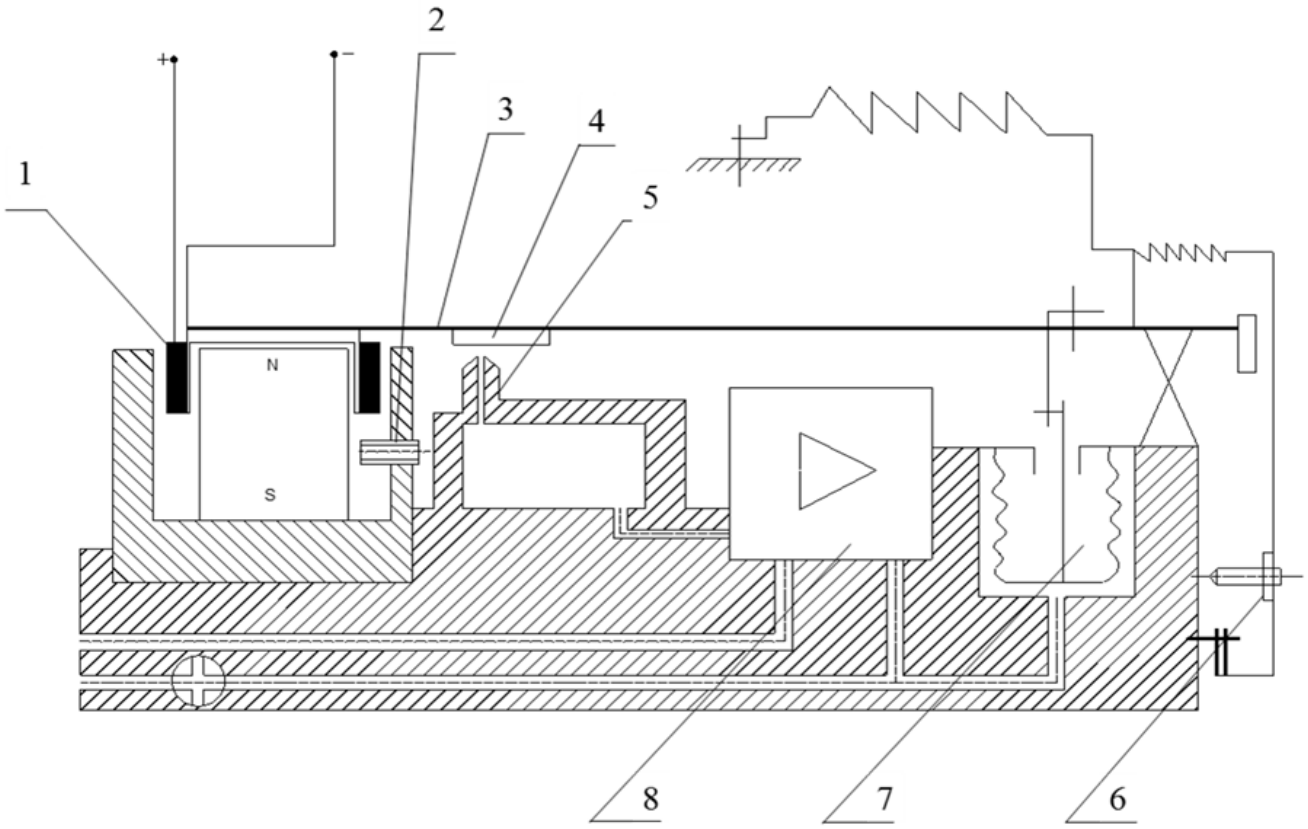


Рисунок 12.3 – Принципиальная схема электропневматического преобразователя

Магнитное поле, возникающее при прохождении тока через катушку 1, взаимодействует с полем постоянного магнита и развивает усилие, которое прямо пропорционально величине входного тока. Под действием этого усилия рычаг 3, поворачиваясь вокруг упругой опоры, вследствие чего изменяется зазор между соплом 5 и заслонкой 4, что приводит к изменению давления воздуха в управляющей камере усилителя 8 до тех пор, пока его выходное давление через сильфон обратной связи 7 не восстановит равновесия на рычаге. Таким образом, реализуется прямо пропорциональная зависимость между входным токовым сигналом I и выходным давлением P преобразователя. Настройка нуля осуществляется вращением винта 6. Шунт 2 необходим для точной настройки диапазона.

В данной лабораторной установке используется **электропневматический преобразователь типа ЕР-РЗ**, выходной пневматический сигнал которого составляет 0,2-1кгс/см2.

Пневматический исполнительный механизм – это устройство, которое служит в качестве исполнительного органа для регулирования температуры, давления и т.д. В нашем случае пневматический исполнительный механизм необходим для имитации нагрузки, путём изменения количества подаваемого воздуха. Схема мембранного исполнительного механизма представлена на рисунке 12.4.

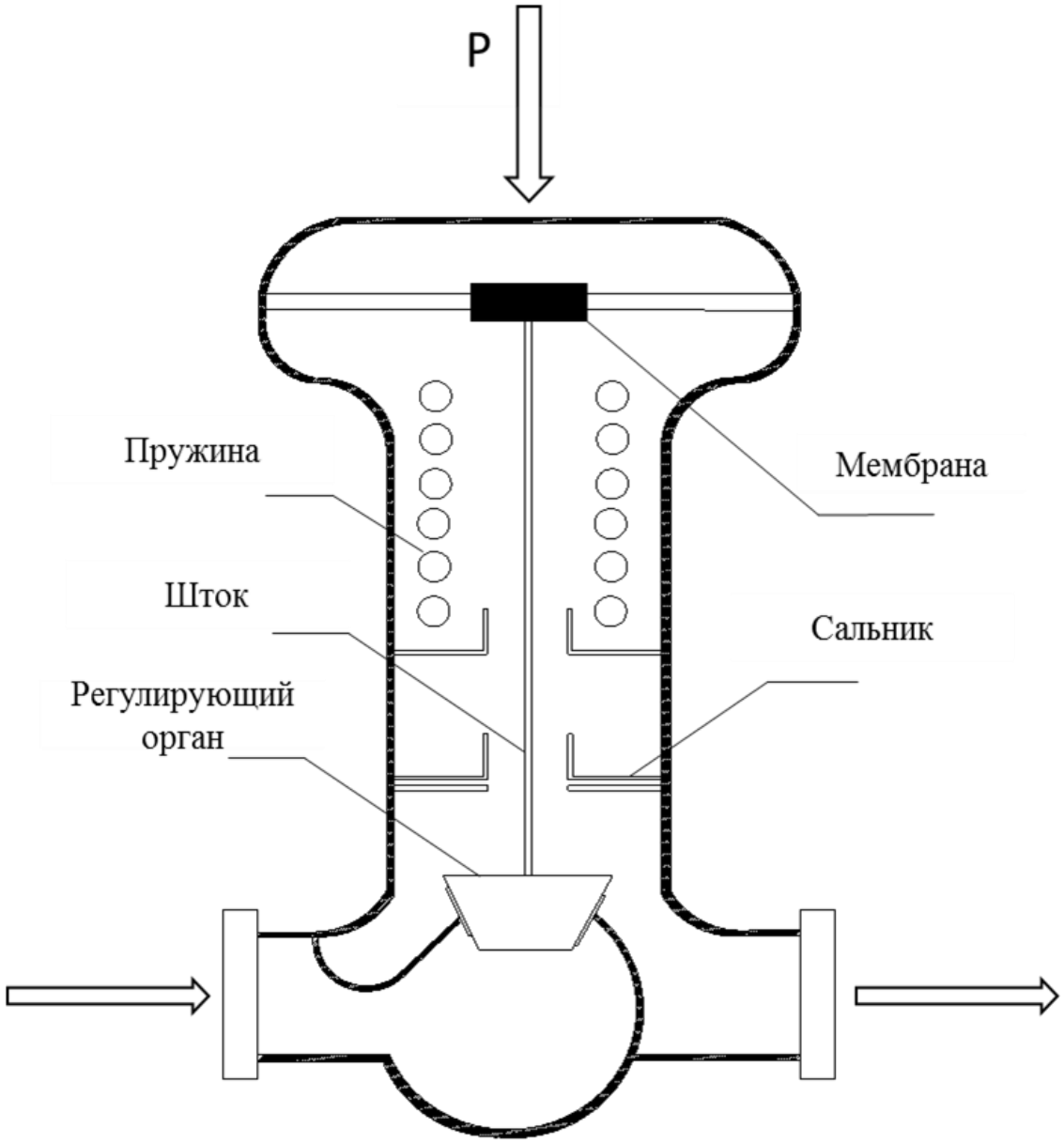


Рисунок 12.4 – Схема мембранного исполнительного механизма

Перемещение выходного штока, который соединен с регулирующим органом, в одну сторону осуществляется силой, создаваемой давлением Р, в другую — усилием пружины. Сигнал Р поступает в герметичную мембранную «головку», в которой находится мембрана из прорезиненной ткани толщиной 2-4 мм с жестким центром. Снизу на мембрану давит пружина. В исполнительном механизме давление управляющего воздуха воздействует на мембрану, зажатую по периметру между крышками привода, и создает усилие, которое уравнивается пружиной.

Плата PCI-1710 является многофункциональной устройством сбора и обработки сигналов, устанавливается на шину PCI. Передовые схемные решения обеспечивают высокое качество и выполнение пяти основных функций измерений

и контроля: цифровой ввод, 12-битное аналого-цифровое преобразование, цифро-аналоговое преобразование, счетчик/таймер и цифровой вывод.

Встроенный в плату буфер FIFO, позволяет хранить до 4000 измеренных значений аналого-цифрового преобразования. Микросхема счетчика: 82C54.

На плате PCI-1710 так же присутствует программируемый счетчик, который служит для генерации импульсов запуска аналого-цифрового преобразования.

Основные характеристики плата сбора данных PCI-1710:

- шесть однопроводных или восемь дифференциальных аналоговых входов,
- программно управляемый коэффициент усиления каждого канала,
- 12-битный АЦП с частотой выборки до 100 кГц,
- встроенный буфер FIFO на 4000 значений,
- автоматический опрос каналов и установка коэффициента усиления,
- шестнадцать цифровых входов и шестнадцать цифровых выходов,
- два 12-битных аналоговых выхода,
- программно управляемый счетчик и схема запуска.

На рисунке 12.5 показана структурная схема этой платы, на рисунке 12.6 – схема подключения установки к плате, на рисунке 12.7 – функциональная схема автоматизации установки и спецификация к ней.

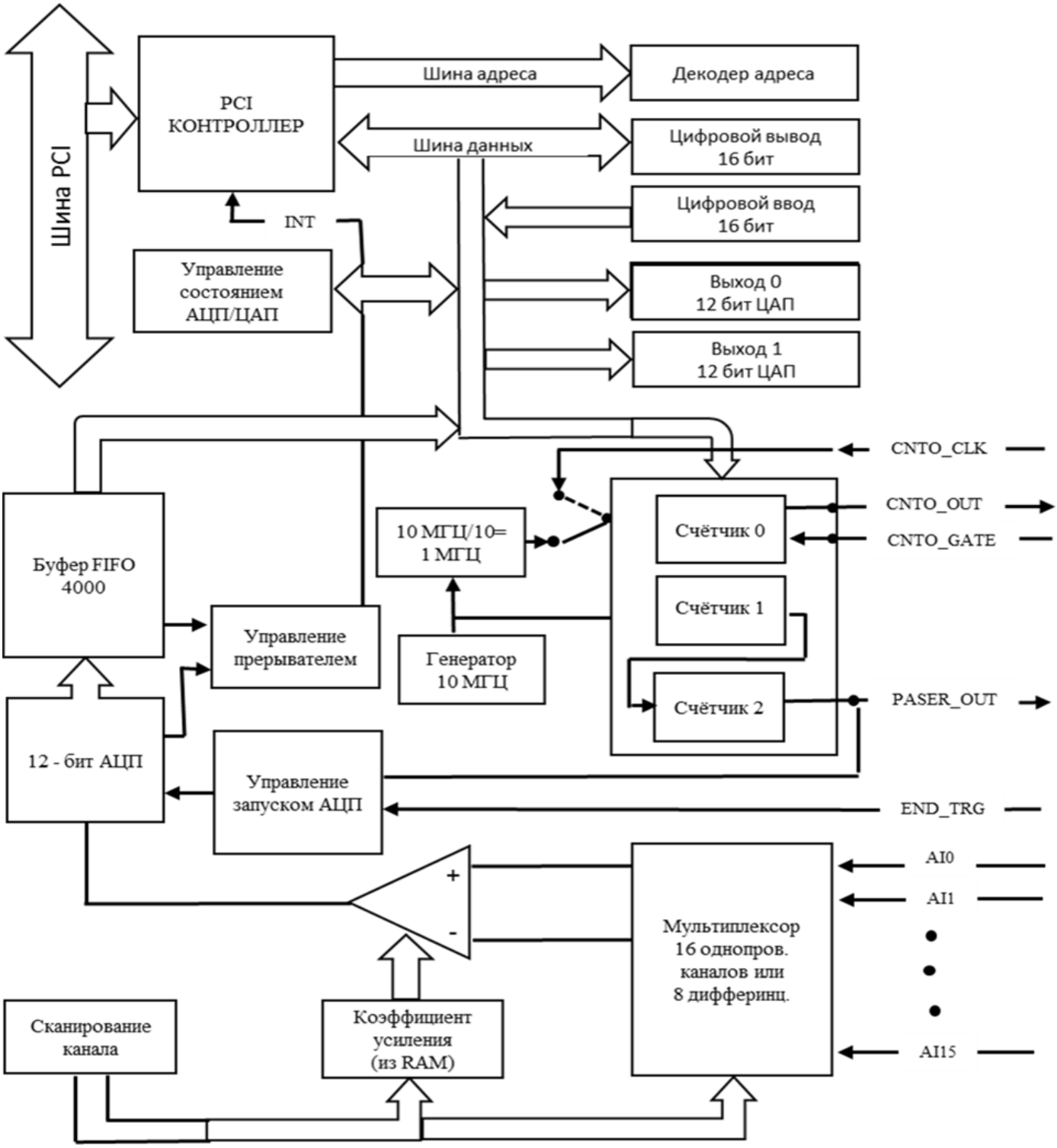
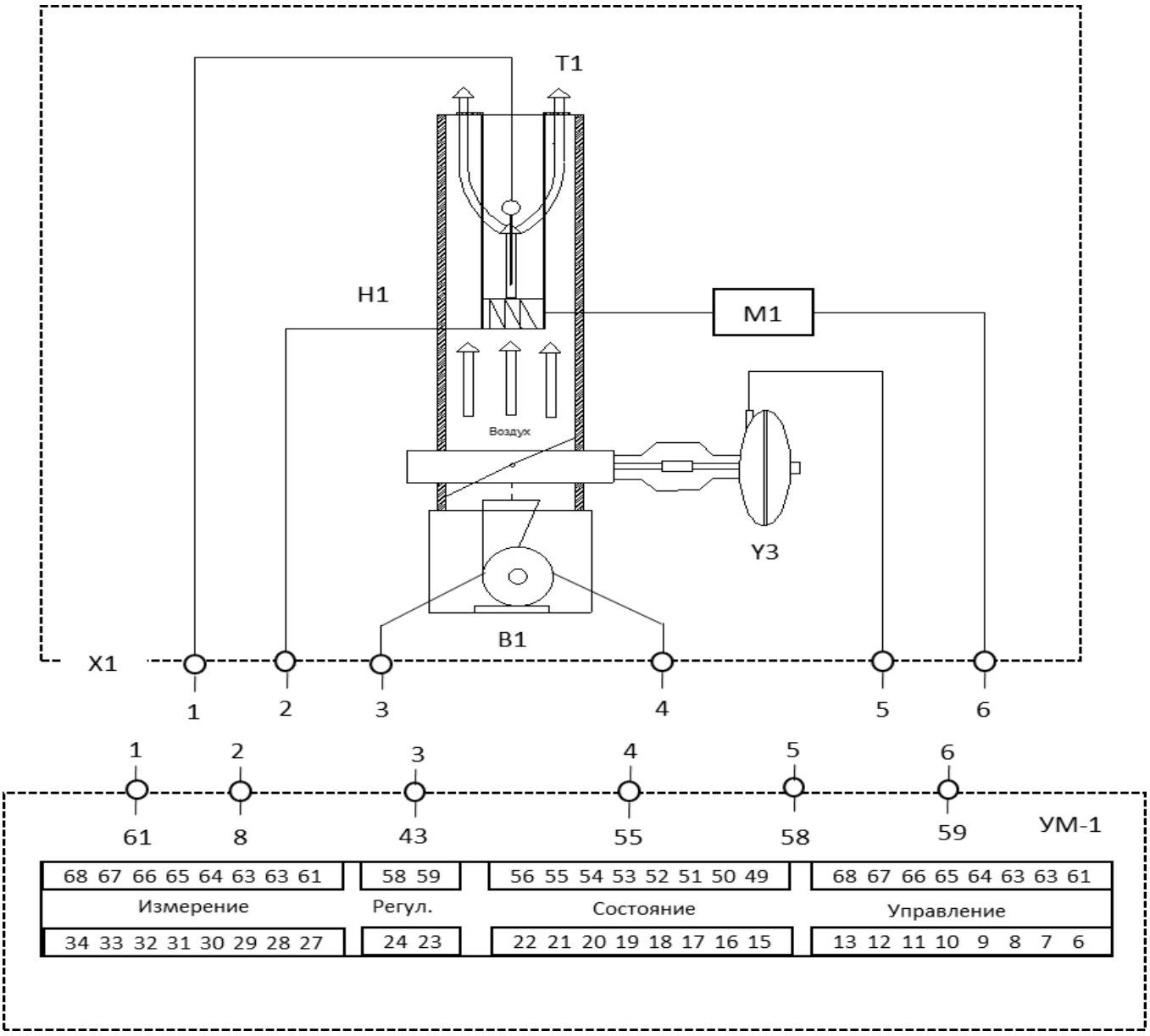


Рисунок 12.5 – Структурная схема платы PCI-1710



Цепь	Назначение	Поз.	Характеристики
1	Измерение Твозд. На выходе из устан.	T1	Измерение от 0 до 45 °С.
2	Управление нагревателем	H1	Включение/выключение
3	Управление вентилятором	B1	Пуск/стоп
4	Контроля состояния вентилятора	B1	Стоит/работает
5	Регулирование положения клапана	Y3	0 – 100 %
6	Регулирование мощности ТЭНа	M1	0 – 10 В.

Рисунок 12.6 – Схема подключения управляющего модуля

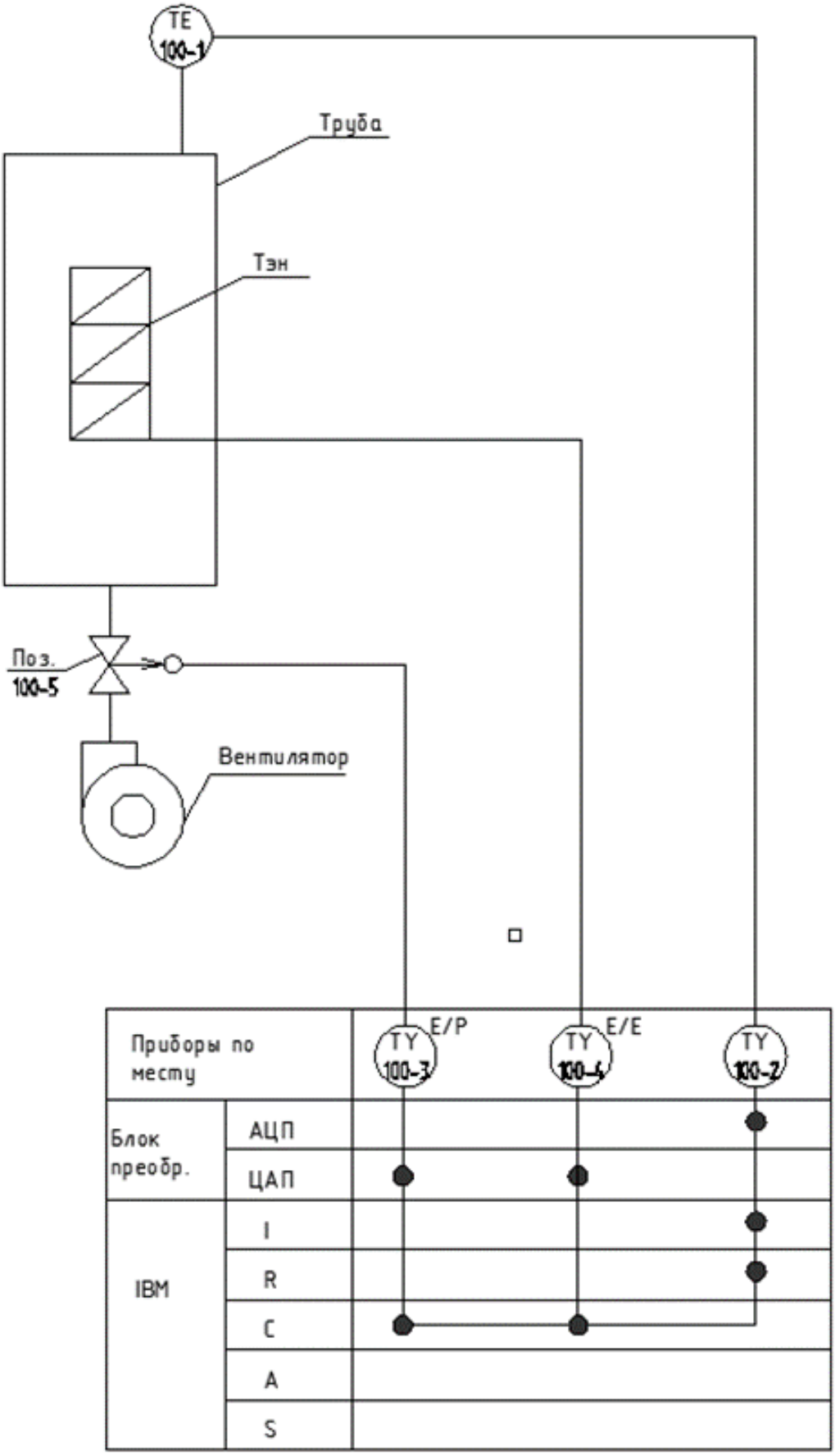


Рисунок 12.7 – Функциональная схема автоматизации установки

В таблице 12.1 приведена спецификация к ФСА (рисунок 12.7).

Таблица 12.1 – Спецификация

Позиция	Наименование техническая характеристкика	Тип марка	Количество	Примеча-ние
1	2	3	4	5
TE 100-1	Термометр сопротивления	Pt100	1	
TY 100-2	Измерительный преобразователь	ИПМ 0104	1	
TY 100-3	Электропневматический преобразователь	ЕР-Р3	1	
TY 100-4	Тиристорный усилитель мощности	У13Н	1	
Поз. 100-5	Регулирующий клапан с МИМ Ру=2 МПа Dy=20 мм Kvy=10 $м^3$/ч Ход=20 мм	“НО”	1	

В таблице 12.2 показаны характеристики оборудования и аппаратуры.

Таблица 12.2 – Характеристики оборудования и аппаратуры

Поз.	Наименование и тип	Шт.	Характеристики
T1	Датчик температуры	1	Диапазон измерения 0 – 100; 50-200 °С.
H1	Нагреватель	1	Нихром 0.5 мм2, ~220 В.
Y3	Клапан регулирующий	1	Ход штока 20 мм, Тзакр. = 3 мин.
B1	Вентилятор	1	1500 об/мин, ~3*127 В.
M1	Тиристор. усил. мощн.	1	Потреб. мощность 15 В*А, ~220 В.

Для проведения лабораторных работ была создана программа на языке C#. Она реализует алгоритмы работы П, ПИ и ПИД регуляторов, а также позволяет сохранять графики переходных процессов. Окно программы представлено на рисунке 12.8, полный исходный код программы приведен приложении 2.

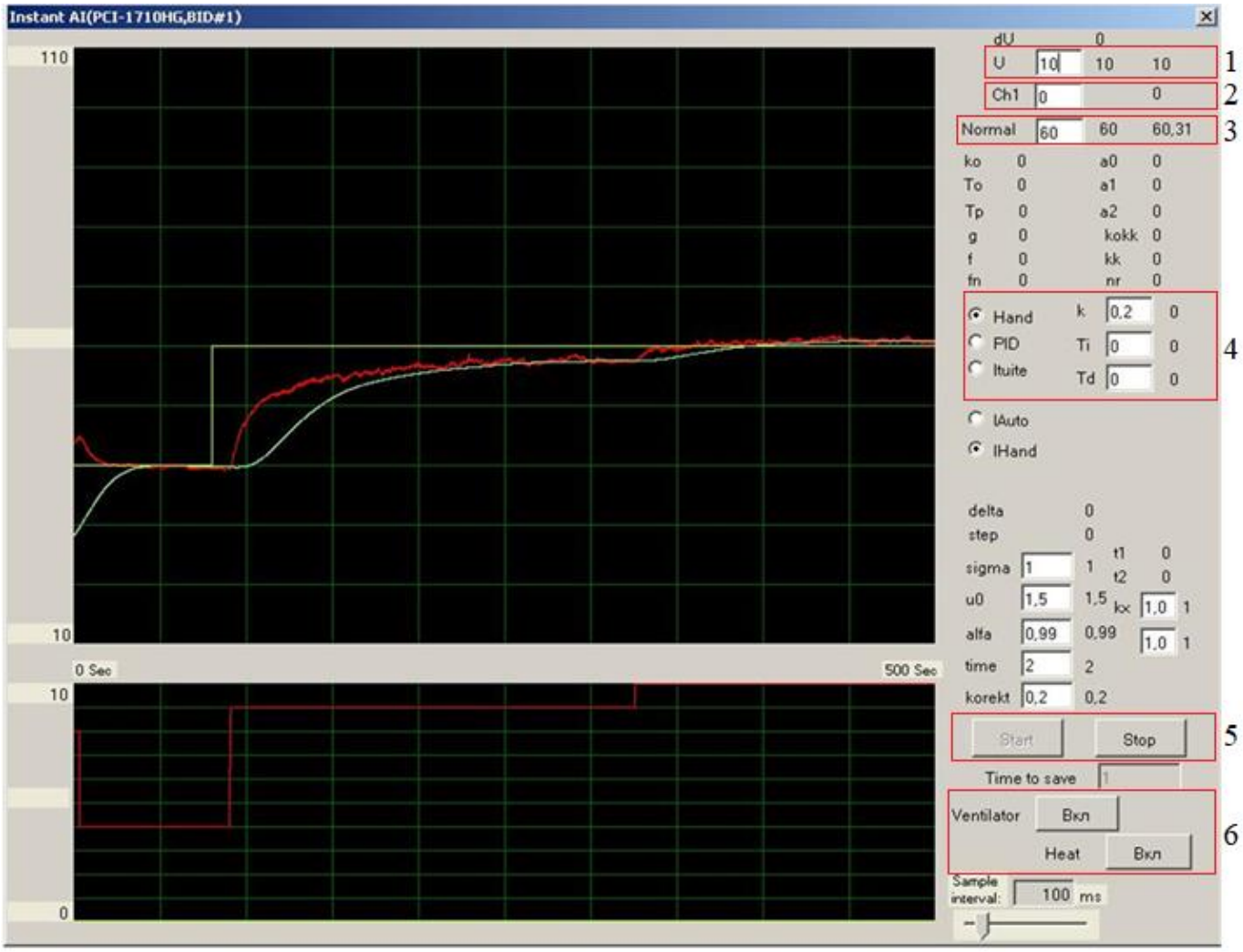


Рисунок 12.8 – Главное окно программы

Разберем каждый элемент программы, необходимый для выполнения лабораторных работ:

1 – окно U, где можно ввести значение управляющего воздействия на объект управления, [В];

2 – окно Ch1;

3 – окно Normal, где можно задать желаемое значение температуры (уставку), [ºС]. Причем, в этой строке отображается как заданное ранее значение, так и текущее значение, регистрируемое датчиком;

4 – окно настройки режима управления. Здесь можно выбрать режим управления (ручной (Hand), автоматический (PID) или интеллектуальный (Ituite)); в случае, если выбран автоматический режим управления, то справа в этой же строке необходимо задать настройки ПИД регулятора. Важно отметить, что настройки регулятора задаются с учетом того, что передаточная функция ПИД регулятора имеет вид (настройки независимые):

$$W_{PID}(s) = K_p + \frac{1}{T_i \cdot s} + T_d \cdot s.$$

Важно! При расчете настроек регулятора по формулам из главы 10 данного пособия учтите, что надо будет пересчитать настройки и в программу подставлять значения:

$$T_i = T_{iz} / K_p ,$$

$$T_d = T_{pr} \cdot K_p .$$

5 – кнопки Start и Stop, позволяют запустить или остановить процесс управления;

6 – кнопки Ventilator и Heat, позволяют включить/выключить вентилятор и нагрев.

ЛАБОРАТОРНАЯ РАБОТА №1

ИЗУЧЕНИЕ СОСТАВНЫХ ЧАСТЕЙ ЛАБОРАТОРНОЙ УСТАНОВКИ

Цель работы: изучить структурную схему лабораторной установки и понять функциональные особенности каждого элемента установки.

Задание:

- изучить параграф 12 этого пособия;
- осмотреть лабораторную установку;
- сопоставить приведенную в пособии ФСА и структурную схему с самой установкой.

Протокол должен содержать:

1. Титульный лист (см. в конце данного пособия);
2. Цель работы;
3. Описание каждого элемента установки;
4. Рисунки 12.1, 12.2, 12.3, 12.4, 12.5, 12.6;
5. Выводы по работе.

Вопросы к защите лабораторной работы:

1. Принцип работы лабораторной установки.
2. Как реализована возможность изменения нагрузки?
3. Нагревательный элемент в установке, его описание и параметры.
4. Регулирующая заслонка, ее конструкция и параметры.
5. Центробежный вентилятор – описание, параметры, предназначение.
6. Какая технологическая переменная является управляемой? Каким датчиком ее измеряют в установке и почему?
7. Какой измерительный преобразователь применен в установке? Классификация.
8. Блок питания в установке, его характеристики.
9. Магнитный пускатель. Принцип его работы, назначение.
10. Усилитель мощности.
11. Электропневматический преобразователь, его принципиальная схема.
12. Исполнительный механизм – схема используемого в установке, классификация.
13. Устройство сбора и обработки сигналов (плата PCI-1710).
14. Схема подключения управляющего модуля.

ЛАБОРАТОРНАЯ РАБОТА №2

ОПРЕДЕЛЕНИЕ СТАТИЧЕСКОЙ ХАРАКТЕРИСТИКИ ОБЪЕКТА УПРАВЛЕНИЯ

Цель работы: получение навыков экспериментального определения статической характеристики объекта управления.

Задание:

1. В окне программы нажать кнопку Start.
2. Нажать кнопку Ventilator.
3. Нажать кнопку Heat. Загорится красная лампочка.
4. В окно U ввести значение 1В.
5. Подождать пока температура объекта установится.
6. Повторить пункты 4 и 5, изменяя напряжение (U) с шагом от 1В до 10В.
7. Выключить установку (п.3, п.2, stop).
8. Выйти из программы.
9. Построить график зависимости температуры T от управляющего воздействия U.

Протокол должен содержать:

1. Титульный лист (см. в конце данного пособия);
2. Цель работы;
3. График статической характеристики объекта управления;
4. Выводы по работе.

Вопросы к защите лабораторной работы:

1. Определение статической характеристики объекта управления.
2. Что можно определить по графику статической характеристики?
3. Самовыравнивание, коэффициент усиления.
4. Объекты статические и астатические – вид переходных характеристик, понятие.
5. Порядок действий при снятии статической характеристики.
6. Элементная база систем управления. Краткая классификация.
7. Функциональные схемы автоматизации. Условные обозначения.
8. Классификация систем управления по типу управляющего устройства.
9. Замкнутые и разомкнутые САР. Примеры функциональных схем, описание.
10. Структурная схема замкнутой САР. Описание каждого элемента.

ЛАБОРАТОРНАЯ РАБОТА №3

ОПРЕДЕЛЕНИЕ ДИНАМИЧЕСКИХ ХАРАКТЕРИСТИК ОБЪЕКТА

Цель работы: используя методы аппроксимации кривых разгона, определить передаточную функцию объекта управления; выявить наличие или отсутствие зависимости значения постоянной времени объекта от управляющего воздействия.

Задание:

1. В окне программы нажать кнопку Start.
2. Нажать кнопку Ventilator.
3. Нажать кнопку Heat. Загорится красная лампочка.
4. В окно U ввести значение 1В.
5. Подождать, пока температура объекта установится.
6. Определить передаточную функцию объекта управления, используя методы аппроксимации кривой разгона (параграф 8 пособия).
7. Повторить пункты 4 и 5, изменяя напряжение (U) с шагом от 1В до 10В.
8. Используя программный эмулятор инерционности, имитировать изменение инерционности объекта.
9. Повторить пункты 4-7.
10. Выключить установку (п.3, п.2, stop).
11. Выйти из программы.
12. Построить графики зависимости постоянной времени объекта управления от управляющего воздействия U.

Протокол должен содержать:

1. Титульный лист (см. в конце данного пособия);
2. Цель работы;
3. Кривые разгона объекта управления, полученные при различных значениях управляющего воздействия;
4. График зависимости постоянной времени объекта управления от управляющего воздействия U.
5. Выводы по работе.

Вопросы к защите лабораторной работы:

1. Переходная характеристика – определение. Типовые возмущающие воздействия на объект.
2. Виды переходных характеристик в зависимости от типа объекта (статический, астатический).
3. Коэффициент усиления объекта – определение, формула. Отличие от коэффициента передачи.
4. Постоянная времени объекта – определение, формула.

5. Время запаздывания объекта – определение, формула.
6. Самовыравнивание – определение, формула.
7. Передаточные функции статического и астатического объекта.
8. Методы аппроксимации кривой разгона – графический метод.
9. Методы аппроксимации кривой разгона – метод Ормана.
10. Типовые звенья – усилительное, интегрирующее, инерционное, колебательное, форсирующее. Передаточные функции, переходные характеристики.

ЛАБОРАТОРНАЯ РАБОТА №4

РАСЧЕТ НАСТРОЕК ПИД РЕГУЛЯТОРА

Цель работы: расчет настроек ПИД регулятора на основании полученной ранее передаточной функции объекта управления; получение графика переходного процесса.

Задание:

1. По формулам Копеловича (таблица 10.1) рассчитать настройки ПИД регулятора. **Учесть требования к расчету**, указанные в 12 параграфе пособия (см. стр. 56)!
2. Запустить установку.
3. Ввести в окне к Tu и Td собственные коэффициенты передачи, время интегрирования и время дифференцирования.
4. Получить график вывода температуры объекта на задание.
5. Выключить установку.
6. Записать файл с расширением .slx.

Протокол должен содержать:

1. Титульный лист (см. в конце данного пособия);
2. Цель работы;
3. Расчет настроек ПИД регулятора;
4. График переходного процесса по заданию.
5. Выводы по работе.

Вопросы к защите лабораторной работы:

1. Определение системы автоматического управления (САУ). Отличие от АСУТП.
2. Классификация САР по методу управления.
3. Классификация САР по характеру использования информации.
4. Классификация САР по результатам работы в установившемся состоянии.
5. Классификация САР по характеру изменения воздействий во времени.
6. Законы регулирования аналоговых САР. П – закон. Достоинства, недостатки.
7. Законы регулирования аналоговых САР. И – закон. Достоинства, недостатки.
8. Законы регулирования аналоговых САР. ПИ – закон. Достоинства, недостатки.
9. Законы регулирования аналоговых САР. ПД – закон. Достоинства, недостатки.
10. Законы регулирования аналоговых САР. ПИД – закон. Достоинства, недостатки.

ЛАБОРАТОРНАЯ РАБОТА №5

ОЦЕНКА ПРЯМЫХ ПОКАЗАТЕЛЕЙ КАЧЕСТВА СИСТЕМ УПРАВЛЕНИЯ

Цель работы: получение графиков переходных процессов при регулировании по заданию и по возмущению; расчет показателей качества регулирования.

Задание:

1. Запустить установку;
2. Применить параметры ПИД регулятора, полученные в предыдущей работе;
3. Получить переходный процесс по заданию;
4. Ввести в систему возмущающее воздействие, используя заслонку;
5. Получить график переходного процесса по возмущению;
6. Рассчитать показатели качества полученных процессов (см. параграф 11).

Протокол должен содержать:

1. Титульный лист (см. в конце данного пособия);
2. Цель работы;
3. Графики переходных процессов по заданию и по возмущению;
4. Рассчитанные показатели качества;
5. Выводы по работе.

Вопросы к защите лабораторной работы:

1. Методы настройки регуляторов – аналитические. Примеры;
2. Методы настройки регуляторов – экспериментальные. Примеры;
3. Статическая ошибка регулирования;
4. Максимальная динамическая ошибка регулирования;
5. Время регулирования;
6. Перерегулирование;
7. Декремент затухания;
8. Степень затухания;
9. Частота колебаний;
10. Интегральный квадратичный показатель качества переходного процесса.

Список использованных источников

1. Проектирование систем автоматизации технологических процессов [Текст] : справ. пособ. / ред. А. С. Клюев. - 2-е изд., перераб. и доп. - М. : Энергоатомиздат, 1990. - 464 с.
2. Дудников Е.Г. Автоматическое регулирование в химической промышленности. М.: Химия, 1987. - 368 с.
3. Обозначения условные приборов и средств автоматизации в схемах. Государственный стандарт 21.208-2013. М.: Стандартинформ, 2015. – 30 с.
4. Chien K. L., Hrones J. A., Reswick J. B. On automatic control of generalized passive systems // Trans. ASME. 1952. Vol. 74. P. 175-185.
5. Ziegler J. G., Nichols N. B. Optimum settings for automatic controllers // Trans. ASME. 1942. Vol. 64. P. 759-768.
6. Метод Коэна-Куна. [Электронный ресурс]. URL: https://www.dataforth.com/tuning-control-loops-for-fast-response.aspx (дата обращения 10.04.2021).

ПРИЛОЖЕНИЕ 1 – ОБРАЗЕЦ ОФОРМЛЕНИЯ ТИТУЛЬНОГО ЛИСТА

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Новомосковский институт (филиал)
федерального государственного образовательного учреждения высшего образования
«Российский химико-технологический университет
имени Д.И. Менделеева»

Кафедра

Автоматизация производственных процессов
Лабораторная работа №___
По дисциплине «_____»

Преподаватель: _____

Выполнил(а): _____
_____

Группа: _____

Выполнение:

Защита:

г. Новомосковск, 202_

ПРИЛОЖЕНИЕ 2 – ИСХОДНЫЙ КОД ПРОГРАММЫ

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Automation.BDaq;

namespace AI_InstantAI
{
  public partial class InstantAIForm : Form
  {
    #region fields

    SimpleGraph m_simpleGraph, u_simpleGraph;
    ListViewItem m_firstItem = new ListViewItem();
    ListViewItem m_secondItem = new ListViewItem();

    double a=32.258, b = -0.85;
    int stringLength = 5;
    public
      double alfa, yf, mtime;

    class TMiddle
    {
      private double real;
      private double count;
      public TMiddle()
      {
        real = 0;
        count = 0;
      }
      public void AddMean(double mean)
      {
        real += mean;
        count++;
      }
      public double getMiddle()
      {
        double middle = real/count;
        count = 0;
        real = 0;
        return (middle);
```

```
    }
  }
  class TDifferential
  {
    private double x0, x1, x2, dt_0, dt_1,lmean, d_0;
    private PerformanceCounter pc;
    bool second;
    public TDifferential()
    {
      x0 = x1 = x2 = 0;
      pc = new PerformanceCounter();
      dt_0 = dt_1 = 0;
      lmean = 0;
      second = false;
    }
    public double Parametr
    {
      set
      {
        x2 = x1;
        x1 = x0;
        x0 = value;
        if (second)
        {
          pc.Stop();
          dt_1 = dt_0;
          dt_0 = pc.Duration;
          if ((dt_1 != 0) && (dt_0 != 0))
          {
            d_0 = dt_0 / dt_1;
            lmean = ((1 + d_0) * (1 + d_0) / d_0 * x1 - ((2 + d_0) * x2) - (x0 /
d_0)) / (dt_1 + dt_0);
          }
        }
        second = true;
        pc.Start();
      }
    }
    public double Mean
    {
      get
      {
        return lmean;
      }
    }
  }
```

```
class TIntegral
{
   private double min, max, lmean,x0,x1;
   private PerformanceCounter pc;
   private bool second;
   public TIntegral(double in_min, double in_max)
   {
      min = in_min;
      max = in_max;
      lmean = 0;
      x0 = x1 = 0;
      second = false;
      pc = new PerformanceCounter();

   }
   public double Parametr
   {
      set
      {
         x1 = x0;
         x0 = value;
         if (second)
         {
            pc.Stop();
            lmean += ((x0 + x1) / 2.0 * pc.Duration);
            if (lmean < min)
               lmean = min;
            if (lmean > max)
               lmean = max;
         }
         second = true;
         pc.Start();
      }
   }
   public double Mean
   {
      get
      {
         return lmean;
      }
   }
}

class TPID
{
```

```
        public double k, ti, td;
        private double k_component, ti_component, td_component, ltask, lparametr, min,
max , lcontrol;
        TIntegral int1;
        TDifferential diff1;
        public TPID (double in_k, double in_ti, double in_td, double in_min, double
in_max)
        {
           k = in_k;
           ti = in_ti;
           td = in_td;
           k_component = ti_component = td_component = 0;
           min = in_min;
           max = in_max;
           int1 = new TIntegral(-10000, 10000);
           diff1 = new TDifferential();
        }
        private void Calculate()
        {
           double e = ltask - lparametr;
           k_component = k * e;
           int1.Parametr = e;
           diff1.Parametr = e;
           if (ti > 0)
           {
              ti_component = int1.Mean / ti;
           }
           else
           {
              ti_component = 0;
           }
           td_component = td*diff1.Mean;
           lcontrol = k_component + ti_component + td_component;
           if (lcontrol > max)
              lcontrol = max;
           if (lcontrol < min)
              lcontrol = min;
        }
        public double Control
        {
           get
           {
              return lcontrol;
           }
        }
        public double Task
```

```
    {
        set
        {
            ltask = value;
            Calculate();
        }
    }
    public double Parametr
    {
        set
        {
            lparametr = value;
            Calculate();
        }
    }
}

class TCicleBuffer
{
    private int maxElements, writeCount, readCount, lcount;
    private double [] buffer;
    //bool begin;

    public TCicleBuffer(int in_max)
    {
        maxElements = in_max;
        buffer = new double[maxElements];
        writeCount = readCount = lcount = 0;
        //begin = false;
        for (int i = 0; i < maxElements; ++i)
        {
            buffer[i] = 0;
        }
    }
    public double mean
    {
        set
        {
            buffer[writeCount]= value;
            writeCount++;
            if (writeCount > maxElements - 1)
                writeCount = 0;
            lcount ++;
        }
    }
    public void maenArray(ref double[] localArray)
```

```
        {
            int localReadCount = writeCount-1;
            if (localReadCount < 0)
            {
                localReadCount = maxElements - 1;
            }
            for (int i = localArray.Length - 1; i >= 0; i--)
            {
                localArray[i] = buffer[localReadCount];
                localReadCount --;
                if (localReadCount < 0)
                {
                    localReadCount = maxElements - 1;
                }
            }
        }
        public int count
        {
            get { return lcount; }
        }
    }

    class TGamma
    {
        const int count = 10000;
        const double minGamma = 0.001, maxGamma = 10;
        private double[] fa, gamma;
        public TGamma()
        {
            fa = new double[count];
            gamma = new double[count];
            for (int i = 0; i < count; ++i)
            {
                gamma[i] = minGamma + ((maxGamma - minGamma)*(double)i/(double)(count-1));
                fa[i] = gamma[i] * Math.Exp(-gamma[i]) / (1 - Math.Exp(-gamma[i]));
            }
        }
        public double gh(double fe)
        {
            double gr=0;

            for (int i = 1; i < count; ++i)
            {
                if ((fe < fa[i - 1]) && (fe > fa[i]))
                {
```

```
                gr = gamma[i - 1] + ((gamma[i] - gamma[i - 1]) / (fa[i - 1] - fa[i]) * (fe - fa[i]));
                break;
            }
        }
        return gr;
    }

}

class TGamma2
{
    const int count = 10000;
    const double minTetta = 0.001, maxTetta = 10;
    private double[] Tetta, gamma2;
    public TGamma2()
    {
        Tetta = new double[count];
        gamma2 = new double[count];
        for (int i = 0; i < count; ++i)
        {
            Tetta[i] = minTetta + ((maxTetta - minTetta) * (double)i / (double)(count - 1));
            gamma2[i] = (1 - Math.Exp(-Tetta[i])) / (1 - ((1 - Math.Exp(-Tetta[i])) / Tetta[i])) - 1;
        }
    }
    public double getTetta(double getGamma2)
    {
        double realTetta = 0;

        for (int i = 1; i < count; ++i)
        {
            if ((getGamma2 < gamma2[i - 1]) && (getGamma2 > gamma2[i]))
            {
                realTetta = Tetta[i - 1] + ((Tetta[i] - Tetta[i - 1]) / (gamma2[i - 1] - gamma2[i]) * (getGamma2 - gamma2[i]));
                break;
            }
        }
        return realTetta;
    }

}

class TIntuite
```

```
    {
        private double lt0, ltask, lparametr, deltaControl, deltaControl_1, lparametr_1, min, max, ldelta, ldelta_1, lcontrol, lcontrol_1, ltime, timerange,ltime0,flag;
        private PerformanceCounter pc;
        public double sigma, u0, lmidtime, korrect, Tob, gamma, deltaPipe, deltaPipe_1;//, A, B, C, D, t1, t2;
        private int lstep, lcount, rangecount;
        public double To, ko, ko_1, futureParametr, du2, kx1, kx2;

        private TCicleBuffer cb1, cb2, cbProisv;
        public bool auto;
        private double u1, u2, sa1; //trubkaCount, sa1;//, beginDelta;
        private double y, sumy, gamma2, maxgamma2, normGamma2, tetta, stabKorrekt;
        private int summCount, summaAllCount;
        private TGamma2 tableGamma2;

        private double k1
        {
           get
           {
              return (kx1 - ((kx1 - kx2) / 5 * du2));
           }
        }
        public double summa1
        {
           get
           {
              return sa1;
           }
        }

        public TIntuite(double in_min, double in_max, double in_sigma, double in_u0, double in_midtime, double in_korrect, double in_gamma)
        {
           min = in_min;
           max = in_max;
           pc = new PerformanceCounter();
           lcontrol = 0;
           sigma = in_sigma;
           ldelta_1 = ldelta = 0;
           lparametr_1 = 0;
           u0 = in_u0;
           lmidtime = in_midtime;
           lstep = 0;
           deltaControl = deltaControl_1 = 0;
           ltime = 0;
```

```
    cb1 = new TCicleBuffer(10000);
    cb2 = new TCicleBuffer(10000);
    cbProisv = new TCicleBuffer(10000);
    lcount = rangecount = 0;
    To = 1;
    ko = 1;
    futureParametr = 0;
    timerange = 0;
    korrect = in_korrect;
    gamma = in_gamma;
    kx1 = 1;
    kx2 = 1;
    auto = true;
    deltaPipe = 0;

    tableGamma2 = new TGamma2();
    sumy = gamma2 = normGamma2 = 0;
    maxgamma2 = 1;
    summCount = summaAllCount = 0;
    stabKorrekt = 0.1;
}

public double Tetta
{
    get
    {
        return tetta;
    }
}
public double fGamma2
{
    get
    {
        return gamma2;
    }
}
public double fnGamma2
{
    get
    {
        return normGamma2;
    }
}
public double Control
{
    get
```

```
    {
        return lcontrol*k1;
    }
}
public double DeltaControl
{
    get
    {
        return deltaControl;
    }
}
public double Delta
{
    get
    {
        return ldelta_1;
    }
}
public double Step
{
    get
    {
        return lstep;
    }
}
public double Task
{
    set
    {
        ltask = value;
        //Calculate();
    }
}
public void TwoStep(double in_u)
{
    lstep = 0;
    if (in_u > u0)
        u1 = u0 + 2 * (in_u - u0);
    else
        u1 = u0 - 2 * ((u0 - in_u));
    u0 = in_u;
    if (u1 > max)
        u1 = max;
    if (u1 < min)
        u1 = min;
}
```

```
        public double Parametr
        {
            set
            {
                lparametr = value;
                if (lstep > 0) //не нулевой шаг остановка/запуск таймера для запоминания отсчёта времени
                {
                    pc.Stop();
                    ldelta += pc.Duration;//время с начала шага

                    ltime += pc.Duration;//время с начала интервала рассчёта
                    pc.Start();

                    lcount++;//количество точек на интервале

                    if (ltime >= lmidtime) //прошло времени больше чем заданный интервал рассчёта
                    {
                        rangecount = lcount;
                        timerange = ltime;
                        lcount = 0;
                        ltime = 0;
                    }


                }
                else
                {
                    pc.Start();
                }

                if (lstep > 1)
                {
                    summCount++;
                    summaAllCount++;
                    y = lparametr - lparametr_1;
                    sumy += y;
                    if (sumy > 0.0001)
                    {
                        gamma2 = (y * summCount / sumy) - 1;
                    }
                    if (gamma2 < 0)
                    {
                        gamma2 = 1;
                    }
```

```
            else
            {
                if ((gamma2 > maxgamma2) && (gamma2 > 1))
                {
                    maxgamma2 = gamma2;
                }
            }
            normGamma2 = gamma2 / maxgamma2;

        }

        switch (lstep)
        {
            case 0:
                lt0 = lparametr_1 = lparametr;
                ldelta_1 = ldelta = 0;
                lstep = 1;
                break;
            case 1:
                if (ldelta > (2 * lmidtime))
                {
                    deltaControl = lcontrol = u0;
                    ldelta_1 = ldelta = 0;
                    lt0 = lparametr_1 = lparametr;

                    sumy = gamma2 = normGamma2 = maxgamma2 = 0;
                    summCount = summaAllCount = 0;

                    lstep = 2;
                }
                break;
            case 2:
                if ((ldelta > (2 * lmidtime)) && (normGamma2 < korrect))
                {
                    tetta = tableGamma2.getTetta(normGamma2);
                    futureParametr = lparametr_1 + ((lparametr - lparametr_1) / (1 -
Math.Exp(-tetta)));

                    ko_1 = ko;
                    To = ldelta / tetta;
                    if (Math.Abs(lparametr - lparametr_1) > Math.Abs((sigma) * 0.1))
                    {
                        ko = (futureParametr - lt0) / deltaControl;
                    }
```

```
                lcontrol_1 = lcontrol;
                deltaControl_1 = deltaControl;
                lcontrol = ltask / (ko);
                deltaControl = lcontrol - lcontrol_1;
                lparametr_1 = lparametr;

                sumy = gamma2 = normGamma2 = 0;
                maxgamma2 = 1;
                summCount = summaAllCount = 0;
                ldelta_1 = ldelta;
                ldelta = 0;
            }
            if ((ldelta > (2 * lmidtime)) && (normGamma2 < stabKorrekt))
            {
                lstep = 3;
            }
            break;
        case 3:
            if (ldelta > (2 * lmidtime))
            {
                if (normGamma2 < stabKorrekt)
                {
                    lstep = 4;
                }
            }
            break;
        case 4:
            if (ldelta > (2 * lmidtime))
            {
            }
            break;
        default:
            break;
    }


  }
 }
}

TMiddle MiddleTemp = new TMiddle();
TMiddle MiddleU = new TMiddle();

private const int doPort0 = 1;
private int doStatus = 0;
private System.IO.StreamWriter file;
```

```
    public const int CHANNEL_COUNT_MAX = 16;
    double[] m_dataScaled = new double[CHANNEL_COUNT_MAX];
    int chanCountSet = 1, chanOutCountSet = 1;

    double[] out_dataScaled = new double[2];

    int chanalStart = 14;
    double Normal = 0, Control = 0;
    TPID pid1;
    TIntuite intuite1;
    #endregion

   public InstantAIForm()
   {
     InitializeComponent();
   }

   public InstantAIForm(int deviceNumber)
   {
      InitializeComponent();
      instantAiCtrl1.SelectedDevice = new DeviceInformation(deviceNumber);
      m_instantAoCtrl.SelectedDevice = new DeviceInformation(deviceNumber);
      instantDoCtrl1.SelectedDevice = new DeviceInformation(deviceNumber);

   }

   public InstantAIForm(string deviceDescription)
   {
      InitializeComponent();
      instantAiCtrl1.SelectedDevice = new DeviceInformation(deviceDescription);
      m_instantAoCtrl.SelectedDevice = new DeviceInformation(deviceDescription);
      instantDoCtrl1.SelectedDevice = new DeviceInformation(deviceDescription);
   }

   private void InstantAiForm_Load(object sender, EventArgs e)
   {
     if (!instantAiCtrl1.Initialized)
      {
         MessageBox.Show("No device be selected or device open failed!",
"instantAiCtrl1");
         this.Close();
         return;
      }
     if (!m_instantAoCtrl.Initialized)
     {
```

```
            MessageBox.Show("No device be selected or device open failed!", "m_instantAoCtrl");
            this.Close();
            return;
        }
        if (!instantDoCtrl1.Initialized)
        {
            MessageBox.Show("No device be selected or device open failed!", "instantDoCtrl1");
            this.Close();
            return;
        }
        doStatus = 0;
         //set title of the form.
        this.Text = "Instant AI(" + instantAiCtrl1.SelectedDevice.Description + ")";

        for (int i = 0; i < 2; ++i)
            out_dataScaled[i] = 0;

        button_start.Enabled = true;
        button_stop.Enabled = false;
        button_pause.Enabled = false;

        m_simpleGraph = new SimpleGraph(pictureBox.Size, pictureBox);
        u_simpleGraph = new SimpleGraph(pictureBox1.Size, pictureBox1);
        timer_getData.Interval = trackBar.Value;
        timer_outData.Interval = trackBar.Value;

        textBox.ReadOnly = true;
        textBox.Text = trackBar.Value.ToString();

        int chanCount = (instantAiCtrl1.ChannelCount <= CHANNEL_COUNT_MAX) ? instantAiCtrl1.ChannelCount : CHANNEL_COUNT_MAX;

        AnalogInputChannel[] channels = instantAiCtrl1.Channels;

        for (int i = 0; i < chanCount; ++i)
        {
            channels[i].SignalType = AiSignalType.SingleEnded;
            ValueRange[] ranges = instantAiCtrl1.Features.ValueRanges;
            channels[i].ValueRange = ValueRange.V_0To10;
        }

        AnalogChannel[] out_chnnels = m_instantAoCtrl.Channels;
        for (int i = 0; i < 2; ++i)
        {
```

```
        ValueRange[] out_ranges = m_instantAoCtrl.Features.ValueRanges;
        out_chnnels[i].ValueRange = ValueRange.V_0To10;
    }

    yf = -10000;
    double u0;
    double.TryParse(tBu0.Text, out u0);
    lu0.Text = u0.ToString();
    double sigma1;
    double.TryParse(tbSigma.Text, out sigma1);
    lsigma.Text = sigma1.ToString();

    double.TryParse(tBalfa.Text, out alfa);
    lalfa.Text = alfa.ToString();

    double.TryParse(tBmtime.Text, out mtime);
    lmtime.Text = mtime.ToString();

    double kx1;
    double.TryParse(tBkx1.Text, out kx1);
    lkx1.Text = kx1.ToString();
    double kx2;
    double.TryParse(tBkx2.Text, out kx2);
    lkx2.Text = kx2.ToString();

     ConfigureGraph();

}

private void ConfigureGraph()
{
     m_simpleGraph.XCordTimeDiv = 50000.0;
     u_simpleGraph.XCordTimeDiv = 50000.0;
     string[] X_rangeLabels = new string[2];
     Helpers.GetXCordRangeLabels(X_rangeLabels, 500.0, 0.0, TimeUnit.Second);
    label_XCoordinateMax.Text = X_rangeLabels[0];
    label_XCoordinateMin.Text = X_rangeLabels[1];

    m_simpleGraph.YCordRangeMax = 110.0;
    m_simpleGraph.YCordRangeMin = 10.0;

    u_simpleGraph.YCordRangeMax = 10.0;
    u_simpleGraph.YCordRangeMin = 0.0;

    ValueUnit unit = (ValueUnit)(-1); // Don't show unit in the label.
    string[] Y_CordLables = new string[3];
```

```
        Helpers.GetYCordRangeLabels(Y_CordLables,
m_simpleGraph.YCordRangeMax, m_simpleGraph.YCordRangeMin, unit);
        label_YCoordinateMax.Text = Y_CordLables[0];
        label_YCoordinateMin.Text = Y_CordLables[1];
        label_YCoordinateMiddle.Text = Y_CordLables[2];

         string[] Y1_CordLables = new string[3];
         Helpers.GetYCordRangeLabels(Y1_CordLables,
u_simpleGraph.YCordRangeMax, u_simpleGraph.YCordRangeMin, unit);
         label_1_YCoordinateMax.Text = Y1_CordLables[0];
         label_1_YCoordinateMin.Text = Y1_CordLables[1];
         label_1_YCoordinateMiddle.Text = Y1_CordLables[2];

         m_simpleGraph.Clear();
         u_simpleGraph.Clear();
      }

      private void timer_getData_Tick(object sender, EventArgs e)
      {
        PerformanceCounter performanceCounter = new PerformanceCounter();
        ErrorCode err;

        performanceCounter.Start();
        err = instantAiCtrl1.Read(chanalStart, chanCountSet, m_dataScaled);
        CheckError(err);
        m_dataScaled[0] = (m_dataScaled[0] + b) * a;

        m_dataScaled[1] = Normal;
         //Проверка начала работы!!!
        if (yf == -10000)
        {
           yf = m_dataScaled[0];
        }
        else
        {
           yf = alfa * yf + ((1 - alfa) * m_dataScaled[0]);//alfa=2/(n+1);
        }
        m_dataScaled[2] = yf;

        int length = m_dataScaled[0].ToString().Length;
        if (stringLength < length) length = stringLength;
        lTemp.Text = m_dataScaled[0].ToString().Substring(0, length);

        if (radioAuto.Checked)
        {
           pid1.Parametr = yf;
```

```
        }
        if (radioInt.Checked)
        {
            intuite1.Parametr = m_dataScaled[0];
            m_dataScaled[3] = 0;// intuite1.a1 * 10 + 35;
            m_dataScaled[4] = intuite1.futureParametr;
            m_dataScaled[5] = 0;// intuite1.summa1 * 1 + 35;


            length = intuite1.Delta.ToString().Length;
            if (stringLength < length) length = stringLength;
            lDelta.Text = intuite1.Delta.ToString().Substring(0, length);
            lStep.Text = intuite1.Step.ToString();
        }
        m_simpleGraph.Chart(m_dataScaled, chanCountSet + 5, 1, 1.0 * trackBar.Value /
1000);
        u_simpleGraph.Chart(out_dataScaled,  chanOutCountSet+1,  1,  1.0  *
trackBar.Value / 1000);

        MiddleTemp.AddMean(yf);

    }

    private void trackBar_Scroll(object sender, EventArgs e)
    {
        m_simpleGraph.Clear();
        u_simpleGraph.Clear();
        timer_getData.Interval = trackBar.Value;
        timer_outData.Interval = trackBar.Value;
        textBox.Text = trackBar.Value.ToString();
    }

    private void button_start_Click(object sender, EventArgs e)
     {
        timer_getData.Start();
        timer_outData.Start();

        button_start.Enabled = false;
        button_pause.Enabled = true;
        button_stop.Enabled = true;
        tbTime.Enabled = false;

        int sec=1;
        int.TryParse( tbTime.Text, out sec);
        timerSaveData.Interval = sec*1000;
```

```
 double sigma1;
 double.TryParse(tbSigma.Text, out sigma1);
 lsigma.Text = sigma1.ToString();

 double u0;
 double.TryParse(tBu0.Text, out u0);

 double.TryParse(tBmtime.Text, out mtime);
 lmtime.Text = mtime.ToString();

 double korrect;
 double.TryParse(tBkorrekt.Text, out korrect);
 lkorrect.Text = korrect.ToString();

double levelGamma;
double.TryParse(tBgamma.Text, out levelGamma);

 double.TryParse(textBoxN.Text, out Normal);
 lNormal.Text = Normal.ToString();
 if (radioAuto.Checked)
 {
    pid1 = new TPID(0.2, 0, 0, 0, 10);
    tbk.Text = lk.Text = pid1.k.ToString();
    tbTi.Text = lTi.Text = pid1.ti.ToString();
    tbTd.Text = lTd.Text = pid1.td.ToString();
    pid1.Task = Normal;
 }


if (radioInt.Checked)
 {
    double kx1;
    double.TryParse(tBkx1.Text, out kx1);
    lkx1.Text = kx1.ToString();
    double kx2;
    double.TryParse(tBkx2.Text, out kx2);

    intuite1 = new TIntuite(0, 10, sigma1, u0, mtime, korrect, levelGamma);
    intuite1.Task = Normal;
    intuite1.lmidtime = mtime;
    intuite1.du2 = out_dataScaled[1];
    intuite1.kx1 = kx1;
    intuite1.kx2 = kx2;

    intuite1.auto = rIntAuto.Checked;
 }
```

```
        DateTime dt = DateTime.Now;
        string fname;
        fname = Environment.CurrentDirectory + "\\";
        fname += dt.Year.ToString() + "-" + dt.Month.ToString() + "-" + dt.Day.ToString()
+ "-" + dt.Hour.ToString() + "-" + dt.Minute.ToString() + "-" + dt.Second.ToString() +
".csv";
        file = new System.IO.StreamWriter(fname, true, Encoding.UTF8);
          string line = "Вермя;Температура;Задание;Управление;Задвижка";
          file.WriteLine(line);

          timerSaveData.Start();
      }

    private void button_pause_Click(object sender, EventArgs e)
    {
      timer_getData.Stop();

      timer_outData.Stop();

      button_start.Enabled = true;
      button_pause.Enabled = false;
    }

    private void button_stop_Click(object sender, EventArgs e)
     {
        timer_getData.Stop();
        timerSaveData.Stop();
        timer_outData.Stop();
        button_start.Enabled = true;

        button_stop.Enabled = false;
        button_pause.Enabled = false;
        Array.Clear(m_dataScaled, 0, chanCountSet);
        m_simpleGraph.Clear();
        u_simpleGraph.Clear();
        tbTime.Enabled = true;
        file.Flush();
        file.Close();
      }

    private void HandleError(ErrorCode err)
    {
```

```
        if (err != ErrorCode.Success)
        {
          MessageBox.Show("Sorry ! some errors happened, the error code is: " +
err.ToString(), "AI_InstantAI");
        }
      }

      private void timer_outData_Tick(object sender, EventArgs e)
      {
        PerformanceCounter performanceCounter = new PerformanceCounter();
        performanceCounter.Start();

        if (radioAuto.Checked)
        {
          out_dataScaled[0]=pid1.Control;
        }
        if (radioInt.Checked)
        {
          out_dataScaled[0] = intuite1.Control;
          int length;

          length = intuite1.DeltaControl.ToString().Length;
          if (stringLength < length) length = stringLength;
          ldU.Text = intuite1.DeltaControl.ToString().Substring(0, length);

          length = intuite1.ko.ToString().Length;
          if (stringLength < length) length = stringLength;
          lko.Text = intuite1.ko.ToString().Substring(0, length);

          length = intuite1.Tob.ToString().Length;
          if (stringLength < length) length = stringLength;
          lTo.Text = intuite1.Tob.ToString().Substring(0, length);

          length = intuite1.futureParametr.ToString().Length;
          if (stringLength < length) length = stringLength;
          lTp.Text = intuite1.futureParametr.ToString().Substring(0, length);

          length = intuite1.Tetta.ToString().Length;
          if (stringLength < length) length = stringLength;
          lgamma.Text = intuite1.Tetta.ToString().Substring(0, length);

          length = intuite1.fGamma2.ToString().Length;
          if (stringLength < length) length = stringLength;
          lfun.Text = intuite1.fGamma2.ToString().Substring(0, length);
```

```
            length = intuite1.fnGamma2.ToString().Length;
            if (stringLength < length) length = stringLength;
            lfunn.Text = intuite1.fnGamma2.ToString().Substring(0, length);
        }
        if (radioHand.Checked)
        {
            out_dataScaled[0] = Control;
        }

        int len = out_dataScaled[0].ToString().Length;
        if (stringLength < len) len = stringLength;
        lControl.Text = out_dataScaled[0].ToString().Substring(0, len);

        ErrorCode err;
        err = m_instantAoCtrl.Write(0, 2, out_dataScaled);
        CheckError(err);

        err = instantDoCtrl1.Write(doPort0, (byte)doStatus);
        CheckError(err);
        performanceCounter.Stop();

        MiddleU.AddMean(out_dataScaled[0]);
    }

    private void CheckError(ErrorCode err)
    {
        if (err != ErrorCode.Success)
        {
            timer_outData.Stop();
            MessageBox.Show("Error: " + err.ToString());
             button_start.Enabled = true;
             button_stop.Enabled = false;
             button_pause.Enabled = false;
             Array.Clear(m_dataScaled, 0, chanCountSet);
             m_simpleGraph.Clear();
             u_simpleGraph.Clear();
        }
    }

    private void buttonWild_Click(object sender, EventArgs e)
    {
        int state = doStatus & 0x1;
        if (doStatus == 1)
        {
            doStatus = 0;
```

```
            buttonWild.Text = "Выкл";
            buttonHeat.Enabled = false;
        }
        else
        {
            doStatus = 1;
            buttonWild.Text = "Вкл";
            buttonHeat.Enabled = true;
        }

    }

    private void buttonHeat_Click(object sender, EventArgs e)
    {
        int state = doStatus & 0x8;
        if (doStatus == 9)
        {
            doStatus = doStatus & 0x1;
            buttonHeat.Text = "Выкл";
        }
        else
        {
            doStatus = 9;
            buttonHeat.Text = "Вкл";
        }
    }

    private void timerSaveData_Tick(object sender, EventArgs e)
    {
        string line;
        DateTime dt = DateTime.Now;
        line = dt.Hour.ToString() + ":" + dt.Minute.ToString() + ":" + dt.Second.ToString()
+ ";";

        line += MiddleTemp.getMiddle().ToString() + ";";
        line += Normal + ";";
        line += MiddleU.getMiddle().ToString() + ";";
        line += out_dataScaled[1].ToString() + ";";

        file.WriteLine(line);

    }

    private void labelO1_Click(object sender, EventArgs e)
    {
```

```
}

private void textBoxOut0_TextChanged(object sender, EventArgs e)
{

}

private void tbTime_TextChanged(object sender, EventArgs e)
{

}

private void textBoxOut0_KeyPress(object sender, KeyPressEventArgs e)
{
   if (e.KeyChar == 13)
   {
      double.TryParse(textBoxOut0.Text, out Control);
      //lControl.Text = Control.ToString();
      int length = Control.ToString().Length;
      if (stringLength < length) length = stringLength;
      lCh0.Text = Control.ToString().Substring(0, length);
   }
}

private void textBoxOut1_KeyPress(object sender, KeyPressEventArgs e)
{
   if (e.KeyChar == 13)
   {
      double.TryParse(textBoxOut1.Text, out out_dataScaled[1]);
      if (radioInt.Checked)
      {
         intuite1.du2 = out_dataScaled[1];
      }
      int length = out_dataScaled[1].ToString().Length;
      if (stringLength < length) length = stringLength;
      lCh1.Text = out_dataScaled[1].ToString().Substring(0, length); ;
   }
}

private void textBoxN_KeyPress(object sender, KeyPressEventArgs e)
{
   if (e.KeyChar == 13)
   {
      double.TryParse(textBoxN.Text, out Normal);
      if (radioAuto.Checked)
        pid1.Task = Normal;
```

```
        if (radioInt.Checked)
            intuite1.Task = Normal;
        lNormal.Text = Normal.ToString();
    }
}

private void tbk_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        double k1;
        double.TryParse(tbk.Text, out k1);
        pid1.k = k1;
        lk.Text = pid1.k.ToString();
    }
}

private void tbTi_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        double Ti1;
        double.TryParse(tbTi.Text, out Ti1);
        pid1.ti = Ti1;
        lTi.Text = pid1.ti.ToString();
    }
}

private void tbTd_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        double Td1;
        double.TryParse(tbTd.Text, out Td1);
        pid1.td = Td1;
        lTd.Text = pid1.td.ToString();
    }
}

private void tbSigma_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        double sigma1;
        double.TryParse(tbSigma.Text, out sigma1);
        intuite1.sigma = sigma1;
```

```
        lsigma.Text = sigma1.ToString();
    }
}

private void tBu0_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        double u0;
        double.TryParse(tBu0.Text, out u0);
        lu0.Text = u0.ToString();
    }
}

private void tBalfa_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        double.TryParse(tBalfa.Text, out alfa);
        lalfa.Text = alfa.ToString();
    }
}

private void tBmtime_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        double.TryParse(tBmtime.Text, out mtime);
        if (radioInt.Checked)
        {
            intuite1.lmidtime = mtime;
        }
        lmtime.Text = mtime.ToString();
    }
}


private void tBkorrekt_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        double korrect;
        double.TryParse(tBkorrekt.Text, out korrect);
        if (radioInt.Checked)
        {
            intuite1.korrect = korrect;
```

```
        }
        lkorrect.Text = korrect.ToString();
    }

}

private void radioInt_CheckedChanged(object sender, EventArgs e)
{
    if (radioInt.Checked)
    {
        double sigma1;
        double.TryParse(tbSigma.Text, out sigma1);

        double u0;
        double.TryParse(tBu0.Text, out u0);

        double.TryParse(tBmtime.Text, out mtime);
        lmtime.Text = mtime.ToString();

        double korrect;
        double.TryParse(tBkorrekt.Text, out korrect);
        lkorrect.Text = korrect.ToString();

        double levelGamma;
        double.TryParse(tBgamma.Text, out levelGamma);

        intuite1 = new TIntuite(0, 10, sigma1, u0, mtime, korrect, levelGamma);
        intuite1.Task = Normal;
        intuite1.lmidtime = mtime;

        double kx1;
        double.TryParse(tBkx1.Text, out kx1);
        lkx1.Text = kx1.ToString();
        double kx2;
        double.TryParse(tBkx2.Text, out kx2);
        lkx2.Text = kx2.ToString();
        intuite1.kx1 = kx1;
        intuite1.kx2 = kx2;

            intuite1.auto = rIntAuto.Checked;
    }
}

private void radioAuto_CheckedChanged(object sender, EventArgs e)
{
    if (radioAuto.Checked)
```

```
    {
        pid1 = new TPID(0.2, 0, 0, 0, 10);
        tbk.Text = lk.Text = pid1.k.ToString();
        tbTi.Text = lTi.Text = pid1.ti.ToString();
        tbTd.Text = lTd.Text = pid1.td.ToString();
        pid1.Task = Normal;
    }
}

private void tBkx1_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        double kx1;
        double.TryParse(tBkx1.Text, out kx1);
        if (radioInt.Checked)
        {
            intuite1.kx1 = kx1;
        }
        lkx1.Text = kx1.ToString();
    }
}

private void tBkx2_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        double kx2;
        double.TryParse(tBkx2.Text, out kx2);
        if (radioInt.Checked)
        {
            intuite1.kx2 = kx2;
        }
        lkx2.Text = kx2.ToString();
    }
}

private void rIntAuto_CheckedChanged(object sender, EventArgs e)
{
    if (rIntAuto.Checked)
    {
        if (radioInt.Checked)
        {
            intuite1.auto = true;
        }
    }
```

```
    }

    private void rIntHand_CheckedChanged(object sender, EventArgs e)
    {
        if (rIntHand.Checked)
        {
            if (radioInt.Checked)
            {
                intuite1.auto = false;
            }
        }

    }

    private void tBgamma_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (e.KeyChar == 13)
        {
            double gamma;
            double.TryParse(tBgamma.Text, out gamma);
            if (radioInt.Checked)
            {
                intuite1.gamma = gamma;
            }
        }

    }

    private void label32_Click(object sender, EventArgs e)
    {

    }

  }

  public static class ConstVal
   {
      public const int Channel_Start = 0;
      public const int Channel_Count = 3;
   }
}
```