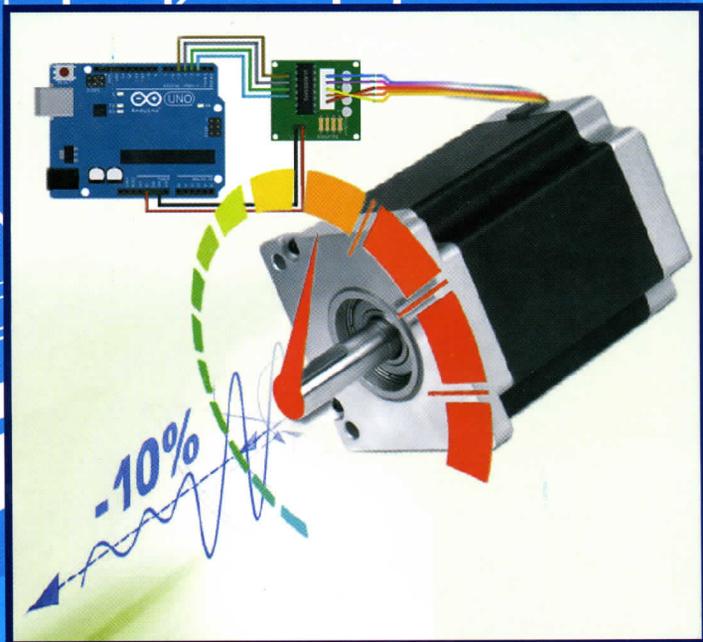




# МОДЕЛИРОВАНИЕ СИСТЕМ УПРАВЛЕНИЯ В SIMULINK



Министерство образования и науки Украины  
ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ АВТОМОБИЛЬНО-  
ДОРОЖНЫЙ УНИВЕРСИТЕТ

Богомолов В.А., Гурко А.Г., Клименко В.И.  
Леонтьев Д.Н., Красюк А.Н.

# **МОДЕЛИРОВАНИЕ СИСТЕМ УПРАВЛЕНИЯ В SIMULINK**

**Учебное пособие**

Рекомендовано Ученым советом Харьковского национального  
автомобильно-дорожного университета

Харьков  
ХНАДУ  
2018

УДК 681.51.011  
М 74

Рецензенти: О.В. Полярус, д.т.н., професор кафедри метрології та безпеки життєдіяльності, Харківський національний автомобільно-дорожній університет;  
Д.О. Волонцевич, д.т.н., професор, завідувач кафедри інформаційних технологій і систем колісних та гусеничних машин імені О.О. Морозова, Національний технічний університет «ХПІ»  
І.К. Шаша, д.т.н., професор кафедри №6 експлуатації та ремонту автомобілів та бойових машин Національної академії Національної гвардії України

Авторский коллектив:  
В.А. Богомолов, д.т.н., профессор;  
А.Г. Гурко, к.т.н., доцент;  
В.И. Клименко, к.т.н., профессор;  
Д.Н. Леонтьев, к.т.н., доцент;  
А.Н. Красюк, к.т.н., ассистент.

Наведено відомості про використання однієї з найефективніших систем комп'ютерної математики MATLAB для аналізу та синтезу систем автоматичного керування. Матеріал ґрунтується на версії Simulink 7.8, що входить до складу пакета MATLAB R2011b. Призначено для студентів та аспірантів механічних, електромеханічних спеціальностей, і буде корисним усім, хто цікавиться моделюванням систем керування.

М 74            Моделирование систем управления в Simulink: уч. пособ. /  
В.А. Богомолов, А.Г. Гурко, В.И. Клименко и др. – Харьков : ХНАДУ,  
2018. – 220 с.  
**ISBN 978-966-303-693-9**

Приведено сведения по использованию одной из самых эффективных систем компьютерной математики MATLAB для анализа и синтеза систем автоматического управления. Материал основывается на версии Simulink 7.8, входящего в состав пакета MATLAB R2011b. Предназначена для студентов и аспирантов механических и электромеханических специальностей, и будет полезна всем, кто интересуется моделированием систем управления.

УДК 681.51.011

**ISBN 978-966-303-693-9**

© Богомолов В.А., Гурко А.Г., Клименко В.И.,  
Леонтьев Д.Н., Красюк А.Н., 2018.  
© ХНАДУ, 2018.

# СОДЕРЖАНИЕ

## ВВЕДЕНИЕ

1. НАЧАЛЬНЫЕ СВЕДЕНИЯ О ТЕОРИИ УПРАВЛЕНИЯ.....	7
1.1. Основные сведения об управлении .....	7
1.2 Математическое моделирование процесса управления.....	15
2. МАТЕМАТИЧЕСКОЕ ОПИСАНИЕ ЛИНЕЙНЫХ СИСТЕМ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ.....	19
2.1. Понятие математической модели .....	19
2.2 Примеры построения моделей физических систем.....	21
2.2.1 Механические системы с линейным перемещением .....	22
2.2.2 Механические системы с вращательным движением.....	60
2.2.3 Электрические системы.....	65
2.2.4 Электромеханические системы.....	68
2.3 Линеаризация.....	78
3. СОЗДАНИЕ ПОДСИСТЕМ .....	82
3.1. Формирование подсистемы.....	82
3.2 Маскирование подсистемы.....	87
3.2.1 Общие сведения.....	87
3.2.2 Создание графического изображения подсистемы .....	89
3.2.3 Задание параметров подсистемы .....	94
3.2.4 Создание справочной информации .....	101
4. УСТОЙЧИВОСТЬ ЛИНЕЙНЫХ СИСТЕМ.....	107
4.1. Общие сведения.....	107
4.2 Устойчивость установившихся режимов.....	109
4.2.1 Понятие устойчивости переходного процесса .....	109
4.2.2 Исследование устойчивости.....	111
4.2.3 Критерий устойчивости Найквиста .....	123
4.3 Точность системы в установившемся режиме.....	132
4.4 Требования к качествам переходных процессов.....	133
5. СИНТЕЗ ЛИНЕЙНЫХ СИСТЕМ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ.....	141
5.1. Постановка задачи синтеза.....	141
5.2 Основные этапы решения задачи синтеза.....	141
5.3 Классификация регуляторов .....	146

5.3.1 П-регулятор.....	146
5.3.2 Интегрирующий регулятор .....	168
5.3.3 Дифференцирующий регулятор.....	172
5.4 ПИД-регуляторы .....	173
5.5. ПИ-регуляторы .....	177
5.6 ПД-регуляторы .....	179
5.7 Достоинства и недостатки ПИ- и ПИД-регуляторов .....	180
5.8 Практическая реализация ПИД-регулятора.....	183
5.9 Пример синтеза ПИД-регулятора в Simulink.....	185
6. МОДЕЛИРОВАНИЕ НЕЛИНЕЙНЫХ СИСТЕМ.....	197
6.1. Нелинейные блоки библиотеки Discontinuities...	204
6.2 Создание сложных нелинейностей.....	223
ЛИТЕРАТУРА.....	230

## ВВЕДЕНИЕ

Современный мир невозможно представить без систем автоматического управления. Бытовая техника и общественный транспорт, станки и сложные производственные комплексы имеют в своём составе множество контуров автоматического управления. Мы так к ним привыкли, что часто их не замечаем и даже не задумываемся об их наличии. Сложность современных систем управления возрастает: увеличивается количество контуров управления, усложняются алгоритмы управления. Созданием подобных систем занимаются специалисты, с использованием современной и робастной<sup>1</sup> теории управления.

В то же время многие из систем управления достаточно просты и могут эффективно работать на принципах, рассматриваемой классической теорией управления. Ни в коем случае не стремясь умалить сложность и значимость этой части теории управления, хочется отметить, что разработка простейших систем часто требует лишь основных теоретических знаний и умения работать с такими мощнейшими программными продуктами как MATLAB, и наиболее популярным его приложением – Simulink. Такие знания позволяют современному инженеру проектировать свои системы и устройства с простейшими контурами автоматического управления без привлечения специалистов по автоматизации.

Данная книга является, своего рода пособием по «быстрому старту», позволяющему изучить основы классической теории управления и получить навыки исследования и проектирования простейших систем в Simulink. Поэтому теоретический материал дается в упрощенном виде, для того чтобы книгой мог воспользоваться человек никогда не сталкивавшийся с моделированием технических систем.

---

<sup>1</sup> **Робастный** – [англ. robust < лат. robuste прочно, крепко] устойчивый к помехам. Робастное управление – совокупность методов теории управления, целью которых является синтез такого регулятора, который обеспечивал бы хорошее качество управления, если объект управления сильно отличается от расчётного или его математическая модель неизвестна.

В основу данного учебного пособия положены курсы лекций по теории управления, читавшиеся студентам и аспирантам Харьковского национального автомобильно - дорожного университета. Книга предназначена, в основном, для студентов и аспирантов, механических и электромеханических специальностей, для которых теория автоматического управления не является профильным предметом. Однако, она будет полезна и студентам, обучающимся по специальности «Автоматизированное управление технологическими процессами», а также студентам и инженерам близких инженерных специальностей и всем, кто желает познакомиться с основами теории управления и получить навыки использования такого мощного инструмента как Simulink.

# НАЧАЛЬНЫЕ СВЕДЕНИЯ О ТЕОРИИ УПРАВЛЕНИЯ

## 1.1. Основные сведения об управлении

Прежде чем начать моделирование систем управления необходимо определиться, что же заложено в сам термин «управление». В самом широком смысле *управление представляет собой совокупность разнокачественных воздействий на объект управления, в результате которых достигается в той или иной степени цель субъекта*. При этом в качестве объекта управления могут выступать: социальные, экономические, технические, биологические и другие процессы. Фактически, во всех частных отраслях прикладного знания, речь идёт об управлении теми или иными процессами, относящимися к их «предметной области», хотя управление этими процессами и не называется *управлением*: медицина лечит, а не управляет здоровьем и болезнями; химия «химичит», а не управляет синтезом и распадом химических соединений; архитектура и строительство что-то воздвигают, а не управляют проектированием и возведением объектов и т.п. [1].

Главным в процессе управления является цель. Цель – это желаемый мысленный конечный результат, т.е. то, что должно быть достигнуто при «идеальном» управлении. Цель – первична, она – основополагающая всего процесса управления [2]. Поэтому, процесс управления начинается с постановки цели и заканчивается при её достижении. В случае если цель не была достигнута или не в той степени, цель корректируется и процесс повторяется. Схематически процесс управления можно представить следующим образом (рис. 1.1).

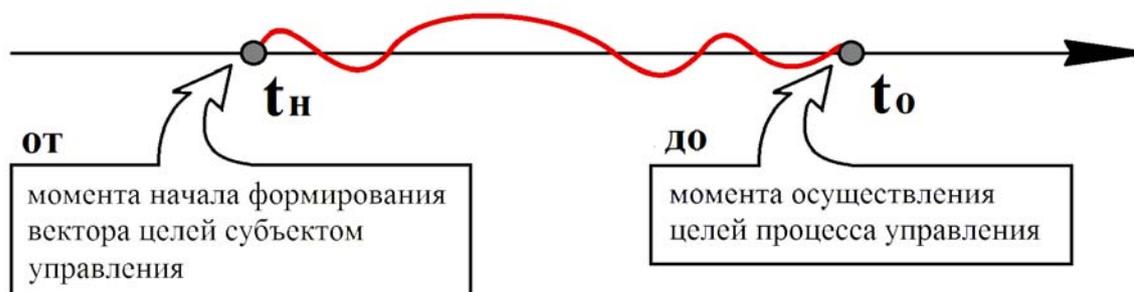


Рис. 1.1. Процесс управления в общем виде

Достижение заданной цели всегда осуществляется путём воздействия на объект управления, которым может быть любой материальный объект в мироздании. Поэтому осуществление процесса управления всегда происходит по схеме, представленной на рис. 1.2 [1].

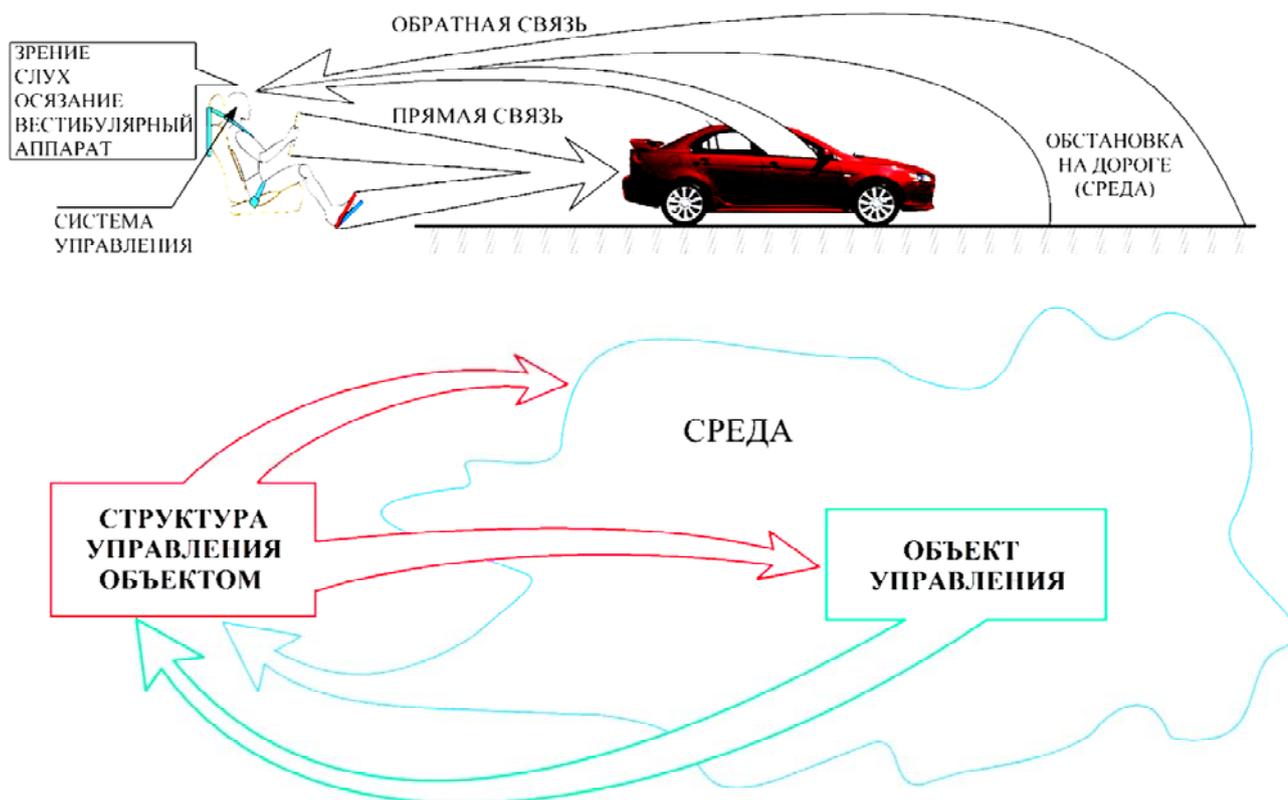


Рис. 1.2. Общая схема процесса управления

В процессе управления всегда присутствуют три составляющие: *субъект управления* или *структура управления объектом*, в зависимости от решаемой задачи, сам *объект управления* и *среда*. Рассмотрим пример: управление автомобилем, который в данном случае является *объектом управления*, водитель (человек) – *субъект управления*, среда – окружающая обстановка вокруг автомобиля (включает: дорожное покрытие, погоду, светофоры, дорожную разметку, другие транспортные средства и т.д.). Водитель, воздействуя на органы управления (рулевое колесо, педаль газа и тормоза, рычаг коробки передач) оказывает управляющее воздействие на автомобиль, что образует *прямую связь*, а через органы чувств воспринимает «поведение» автомобиля и окружающую обстановку – *обратная связь*. В процессе управления

автомобилем водитель соразмеряя цель управления, информацию о самом автомобиле и информацию от обратной связи принимает решение о дальнейшем управляющем воздействии.

В теории управления возможна постановка всего двух задач [1]:

– первая задача заключается в том, что мы хотим управлять объектом в процессе его функционирования сами – задача **УПРАВЛЕНИЯ**.

– вторая задача заключается в том, что мы не хотим управлять объектом в процессе его функционирования, но хотим, чтобы объект, без нашего непосредственного вмешательства в процесс, самоуправлялся в приемлемом для нас режиме – задача **САМОУПРАВЛЕНИЯ**. Например, если человек сам управляет автомобилем, то осуществляется процесс **УПРАВЛЕНИЯ**. В случае если человек едет в автомобиле управляемый, либо личным водителем, либо водителем такси, тогда решается задача **САМОУПРАВЛЕНИЯ**, т.е. человек не управляет автомобилем лично, но он едет туда, куда ему нужно.

Также следует понимать и учитывать то, что в управлении всегда присутствует иерархия, т.е. субъект управления + объект управления (некий частный процесс управления) может объектом управления для более высшего уровня иерархии, образно это можно представить следующим рисунком [1].



Рис. 1.3. Иерархия процесса управления

Для лучшего понимания рассмотрим пример – выпекание хлеба в хлебопечке. В этом случае человек является субъектом управления, тесто – объект управления. Процесс управления заключается в том, чтобы положить необходимые ингредиенты в нужном количестве в хлебопечку и включить нужную программу (происходит процесс управления на данном уровне иерархии). Когда программа в хлебопечке запущена, блок управления начинает управлять нагревательным и вымешивающими элементами, таким образом, чтобы через заданный промежуток времени получился готовый хлеб – происходит решение задачи управления на данном уровне иерархии, но для высшего уровня иерархии (для человека) решается задача самоуправления.

Важно знать две аксиомы теории управления [1, 3]:

1) Управлять возможно только объективно существующим процессом. В связи с этим моделировать также возможно только объективно существующий процесс.

2) Объекты, не обладающие устойчивостью в смысле предсказуемости их поведения в достаточной для этого мере, в принципе не поддаются управлению и не могут быть введены в режим самоуправления.

Для осознанной постановки и решения каждой из задач и обеих задач совместно (когда одна сопутствует другой) необходимы три набора информации (рис. 1.4):



Рис. 1.4. Структурирование информации, описывающее процесс управления.

1) Вектор целей управления, представляющий собой описание идеального режима функционирования (поведения) объекта.

2) Вектор (текущего) состояния контрольных параметров, описывающий реальное поведение объекта по параметрам, входящих в вектор целей.

3) Вектор ошибки управления, представляющий собой «разность» (в кавычках потому, что разность не обязательно привычная алгебраическая) между вектором целей и вектором (текущего) состояния.

Так определяется как текущее, так и конечное качество управления – первый критерий оценки систем управления и процесса управления в целом.

Рассмотрим пример, в котором учтём выше изложенный материал – выполнение экипажем танка боевой задачи (рис. 1.5). В качестве субъекта управления выступает командир танка, который оказывает управляющее воздействие на механика-водителя. Механик-водитель оказывает управляющее воздействие непосредственно на боевую машину (цепочка передачи информации от командира танка до механика-водителя – задача управления; от командира танка до боевой машины – задача самоуправления). При этом как у командира танка, так и у механика-водителя есть свой вектор целей, вектор текущего состояния и вектор ошибки.

В процессе реального управления осуществляется замыкание информационных потоков с вектора целей на вектор ошибки (или эквивалентное ему замыкание на вектор состояния). Иными словами, в процессе управления информация о векторе состояния (или векторе ошибки управления) соотносится с вектором целей и на основе этого соотношения вырабатывается и осуществляется *управляющее воздействие*.

Теперь рассмотрим, как формируется *управляющее воздействие*. Итак, любой объект управления обладает физическими параметрами, такими как: масса, скорость, геометрические размеры, температура и т.д. Эти физические параметры, в процессе управления, можно разделить на три группы:

1) *Контрольные* – это те параметры, по которым однозначно определяется поведение и/или состояние объекта управления;

2) *Управляемые* – это параметры, которые могут быть непосредственно изменены воздействием со стороны субъекта, что повлечёт за собой и изменение контрольных параметров. В изменении значений непосредственно управляемых параметров выражается *управляющее воздействие*;

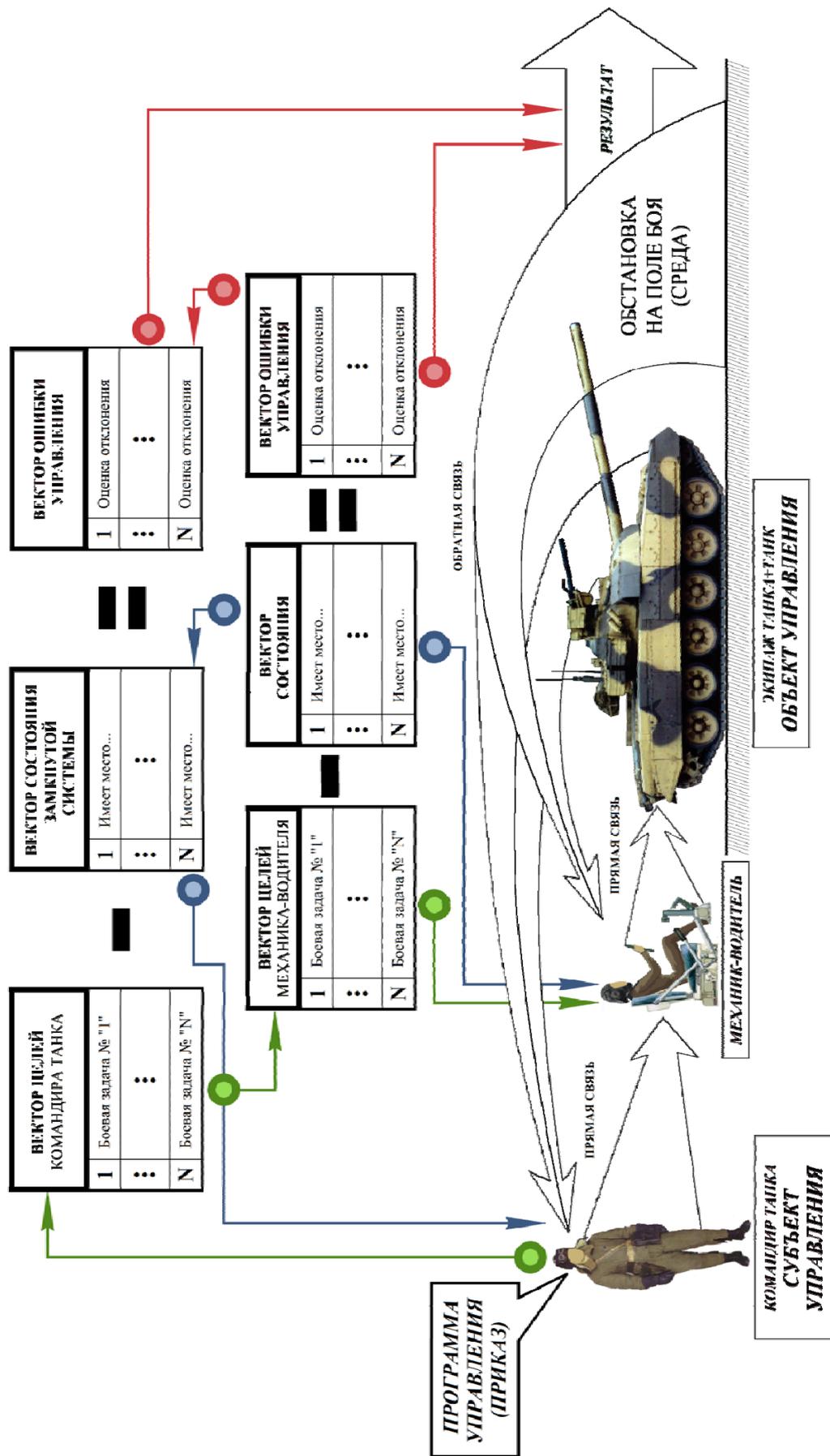


Рис. 1.5. Структурирование информации, описывающее процесс управления боевой машиной (танком) при выполнении боевой задачи.

3) *Свободные* – это параметры, которые изменяются при изменении непосредственно управляемых, но не входят в перечень контрольных параметров, составляющих вектор целей управления. Все объективно возможные значения свободных параметров в процессе управления признаются допустимыми.

Так, например, для корабля:

- угол курса – контрольный параметр;
- угол перекладки руля – (непосредственно) управляемый параметр;

- угол дрейфа (между вектором скорости, т.е. направлением движения в текущий момент времени и плоскостью симметрии корабля, называемой диаметральной плоскостью) – свободный параметр.

Или другой пример: для систем «искусственного климата» контрольным параметром может быть температура воздуха в помещении, а относительная влажность воздуха в нём может быть свободным параметром. Включение относительной влажности воздуха в список контрольных параметров потребует дополнения системы устройствами поглощения избыточной влаги и увлажнения чрезмерно обезвоженного воздуха.

Следующим шагом в понимании процесса управления необходимо рассмотреть режимы работы. В процессе управления любой объект управления может находиться только в двух режимах – это балансировочный (рис. 1.6) и манёвр (рис. 1.7).

В мироздании любой процесс носит колебательный характер, поэтому в процессе управления контрольные параметры являются не конкретным значением, а диапазоном значения от и до конкретной величины. Название балансировочного режима пошло от того, что контрольные параметры как бы балансируют между верхними и нижними значениями контрольных параметров. В технической литературе такой режим принято называть *стационарным*.

В случае если объект управления переводиться из одного балансировочного режима в другой, тогда наступает режим манёвра. Существует два типа манёвра – это слабый и сильный, их отличие друг от друга условное и определяется субъективным выбором эталонного времени, и единицы измерения времени.

Это разделение манёвров на сильные и слабые происходит из того, что во многих случаях моделирование слабых манёвров может быть существенно упрощено, за счёт пренебрежения целым рядом факторов, без потери качества управления. В технической литературе слабый манёвр рассматривается как стационарный процесс, а сильный манёвр – как динамический процесс.

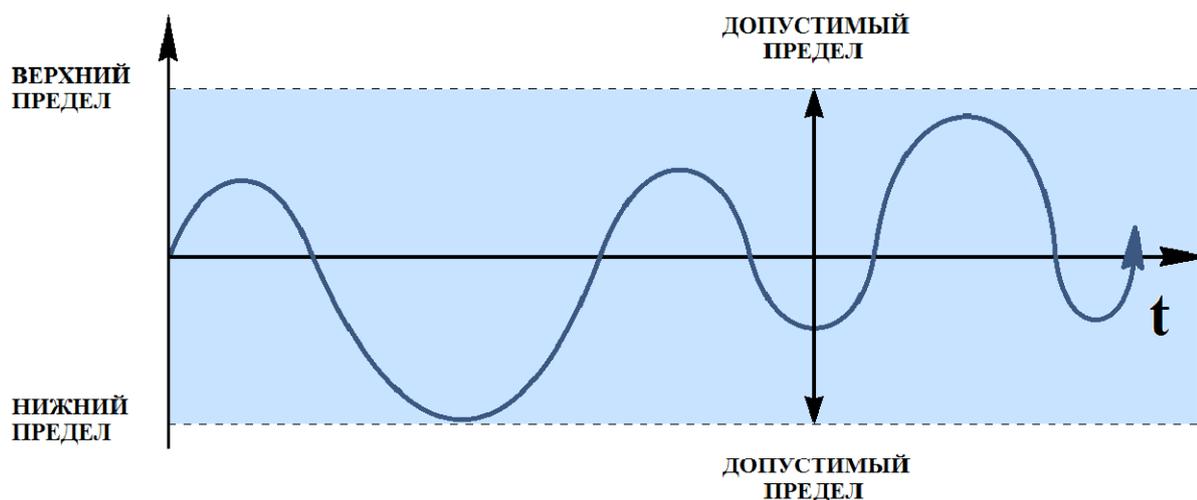


Рис. 1.6. Балансировочный режим работы

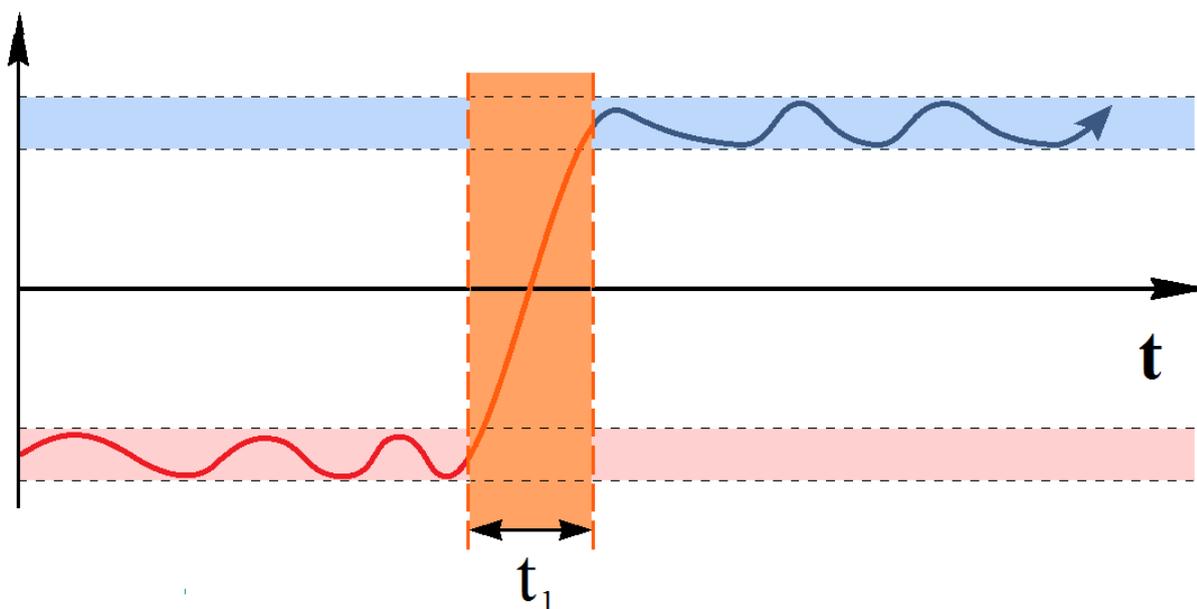


Рис. 1.7. Режим работы – манёвр

## 1.2 Математическое моделирование процесса управления.

Начать рассмотрение вопроса моделирования процесса управления необходимо с определения цели самого моделирования. Здесь вариантов может быть множество, но должно соблюдаться правило «ясно, конкретно, понятно», например: установить влияние параметра «А» на выходные параметры системы «X, Y, Z» или определить значения параметров «А» и «В», при которых система будет иметь наивысшее быстродействие и т.п. Очень частой ошибкой является формирование цели в виде: «Построение математической модели системы «X»», что представляет собой недостижимую цель, поэтому возникают большие сложности при составлении математических моделей.

После определения цели необходимо составить структурную схему исследуемого процесса, хотя очень часто цель может ставиться на основании такой структурной схемы. В построении нет четких правил, очень удобно использовать общую схему управления (рис. 1.2), например, управление направлением движения автомобиля (приведено на рис. 1.8 и рис. 1.2)

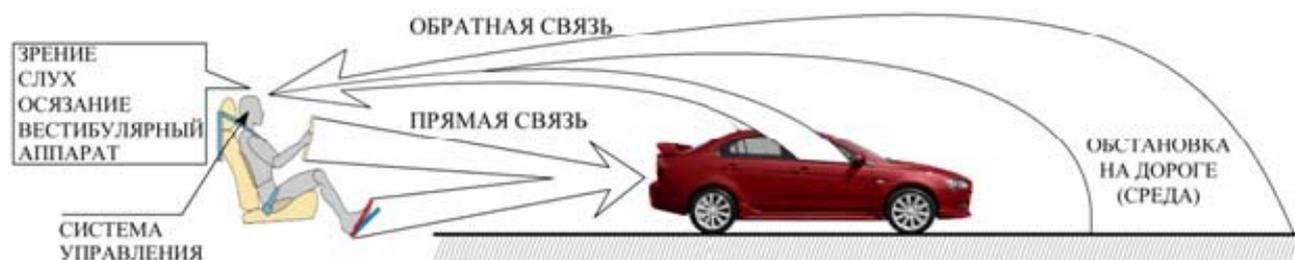


Рис. 1.8. Общая схема процесса управления движением автомобиля.

На данном этапе необходимо определить иерархию взаимодействующих компонентов, контролируемые и управляемые параметры.

После этого необходимо приступить к построению функциональной схемы. Рассмотрим типовую функциональную схему системы управления (рис. 1.9). На функциональных схемах при помощи блоков отображают элементы системы управления с указанием выполняемых ими функций, а также связи между этими элементами. При помощи задающего устройств вырабатывается

желаемое значение входной величины  $x(t)$ , которое поступает на один из входов сумматора, на другой вход сумматора по цепи обратной связи подается измеренное с помощью датчика действительное значение выходной величины  $y_{из}(t)$ . На выходе сумматора образуется сигнал ошибки (отклонения)  $\varepsilon(t)$ , который является разностью между заданным и действительными значениями параметров:  $\varepsilon(t) = x(t) - y_{из}(t)$ . Управляющее устройство в зависимости от величины и знака ошибки вырабатывает сигнал управления  $u(t)$ . Этот сигнал подается на исполнительное устройство, что формирует управляющее воздействие на объект управления.

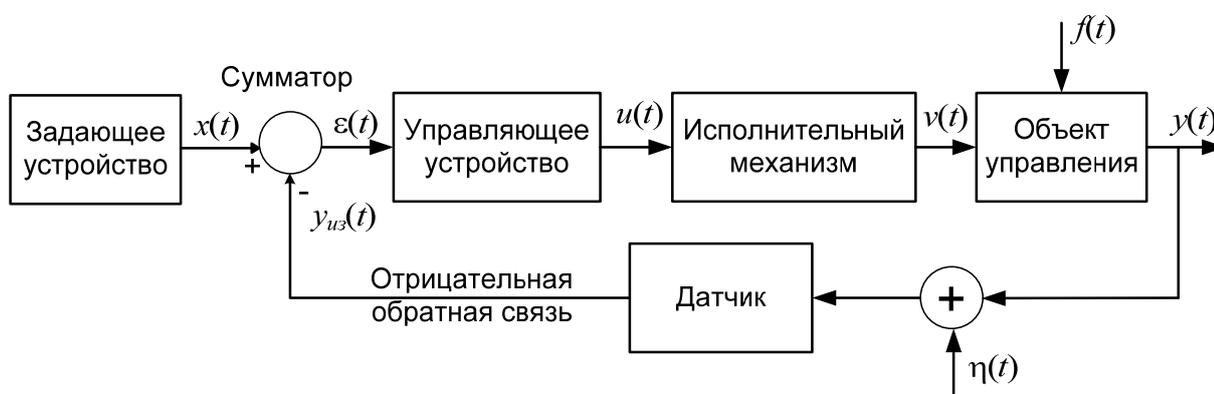


Рис. 1.9. Типовая функциональная схема замкнутой системы управления

При действии внешних возмущений  $f(t)$  выходная величина  $y(t)$  объекта начинает отклоняться от желаемого значения  $x(t)$ , на входе управляющего устройства формируется сигнал ошибки  $\varepsilon(t)$ , и управляющее устройство выдает такое управление  $u(t)$ , чтобы свести эту ошибку к нулю или к конструктивно возможному минимуму. К сожалению, процесс измерения сопровождается шумом  $\eta(t)$ , что усложняет расчет управления  $u(t)$ . Кроме того, обратная связь борется с теми возмущениями, которые уже подействовали на объект, поэтому при часто меняющихся возмущающих воздействиях эффективность замкнутой системы управления снижается.

Применительно к управлению направлением движения автомобиля (рис. 1.8) функциональная схема будет иметь вид рис. 1.10.

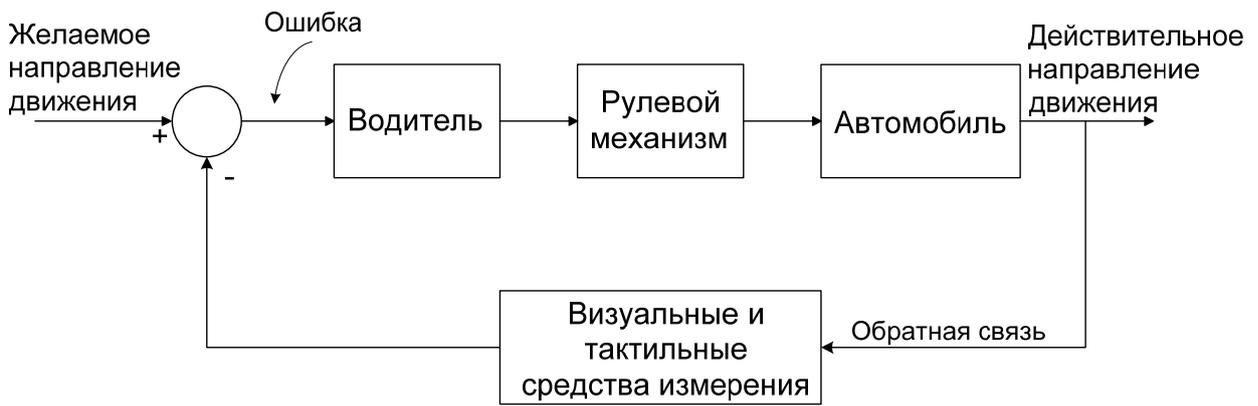


Рис. 1.10. Функциональная схема управления направлением движения автомобиля

Рассмотрим ещё один пример – управление высотой кузова транспортного средства, структурная схема такой системы имеет следующий вид (рис. 1.11).

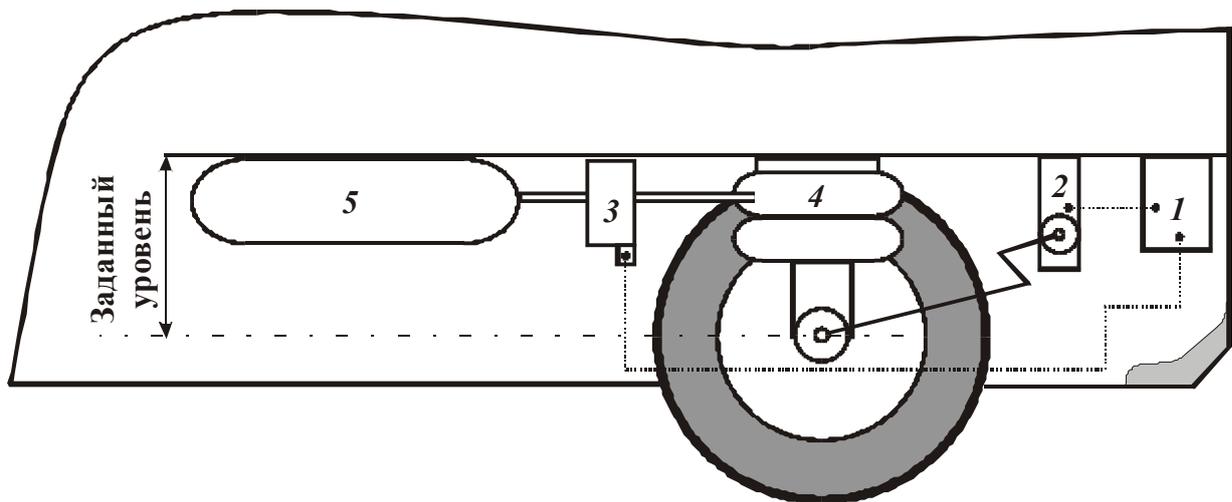


Рис. 1.11. Структурная схема управления пневматической подвеской:  
 1 – электронный блок управления; 2 – датчик высоты; 3 – электропневматический модулятор; 4 – пневматический упругий элемент; 5 – ресивер.

Целью работы данной системы является обеспечение требуемой высоты кузова при движении транспортного средства по неровностям дороги. Упрощенно система работает следующим образом. Под действием загрузки пневмобаллон сжимается, что регистрирует датчик высоты кузова. Сигнал от этого датчика подается на электронный блок управления (ЭБУ). ЭБУ подает команду на исполнительный элемент – модулятор, который,

например, приводится в действие электродвигателем. Тогда ЭБУ подает команду электродвигателю повернуть свой вал на некоторый угол  $\varphi$ , чтобы при помощи золотника открыть соответствующий клапан и либо пропустить дополнительное количество воздуха к пневмобаллону, либо выпустить лишний воздух в атмосферу. В данном случае функциональная схема будет иметь следующий вид (рис. 1.12.).

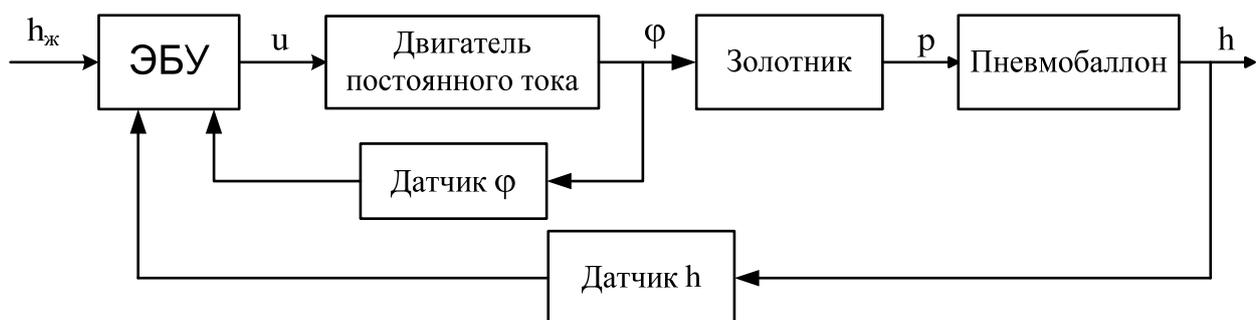


Рис. 1.12. Функциональная схема системы управления пневматической подвеской

После составления функциональной схемы можно переходить к составлению математической модели.

## 2. МАТЕМАТИЧЕСКОЕ ОПИСАНИЕ ЛИНЕЙНЫХ СИСТЕМ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ

### 2.1. Понятие математической модели

При работе с любыми техническими системами, инженер сталкивается с двумя основными задачами: задачей анализа и задачей синтеза. Задача анализа заключается в определении того, как будет вести себя система в той или иной ситуации, а задача синтеза – в разработке наиболее простой структуры системы автоматического управления и расчета ее параметров, обеспечивающих заданные показатели качества и точности. Конечно, анализировать поведение системы можно непосредственно в процессе её работы, однако, во-первых, это может дорого стоить, потребовать слишком много времени и, вообще, вывести систему из строя, а, во-вторых, на этапе проектирования также надо анализировать качество создаваемой системы, однако её ведь еще нет в природе! Выходом из сложившейся ситуации является использование математической модели системы. Математические модели используются и при синтезе закона управления, реализуемого управляющим устройством.

*Математическая модель реального устройства* - это формулы, устанавливающие между математическими описаниями  $x(t)$  и  $y(t)$  его входной и выходной величин соответствие, близкое к тому, какое существует между  $x_{\text{реал}}(t)$  и  $y_{\text{реал}}(t)$ . Математическая модель должна быть достаточно простой и в то же время отражать существенные черты связи между  $x_{\text{реал}}(t)$  и  $y_{\text{реал}}(t)$ .

В общем случае для непрерывных систем математическая модель описывается дифференциальными или интегральными уравнениями, т.е. поведение системы управления нельзя точно описать, не зная его предыстории (интегральная составляющая) и того, как *происходят изменения с течением времени* (дифференцирующая составляющая).

Обычно математическую модель линейной системы сводят к дифференциальному уравнению. Вид этого уравнения зависит от свойств, которыми обладает система: *дифференциальные уравнения отражают фундаментальные законы природы, определяющие процессы в моделируемой системе.*



Рис. 2.1. Реальное устройство и его математическая модель

Следует отметить, что дифференциальные уравнения отображают *динамику* системы, т.е. описывают тот режим ее работы, когда входные и выходные сигналы являются функциями времени. Если входные и выходные сигналы не изменяются во времени, это соответствует статическому режиму работы системы. Этот режим описывается при помощи *статической характеристики*.

*Статическая характеристика* представляет собой зависимость выходного сигнала от входного в установившемся режиме. Таким образом, статическая характеристика представляет собой дифференциальное уравнение нулевого порядка, т.е. алгебраическое уравнение.

Системы управления преимущественно работают в динамических режимах и анализ статических режимов необходим до того, как рассматривается динамика системы.

Общий порядок построения дифференциального уравнения, моделирующего поведение системы следующий [4]:

1. Определяются входная и выходная величины звена.
2. Устанавливается физический закон (законы), в соответствии с которым протекают процессы в системе.

3. Внешняя сила, энергия, входящий поток вещества и т.п. выражаются через входную переменную и ее производные, а сила сопротивления, накапливаемая энергия, выходной поток вещества и т.п. – через выходную переменную и ее производные. При этом все входные переменные и их производные записываются справа от знака «=», а все выходные переменные и их производные – слева.

Построенное из таких физических соображений уравнение устанавливает уже чисто математическое соотношение между входной и выходной величинами реальной системы, то есть и является её математической моделью.

При построении математической модели указанным выше способом часто сталкиваются со следующей проблемой. Дело в том, что большинство реальных систем управления очень сложные, они имеют в своем составе множество элементов и устройств, работающих на различных физических принципах и на основе различных фундаментальных законов. В этом случае практически невозможно сразу построить математическую модель сложной системы. Поэтому при разработке математической модели сколь-нибудь сложной системы её целесообразно представить при помощи *структурной схемы или графа*.

Структурной схемой называется изображение системы в виде совокупности звеньев и связей между ними. Звенья представляются в виде прямоугольных блоков, внутри которых указывается их математическая модель. Вход и выход блока обозначается стрелками. Каждое из звеньев имеет единую физическую природу: механическую, электрическую, гидравлическую и т.п. Конечно, иногда в таком представлении необходимости нет. Кроме того, несколько физических элементов могут быть представлены одним звеном.

При этом все выделяемые элементарные звенья должны обладать направленностью действия, при которой процессы, происходящие в звене, непосредственно не влияют на входное воздействие. Математическое описание каждого такого звена может быть получено без учета его связи с другими звеньями. Соответственно математическое описание всего устройства будет представляться совокупностью уравнений отдельных звеньев, дополненных уравнениями связей, определяемых структурой устройства.

## **2.2 Примеры построения моделей физических систем**

Рассмотрим примеры построения дифференциальных уравнений физических элементов различной физической природы.

### *2.2.1 Механические системы с линейным перемещением*

Рассмотрим сначала модели элементов, из которых состоят механические системы с линейным перемещением [5]. Параметрами механических элементов являются *масса* (рис. 2.2, а), *демпфирование (трение)* (рис. 2.2, б) и *упругость (эластичность)* (рис. 2.2, в).

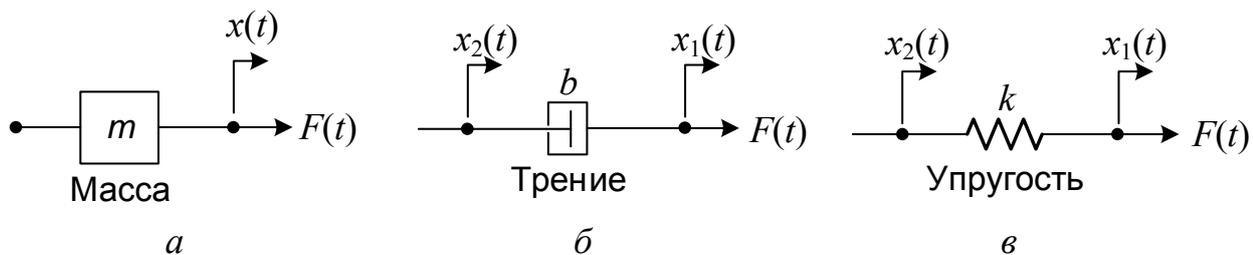


Рис. 2.2. Элементы механических систем с линейным перемещением:  
 $a$  – масса;  $b$  – трение;  $v$  – упругость

Рассмотрим, как составлять уравнения для элемента масса. На рис. 2.2,  $a$   $F(t)$  – приложенная сила,  $x(t)$  – перемещение, а  $m$  – масса. Тогда, в соответствии со вторым законом Ньютона (сумма сил, действующих на тело, равна произведению массы тела на его ускорение), запишем:

$$F(t) = ma(t) = m\dot{v}(t) = m\ddot{x}(t), \quad (2.1)$$

где  $v(t)$  – скорость,  $a(t)$  – ускорение.

Предполагается, что масса является жесткой, т.е. левая точка, с которой она соединена, не может перемещаться относительно правой точки. Следовательно, положение левой точки также равно  $x(t)$ . Для двух других механических элементов левая точка может перемещаться относительно нижней, поэтому для описания движения этих элементов потребуется две переменные.

Физическим аналогом элемента трения (рис. 2.2,  $b$ ) может служить вязкое трение в масле, воздухе и т.д. Сила трения прямо пропорциональна *относительной* скорости элемента, поэтому уравнение, описывающее его поведение во времени имеет вид:

$$F(t) = b[\dot{x}_1(t) - \dot{x}_2(t)], \quad (2.2)$$

где  $b$  – коэффициент демпфирования.

Уравнение движения упругого элемента или пружины (рис. 2.2,  $v$ ) определяется законом Гука, т.е. сила прямо пропорциональна разности положений концов пружины:

$$F(t) = k[x_1(t) - x_2(t)], \quad (2.3)$$

где  $k$  – коэффициент пропорциональности, характеризующий жесткость пружины.

Приведенные уравнения справедливы для сил и перемещений, направление которых на рис. 2.2 обозначено стрелками. Если какое-либо направление меняется на обратное, то в уравнениях надо изменить знак соответствующего члена. Кроме того, полагают, что элементы трения и упругости имеют нулевую массу.

### Пример 1.

Рассмотрим простейший пример [6]. Пусть тележка массой  $m$  перемещается по некоторой горизонтальной поверхности (рис. 2.3) под действием силы  $F(t)$ ; положение тележки характеризуется переменной  $x(t)$ .

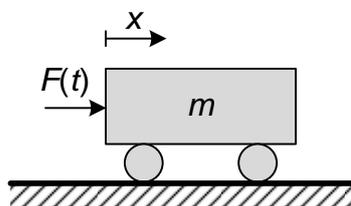


Рис. 2.3. Движущаяся без трения тележка

Предполагается, что тележка перемещается без трения. Тогда уравнение второго закона Ньютона можно записать в виде:

$$\ddot{x}(t) = \frac{F(t)}{m}. \quad (2.4)$$

Согласно приведенному на с. 20 порядку построения дифференциальных уравнений, в правой части уравнения (2.4) мы оставили действующую силу  $F(t)$ , а в левую перенесли вторую производную выходной величины  $\ddot{x}(t)$ .

Теперь, зная массу тележки  $m$ , мы можем определить её положение при любом значении действующей силы  $F(t)$ . Для этого достаточно решить уравнение (2.4), дважды проинтегрировав обе его части. Однако можно воспользоваться возможностями Simulink, который использует графическое описание модели в виде структурных схем. Если читатель помнит, то именно структурные схемы позволяют существенно упростить процесс составления математических моделей.

Попробуем записать уравнение (2.4) в виде структурной схемы, а затем, используя возможности Simulink для его решения.

В самом общем случае, структурная схема, соответствующая уравнению (2.4) изображена на рис. 2.4.

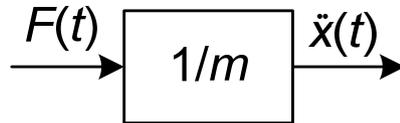


Рис. 2.4. Структурная схема для уравнения (2.4) движения тележки

Однако в данном случае нас интересует не ускорение тележки  $\ddot{x}(t)$ , а её положение  $x(t)$ ; именно эта переменная является выходной величиной, поэтому схема на рис. 2.4 нас не устраивает. Попробуем изменить её.

Подобные структурные схемы удобно начинать с конца. Изобразим в правой части поля схемы стрелку, отображающую выход системы. Поскольку выходная переменная  $x(t)$ , то именно её мы укажем возле стрелки. Каким образом мы можем получить перемещение  $x(t)$ ? Дважды проинтегрировав ускорение  $\ddot{x}(t)$ ! Поместим слева от выходной стрелки два блока интеграторов, соединим их между собой стрелками и обозначим их соответствующими переменными. Полученная структурная схема (рис. 2.5) связывает ускорение тележки с его перемещением.

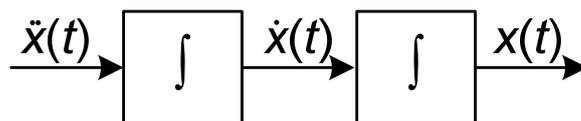


Рис. 2.5. Структурная схема, связывающая перемещение тележки и её ускорение

Теперь объединим схемы на рис. 2.4 и рис. 2.5. Мы получили структурную схему (рис. 2.6), связывающую силу, действующую на тележку с её перемещением (а, заодно, и скоростью).

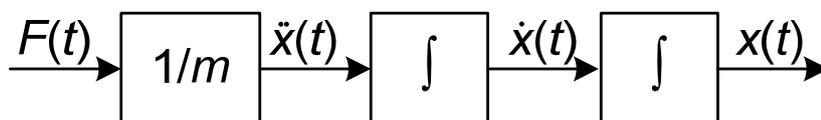


Рис. 2.6. Структурная схема для вычисления положения тележки

Соберем аналогичную схему в Simulink. Для этого нам понадобятся два стандартных блока: блок усилителя Gain и блок интегратора Integrator.

Блок усилителя Gain<sup>1</sup> рис. 2.7 расположен в библиотеке Simulink Math Operations (Математические операции). Выходная величина этого блока прямо пропорциональна входной величине:

$$y(t) = k \cdot x(t), \quad (2.5)$$

где  $k$  - коэффициент усиления.

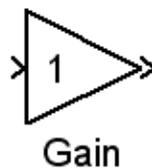


Рис. 2.7. Блок Gain

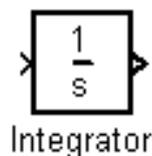
Блок Integrator (Интегратор) располагается в библиотеке элементов Continuous (Непрерывные).

Уравнение, связывающее выходной и входной сигналы блока имеет вид:

$$y(t) = \int_{t_0}^t x(t) dt + y_0, \quad (2.6)$$

где  $x(t)$  – входной сигнал,  $y(t)$  – выходной сигнал,  $y_0$  – начальные условия.

Внутри пиктограммы блока изображена передаточная функция интегрирующего звена  $1/s$  (рис. 2.8).



---

<sup>1</sup> Подробнее об этом и других блоках, а также приемах работы в Simulink можно прочитать в специальной литературе, например, в [7, 8], а также в меню **Help** пакета Simulink.

Рис. 2.8. Изображение блока Integrator

В Simulink 7.8, входящего в состав пакета MATLAB R2011b, который использовался при написании данного пособия, указанные блоки можно также найти в библиотеке Commonly Used Blocks (Часто используемые блоки).

Итак, попробуем собрать схему, изображенную на рис. 2.6 в Simulink. Для этого нужно сначала открыть окно новой системы при помощи кнопки  панели инструментов либо выбрать **File** → **New** → **Model**. Открывшемуся окну с будущей моделью по умолчанию присваивается имя Untitled. Чтобы сохранить создаваемую модель, нажмите кнопку  (**File** → **Save**). В открывшемся окне можно задать свое имя файла, например Model\_1 (модель 1). Обратите внимание, что Simulink (как впрочем, и вообще MATLAB) «не любит» имена, начинающиеся с цифр или содержащие кириллицу. Кроме того, во избежание недоразумений, настоятельно рекомендуем сохранять модели сразу же, в самом начале ее создания. Все дальнейшие изменения можно сохранить повторным нажатием кнопки . При сохранении автоматически создается файл в формате \*.mdl<sup>1</sup>, содержащий всю информацию, необходимую для открытия модели в следующих сеансах работы Simulink.

Теперь приступим непосредственно к созданию модели.

– Выберем и, удерживая левую кнопку мыши, переместим их в окно модели из библиотеки Commonly Used Blocks блоки Gain и два блока Integrator (можно переместить один блок Integrator, а затем воспользоваться функцией копирования блока).

– Разместим эти блоки так, как показано на рис. 2.9.

- Соединим блоки между собой. Для этого поместим курсор на выходной порт (который обозначается символом «>») в правой части блока – источника сигнала (рис. 2.10). При этом курсор примет форму крестика.

- Удерживая нажатой левую кнопку мыши, переместим курсор к входному порту (который обозначен символом «>») на левой стороне блока-приемника. (Второй способ – выделить блок-источник левой кнопкой мыши и, удерживая клавишу <Ctrl>, щелкнуть мышью на блоке-приемнике). Когда соединение будет установлено, на

---

<sup>1</sup> В более поздних версиях Simulink – в формате \*.sfx, однако формат \*.mdl также сохраняется.

соединительной линии появится стрелка, указывающая направление передачи информации (рис. 2.10).

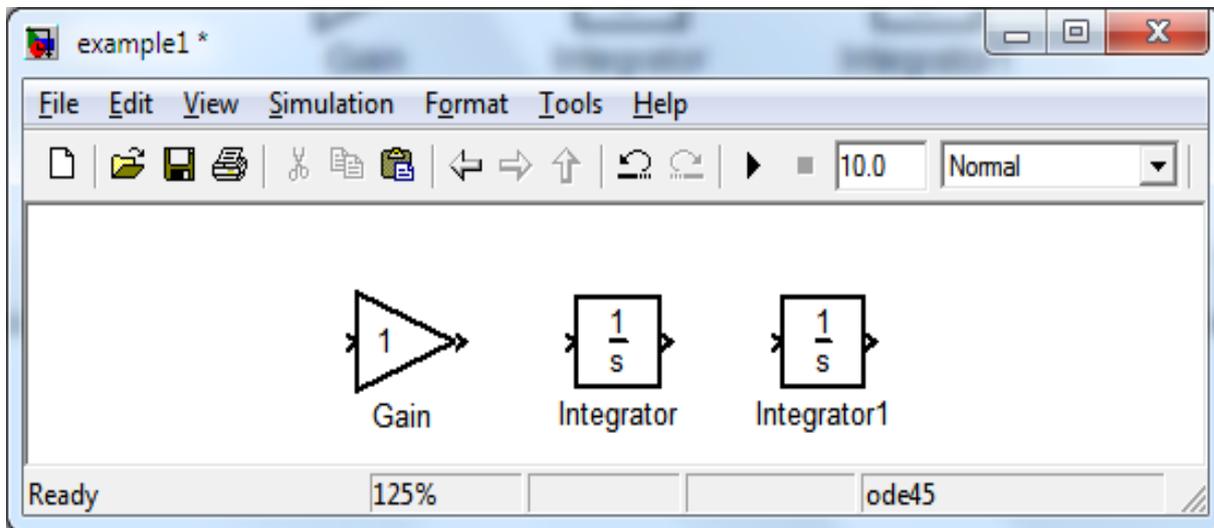


Рис. 2.9. Подготовка модели

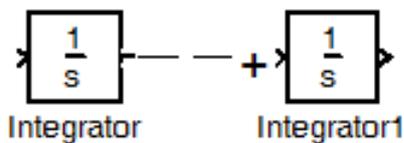


Рис. 2.10. Соединение двух блоков

В результате мы, получим схему, изображённую на рис. 2.11.

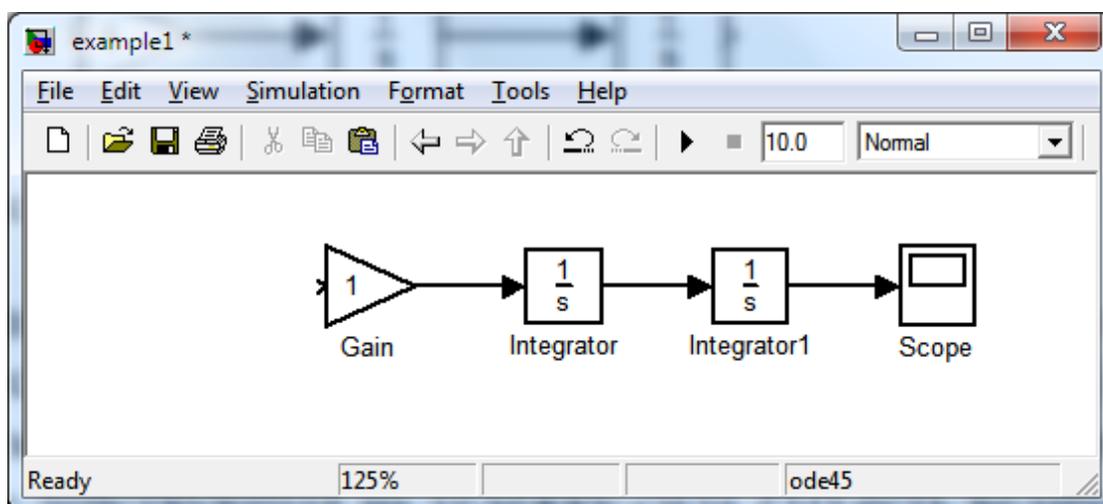


Рис. 2.11. Вид модели после соединения блоков

Однако модель на рис. 2.11 является неполной: на ней необходимо поместить блок, моделирующий источник силы  $F(t)$  и блок, в котором мы будем наблюдать результаты моделирования. Если принять, что действующая на тележку сила  $F(t)$  есть величина постоянная, то в качестве блока-источника удобно принять блок Constant (Константа) из библиотеки источников сигналов Sources. Для отображения результатов моделирования чаще всего используется блок индикатора Scope. Указанные блоки также дублируются в библиотеке Commonly Used Blocks. Выберем эти блоки и также добавим их в нашу модель (рис. 2.12). Она почти готова. Осталось только ввести численные значения массы  $m$  тележки и силы  $F(t)$ .

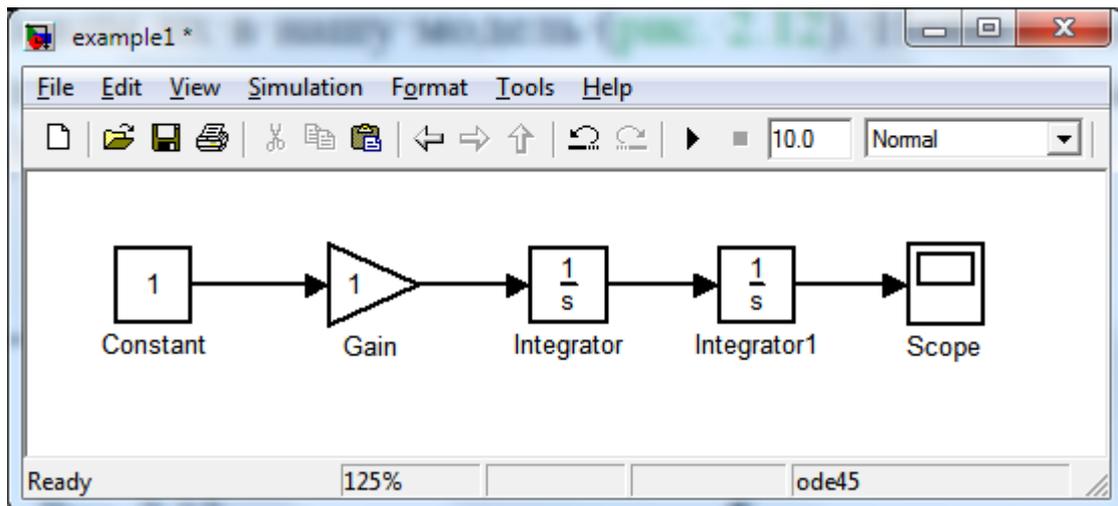


Рис. 2.12. Вид готовой модели

Пусть, к примеру,  $m = 2$  кг и  $F = 1$  Н. Чтобы ввести эти значения, необходимо открыть меню свойств соответствующих блоков дважды щелкнув левой кнопкой мыши на их пиктограммах окне модели.

Блок Constant по умолчанию уже имеет значение константы (Constant value), равное 1 (рис. 2.13), а в текстовом поле **Gain** меню параметров блока Gain укажем значение  $1/2$  (рис. 2.14).

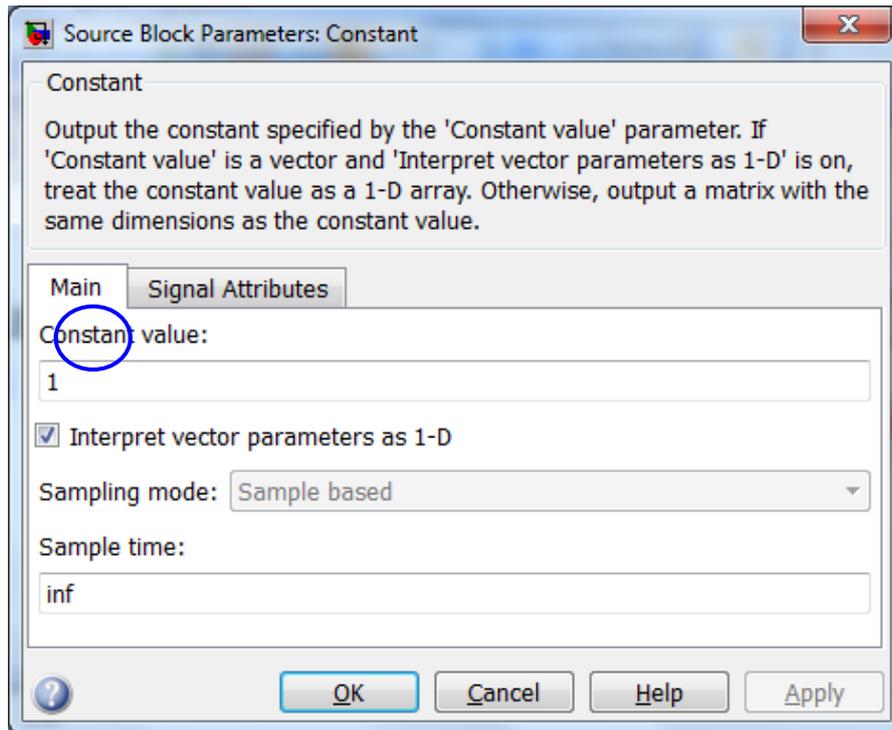


Рис. 2.13. Окно параметров блока Constant

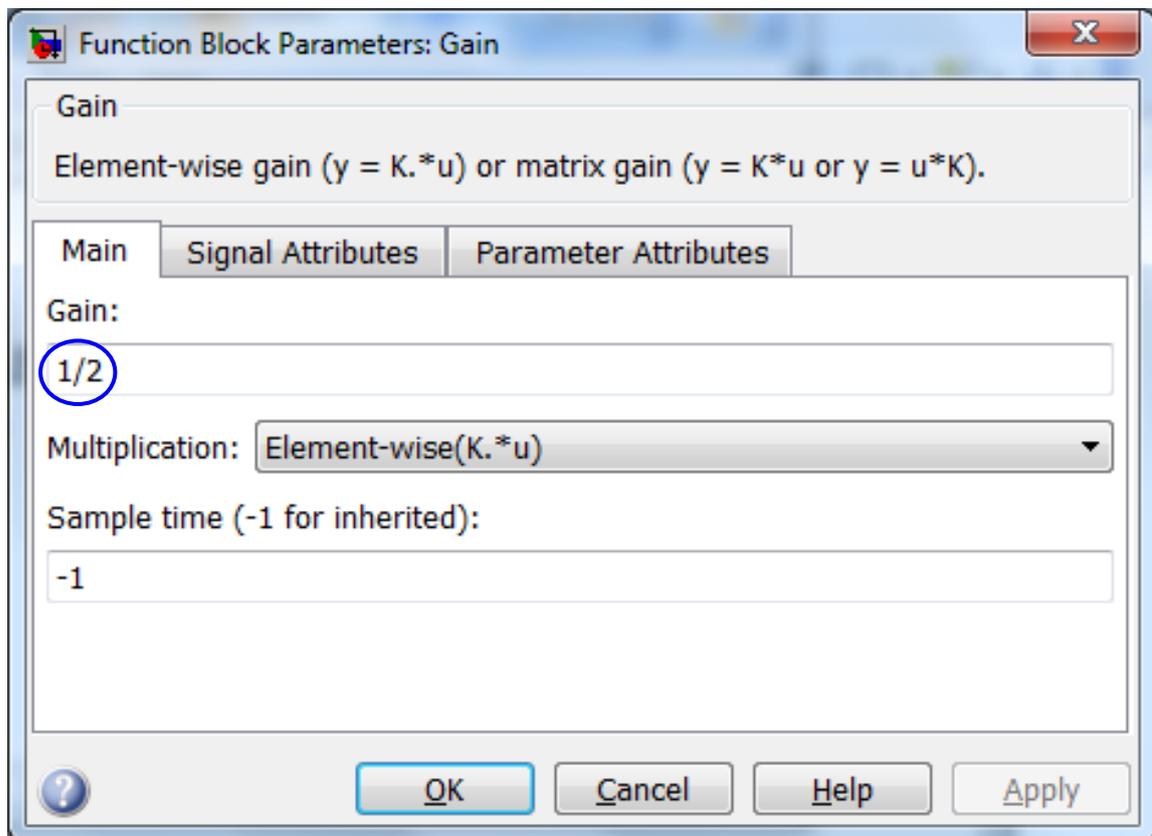


Рис. 2.14. Изменение значения **Gain** в окне параметров блока Gain

В результате должна получиться модель, показанная на рис. 2.15.

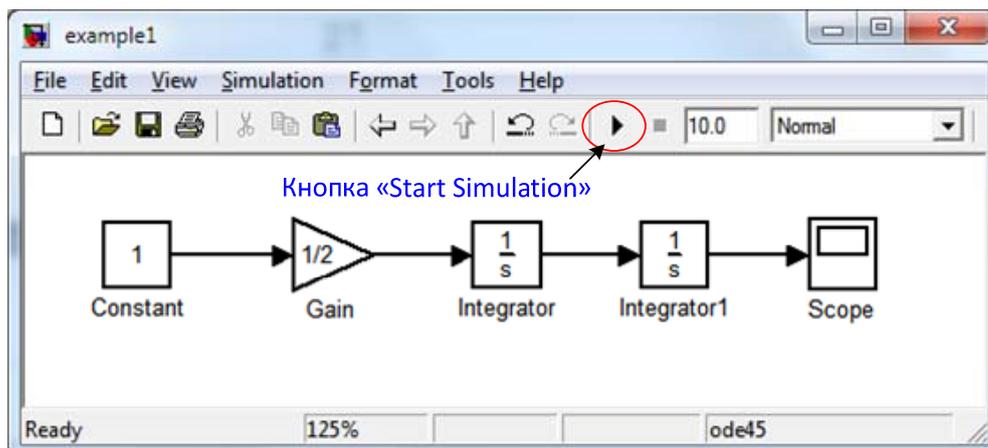


Рис. 2.15. Окончательная модель перемещения тележки

Для начала моделирования нужно выбрать опцию **Simulation** → **Start** из меню или просто нажать кнопку «**Start simulation**»  на панели инструментов (рис. 2.15). Чтобы наблюдать решение уравнения (2.4), откроем индикатор Scope двойным щелчком мыши, при этом на экране появится изображение, приведенное на рис. 2.16. На графике по оси абсцисс отложено время (по умолчанию – 10 с), а по оси ординат – перемещение тележки в метрах.

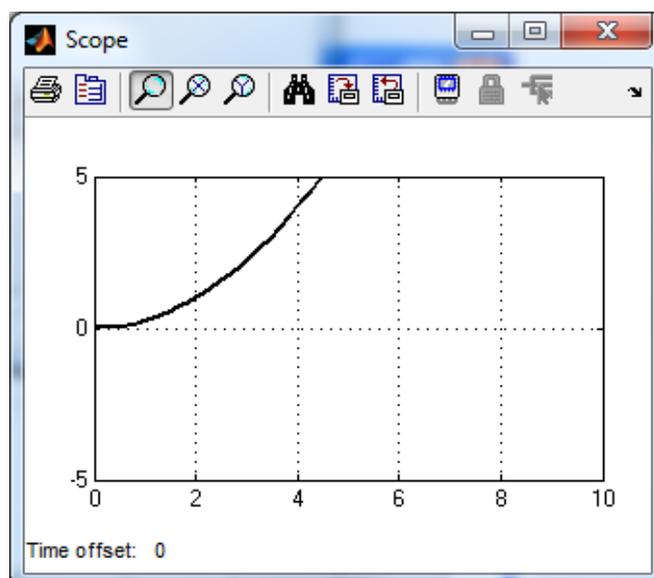
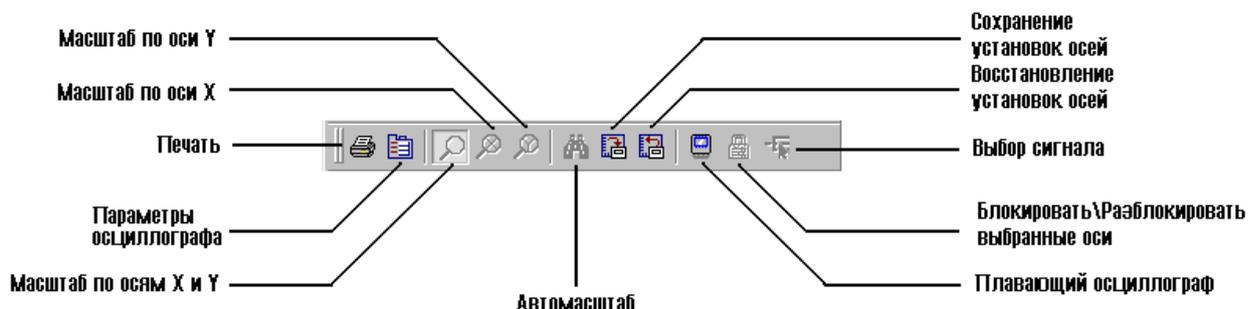


Рис. 2.16. Результат моделирования

Окно индикатора Score имеет свою панель инструментов (рис. 2.17). Если подвести указатель мыши к какой-либо кнопке, всплывает подсказка об ее назначении. Некоторые параметры блока Score мы будем рассматривать ниже.



Рисуно 2.17. Панель инструментов блока Score

Попробуем изменить масштаб изображения. Для этого достаточно нажать на кнопку  Autoscale (Автомасштаб) с изображением. Теперь график стал гораздо нагляднее (рис. 2.18).

Как видим, использование возможностей Simulink для моделирования физических систем довольно просто и удобно.

### Пример 2.

Модель на рис. 2.4 является идеальным случаем.

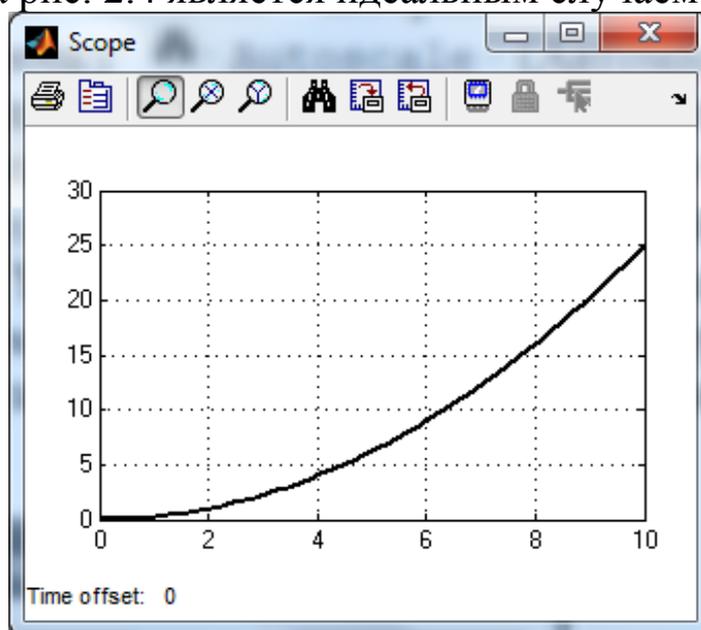


Рис. 2.18. График решения уравнения (2.4) в новом масштабе

В действительности на тележку будут действовать силы сопротивления, например, сила трения. Рассмотрим пример моделирования движения автомобиля. Пусть автомобиль массой  $m$  кг перемещается по горизонтальной дороге под действием силы  $F(t)$ , Н, развиваемой двигателем (рис. 2.19). Движению автомобиля препятствует сила сопротивления, которая зависит от ветра, коэффициента сцепления колес с дорогой и т.д. Эта сила сопротивления характеризуется коэффициентом демпфирования  $b$ , Нс/м. Автомобиль движется со скоростью  $v(t)$ , м/с. Примем для нашей системы следующие значения параметров:  $m = 1100$  кг,  $b = 40$  Нс/м.

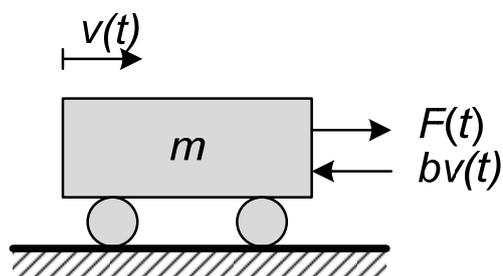


Рис. 2.19. Упрощенная модель движения автомобиля

Запишем уравнение движения данной системы, при помощи алгоритма, приведенного на стр. 20.

1. Входной величиной является внешняя сила  $F(t)$ , а выходной – скорость автомобиля  $v(t)$ .

2. Поскольку рассматривается механическая система с линейным перемещением, то в качестве фундаментального закона будем использовать второй закон Ньютона.

3. Запишем уравнение движения на основе второго закона Ньютона:

$$m\dot{v}(t) = F(t) - bv(t). \quad (2.7)$$

Приведем уравнение (2.7) к виду, удобному для построения структурной схемы:

$$\dot{v}(t) = \frac{F(t) - bv(t)}{m}, \quad (2.8)$$

Построим в Simulink структурную схему, соответствующую этому уравнению. Как мы уже упоминали, схему удобнее строить с конца.

- Перенесем в окно модели блок Integrator (рис. 2.20) из библиотеки Continuous (или Commonly Used Blocks).

- Подведем также линии связи, они будут красного цвета, так как их вторые концы не подсоединены ни к какому блоку.

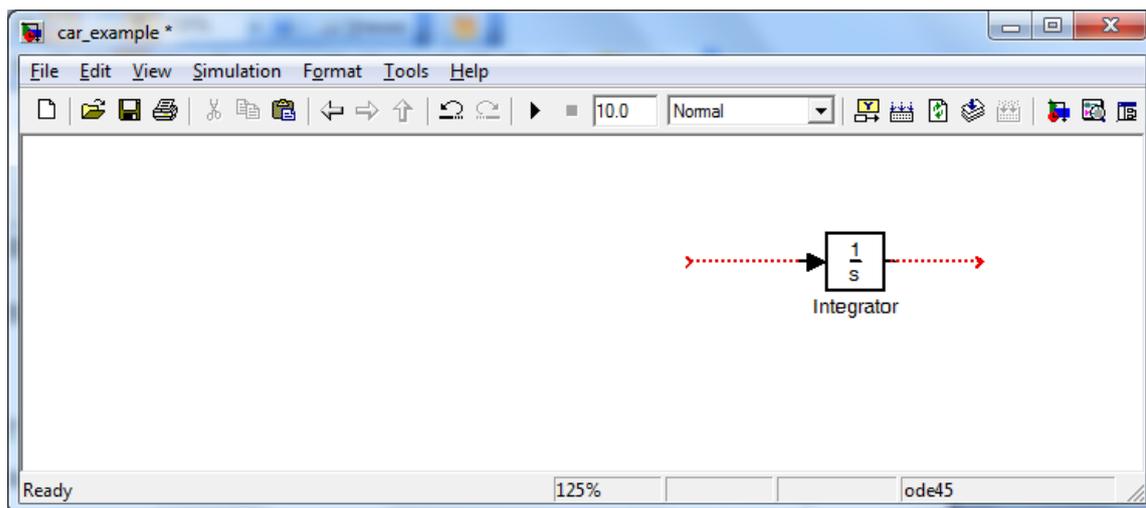


Рис. 2.20. Начало построения модели движения автомобиля

- Выходной сигнал интегратора – это скорость автомобиля  $v(t)$ . Добавим теперь в окно модели блоки Sum и Gain (рис. 2.21).

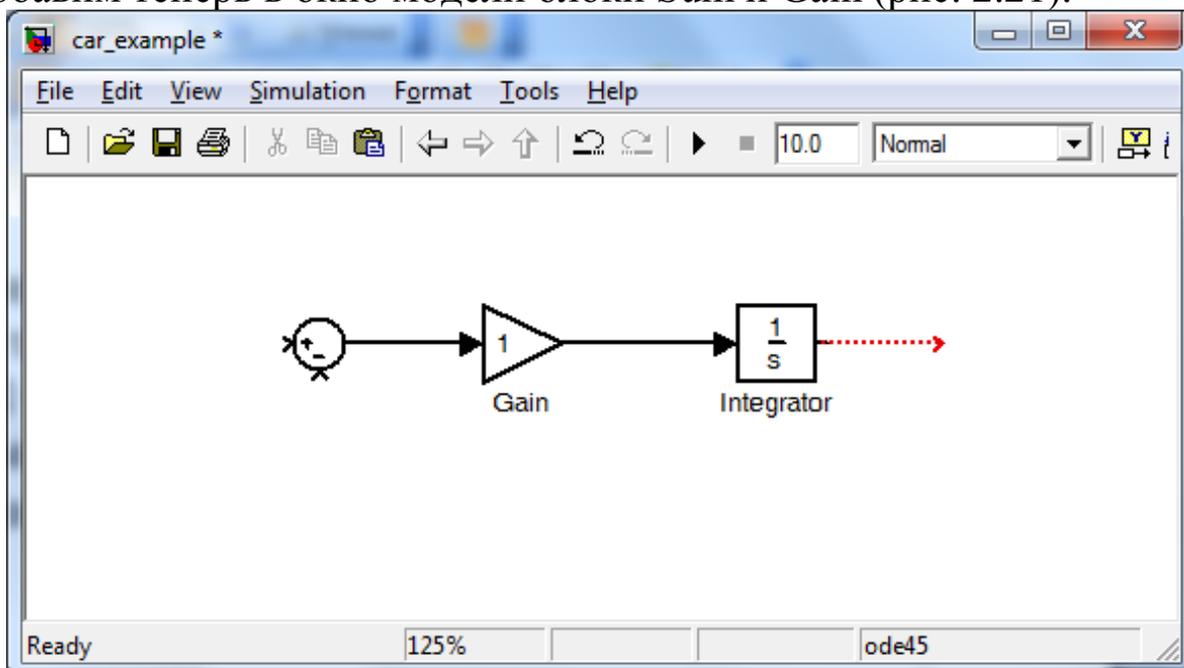


Рис. 2.21. Вид модели после добавления блоков Sum и Gain

Оба этих блока находятся в библиотеке Math Operations и продублированы в Commonly Used Blocks. Блок Sum необходим для того, чтобы сложить силы  $F(t)$  и  $bv(t)$ , а блок Gain потребуется для деления на массу. С блоком Gain мы уже сталкивались в предыдущем примере, поэтому остановимся подробнее на блоке Sum.

Блок Sum служит для нахождения алгебраической суммы двух или более входных переменных, каждой из которых присваивается знак операции сложения «+» или вычитания «-». Так, для блока Sum, показанного на рис. 2.22, справедливо выражение

$$x = u - y. \quad (2.9)$$

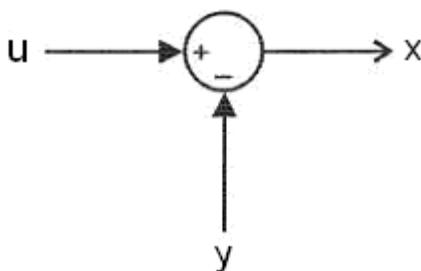


Рис. 2.22. Блок Sum

Блок Sum должен иметь, по крайней мере, один входной и один выходной порты. Окно его свойств приведено на рис. 2.23.

На главной вкладке **Main** окна свойств блока Sum можно указать следующие параметры блока.

**Icon shape** – этот раскрывающийся список позволяет выбрать форму блока:

- **round** – окружность,
- **rectangular** – прямоугольник.

**List of signs** – в этом поле задается список знаков, которые определяют количество входов и задают арифметические действия над соответствующими входными сигналами блока. В списке можно использовать следующие знаки: + (плюс), - (минус) и | (разделитель знаков, его положение в списке определяет, какой входной порт блока будет закрыт). На рис. 2.24 показан вид поля **List of signs** блока Sum, имеющего один положительный и один отрицательный входы.

Количество входов можно также задать константой. В этом случае все входы будут суммирующими. Если в качестве списка

знаков указать цифру 1 (один вход), то блок можно использовать для определения суммы элементов вектора. При этом внутри пиктограммы блока появится знак  $\Sigma$ .

Вернемся, теперь, к блоку Gain. Откроем двойным щелчком окно его параметров и введем коэффициент усиления 1/1100 (деление на 1100 эквивалентно умножению на 1/1100).

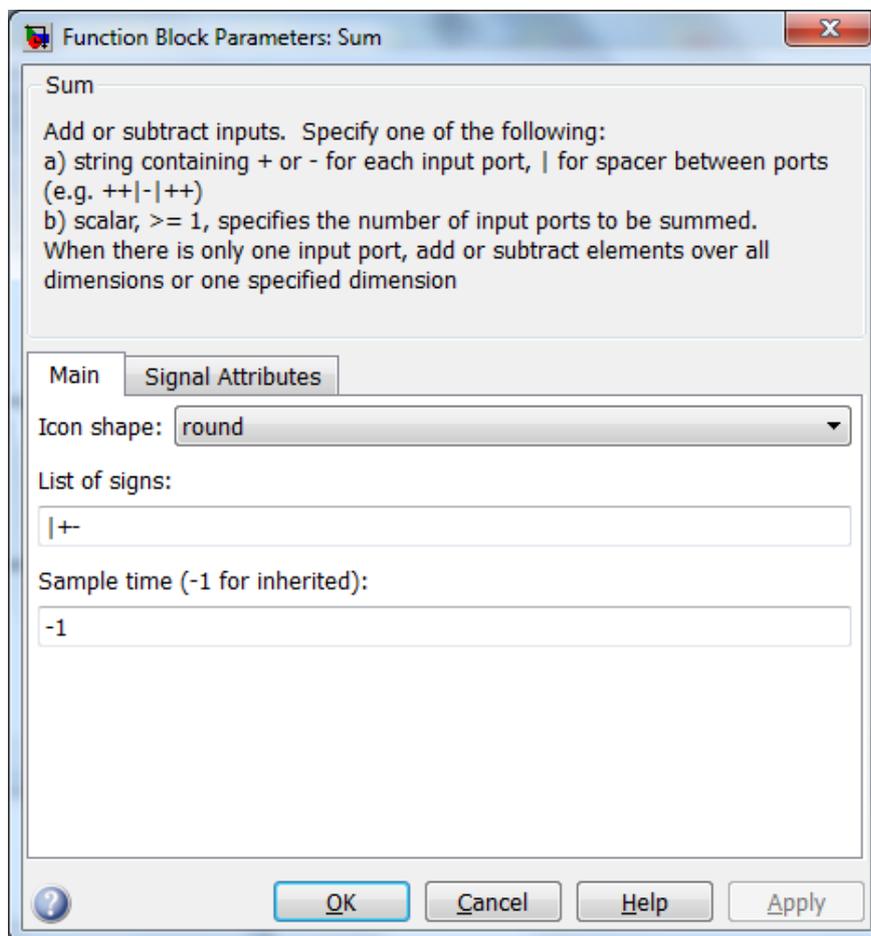


Рис. 2.23. Окно параметров блока Sum

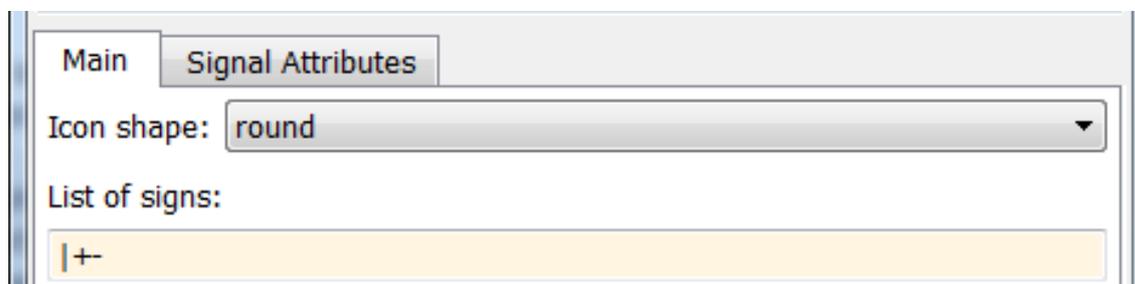


Рис. 2.24. Изменение знаков входов в блоке Sum

Если содержимое блока слишком длинное и не помещается в границах пиктограммы стандартного размера, то внутри

пиктограммы появляется символ или надпись, характеризующий свойства блока; для блока Gain это символ коэффициента усиления  $K$  (рис. 2.25, а). В принципе, можно работать и с таким изображением блока, однако для удобства, лучше изменить его размеры. Чтобы изменить размер, блок нужно выделить, после чего установить курсор мыши на маркеры по его углам. Как только курсор мыши превратится в двунаправленную диагональную стрелку, можно увеличивать, или уменьшать размер блока, удерживая левую кнопку мыши (рис. 2.25, б). Обратите внимание на то, что растягивается только пиктограмма блока, а размеры его названия в виде текстовой надписи не изменяются.

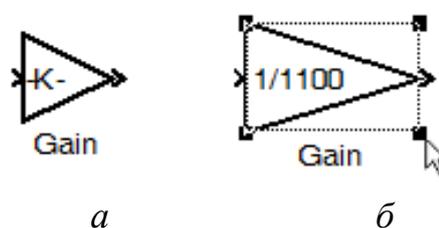


Рис. 2.25. Изменение размера блока

Поскольку наша модель будет достаточно большой и содержать много линий связи, то желательно указать имена сигналов, которые проходят по этим линиям. Для этого нужно дважды щелкнуть левой кнопкой мыши на нужной линии, при этом появится рамка, определяющая область текстового блока, и курсор ввода (рис. 2.26). Теперь можно ввести нужную надпись. Для перехода на новую строку нужно нажать **Enter**. Щелчок мышью вне текстовой области позволяет выйти из режима ввода текста. Введенное имя сигнала можно перетянуть в любое место около соответствующей линии связи.

Как показывает опыт, Simulink, как и вообще MATLAB, не всегда корректно работает с кириллицей. Поэтому без веских оснований пользоваться символами кириллицы не рекомендуется.

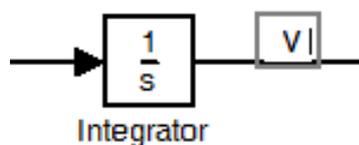


Рис. 2.26. Ввод имени сигнала

Таким же образом можно изменять и имена блоков. Для этого достаточно щелкнуть левой кнопкой мыши на имени блока. Подпишем линии связи так, как показано на рис. 2.27.

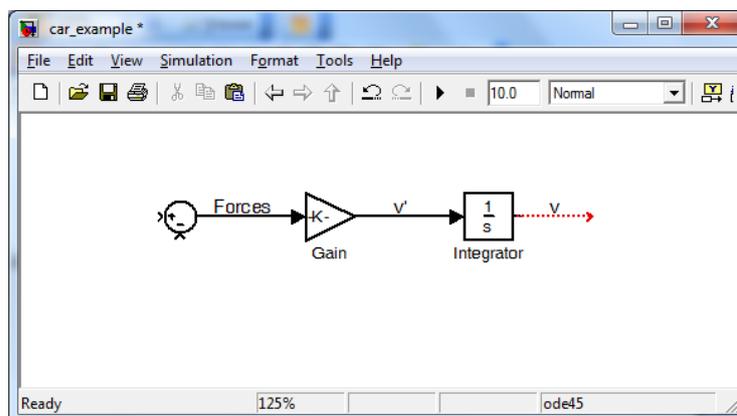


Рис. 2.27. Вид модели после добавления пояснительных надписей

Для получения силы сопротивления, необходимо сделать ответвление от сигнала скорости и умножить его на коэффициент демпфирования  $b$ . Для создания ответвления линии связи в том месте линии связи, где нужно создать ответвление, надо нажать правую кнопку мыши (или удерживая клавишу **<Ctrl>** использовать левую кнопку мыши), и удерживая её, потянуть ответвление в желаемом направлении. При этом указатель мыши принимает вид креста (рис. 2.28).

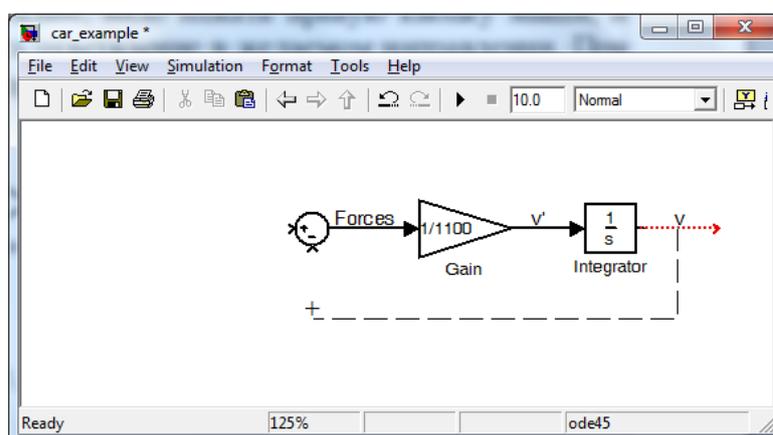


Рис. 2.28. Ответвление линии связи

Теперь в образовавшееся ответвление надо вставить блок усиления Gain с параметром  $b = 40$ , при этом блок должен изменить свою ориентацию. Это можно сделать несколькими способами.

1. Перетянуть новый блок на нужное место посреди соединительной линии (рис. 2.29). При этом блок автоматически примет нужную ориентацию, а соединительная линия найдет входной и выходной порты блока. Однако подобная простая вставка возможна для блоков, имеющих только один вход и один выход.

2. Большинство операций по форматированию блоков доступно в меню **Format** в строке меню или в контекстном меню, которое вызывается щелчком правой кнопки мыши, и набор его команд зависит от элемента, на котором щелкнули мышью, т.е. от контекста. В состав меню **Format** контекстного меню блоков входят команды **Flip block** и **Rotate block**. Команда **Flip block** осуществляет поворот блока на  $180^\circ$ . При этом входные и выходные порты блока меняют свое местоположение. Команда **Rotate block** производит поворот блока на  $90^\circ$  по часовой стрелке (Clockwise) или против часовой стрелки (Counterclockwise). При этом имя блока располагается сбоку от его изображения (рис. 2.30).

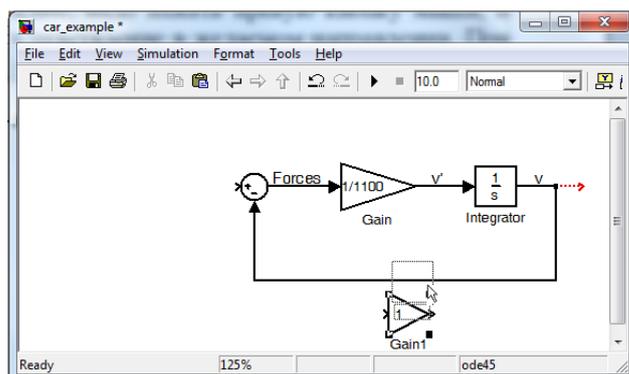


Рис. 2.29. Вставка блока в линию связи

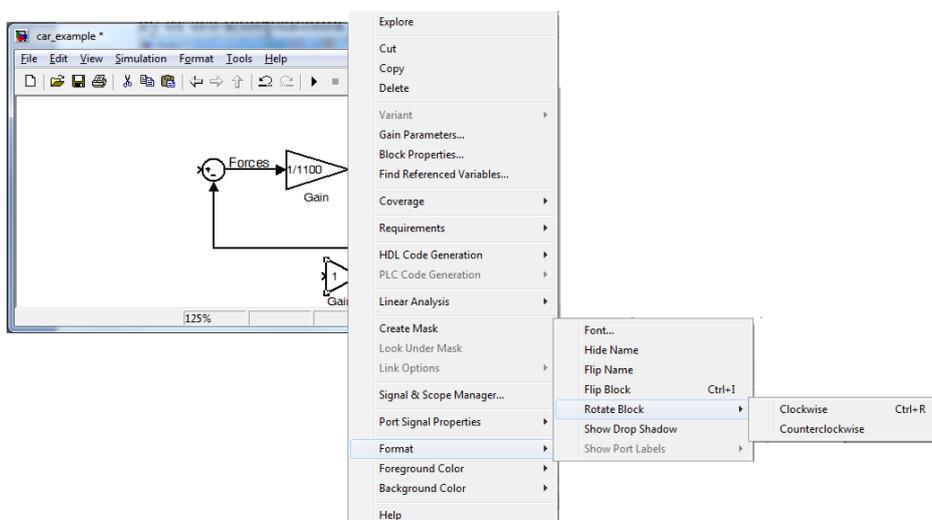


Рис. 2.30. Контекстное меню для блока Gain

Теперь изменим коэффициент усиления блока Gain на значение, равное коэффициенту демпфирования системы (40 Н·с/м). После всех изменений модель должна выглядеть так, как показано на рис. 2.31.

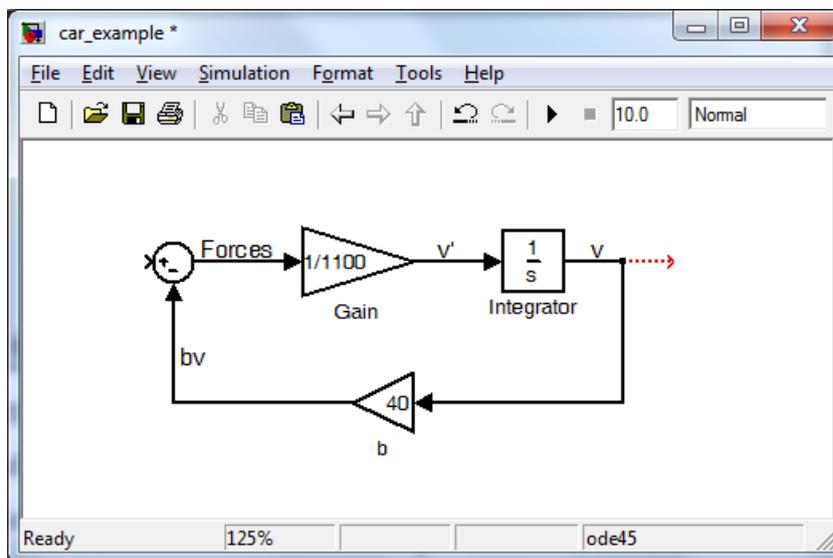


Рис. 2.31. Вид модели после вставки блока в линию обратной связи

Перед запуском моделирования нам еще необходимо установить блок, генерирующий входной сигнал, т.е.  $F$ . Предположим, что в начальный момент времени (при  $t = 0$ ) автомобиль находился в состоянии покоя, и двигатель вырабатывает ступенчатое воздействие  $F(0) = 400$  Н. Это приблизительно соответствует ситуации, когда водитель, стартуя от светофора, быстро нажимает на педаль газа и удерживает её в постоянном положении. Для моделирования такого входного сигнала можно использовать уже известный нам блок Constant, однако, в данном случае, мы применим блок ступенчатого сигнала Step из библиотеки источников сигналов Sources.

- Добавим в модель блок Step и соединим его выход с первым (положительным) входом блока Sum.
- Присвоим имя «F» линии связи блоков Step и Sum.
- Перенесем также в нашу модель блок индикатора Scope из библиотеки приемников сигналов Sinks и подадим на его вход выходной сигнал блока Integrator, чтобы иметь возможность наблюдать скорость автомобиля  $v(t)$ . Simulink-модель теперь должна выглядеть так, как показано на рис. 2.32.

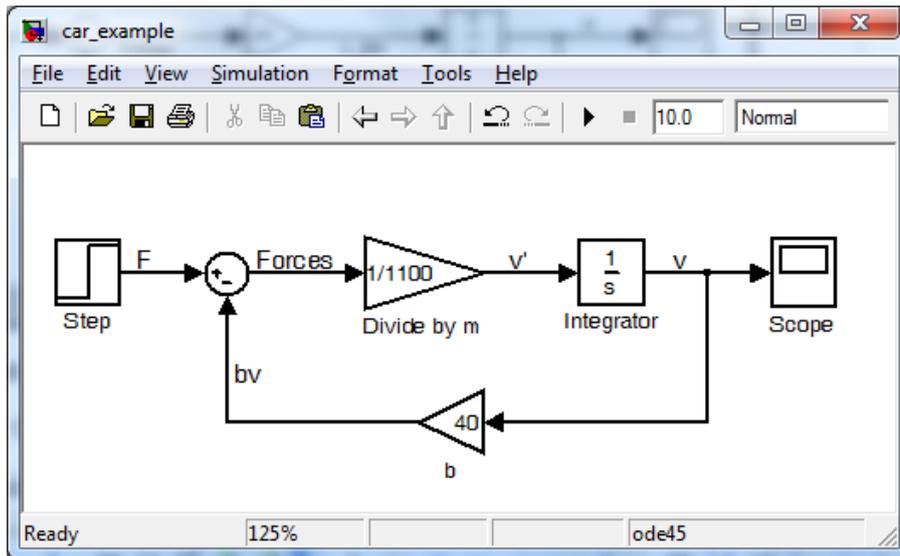


Рис. 2.32. Окончательный вид модели

- Изменим параметры блока Step так, чтобы он генерировал требуемый нам входной сигнал. Дважды щелкнув левой кнопкой мыши на блоке, откроем окно его параметров и установим время подачи сигнала **Step time** на 0 и конечное значение сигнала **Final value** на 400 (рис. 2.33). Значение в поле **Initial value** (начальное значение) оставим равным 0, т.к. сила  $F$  изменяется ступенчато от 0 в момент  $t = 0$ . Значение времени квантования в поле **Sample time** также оставим равным 0, т.к. мы рассматриваем непрерывную, а не дискретную модель.

Теперь запустим моделирование (нажатием кнопки **Start simulation** или выбрав **Simulation** → **Start**). По окончании моделирования дважды щелкнем на блоке индикатора Scope, чтобы наблюдать изменение скорости автомобиля при ступенчатом входном сигнале. После нажатия кнопки автомасштаба **Autoscale** (с изображением бинокля), график должен выглядеть так, как показано на рис. 2.34.

Как видим, этот график не показывает установившегося значения скорости автомобиля  $v(t)$  по причине того, что, по умолчанию, время моделирования составляет 10 секунд. Чтобы наблюдать установившееся значение скорости, изменим время моделирования на 150 секунд. Изменить время моделирования можно в специальном текстовом поле на панели инструментов (рис. 2.35) или в поле **Stop time** выбрав в меню **Simulation** → **Configuration Parameters**.

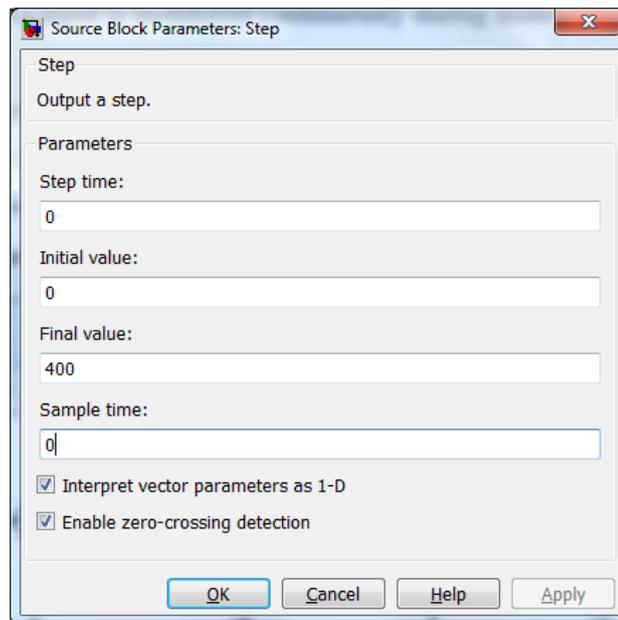


Рис. 2.33. Изменение параметров блока Step

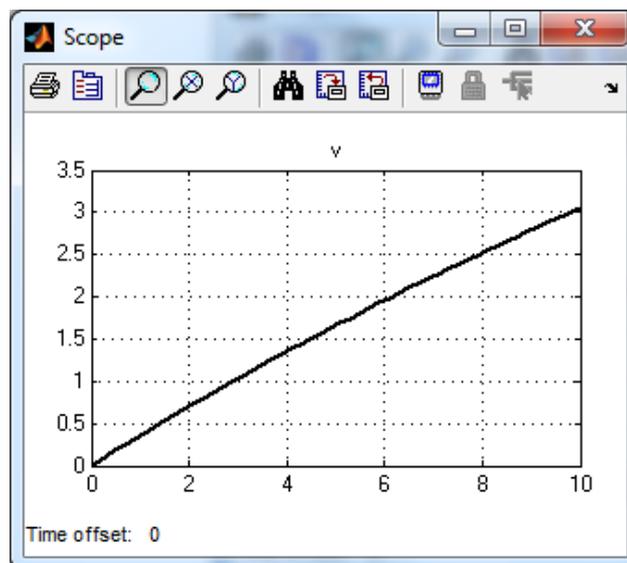


Рис. 2.34. Результат моделирования



Рис. 2.35. Изменение времени моделирования

Теперь перезапустим моделирование, результат которого изображен на рис. 2.36.

Из графика на рис. 2.36 видно, что установившаяся скорость системы составляет 10 м/с.

Теперь рассмотрим реакцию системы не на ступенчатый, а на импульсный входной сигнал. Это приблизительно соответствует случаю, когда водитель нажимает педаль газа и удерживает её в постоянном положении в течение определенного периода времени, а затем отпускает.

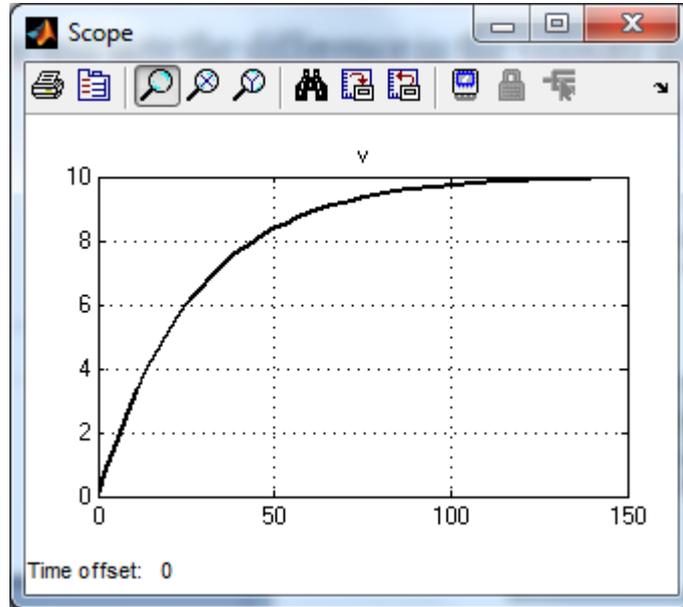


Рис. 2.36. Результат моделирования после изменения времени моделирования

Для построения модели источника импульсного сигнала, добавим в модель еще по блоку Step и Sum, как показано на рис. 2.37.

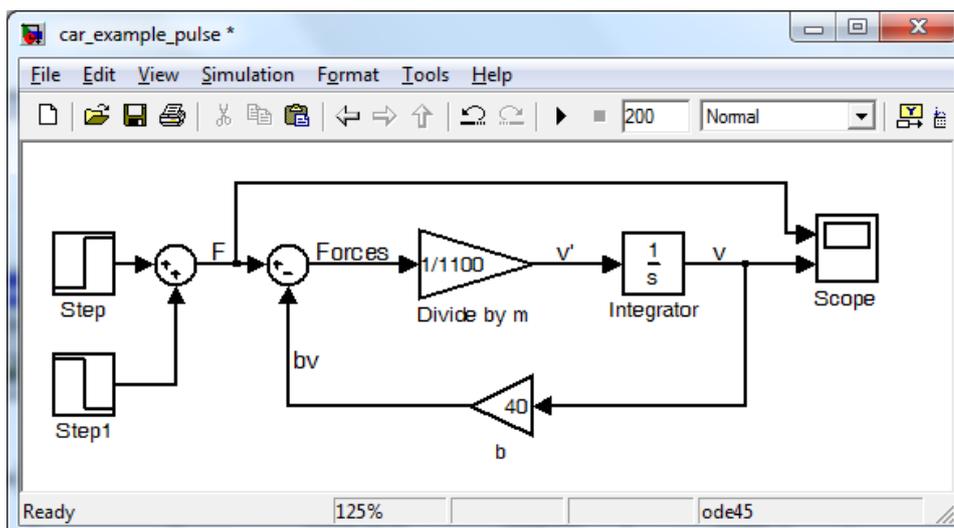


Рис. 2.37. Вид модели после добавления блока Step

Параметры первого блока Step оставим неизменными, какими они были в предыдущей модели, а параметры нового блока Step1 изменим следующим образом:

Step time: 100

Initial value: 0

Final value: -400

Эти значения параметров блока Step1 устраняют входное воздействие от блока Step с момента  $t = 100$ <sup>1</sup>.

Для наблюдения импульсного входного сигнала  $F(t)$  можно использовать новый блок Scope, а можно использовать уже имеющийся блок, добавив к нему еще один вход. Для этого вызовем окно параметров блока Scope нажатием кнопки **Parameters**  на панели инструментов графического окна. На основной вкладке **General** в текстовом поле **Number of axes** (Число осей) окна параметров блока Scope указывается требуемое количество каналов. При этом у индикатора появляется соответствующее число входов и для каждого входа строится соответствующий график. В нашем случае в поле **Number of axes** задаем число 2 (рис. 2.38).

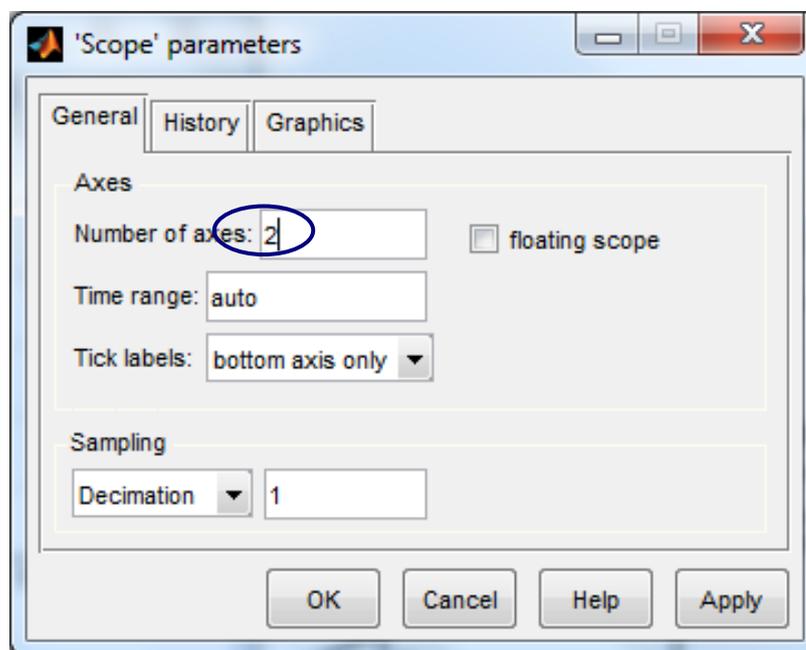


Рис. 2.38. Изменение числа осей блока Scope

<sup>1</sup> Можно также воспользоваться блоком Signal Builder

Изменим в специальном поле на панели инструментов время моделирования на 200 секунд (или выберем **Simulation** → **Configuration Parameters**), и запустим моделирование. После использования функции автомасштаба, графики силы  $F$  и скорости  $v$  должны выглядеть так, как показано на рис. 2.39.

Рассмотрим рис. 2.39 более подробно. Системе для достижения установившегося режима требуется около 100 секунд. В момент времени  $t = 100$  секунд, ошибка составляет около 2,5% от установившегося режима в 10 м. Это можно увидеть, нажав на панели инструментов графического окна кнопку Zoom , и выделив требуемую область (рис. 2.40). Когда сила  $F$  мгновенно падает до 0 в момент времени  $t = 100$  секунд, системе необходимо еще 100 секунд (до  $t = 200$  с) для отработки этого нового входного сигнала. Заметим, также, что установившееся значение скорости при входном сигнале  $F = 0$  равно  $v = 0$ . Физически это означает, что если при движении автомобиля с любой скоростью водитель отпускает педаль газа, автомобиль, в конце концов, остановится за счет действия сил сопротивления движению.

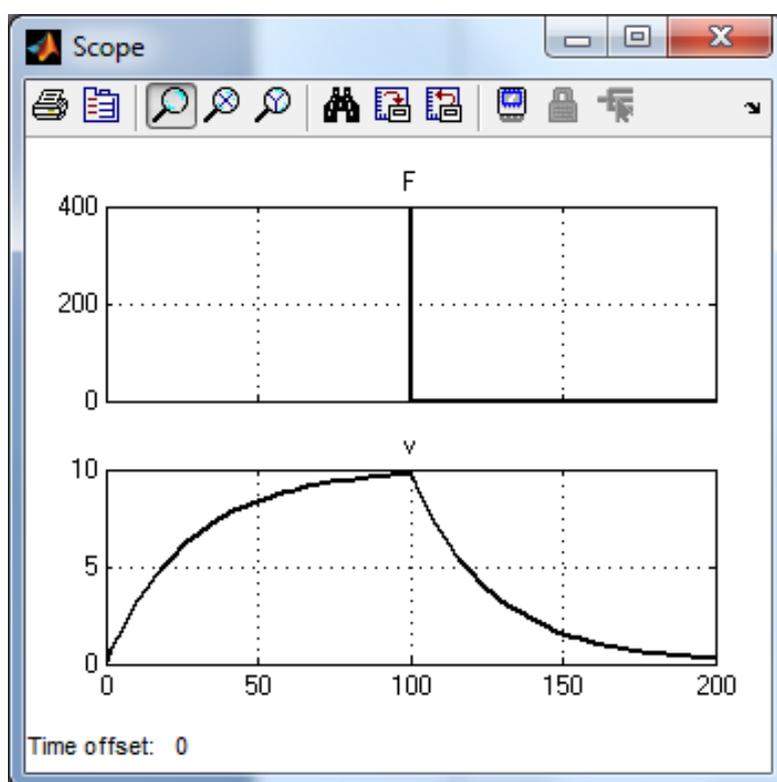


Рис. 2.39. Изменение числа осей блока Scope

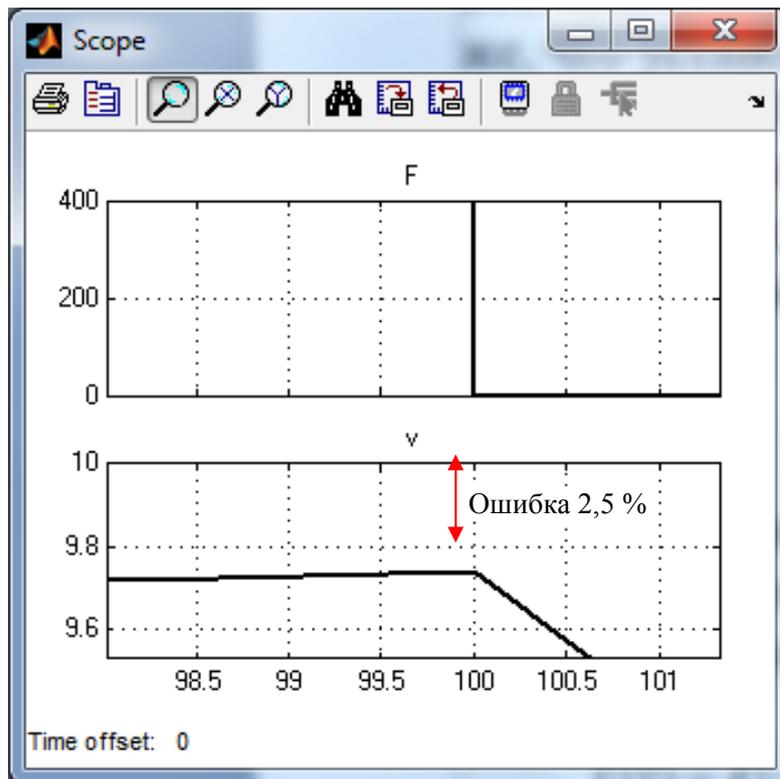


Рис. 2.40. Увеличение области графика и определение ошибки

### Пример 3.

В качестве итогового примера на моделирование механических систем с линейным перемещением построим модель подвески автобуса [9].

Обычно при проектировании системы рассматривается ее модель в расчете на одно колесо (так наз.  $\frac{1}{4}$  автобуса) и подвеска рассматривается в виде одномерной системы «пружина-демпфер». Схема такой системы представлена на рис. 2.41.

Значения приведенные на рис. 2.41 параметров, следующие:

$m_1$  – масса автобуса,  $m_1 = 2500$  кг,

$m_2$  – масса подвески,  $m_2 = 320$  кг,

$k_1$  – коэффициент упругости подвески,  $k_1 = 80\,000$  Н / м,

$k_2$  – коэффициент упругости колеса и шины,  $k_2 = 500\,000$  Н / м,

$b_1$  - коэффициент демпфирования подвески,  $b_1 = 350$  Нс/ м.

$b_2$  - коэффициент демпфирования колеса и шины,  
 $b_2 = 15020$  Нс/ м

$u$  – управляющее воздействие, которое надо найти.

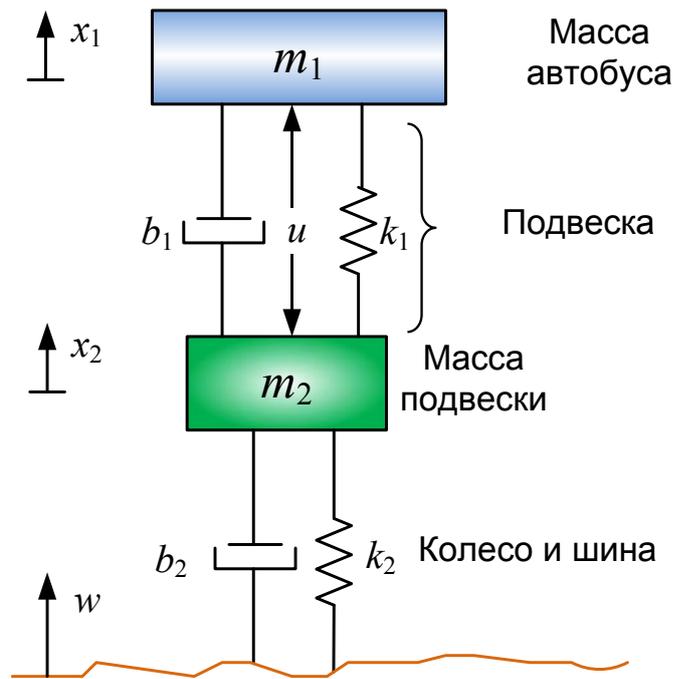


Рис. 2.41. Расчётная схема подвески автобуса

Функциональная схема подвески автобуса будет иметь вид (рис. 2.42).

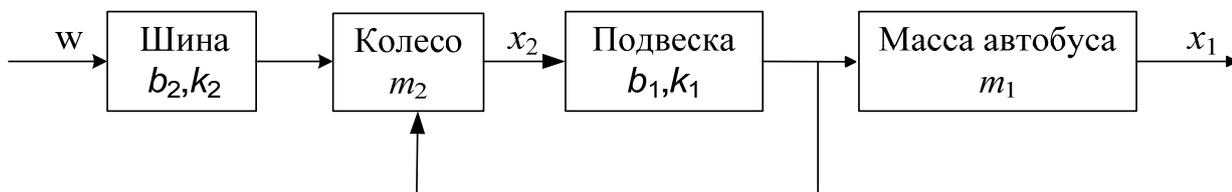


Рис. 2.42. Функциональная схема подвески автобуса

### Требования к системе.

Хорошая подвеска автобуса должна иметь удовлетворительное сцепление с дорогой, обеспечивая при этом комфорт при езде по неровностям дороги. Когда автобус наезжает на неровность дороги, его кузов не должен иметь большую амплитуду колебаний и колебания должны быстро рассеиваться. Так как расстояние  $x_1 - w$  очень трудно измерить, и деформацией шины ( $x_2 - w$ ) можно пренебречь, мы в качестве выходной величины вместо  $x_1 - w$  будем использовать расстояние  $x_1 - x_2$ .

Возмущения от дороги  $w$  в этой задаче будет моделироваться единичной ступенчатой функцией. Например, ступенькой можно представить входной сигнал при выезде автобуса из выбоины.

### Уравнения движения.

В данном случае нам нужны два уравнения, т.к. имеют место два независимых перемещения  $x_1(t)$  и  $x_2(t)$ . Применяя второй закон Ньютона к каждой массе, мы можем получить следующие уравнения динамики:

$$m_1 \ddot{x}_1 = -b_1(\dot{x}_1 - \dot{x}_2) - k_1(x_1 - x_2) + u, \quad (2.9)$$

$$m_2 \ddot{x}_2 = b_1(\dot{x}_1 - \dot{x}_2) + k_1(x_1 - x_2) + b_2(\dot{w} - \dot{x}_2) + k_2(w - x_2) - u. \quad (2.10)$$

### Построение модели в Simulink

Разделим обе части уравнений (2.9 – 2.10) на массы:

$$\ddot{x}_1 = \frac{1}{m_1}(-b_1(\dot{x}_1 - \dot{x}_2) - k_1(x_1 - x_2) + u) \quad (2.11)$$

$$\ddot{x}_2 = \frac{1}{m_2}(b_1(\dot{x}_1 - \dot{x}_2) + k_1(x_1 - x_2) + b_2(\dot{w} - \dot{x}_2) + k_2(w - x_2) - u) \quad (2.12)$$

Построим, теперь схему, позволяющую решать эти уравнения в Simulink.

Начнем с блока Integrator (библиотеки Continuous или Commonly Used Blocks) и нарисуем линии связи от его выхода и к его входу.

Обозначим входную линию как «v1» (скорость) и выходную линию как x1 (положение массы  $m_1$ ). Напомним, что для добавления имени сигнала, проходящего по линии связи, нужно дважды щелкнуть левой кнопкой мыши по этой линии.

Перенесем еще один блок Integrator и соединим его со входом первого блока. Нарисуем входную линию связи и обозначим её как «a1» (ускорение массы  $m_1$ ).

Выделим левой кнопкой мыши и скопируем эти блоки и вставим чуть ниже, заменив имена соответствующих линий связи на «x2», «v2» и «a2» соответственно (рис. 2.43).

Теперь обратимся к уравнениям (2.11) и (2.12). Из них видно, что нам необходимо добавить к модели два блока Gain (для выполнения действий  $1/m_1$  and  $1/m_2$ ) и два блока суммирования Sum (для сложения сил записанных в скобках).

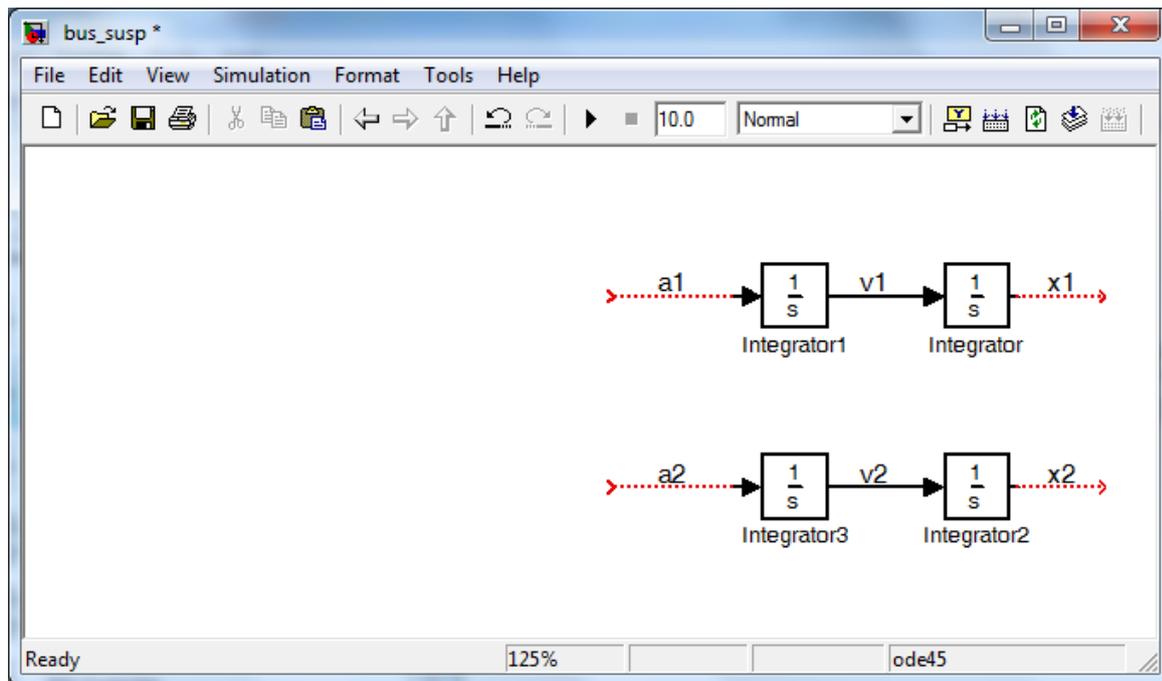


Рис. 2.43. Начало построения модели подвески автобуса

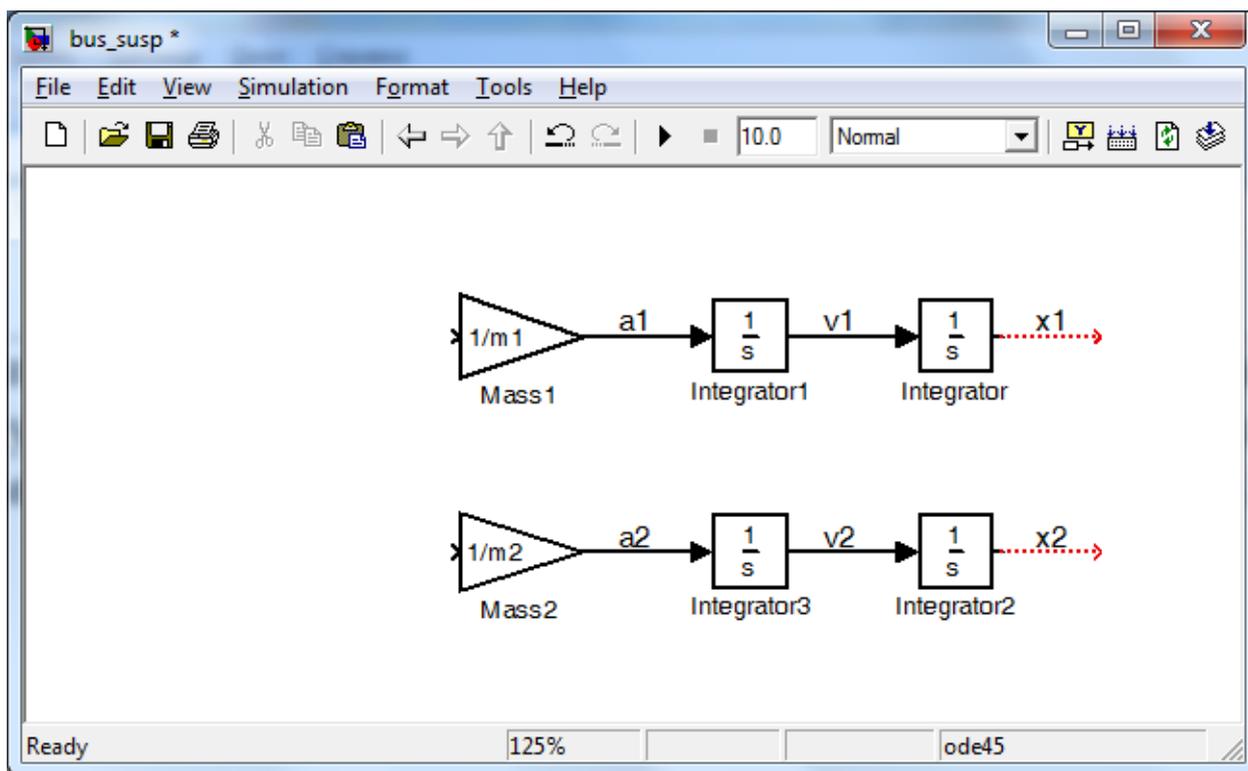


Рис. 2.44. Начало построения модели подвески автобуса

Перенесем в окно модели два блока Gain (из библиотеки Math Operations или Commonly Used Blocks) и соединим их выходы со свободными входами интеграторов (рис. 2.44).

Изменим коэффициент усиления первого блока Gain на  $1/m_1$ , вызвав окно параметров двойным щелчком мыши.

Изменим имя этого блока с Gain на «Mass1», щелкнув мышью на имени «Gain» под блоком.

Подобным образом изменим параметр другого блока Gain на  $1/m_2$  и изменим его имя на «Mass2».

Можно изменить размеры блоков Gain, чтобы в них отображались значения коэффициентов усиления. Напомним, что изменить размер блока можно, выделив его щелчком мыши, и растянув границы блока до желаемого размера.

Как видно из рис. 2.41 и уравнений (2.11)-(2.12) на массу  $m_1$  действуют три силы (сила упругости, сила трения и управляющая сила  $u$ ), а на массу  $m_2$  - пять сил, (две упругие, две трения и управляющая сила  $u$ ).

Переместим два блока Sum (из библиотеки Math Operations или Commonly Used Blocks) в окно модели, и присоединим по одному блоку к блокам Mass 1 и Mass 2.

В поле введем «+--» (рис. 2.45, а), что соответствует трем силам (две из которых будут отрицательными).

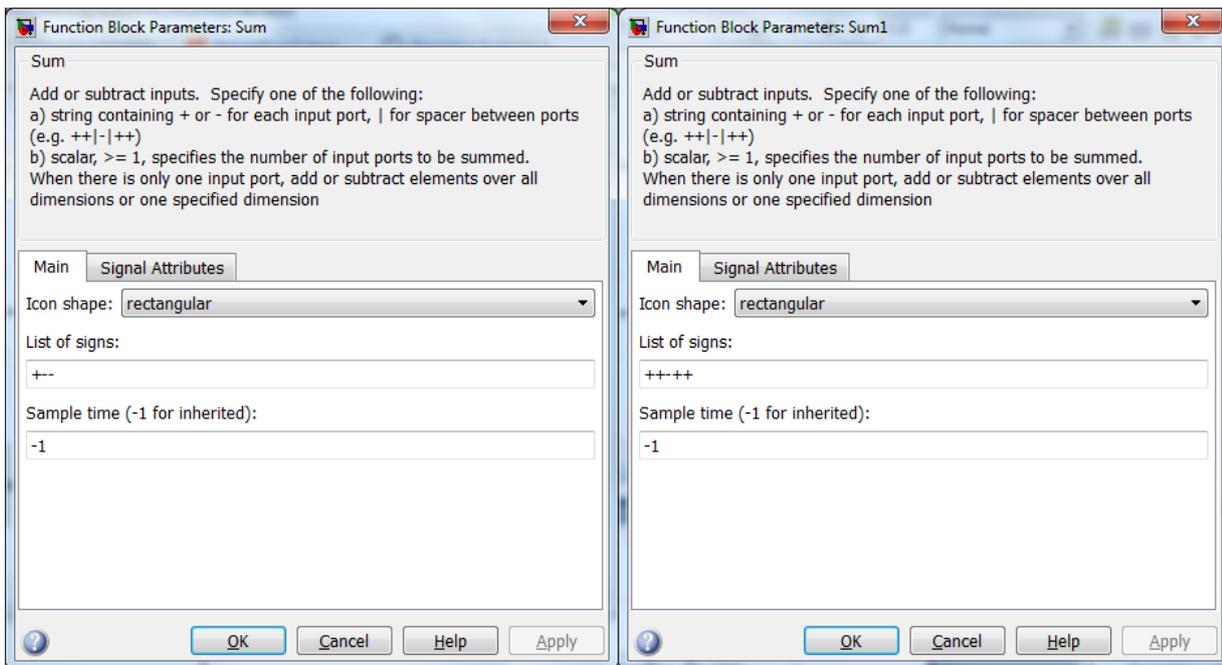
Введем в поле **List of signs** окна свойств блока Sum, соединенного с блоком Mass 2 «++-++» для пяти сил, одна из которых будет отрицательной.

Также для удобства отображения блоков в выпадающем списке Icon shape выберем прямоугольное изображение блоков – rectangular (рис. 2.45, а,б). Рекомендуются также растянуть изображение блоков, чтобы было удобнее подводить к ним входные линии (рис. 2.46).

Теперь мы будем добавлять силы, действующие на каждую массу. Сначала добавим силу, возникающую за счет упругих свойств подвески автобуса. Эта сила пропорциональна разности  $x_1 - x_2$ :  $k_1(x_1 - x_2)$ .

Вставим блок Sum после последней пары интеграторов (он автоматически примет имя Sum2, т.к. блоки Sum и Sum1 уже есть).

Установим в поле **List of signs** знаки «+» и соединим сигнал «x1» с положительным входом и сигнал «x2» – с отрицательным входом. Вид блока по желанию, можно оставить круглым, а можно изменить на прямоугольный.



*a*

*б*

Рис. 2.45. Окно параметров блока Sum

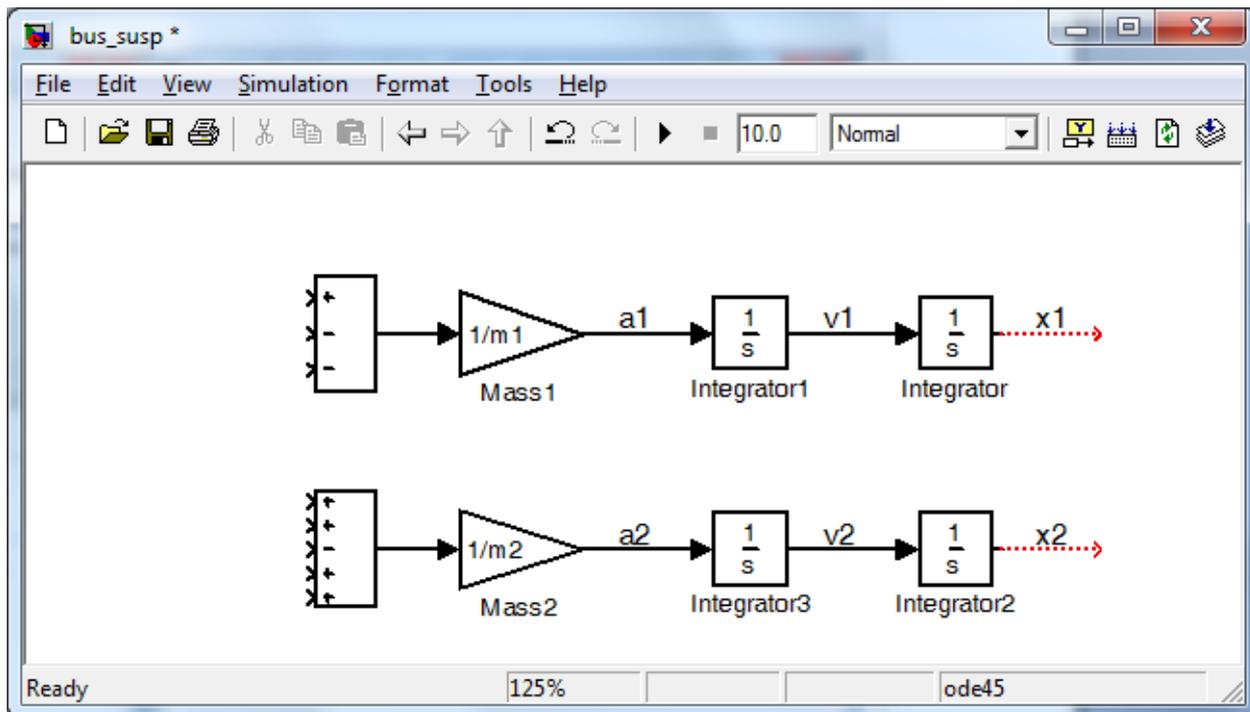


Рис. 2.46. Вид модели после добавления блоков Sum

Вставим блок Gain над блоком Mass1 и отразим его слева направо нажав правой кнопкой мыши и выбрав Format → Flip Block (или комбинацией клавиш Ctrl+I).

Изменим значение коэффициента усиления этого блока Gain на «k1» и дадим ему имя «Spring 1» (англ. spring - пружина).

Отведем линию от выхода блока Sum2 и соединим её со входом блока Spring 1, а его выход – со вторым входом блока Sum, стоящего перед блоком Mass1. Этот вход должен быть отрицательным, т.к. пружина Spring1 тянет массу  $m_1$ , когда  $x_1 > x_2$ .

Сделаем ответвление линии с упругой силой, и соединим её со вторым входом блока Sum напротив блока Mass 2 (рис. 2.47). Этот вход положительный, т.к. Spring1 толкает вверх массу  $m_2$ .

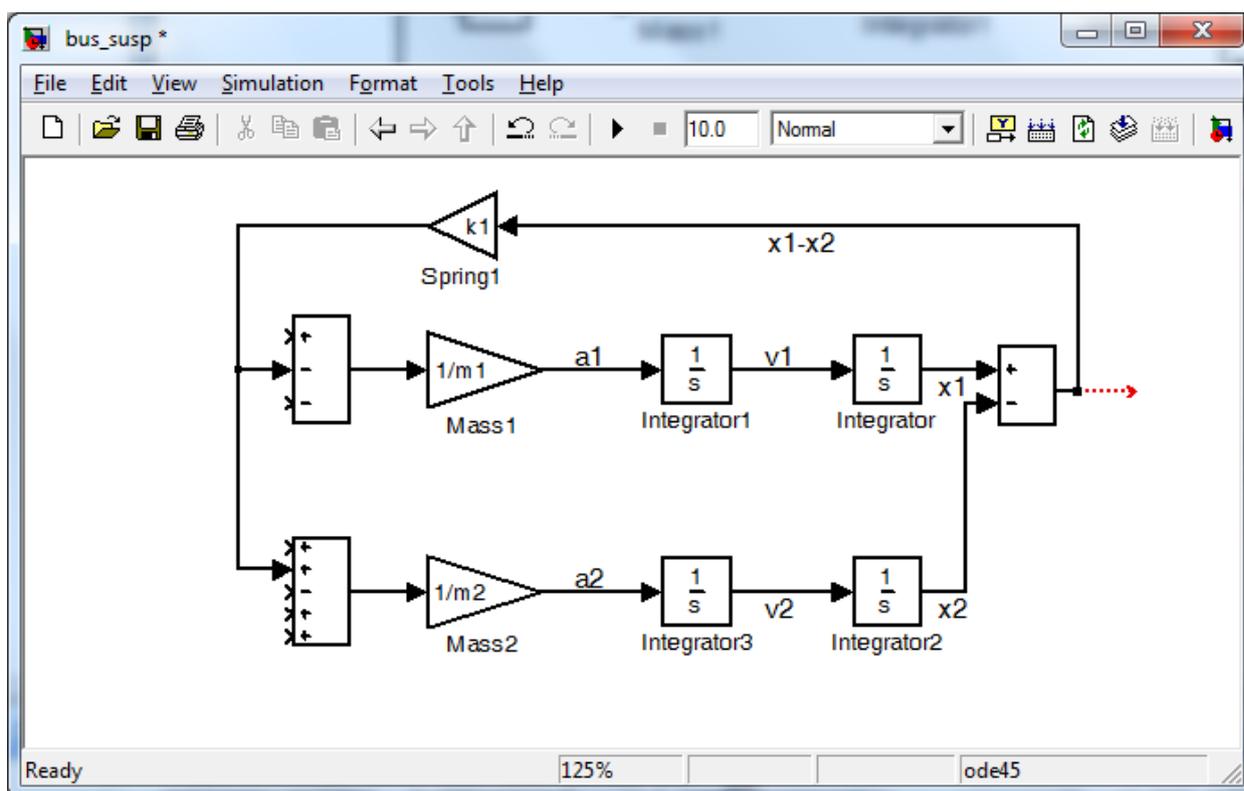


Рис. 2.47. Вид модели после добавления объединения блоков

Добавим, теперь, силу демпфирования, равную  $b_1(v_1-v_2)$ .

Вставим блок Sum под блоком Integrator1 (который соединен с блоком Mass1).

Отразим новый блок Sum слева направо, и изменим его знаки на «+-».

Отведем линию от линии «v1» и соединим её с положительным входом последнего блока Sum.

Отведем линию от линии «v2» и соединим её с отрицательным входом блока Sum.

Слева от этого блока Sum вставим блок Gain и отразим его слева направо.

Изменим значение коэффициента усиления блока Gain на «b1» и обозначим его как «Damper1» (англ. damper - амортизатор).

Соединим выход нового блока Sum с входом усилителя «Damper1».

Соединим выход этого усилителя (демпфирующую силу) с третьим входом блока Sum, расположенного около блока Mass1. Этот вход отрицательный, т.к. сила сопротивления мешает перемещению массы  $m_1$ .

Сделаем отвод выходной линии с блока Damper1 и соединим её с первым, положительным, входом блока Sum около блока Mass2 (рис. 2.49).

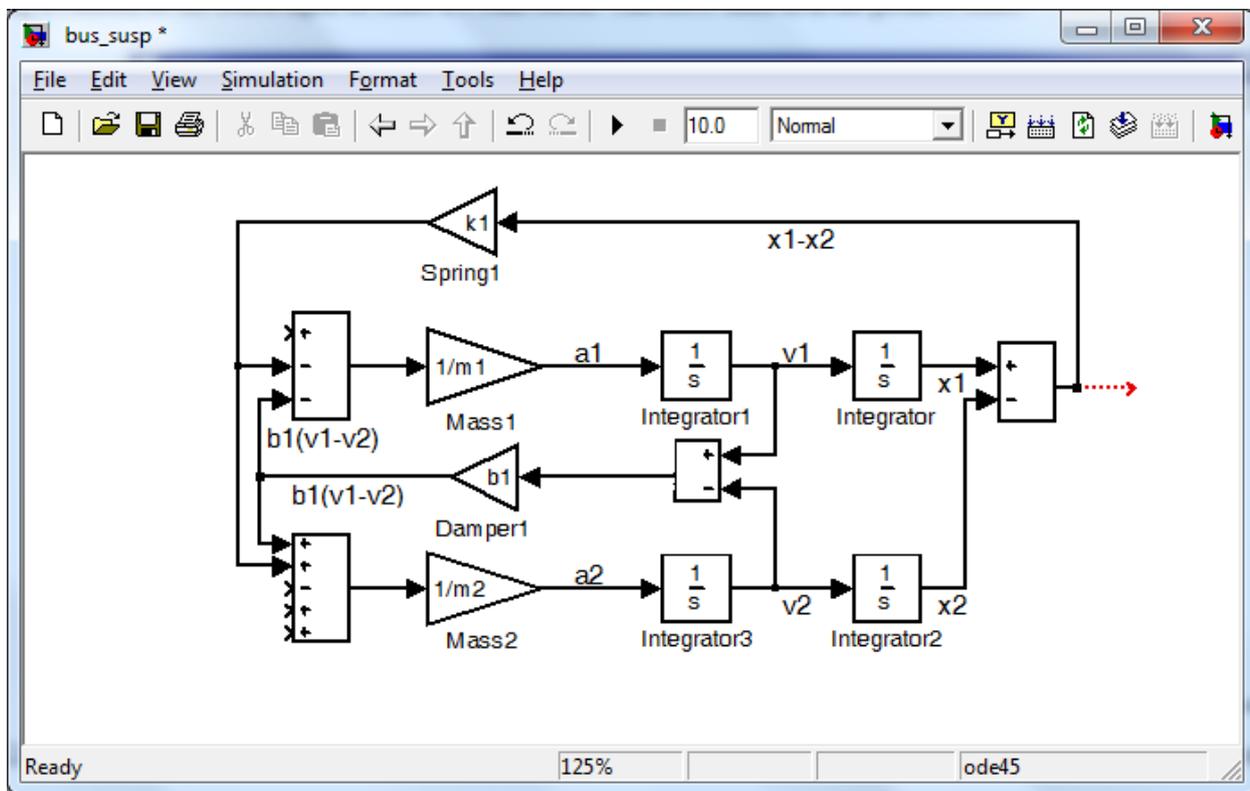


Рис. 2.49. Вид модели после добавления силы демпфирования

Теперь добавим упругую силу, создаваемую шиной колеса. Эта сила действует только на массу  $m_2$ , но зависит от профиля дороги  $w$  (см. 2.41): она равна  $x_2-w$ .

- Вставим блок ступенчатой функции Step (библиотека Sources) в левую нижнюю часть окна модели, и обозначим его  $w$ . Этот блок будет моделировать неровности дороги.

- Установим в параметрах блока время подачи сигнала **Step time** на «0» и его конечное значение **Final value** на «0.1» (будем считать, что автобус заезжает на неровность, высотой 0,1 м).
- Вставим блок Sum справа от блока w ступенчатой функции и установим его знаки на «-+».
- Соединим выход блока Step с положительным входом блока Sum.
- Сделаем ответвление от линии с сигналом «x2» и соединим её с отрицательным входом блока Sum.
- Вставим блок Gain справа от блока Sum и соединим выход блока Sum с входом нового блока Gain.
- Изменим значение коэффициента усиления блока Gain на "k2" и дадим имя блоку «Spring2».
- Соединим выход этого блока (сила упругости шины) с четвертым (положительным) входом блока Sum, находящегося возле блока Mass2 (рис. 2.50).

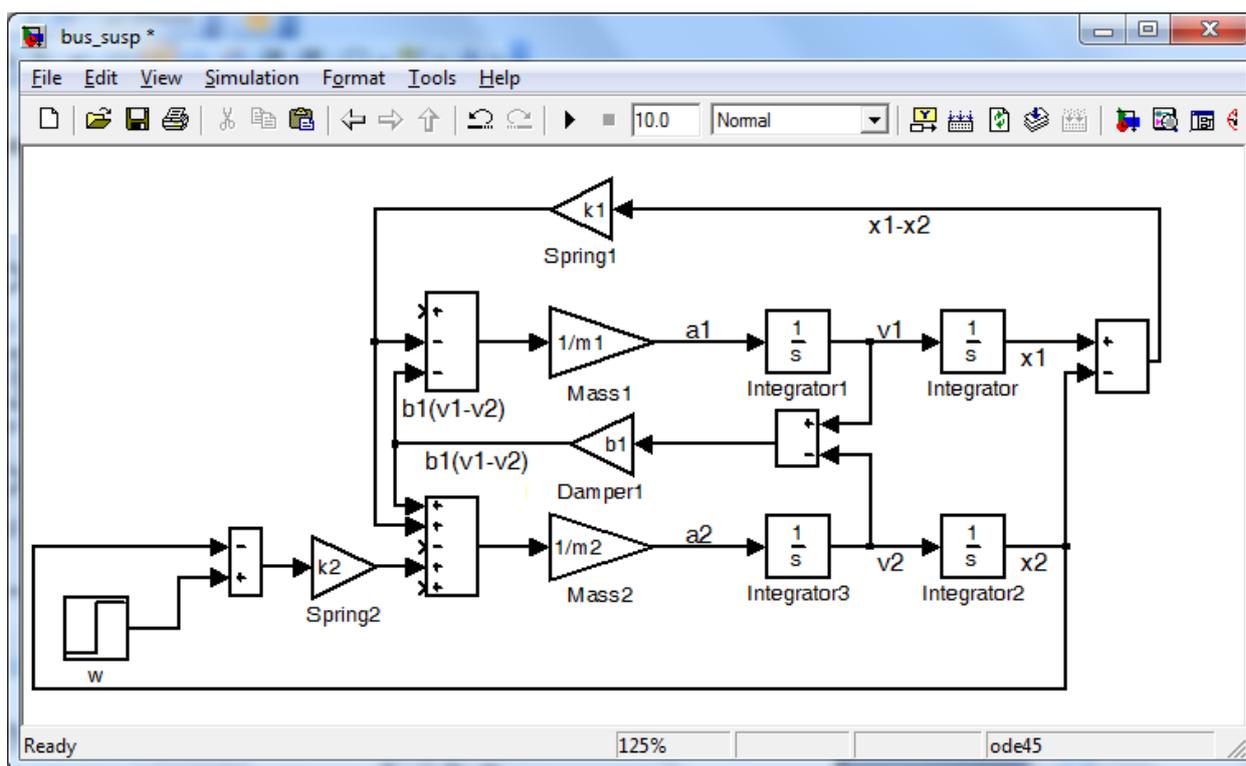


Рис. 2.50. Вид модели после добавления силы, создаваемой шиной колеса

Добавим, теперь, к нашей модели силу демпфирования шины, равную  $b_2 (dw/dt - v_2)$ .

- Вставим справа от блока ступенчатой функции w блок

производной Derivative из библиотеки Continuous. Блок Derivative позволяет получить производную входного сигнала блока, в нашем случае,  $dw/dt$ .

- Для этого, сделаем ответвление от выходного сигнала блока «w», и соединим его с входом блока Derivative.

- После блока Derivative вставим блок Sum и изменим его знаки на «+»-».

- Соединим выход блока производной Derivative с положительным входом нового блока Sum.

- Сделаем ответвление линии «v2» и соединим его с отрицательным входом блока Sum.

- Поставим после блока Sum усилитель Gain, дадим ему имя «Damper2» и изменим значение коэффициента усиления на b2.

- Соединим выход блока Sum с входом блока Damper2, а выход этого блока (силу демпфирования шины) с пятым входом блока Sum, расположенного около блока Mass2 (рис. 2.51).

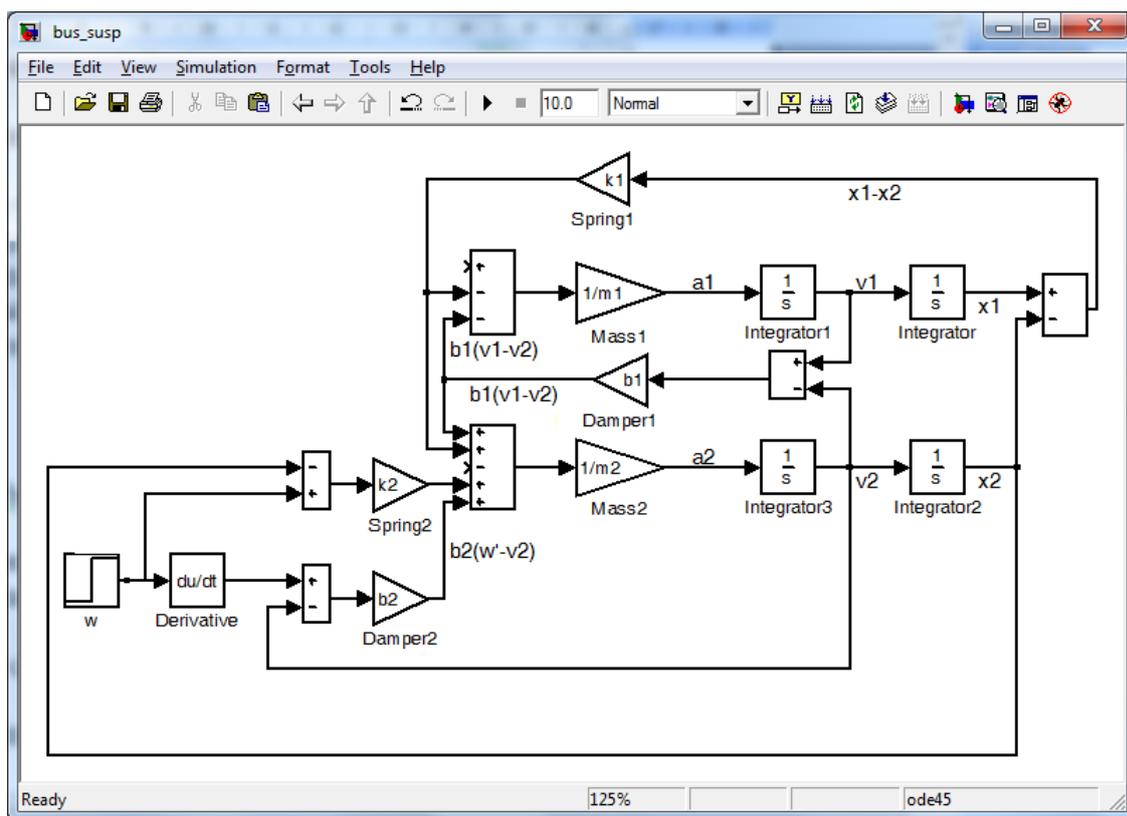


Рис. 2.51. Вид модели после добавления силы демпфирования шины

Последняя сила, действующая на массы  $m_1$  и  $m_2$  - управляющая сила  $u$ , которая стремится обеспечить желаемые колебания в подвеске автобуса. Добавим эту силу к нашей модели.

- Вставим блока Step в левый верхний угол окна модели и переименуем его как « $u$ ».

- Изменим значения параметров **Step time** и **Final value** блока « $u$ » на «0» (т.е. будем пока считать, что сила  $u = 0$ ).

- Соединим выход блока « $u$ » с оставшимся входом (с положительным знаком) блока Sum, находящегося возле блока Mass1.

- Сделаем отвод от этой линии и соединим его с оставшимся (отрицательным) входом блока Sum, расположенного возле блока Mass 2.

- Теперь, чтобы иметь возможность наблюдать выходные колебания  $x_1-x_2$  системы, выберем в окне библиотеки Sinks блок индикатора Scope и соединим его с выходным сигналом самого правого блока Sum. В итоге, наша модель будет иметь следующий вид (рис. 2.52).

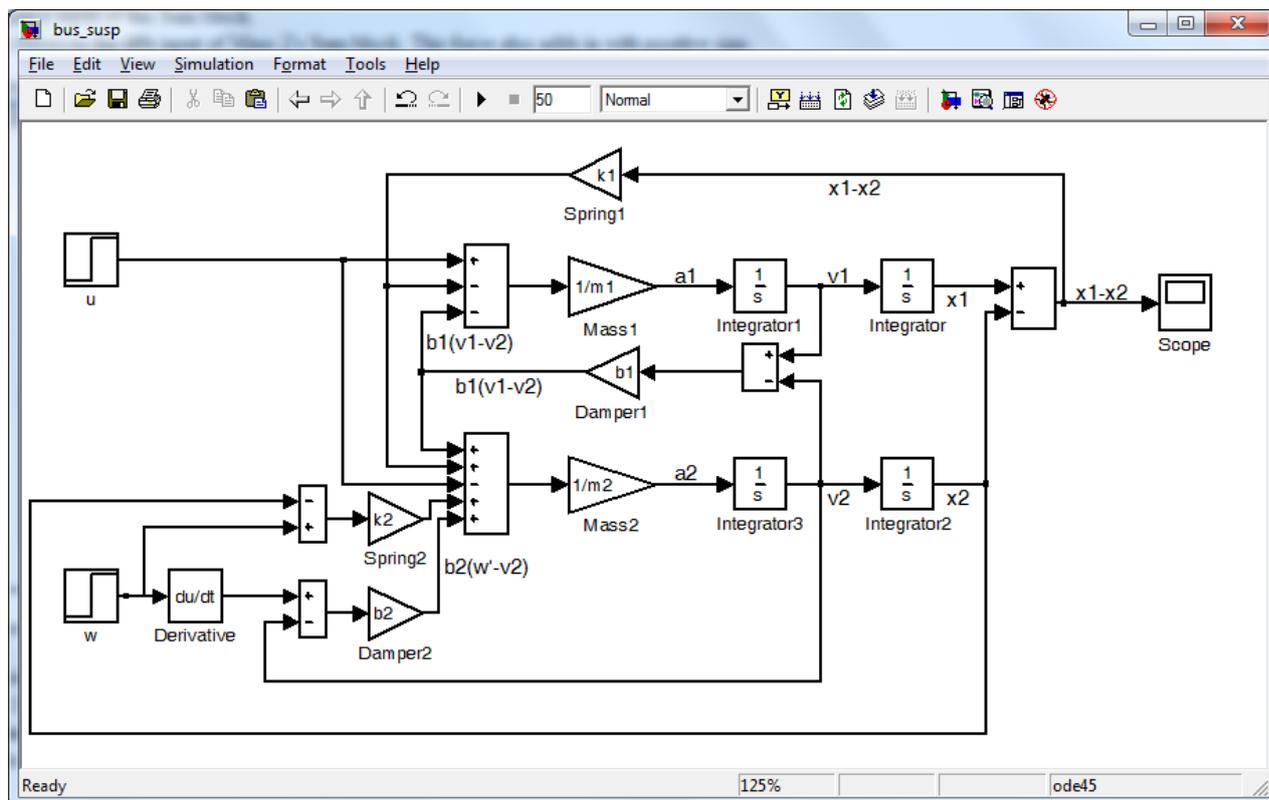


Рис. 2.52. Окончательный вид модели

Перед началом моделирования изменим время моделирования на 50 секунд, а также зададим значения физических параметров системы. Для этого введем следующие команды в командной строке MATLAB:

```
m1=2500; m2=320; k1=80000; k2=500000; b1 = 350; b2 = 15020;
```

Для начала моделирования выберем опцию **Simulation** → **Start** из меню или просто нажмем кнопку **Start** на панели инструментов. Откроем индикатор двойным щелчком мыши, при этом на экране появится изображение, приведенное на рис. 2.53.

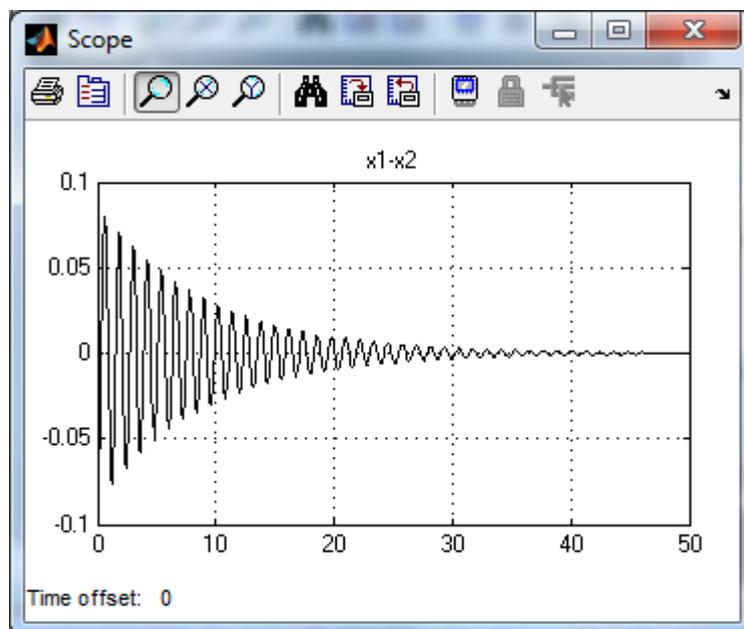


Рис. 2.53. Реакция разомкнутой системы на ступенчатое возмущение в 0,1 м

Из рис. 2.53 мы можем увидеть, что когда автобус проезжает неровность высотой в 10 см, кузов автобуса будет колебаться в течение неприемлемо долгого времени (100 секунд) с большой амплитудой. Люди, сидящие в таком автобусе, будут испытывать дискомфорт от таких колебаний. Более того, такие колебания могут повредить подвеску.

В теории управления в качестве входного воздействия очень часто используют единичную ступенчатую функцию, т.е.  $u = 1(t)$ . Изменим параметры нашей модели, чтобы отследить её реакцию на единичную ступенчатую функцию.

Для этого в окне параметров блока «w» установим значения **Step time** и **Final value**, равными «0» (это соответствует движению автобуса по ровной дороге), а в блоке «u», установим значение **Step time** - «0» и **Final value** - «1». Рис. 2.54 отображает колебания автобуса при его движении по ровной дороге и действии силы  $u=1(t)$ .

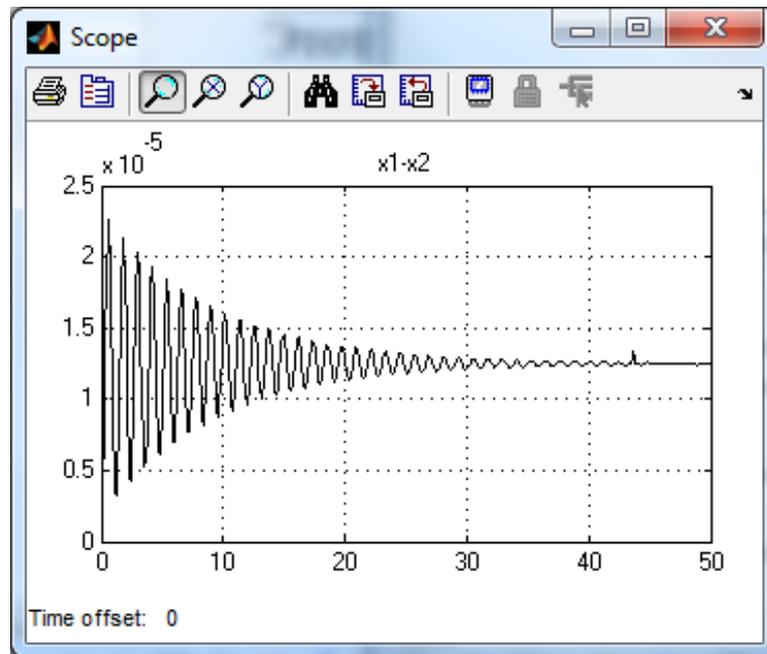


Рис. 2.54. Реакция разомкнутой системы на единичное ступенчатое воздействие

Из этого графика реакции разомкнутой системы мы видим, что система является недодемпфированной (слишком много колебаний), а установившаяся ошибка (разность между 0 и установившемся значением) составляет около 0,013 мм. Для улучшения работы нашей системы необходимо добавить регулятор с обратной связью. Далее мы еще будем обращаться к этой системе управления подвеской для проектирования регулятора.

### 2.2.2 Механические системы с вращательным движением

Уравнения элементов механических систем с вращательным движением аналогичны соответствующим уравнениям системы с линейным перемещением, и при их составлении используются те же рассуждения.

На рис. 2.55 изображены три элемента, характеризующиеся вращательным движением.

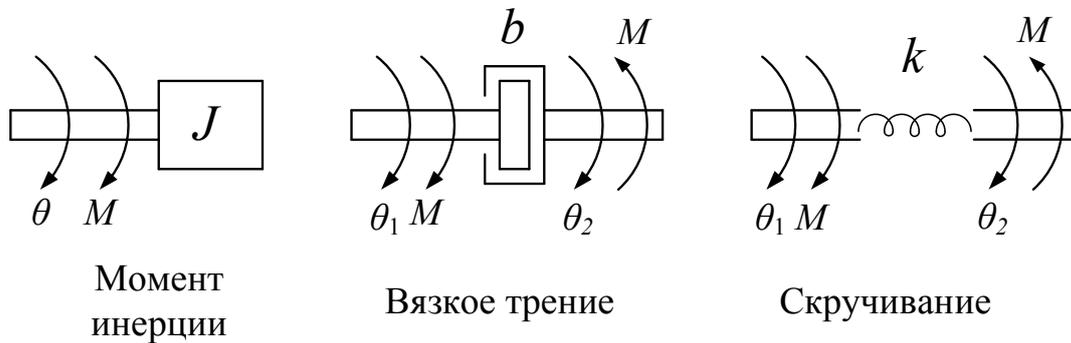


Рис. 2.55. Механические элементы с вращательным движением

Первый элемент, момент инерции, описывается уравнением

$$M(t) = J \frac{d^2\theta(t)}{dt^2} = J \frac{d\omega(t)}{dt}, \quad (2.13)$$

где  $M(t)$  - приложенный вращающий момент;  
 $J$  - момент инерции;  
 $\theta(t)$  - угол поворота;  
 $\omega(t)$  - угловая скорость.

Уравнение (2.13) по форме совпадает с уравнением (2.1) для массы. Фактически момент инерции тела является функцией его массы и геометрических параметров.

Для второго элемента на рисунке 2.21, трения, уравнение имеет вид:

$$M(t) = b \left[ \frac{d\theta_1(t)}{dt} - \frac{d\theta_2(t)}{dt} \right] = b [\omega_1(t) - \omega_2(t)] \quad (2.14)$$

где  $b$  – коэффициент трения.

Здесь предполагается, что момент инерции вращающегося элемента равен нулю, а трение возникает между двумя частями этого элемента.

Уравнение для элементов типа «Скручивающаяся пружина» (рис. 2.55, в) имеет вид:

$$M(t) = k[\theta_1(t) - \theta_2(t)], \quad (2.15)$$

где  $k$  - коэффициент упругости.

Здесь так же предполагается, что момент инерции данного элемента равен нулю. Для составления математической модели реальной системы момент инерции, возможно, придется учитывать.

В основе уравнений, описывающих вращательное движение механической системы, лежит принцип, согласно которому сумма моментов сил относительно оси вращения равна произведению момента инерции на угловое ускорение.

В качестве примера рассмотрим систему, представляющую собой часть руки робота-манипулятора (Рис. 2.56). Такие роботы широко используются в автомобильной промышленности на сборочных, сварочных, покрасочных работах (Рис. 2.57)

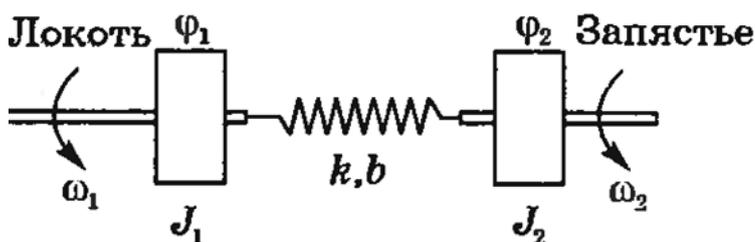


Рис. 2.56. Фрагмент руки промышленного робота

Электродвигатель в локтевом сочленении робота приводит в движение запястье через предплечье. Предплечье обладает определенной гибкостью, отображенной на рис. 2.56 в виде пружины. Пружина имеет коэффициент упругости  $k$  и коэффициент трения  $b$ .

Как и в предыдущем примере, мы имеем два независимых перемещения (углы поворотов  $\varphi_1$  и  $\varphi_2$ ), поэтому математическая модель будет состоять из двух уравнений:

$$\begin{cases} J_1 \ddot{\varphi}_1 = M_1 - b(\omega_1 - \omega_2) - k(\varphi_1 - \varphi_2), \\ J_2 \ddot{\varphi}_2 = b(\omega_1 - \omega_2) + k(\varphi_1 - \varphi_2). \end{cases} \quad (2.16)$$



Рис. 2.57. Роботы в сварочном цехе автозавода [10]

Структурная схема в Simulink, решающая систему (2.16), приведена на рис. 2.58.

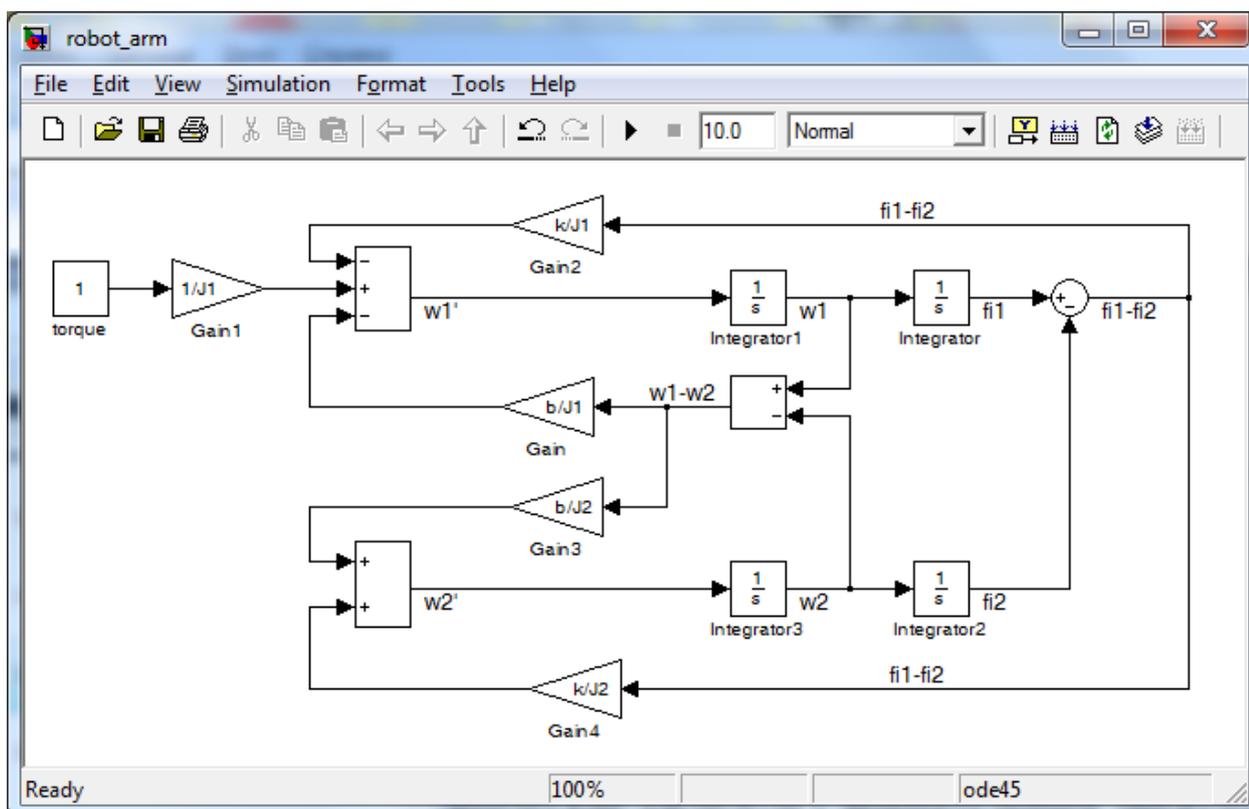


Рис. 2.58. Окончательный вид модели фрагмента руки промышленного робота

В технике часто требуется передавать вращательное движение от одного вала к другому. Например, в автомобиле мощность, развиваемая двигателем, передается вращающимся колесам через коробку передач и дифференциал. Коробка передач позволяет водителю выбирать то или иное передаточное отношение в зависимости от дорожных условий, тогда как дифференциал находится в неизменном положении. В этом случае скорость движения не является постоянной - водитель может менять ее по своему усмотрению. Рассмотрим, поэтому, математические модели некоторых механических преобразователей данного типа.

При рассмотрении зубчатой передачи (рис. 2.59) обычно предполагается, что шестерни являются идеальными, т.е. жесткими, не обладающими инерцией и с зубьями, постоянно находящимися в зацеплении. Тогда расстояние вдоль окружности одной шестерни будет равно расстоянию вдоль окружности другой, или

$$r_1\theta_1 = r_2\theta_2, \quad (2.17)$$

где  $r_1$ ,  $\theta_1$ ,  $r_2$ ,  $\theta_2$  - соответственно радиус и угол поворота первой и второй шестерни. Кроме того, сила, приложенная в точке контакта со стороны одной шестерни, должна вызывать такую же реакцию со стороны другой шестерни. Следовательно,

$$\theta_2 = \frac{r_1}{r_2}\theta_1, \quad \frac{M_1}{r_1} = \frac{M_2}{r_2}. \quad (2.18)$$

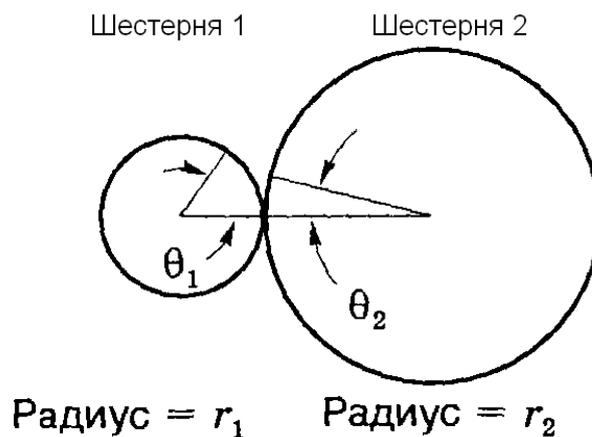


Рис. 2.59. Зубчатая пара

Рассмотрим зубчатую передачу, изображенную на рис. 2.60. Здесь

$$M_2 = J \frac{d^2\theta_2}{dt^2} \quad (2.19)$$

и

$$M_1 = \left(\frac{r_1}{r_2}\right) M_2 = \left(\frac{r_1}{r_2}\right) J \frac{d^2\theta_2}{dt^2} = \left[\left(\frac{r_1}{r_2}\right)^2 J\right] \frac{d^2\theta_1}{dt^2}. \quad (2.20)$$

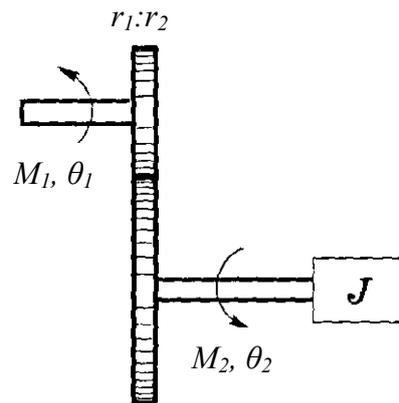


Рис. 2.60. Зубчатая передача

Таким образом, зубчатая передача обладает эффектом преобразования момента инерции через квадрат отношения радиусов шестерен.

Примером устройства, преобразующего вращательное движение в поступательное, является передача зубчатое колесо-рейка. Эта система описывается уравнением

$$x = r\theta, \quad (2.19)$$

где  $r$  – радиус зубчатого колеса;  
 $\theta$  - угол его поворота.

### 2.2.3 Электрические системы

Математические модели электрических цепей обычно строят с использованием законов Кирхгофа.

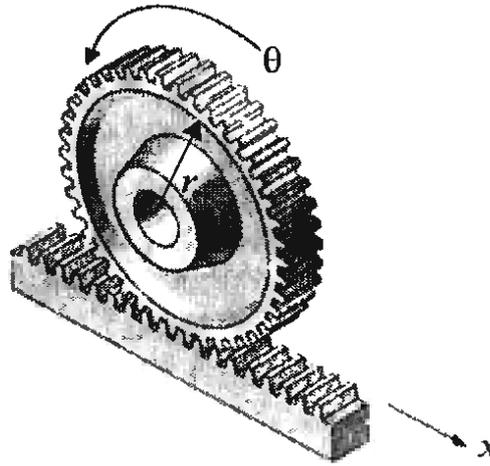


Рис. 2.61. Зубчатое колесо и рейка

Первый закон Кирхгофа (закон для токов) состоит в том, что *алгебраическая сумма токов, сходящихся в любом узле, равна нулю.*

$$\sum_{k=1}^n I_k = 0, \quad (2.20)$$

где  $k$  – число токов, сходящихся в данном узле.

Второй закон Кирхгофа (для напряжений) имеет следующую формулировку: *алгебраическая сумма падений напряжений на отдельных участках замкнутого контура, произвольно выделенного в сложной разветвленной цепи, равна алгебраической сумме ЭДС в этом контуре*

$$\sum_{k=1}^n E_k = \sum_{k=1}^m I_k R_k, \quad (2.21)$$

где  $n$  – число источников ЭДС;

$m$  – число ветвей в замкнутом контуре;

$I_k, R_k$  – ток и сопротивление  $k$ -й ветви.

При составлении также уравнений используют следующие правила:

1) ЭДС положительна, если ее направление совпадает с направлением произвольно выбранного обхода контура;

2) падение напряжения на резисторе положительно, если направление тока в нем совпадает с направлением обхода.

Напомним, также, основные соотношения, связывающие параметры элементов электрических цепей с токами и напряжениями.

Связь напряжения на концах проводника с силой тока и сопротивлением проводника устанавливает закон Ома:

*сила тока  $i(t)$ , текущего по однородному металлическому проводнику (т. е. проводнику, в котором не действуют сторонние силы), пропорционально напряжению  $U(t)$  на концах проводника:*

$$U(t) = R \cdot i(t) \quad (2.22)$$

Мгновенное значение напряжения на конденсаторе:

$$U_C(t) = \frac{1}{C} \int i(t) dt, \quad (2.23)$$

а мгновенное значение напряжения на индуктивности:

$$U_L(t) = L \frac{di(t)}{dt}, \quad (2.24)$$

где  $C$  – ёмкость конденсатора;

$L$  - индуктивность.

#### **Пример 4.**

Получим дифференциальное уравнение, моделирующее ненагруженный четырехполюсник (рис. 2.62), в котором в качестве входной величины рассматривается напряжение  $U_{вх}$  стороннего источника постоянного тока, а в качестве выходной – напряжение  $U_C$  на конденсаторе.

Дифференциальное уравнение, моделирующее четырехполюсник, удобно строить на основе второго закона Кирхгофа и должно отражать равенство сторонней электродвижущей силы, вызывающей в нем процесс зарядки конденсатора, и силы сопротивления четырехполюсника этому процессу.

Электродвижущая сила стороннего источника равна  $U_{вх}$ . При зарядке конденсатора он сам превращается в источник питания с электродвижущей силой, направленной против электродвижущей силы стороннего источника.

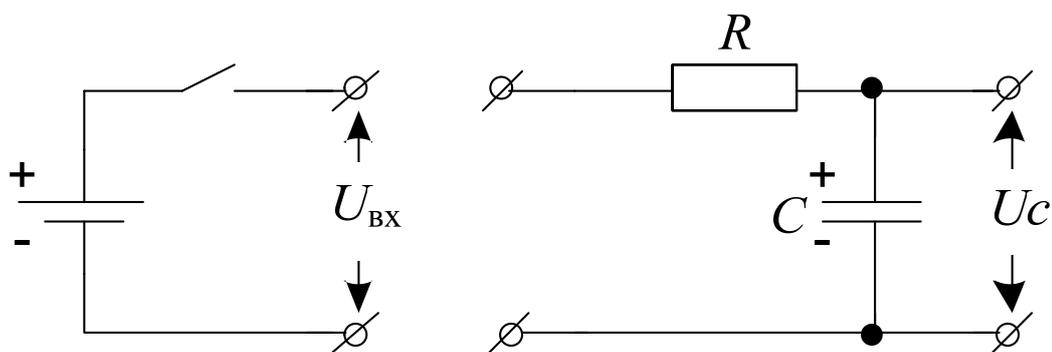


Рис. 2.62. Ненагруженный четырехполюсник  $RC$

Это видно по знаку зарядов на пластинах конденсатора. Электродвижущая сила конденсатора равна  $U_C(t)$ . Это – первая составляющая силы сопротивления.

Вторая составляющая – это сопротивление резистора току зарядки, то есть, падение  $U_R(t)$  напряжения на резисторе.

Таким образом, исходным уравнением, демонстрирующим баланс сил, является уравнение

$$U_C(t) + U_R(t) = U_{\text{вх}}(t). \quad (2.25)$$

Выразим все величины, фигурирующие в (2.25), через входную  $U_{\text{вх}}(t)$  и выходную  $U_C(t)$  величины. В данном примере это касается падения  $U_R(t)$  напряжения на резисторе.

Воспользуемся известными законами электричества:

$$U_R(t) = i(t) \cdot R = \frac{dQ_C(t)}{dt} \cdot R = \frac{dCU_C(t)}{dt} \cdot R = CR \cdot \frac{dU_C(t)}{dt} \quad (2.26)$$

В (2.26)  $i(t)$  – сила тока зарядки,  $Q_C(t)$  – заряд конденсатора (скорость  $dQ_C(t)/dt$  зарядки конденсатора равна силе  $i(t)$  тока зарядки),  $Q_C(t) = C \cdot U_C(t)$ . Поставим (2.26) в (2.25) и получим:

$$CR \cdot \frac{dU_C(t)}{dt} + U_C(t) = U_{\text{вх}}(t). \quad (2.27)$$

Падение напряжения
противо-ЭДС
ЭДС  
на резисторе
конденсатора
стороннего  


источника

Дифференциальное уравнение (2.27) – это и есть математическая модель ненагруженного четырехполюсника  $RC$ . Для исследования процессов, протекающих в четырехполюснике при

изменении  $U_{вх}$ , теперь не обязательно проводить эксперименты над ним. Исследование можно проводить, решая дифференциальное уравнение (2.27), например, при помощи Simulink.

### 2.2.4 Электромеханические системы

Наиболее широко распространенным исполнительным приводом в системах управления является двигатель постоянного тока (ДПТ). Он обеспечивает вращательное движение, а при дополнительном использовании различного рода барабанов, зубчатых колес, тросов – и поступательное движение.

Рассмотрим двигатель постоянного тока независимого возбуждения с инерционной нагрузкой (рис. 2.63). Данную систему можно разбить на две части: электрическую и механическую.

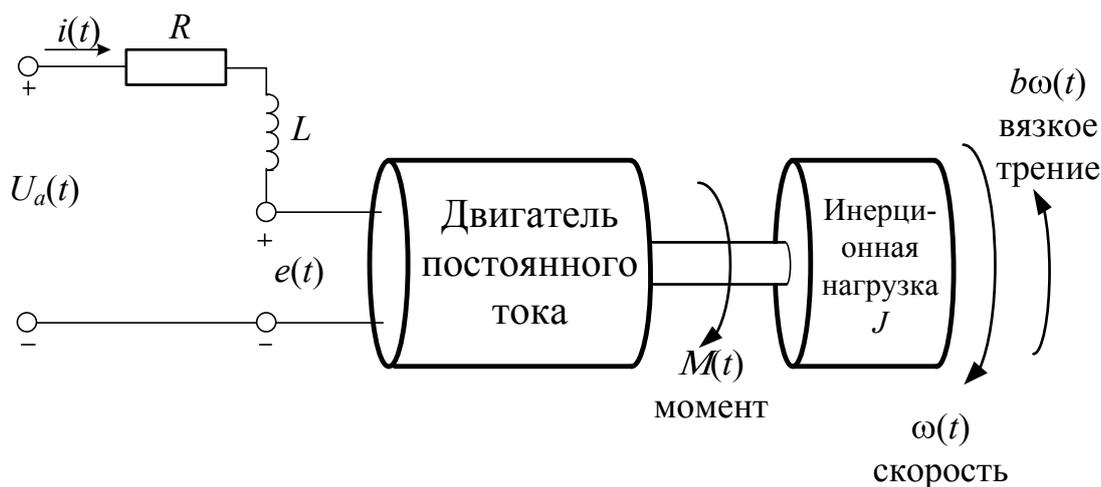


Рис. 2.63. Структурная схема двигателя постоянного тока независимого возбуждения с инерционной нагрузкой

Входной величиной электрической части является напряжение  $U_a(t)$ , подаваемое на якорную обмотку двигателя, выходной – электромагнитный момент  $M_m(t)$ , развиваемый двигателем. Для механической части входом является момент, выходной – угловая скорость  $\omega(t)$  инерционной массы.

При построении модели будем пренебрегать такими второстепенными эффектами, как гистерезис и падение напряжения на щетках, а также будем считать магнитное поле постоянным.

На основании второго закона Кирхгофа электрическая часть двигателя описывается следующим уравнением

$$U_a(t) = Ri(t) + L \frac{di(t)}{dt} + e(t), \quad (2.28)$$

где  $R$  и  $L$  - сопротивление и индуктивность цепи якоря, соответственно;

$i(t)$  - ток якоря;

$e(t)$  - противоЭДС, возникающая в обмотке якоря в результате его вращения в магнитном поле.

В свою очередь, противоЭДС пропорциональна скорости вращения вала

$$e(t) = K_e \cdot \omega(t), \quad (2.29)$$

где  $K_e$  – постоянный коэффициент, который зависит от определенных физических свойств двигателя.

Двигатель преобразует ток, протекающий в цепи якоря, во вращающий момент  $M_m(t)$ , который, если считать магнитный поток постоянным, пропорционален току якоря:

$$M_m(t) = K_m i(t), \quad (2.30)$$

где  $K_m$  – постоянная якоря, которая зависит от физических свойств двигателя, таких как напряженность магнитного поля, числа витков проволоки в катушке и т.д. Обычно  $K_e = K_m$ , хотя имеют разную размерность.

С другой стороны, вращению вала двигателя препятствуют сила вязкого трения  $b\omega$  ( $b$  – коэффициент трения) и возмущающий момент  $M_L(t)$ . Тогда, на основании второго закона Ньютона, механическая часть двигателя описывается следующим уравнением:

$$J \frac{d\omega(t)}{dt} = M_m(t) - M_L(t) - b\omega(t), \quad (2.31)$$

где  $J$  - суммарный момент инерции якоря и нагрузки.

Окончательно, на основании уравнений (2.28) –(2.31), запишем

$$\begin{cases} U_a(t) = Ri(t) + L \frac{di(t)}{dt} + K_e \omega(t), \\ M_L(t) = K_m i(t) - b\omega(t) - J \frac{d\omega(t)}{dt}. \end{cases} \quad (2.32)$$

Построим, теперь, модель двигателя в Simulink. Для удобства можно записать уравнения (2.32) прямо в окне модели. Для этого достаточно дважды щелкнуть левой кнопкой мыши в нужном месте, в результате чего появится рамка, в которой можно вводить текст (рис. 2.64).

$$\begin{cases} \frac{di(t)}{dt} = \frac{1}{L}(U_a(t) - Ri(t) - K_e \omega(t)) \\ \frac{d\omega(t)}{dt} = \frac{1}{J}(K_m i(t) - b\omega(t) - M_L(t)). \end{cases} \quad (2.33)$$

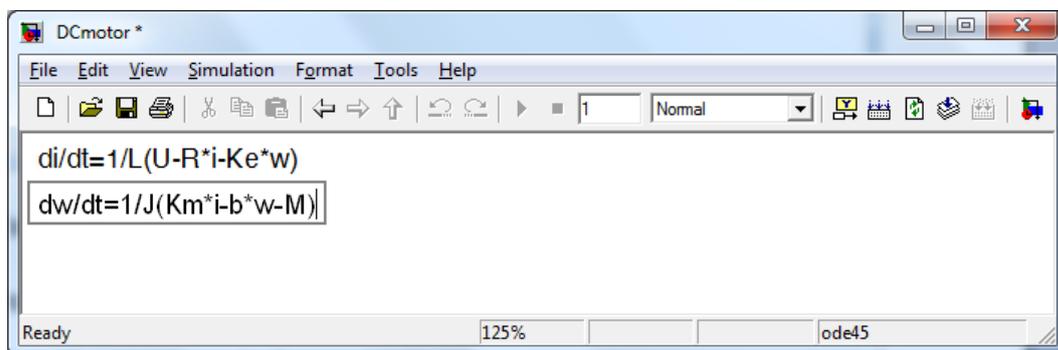


Рис. 2.64. Ввод текстовой надписи в окне Simulink

Как обычно, начнем построение модели с конца.

- Вставим блок Integrator из библиотеки Continuous и нарисуем линии к входу блока и от выходного порта блока.
- Обозначим входную линию как «di/dt», а выходную – как «i».
- Вставим еще один блок Integrator под предыдущим блоком и нарисуем входные и выходные линии связи.
- Обозначим входную линию как «di/dt», а выходную – как «w» (рис. 2.65).

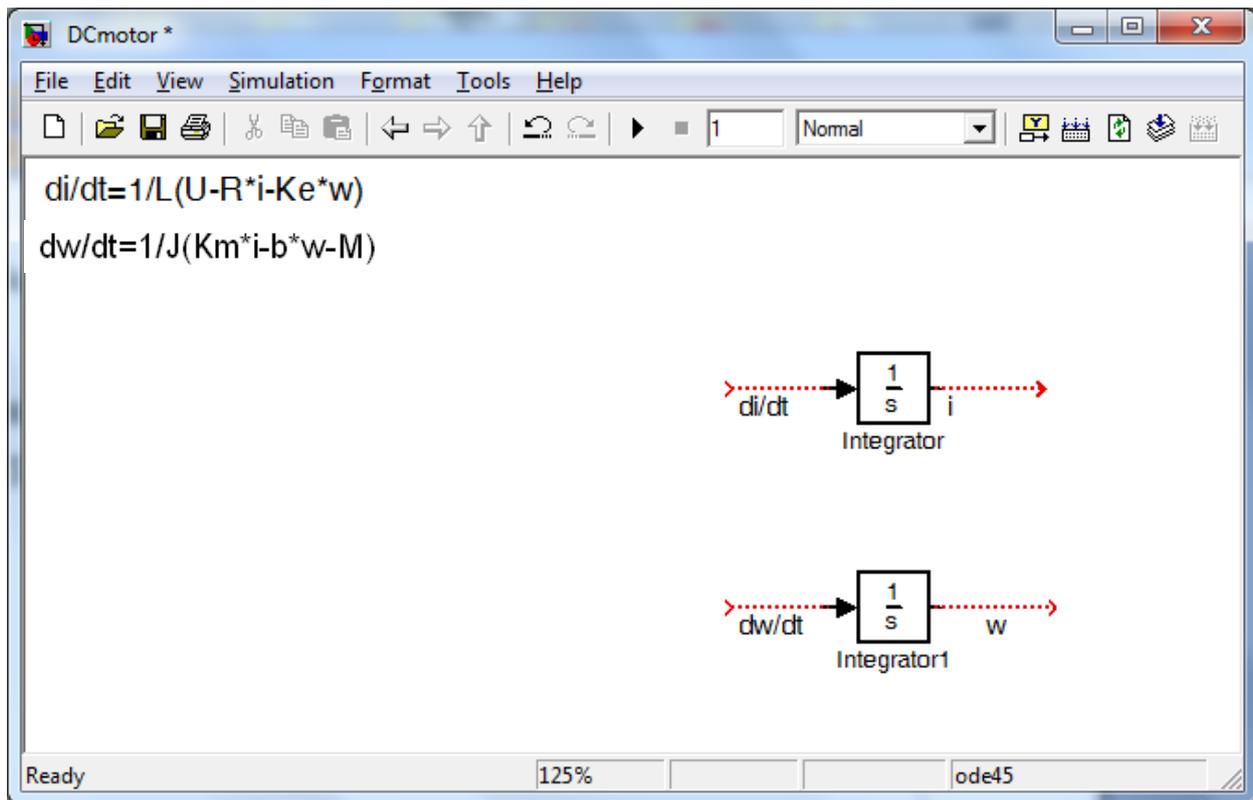


Рис. 2.65. Начало построения модели двигателя постоянного тока

Теперь начнем моделировать законы Ньютона и Кирхгофа (11.6). Для этого выполним следующие операции.

- Перед каждым из интеграторов вставим по блоку Gain и соединим их с входами соответствующих интеграторов.
- Изменим коэффициент усиления верхнего блока Gain на « $1/L$ ».
- Подобным же образом изменим параметры второго блока Gain, присвоив его коэффициенту усиления значение « $1/J$ ».
- Перенесем в окно модели два блока Sum из библиотеки Math Operations и расположим их перед блоками Gain.
- В верхнем сумматоре установим знаки « $-+ -$ » для записи уравнения Кирхгофа.
- В нижнем блоке Sum установим знаки « $+ -$ » для реализации уравнения Ньютона.
- После выполнения описанных выше действий, модель должна иметь вид, подобный рис. 2.66.

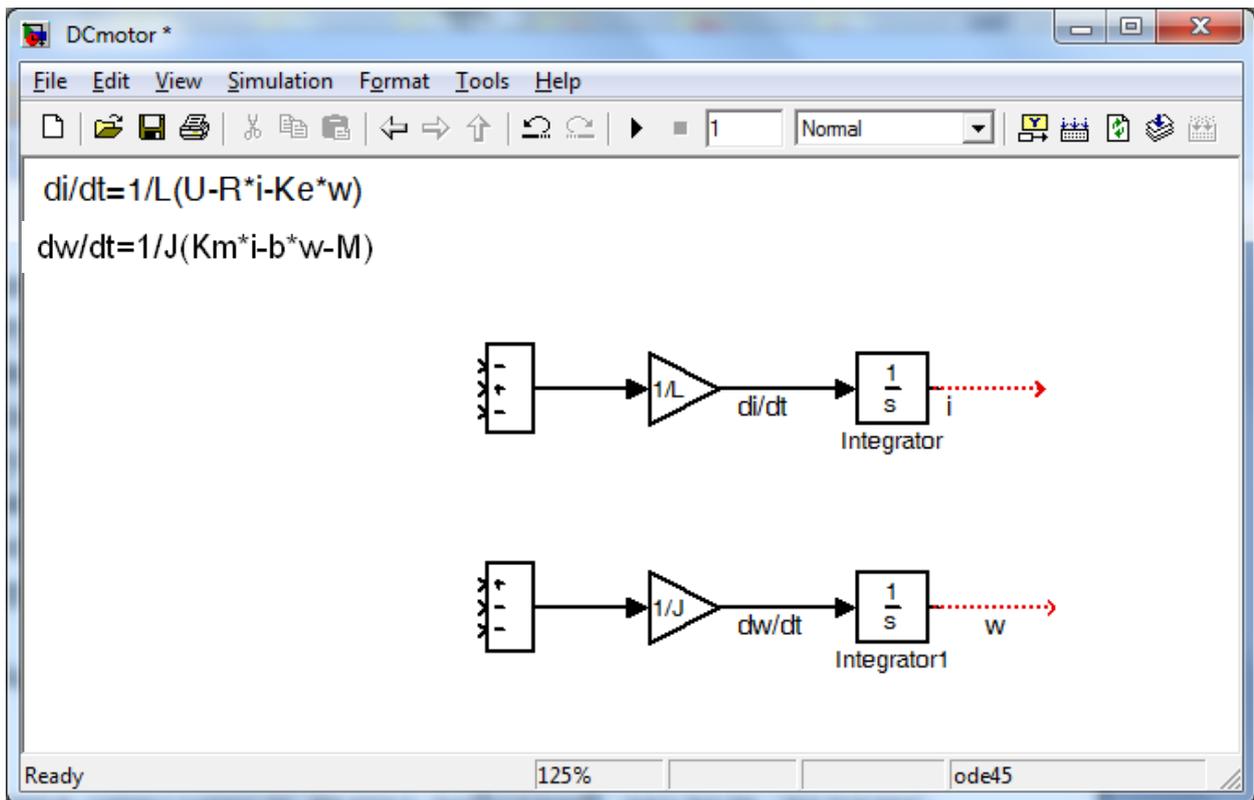


Рис. 2.66. Вид модели после добавления блоков Sum и Gain

Рассмотрим первое уравнение в (2.33). Из него видно, что нам необходимо на входы верхнего сумматора блока подать три сигнала:  $U_a(t)$ ,  $Ri(t)$  и  $K_e\omega(t)$ . Для этого выполним следующие операции.

- Подведем линию «i», выходящую из верхнего блока Integrator к верхнему (отрицательному) входу первого блока Sum.
- Вставим в эту линию блок Gain. Обратите внимание, что блок Gain сам поменяет свою ориентацию.
- Присвоим коэффициенту усиления этого блока Gain значение сопротивления цепи якоря «R».
- Дадим линии связи блока Gain с сумматором имя «R\*i».
- Сделаем ответвление от линии «w», выходящей из нижнего интегратора и соединим с нижним (отрицательным) входом верхнего блока Sum.
- Вставим в эту линию блок Gain и поменяем его коэффициент усиления на «Ke».
- Дадим линии связи этого блока Gain с сумматором имя «Ke\*w». Эта цепь моделирует противоЭДС в якорной обмотке двигателя.

- К оставшемуся положительному входу верхнего сумматора подключим напряжение, подаваемое на якорь двигателя. Для этого будем использовать блок Step из библиотеки источников сигналов Sources.

- Чтобы напряжение на якорную обмотку подавалось в начальный момент времени  $t = 0$ , дважды щелкнем на блоке Step и установим значение в поле **Step Time** на «0».

- Присвоим блоку Step и его выходной линии имена «U» (рис. 2.67).

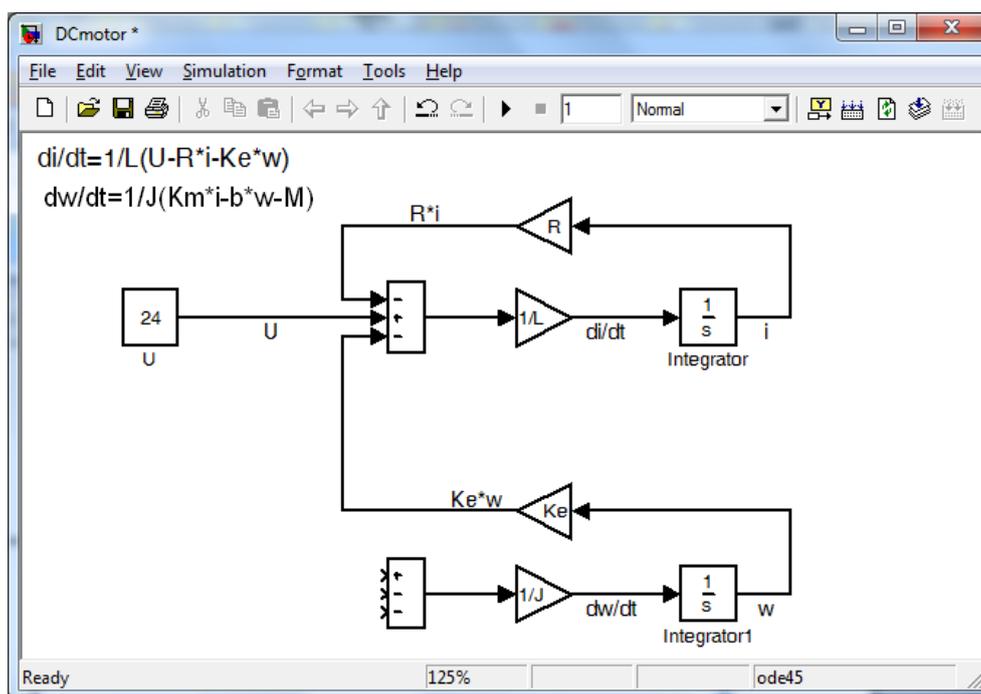


Рис. 2.67 Вид модели после добавления величин  $U_a(t)$ ,  $Ri(t)$  и  $K_e\omega(t)$

Построим теперь модель, соответствующую второму уравнению в (2.33).

- Сделаем ответвление от линии «i» и соединим с верхним (положительным) входом нижнего блока Sum.

- Вставим в эту линию блок Gain и присвоим его коэффициенту усиления значение постоянной якоря «Km».

- Дадим линии связи этого блока Gain с нижним сумматором имя «Km\*i». Эта цепь передает вращающий момент, создаваемый двигателем.

- Сделаем еще одно ответвление от линии «i» и соединим его с нижним (отрицательным) входом нижнего блока Sum.

- Добавим в эту линию связи блок Gain и присвоим его коэффициенту усиления значение коэффициента вязкого трения «b».

- Присвоим имя линии, связывающей блок Gain с нижним входом сумматора имя «b\*w». Эта цепь моделирует сопротивление вращению вала в результате эффектов трения.

Последнее, что нам надо сделать – это добавить к нашей модели момент сопротивления и индикатор для отображения результатов моделирования.

- Для его моделирования момента сопротивления удобно использовать блок ступенчатой функции Step из библиотеки Sources. Добавим блок Step в окно нашей модели.

- Соединим блок Step со вторым (отрицательным) входом нижнего блока Sum.

- Присвоим всем параметрам блока Step нулевые значения (будем пока считать, что момент сопротивления отсутствует).

- Дадим блоку Step и исходящей из него линии имени «M».

- Добавим к модели блок индикатора Scope из библиотеки Sinks и подадим на его вход ответвление от линии «w», выходящей из нижнего интегратора.

После выполнения указанных операций модель двигателя постоянного тока должна иметь вид, изображенный на рис. 2.68.

При моделировании будем использовать следующие параметры двигателя:

- момент инерции ротора  $J = 0,01 \text{ кгм}^2/\text{с}^2$ ;

- коэффициент демпфирования механической системы  $b = 0,1 \text{ Нмс}$ ;

- электромеханическая постоянная двигателя  $K_e = 0,01 \text{ В}\cdot\text{с}/\text{рад}$ ;

- постоянная якоря  $K_m = 0,01 \text{ Нм}/\text{А}$ ;

- сопротивление цепи якоря  $R = 1 \text{ Ом}$ ;

- индуктивность цепи якоря  $L = 0,5 \text{ Гн}$ .

Введем эти значения в командной строке MATLAB:

$J = 0.01$ ;

$b = 0.1$ ;

$Ke = 0.01$ ;  $Km = 0.01$ ;

$R = 1$ ;

$L = 0.5$ ;

Для удобства можно также изменить время моделирования с 10 на 3 с. Запустим моделирование, нажав кнопку  («Start simulation»)

на панели инструментов. По окончании моделирования откроем окно индикатора двойным щелчком мыши, при этом должен появиться график изменения скорости вращения двигателя, приведенный на рис. 2.69.

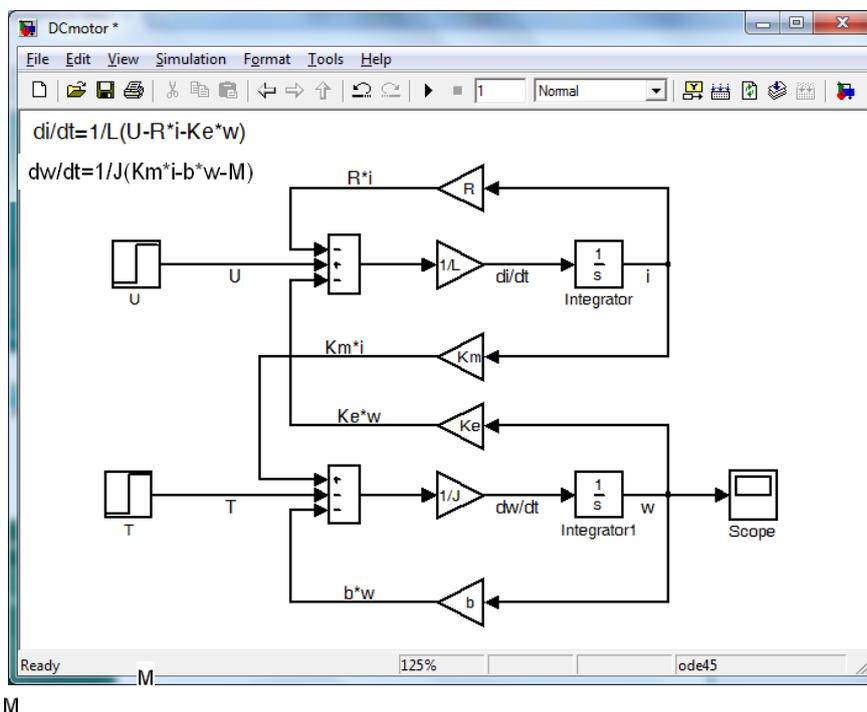


Рис. 2.68. Вид готовой модели двигателя постоянного тока

Для практических расчетов часто приходится определять значения параметров модели двигателя по паспортным данным или каталогам, в которых не всегда имеются сведения об индуктивности и сопротивлении якорной обмотки или о электромагнитной постоянной времени. В этом случае значения параметров могут быть вычислены по паспортным номинальным данным двигателя: мощности  $P_N$  [Вт], напряжению  $U_N$  [В], току якоря  $I_N$  [А], частоте вращения  $n_N$  [об/мин], коэффициенту полезного действия  $\eta$  %.

Если сопротивление обмотки якоря двигателя  $R_a$  не задано, его приближенно рассчитывают, принимая, что в номинальном режиме работы на обмотку якоря приходится определенная часть общих потерь мощности в машине:

$$R_a \approx \frac{\alpha(1-\eta)U_N}{I_N} \quad (2.34)$$

для двигателей параллельного возбуждения  $\alpha=0,5$ ; для двигателей смешанного возбуждения  $\alpha=0,6$ ; для двигателей последовательного возбуждения  $\alpha=0,75$ .

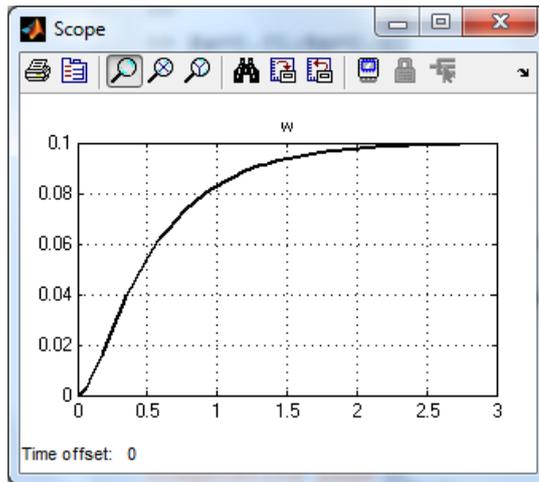


Рис. 2.69. Изменение скорости вращения двигателя постоянного тока

$$L_a \approx \frac{30U_N c_v}{\pi p n_N I_N}, \quad (2.35)$$

где  $c_v$  - эмпирический коэффициент, равный 0,4 или 0,1 соответственно для двигателей при отсутствии или при наличии компенсационной обмотки;  $p$  - число полюсов у двигателя.

Если неизвестен момент инерции ротора, то его можно определить по формуле

$$J = \frac{MD^2}{4}, \quad (2.36)$$

где  $M$  - масса ротора,  $D$  - его диаметр.

Обычно

$$J \geq (5 - 20) \frac{L_a P_N^2}{R_a^2 \omega_N^2 I_a^2}, \quad (2.37)$$

где  $\omega_N$  - номинальная угловая скорость,  $\omega_N = \frac{2\pi n_N}{60}$ .

Электромагнитный коэффициент

$$K_e = \frac{1}{k_d} = \frac{U_N - I_N R_a}{\omega_N}. \quad (2.38)$$

Для маломощных двигателей существенно наличие зоны нечувствительности или напряжения трогания  $U_{тр}$ . Их величина, если пренебречь моментом сопротивления, оценивается в виде

$$U_{mp} \approx I_N R_a. \quad (2.39)$$

## 2.2. Линеаризация

Если коэффициенты в математической модели являются постоянными величинами, то такой элемент является линейным. Большинство реальных систем и устройств нелинейные, т.е. коэффициенты в описывающих их динамику уравнениях являются переменными. Нелинейные элементы сложнее исследовать, однако их модели часто удается линеаризовать при условии *малых отклонений сигналов от стационарных значений*. Многие механические и электрические элементы в достаточно широком диапазоне изменения переменных можно рассматривать как линейные. Этого нельзя сказать о тепловых и гидравлических элементах, которые чаще всего по принципу своего действия являются нелинейными. Рассмотрим пример.

Пусть один из элементов системы управления представляет собой маятник (рис. 2.70). Такой маятник, например, может играть роль чувствительного элемента в датчике вертикали, предназначенном для определения углов наклона объекта.

Запишем уравнение колебаний маятника. Входной величиной является момент  $M$ , действующий на массу  $m$ , а выходной – угол его отклонения от положения равновесия  $\theta_0 = 0^\circ$ . Следовательно:

$$M = mgL \sin \theta, \quad (2.40)$$

где  $g$  – ускорение силы тяжести.

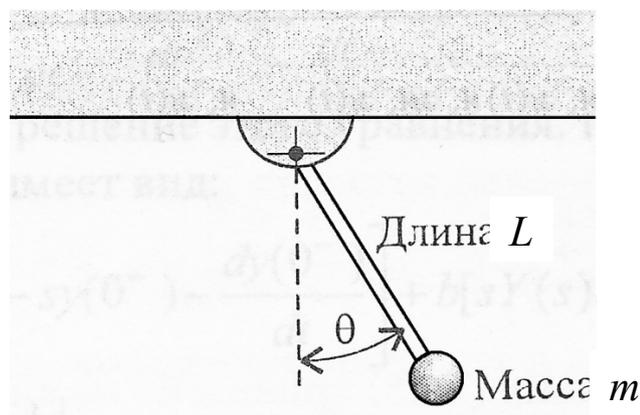


Рис. 2.70. Маятник

Графическое отображение зависимости (2.40) представлено на рис. 2.71.

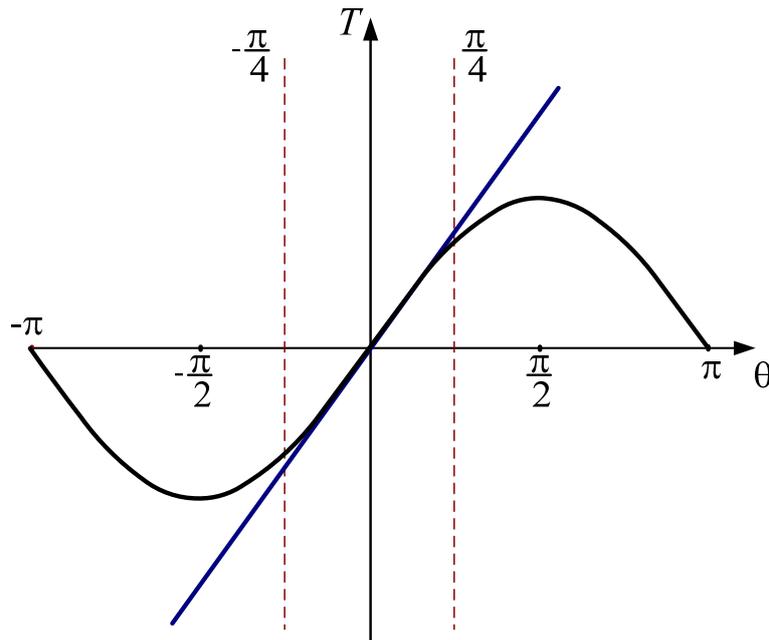


Рис. 2.71. Колебания маятника

Проведем касательную к кривой  $M(t) = f[\theta(t)]$  в рабочей точке, в данном случае в точке равновесия  $\theta_0 = 0^\circ$ . Угол наклона этой касательной характеризуется значением производной в рабочей точке  $\theta_0$ :

$$\left. \frac{df}{d\theta} \right|_{\theta=\theta_0} \quad (2.41)$$

Таким образом, разложив функцию  $M(t) = f[\theta(t)]$  в ряд Тейлора в окрестности рабочей точки и отбросив производные высших порядков, получим

$$M = M(\theta_0) + \left. \frac{dmgL \sin \theta}{d\theta} \right|_{\theta=\theta_0} \cdot \frac{\theta - \theta_0}{1!} = M(\theta_0) + mgL \left. \frac{d \sin \theta}{d\theta} \right|_{\theta=\theta_0} \cdot (\theta - \theta_0). \quad (2.42)$$

Тогда

$$M - M(\theta_0) = mgL \cos \theta_0 \cdot (\theta - \theta_0). \quad (2.43)$$

Учитывая, что  $M_0 = 0$  и  $\theta_0 = 0^\circ$ , окончательно получим

$$M = mgL \cos 0^\circ \cdot \theta = mgL\theta = k\theta, \quad (2.44)$$

где  $k = mgL$  – тангенс угла наклона касательной к кривой  $f(\theta)$  в рабочей точке.

Подобная аппроксимация является достаточно приемлемой в диапазоне  $-\pi/4 \leq \theta \leq \pi/4$ . Так, колебания линейной модели в диапазоне  $\pm 30^\circ$  от положения равновесия отличаются всего на 2% от действительных колебаний маятника.

Рассмотрим другой пример.

Пружина, используемая в автомобильном амортизаторе, создает силу  $f = kx^3$ , где  $x$  – деформация пружины. Получим линейную модель пружины при  $x_0 = 1$ .

Точка равновесия  $x_0$  определяется из условия, что упругая сила пружины равна весу груза  $mg$ , т.е.  $f_0 = mg$ . Если нелинейная пружина характеризуется зависимостью  $f = kx^3$ , то в положении равновесия  $x_0 = \sqrt[3]{mg}$ . В нашем случае,  $x_0 = \sqrt[3]{mg} = 1$ .

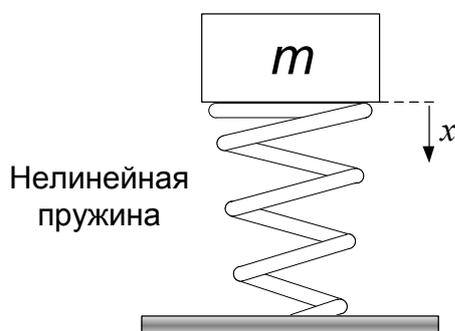


Рис. 2.72. Модель системы с нелинейной пружиной

Разложим функцию  $f(x)$  в ряд Тейлора и отбросим члены высшего порядка малости.

$$f(x) \approx f(x_0) + \left. \frac{df(x)}{dx} \right|_{x=x_0} \cdot (x - x_0),$$

$$f(x) \approx f(x_0) + 3kx^2 \Big|_{x=x_0} \cdot (x - x_0).$$

У нас  $f(x) = kx^3$ , следовательно

$$kx^3 \approx k + 3k(x-1) = 3kx - 2k.$$

Пусть  $k = 1$ , тогда

$$f(x) \approx 3x - 2$$

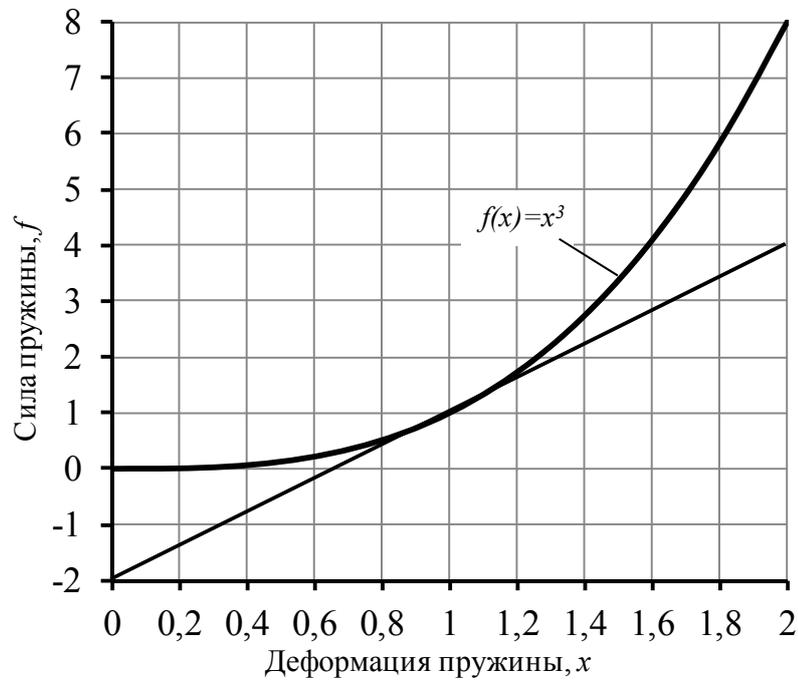


Рис. 2.73. Изменение силы нелинейной пружины

## 3. СОЗДАНИЕ ПОДСИСТЕМ

Модель системы управления может иметь сложную структуру, что существенно затрудняет работу с такой системой. Например, модели подвески автобуса (рис. 2.51) и двигателя (рис.2.67) весьма громоздки, и работать с ними неудобно. Для более наглядного представления структуры модели ее представляют в виде набора отдельных пользовательских блоков, не показывая их содержимого. Такие пользовательские блоки называют *подсистемами*. Использование подсистем уменьшает количество блоков, которые отображаются на экране, облегчает отладку и редактирование модели, повышает ее информативность.

### 3.1. Формирование подсистемы

Роль подсистем и процесс их создания рассмотрим на примерах моделей двигателя постоянного тока и подвески автобуса, представленных на рис. 2.51 и 2.67 соответственно.

Рассмотрим сначала модель двигателя постоянного тока (рис.2.67):

– Нажмем левую кнопку мыши и, не отпуская её, выделим все блоки и линии связи, предназначенные для включения в подсистему (рис.3.1). Иногда предварительно требуется передвинуть некоторые блоки так, чтобы лишние блоки не были выделены.

– Теперь непосредственно формируем подсистему. Это можно сделать несколькими способами:

1) Первый способ заключается в использовании команды **Create subsystem** меню **Edit** окна модели (рис. 3.2). Эту же команду можно выбрать в контекстном меню, которое вызывается щелчком правой кнопки мыши на выделенной части системы (рис. 3.3).

2) Можно также нажать комбинацию клавиш **Ctrl+G**. Модель системы примет вид, представленный на рис. 3.4. На месте выделенных блоков появился блок **Subsystem** с входными портами **In1** и **In2** для сигналов **U** и **M**, и выходным портом **Out1**. В общем случае входных и выходных портов столько же, сколько входных и выходных переменных в подсистеме.

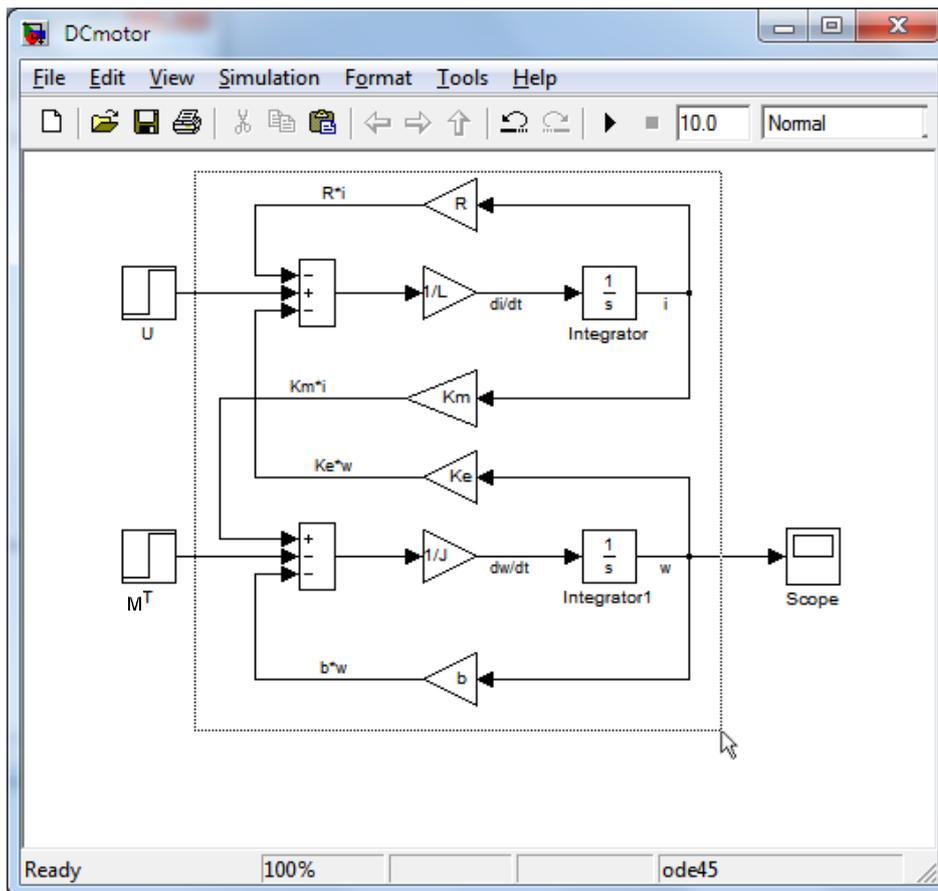


Рис. 3.1. Выделение блоков, входящих в подсистему

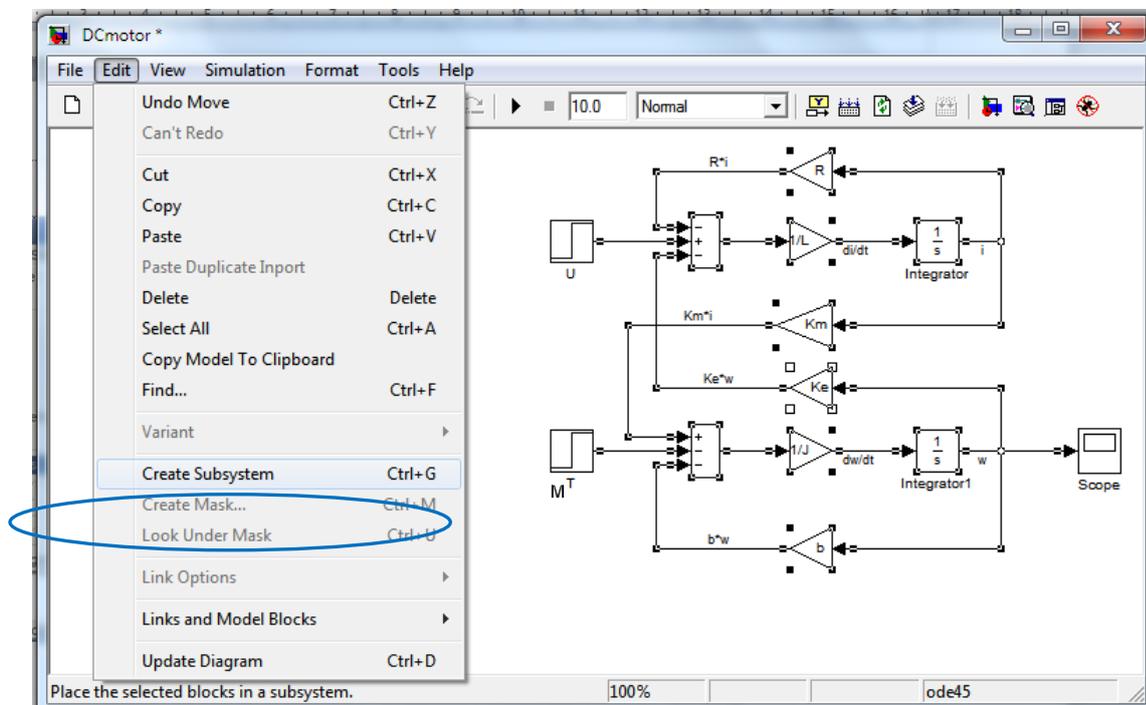


Рис. 3.2. Создание подсистемы из меню **Edit**

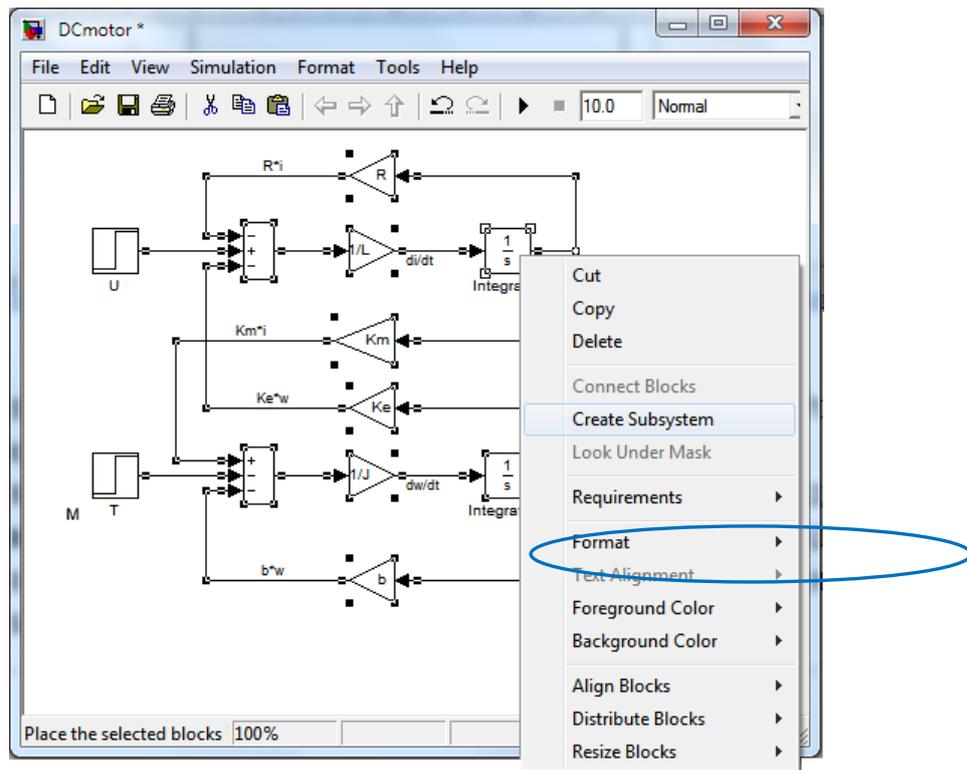


Рис.. 3.3. Создание подсистемы при помощи контекстного меню

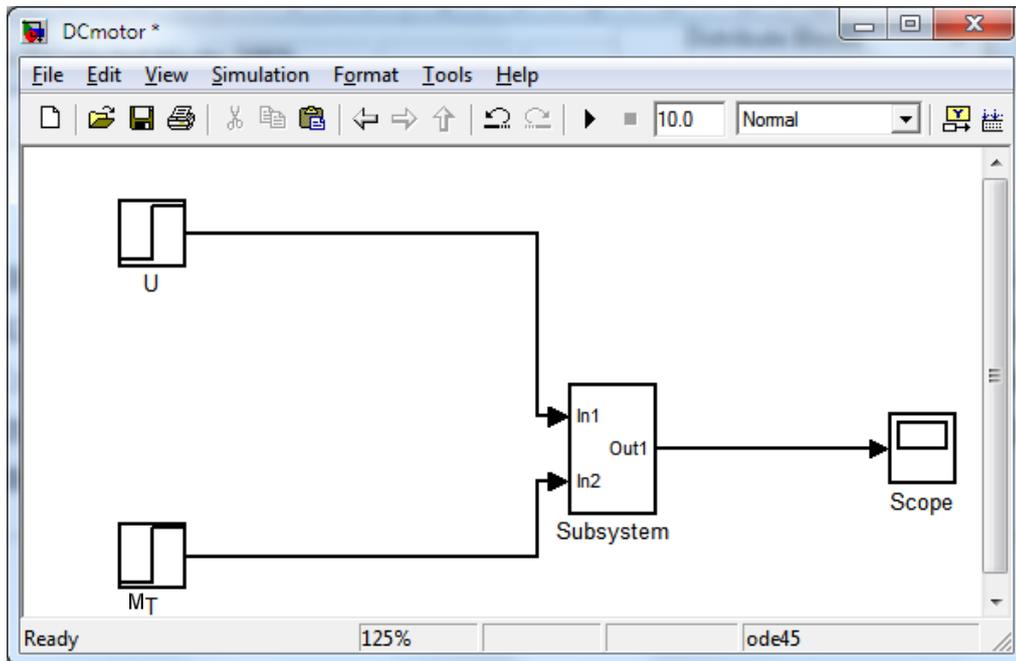


Рис. 3.4. Модель двигателя в виде подсистемы

Если раскрыть окно блока `Subsystem`, дважды щелкнув на нем, то Simulink отобразит блок-схему нашего двигателя (рис. 3.5). Как видно внутри подсистемы появились блоки `In1`, `In2` и `Out` для отображения входов и выходов в систему высшего уровня.

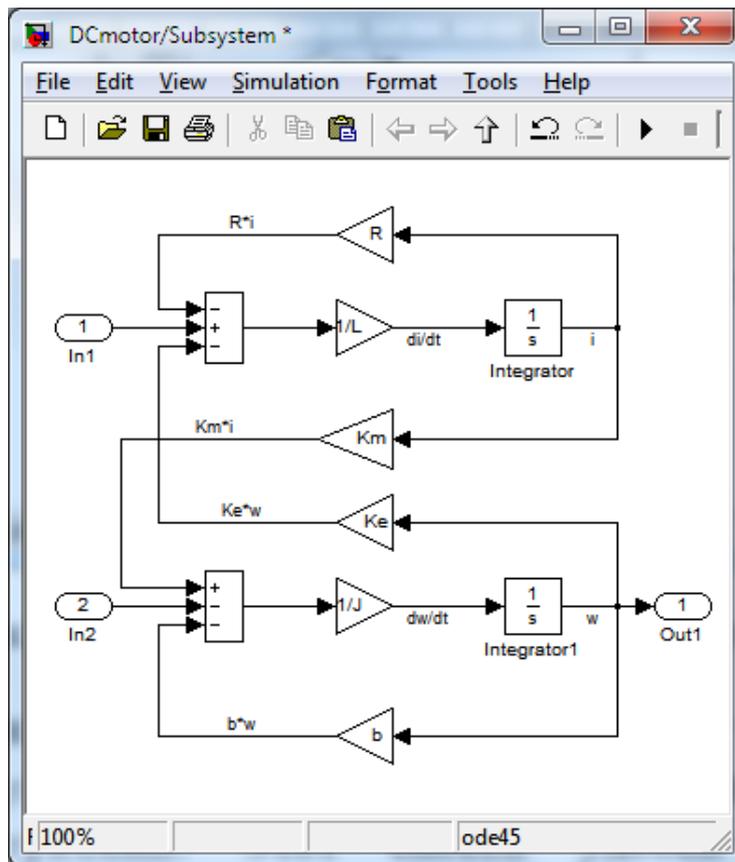


Рис. 3.5. Просмотр содержимого подсистемы

Второй способ создания подсистемы состоит в использовании блока `Subsystem` (Подсистема) из библиотеки блоков `Ports & Subsystems`. Этот способ удобен при создании подсистемы непосредственно в процессе построения модели.

Для создания подсистемы с использованием блока `Subsystem`, требуется перенести его в окно модели и дважды нажать левую кнопку мыши на изображении этого блока. В результате откроется окно подсистемы (рис. 3.6).

Далее необходимо сформировать структурную схему подсистемы. Все входящие в подсистему линии связи должны быть соединены с выходными портами блоков `In`, а все выходящие из подсистемы линии связи должны быть связаны с входными портами блоков `Out`.

Теперь перейдем к оформлению модели. Полученную подсистему желательно переименовать, так как имя `Subsystem` о назначении данного блока ничего не говорит. Дадим подсистеме имя `DC Motor` (двигатель постоянного тока).

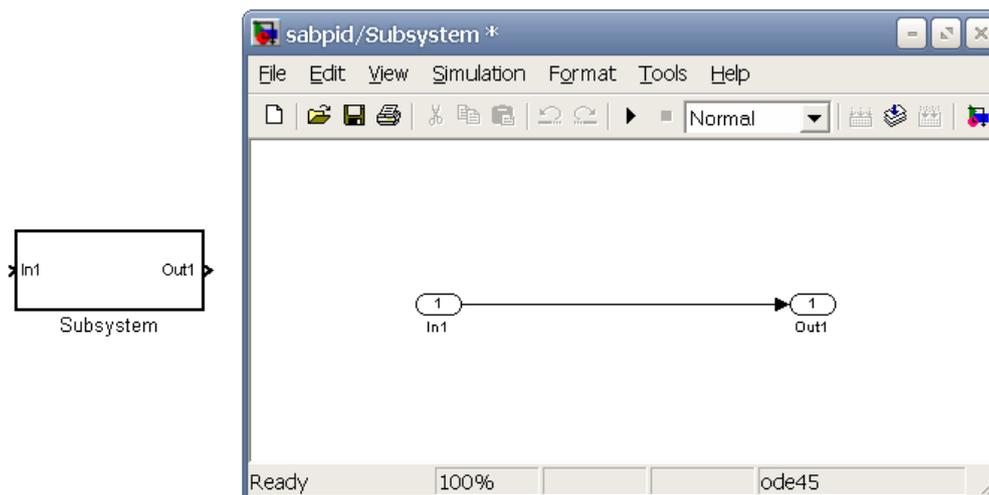


Рис. 3.6. Блок Subsystem из библиотеки Ports & Subsystems

Желательно также переименовать входы In и выходы Out подсистемы, чтобы иметь представление о входных и выходных переменных. Напомним, что войти в режим редактирования имени блока можно, щелкнув левой кнопкой мыши на старом имени. Окончательный вид модели системы управления представлен на рис. 3.7.

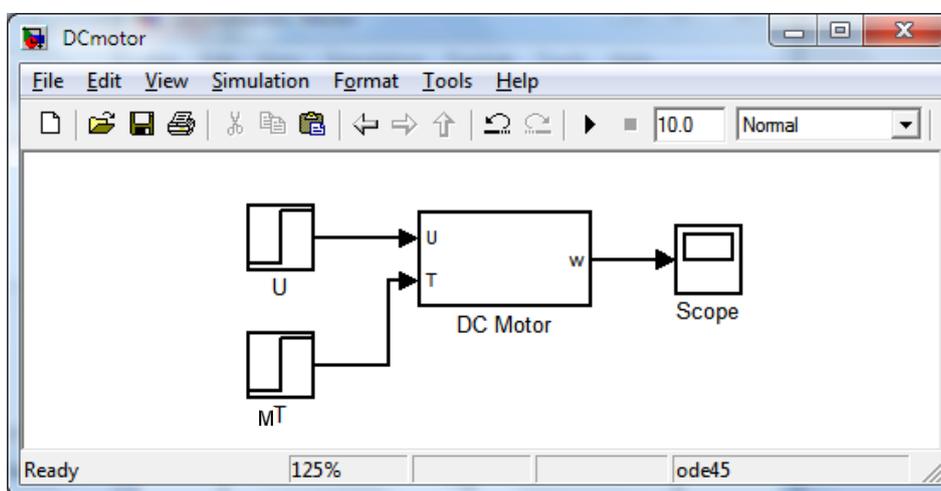


Рис. 3.7. Модель двигателя

Теперь, создадим подсистему модели подвески автобуса (рис.2.51).

– Нажмем левую кнопку мыши и, не отпуская её, выделим все блоки и линии связи, предназначенные для включения в подсистему (кроме блоков Step и Scope).

- Используя комбинацию клавиш **Ctrl+G**, создадим подсистему.
- Дадим созданной подсистеме имя «Bus Suspension».
- Откроем подсистему двойным щелчком мыши и переименуем блок In1 на «u», блок In2 на «w», а блок Out1 на «x1-x2» (рис. 3.8).

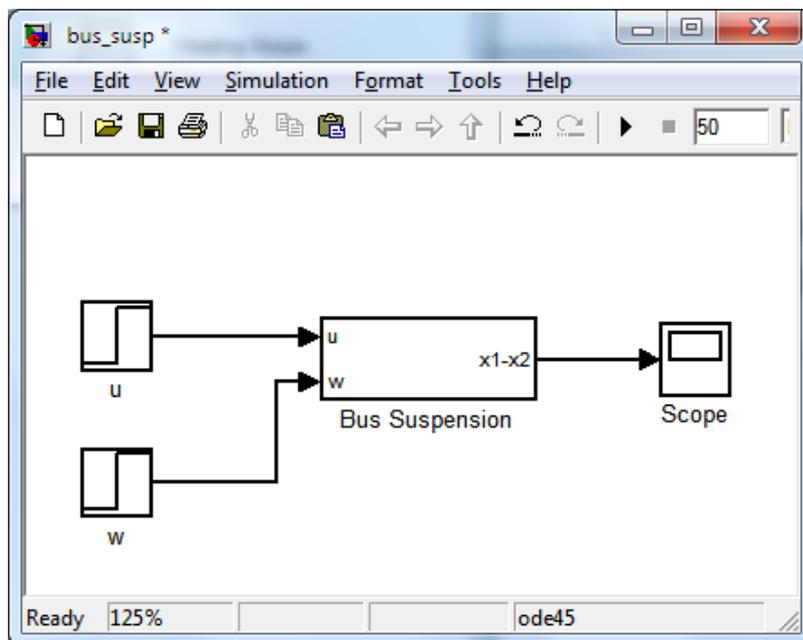


Рис. 3.8. Модель пневмоподвески автобуса

## 3.2. Маскирование подсистемы

### 3.2.1. Общие сведения

Simulink предоставляет пользователю возможность оформлять подсистемы в виде обычных библиотечных блоков. Эта операция называется *маскированием*. Маскированному блоку можно задать своё графическое изображение, сформировать диалоговое окно, в котором можно указывать параметры блока, создать справочную информацию об его назначении и работе. Таким образом, маскирование обеспечивает гибкость при работе с моделью, упрощает построение сложных моделей, повышает наглядность модели, предохраняет подсистему от несанкционированного изменения.

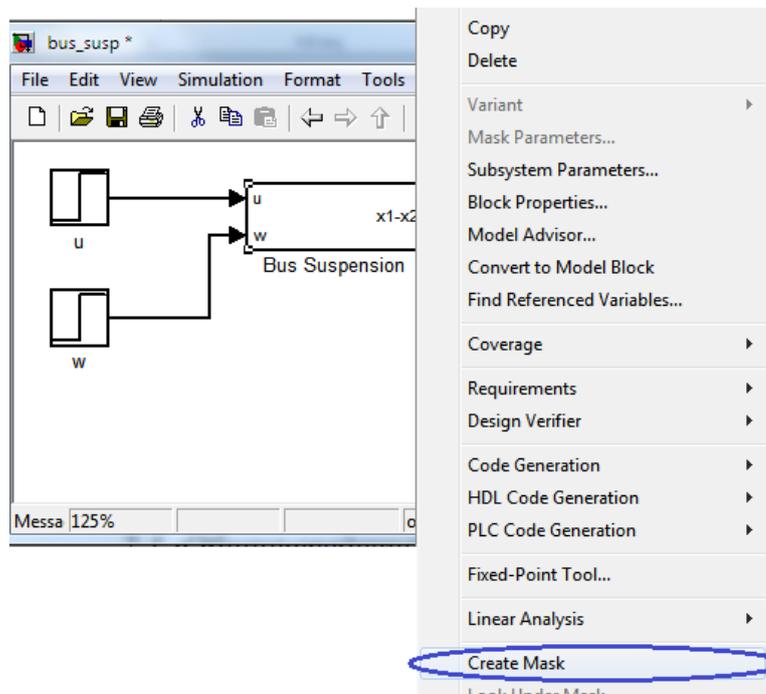


Рис.. 3.9. Маскирование подсистемы при помощи контекстного меню

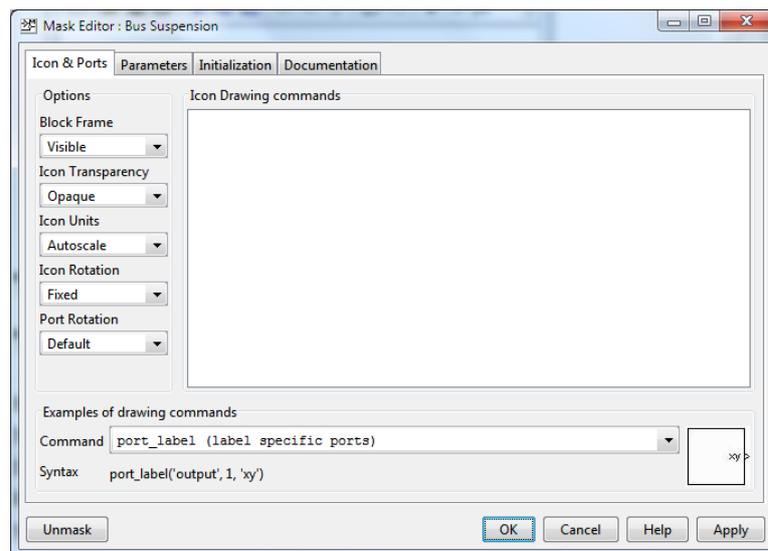


Рис. 3.10. Окно **Mask Editor** для подсистемы Bus Suspension

Для того чтобы маскировать подсистему её необходимо выделить и выбрать команду **Create Mask** из меню **Edit** в окне модели или из контекстного меню, которое вызывается щелчком правой кнопки мыши на блоке подсистемы. Можно также нажать комбинацию клавиш **Ctrl+M**. При этом появится окно редактора маски **Mask Editor** (рис. 3.10).

Окно **Mask Editor** имеет четыре вкладки: **Icon&Ports**, **Parameters**, **Initialization** и **Documentation**. Остановимся немного на назначении каждой из этих вкладок.

### *3.2.2. Создание графического изображения подсистемы*

Первая вкладка окна **Mask Editor**, открытая по умолчанию – вкладка **Icon&Ports**. Как следует из названия, здесь формируется изображение маскированного блока. Это изображение, или иначе, иконка, пиктограмма, может содержать текст, выражение, Рис. или график.

Вкладка **Icon&Ports** разделена на три части. Большую часть вкладки занимает поле **Icon Drawing commands**, в котором при помощи команд на языке MATLAB формируется графическое изображение блока. Примеры написания некоторых из этих команд приводятся в поле **Examples of drawing commands**. Здесь в раскрывающемся списке **Command** даны названия команд и в скобках их назначение, а ниже приводится пример синтаксиса команды. При этом возможный вид изображения блока отображается в правом нижнем углу вкладки **Icon&Ports**. По умолчанию указана команда `port_label`, которая выводит на экран имена входных и выходных портов (для подсистемы `Bus Suspension` это имена «u», «w» и «x1-x2»).

Очень удобной является возможность отображения в блоке подсистемы рисунка. Это существенно повышает наглядность модели.

Для отображения в блоке рисунка, находящегося в файле, используется команда

```
image(imread('имя_файла'))
```

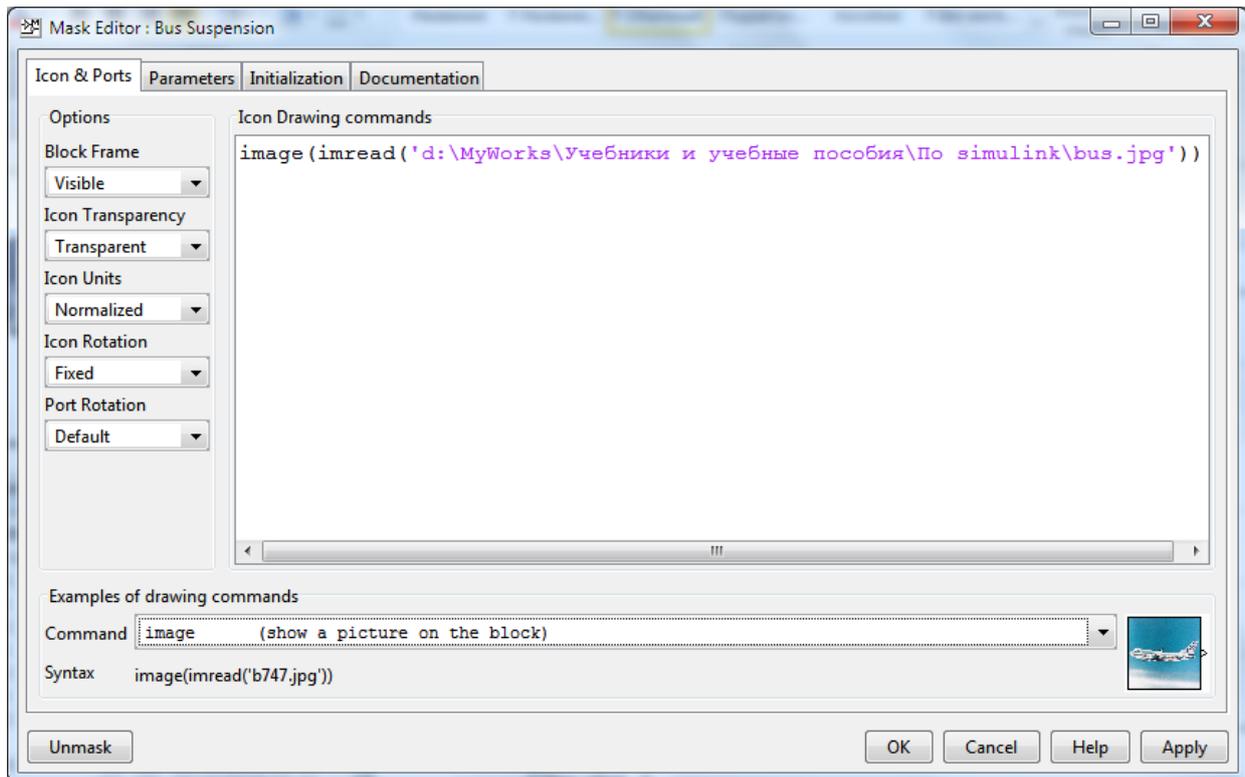
где `имя_файла` – имя графического файла с расширением, причем поддерживаются только RGB изображения, например, с расширением `*.jpg`. `Bitmap` файлы, например, с расширением `*.gif`, необходимо преобразовать в RGB формат.

Для корректной работы команды `imread` файл с рисунком должен находиться в одной папке с файлом модели, в противном случае необходимо указать полный путь. Опыт показывает, что

полный путь к файлу с изображением лучше указывать всегда. На рис. 3.11 показан пример отображения рисунка на блоке подсистемы.



*a*



*б*

Рис. 3.11. Отображение рисунка на блоке подсистемы

Для создания рисунка на блоке можно также использовать специальный редактор, который вызывается следующей командой в рабочей строке MATLAB

```
>> iconedit('имя_модели', 'имя_подсистемы')
```

Например,

```
>> iconedit('bus_susp', 'Bus Suspension')
```

В результате появляется графическое окно **Block Icon Editor** (рис. 3.12) в котором можно создать Рис. для подсистемы. Рис. создается по точкам, расположение которых указывается с помощью мыши.

Между собой точки соединяются прямыми линиями. Для того чтобы начать новую линию необходимо нажать клавишу **n** на клавиатуре. Для отмены создания последней точки используется клавиша **d**. Выход из режима рисования осуществляется клавишей **q**.

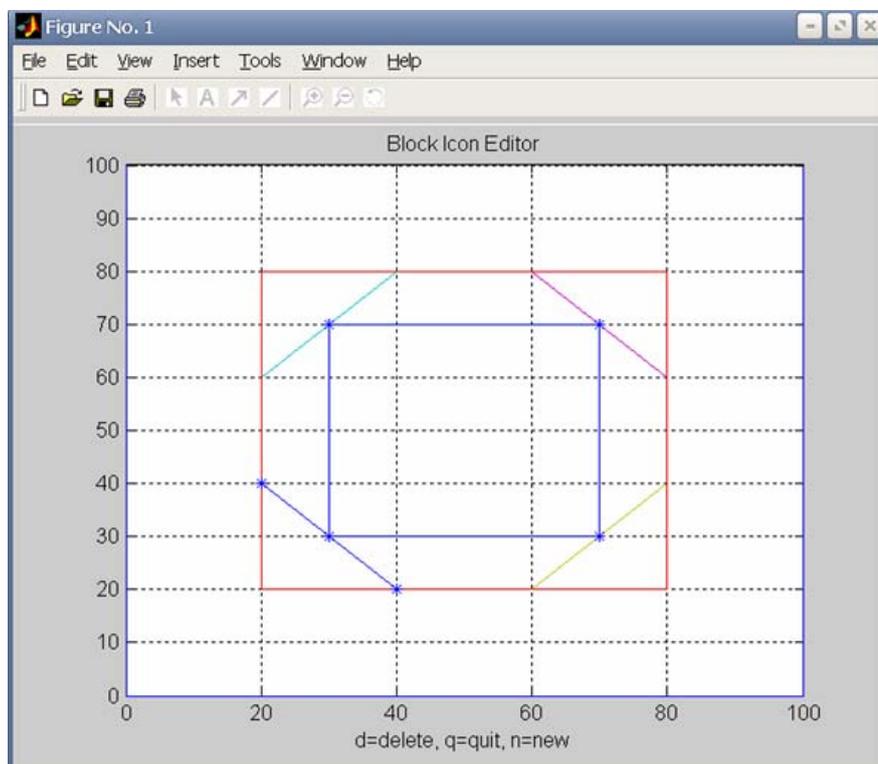


Рис. 3.12. Окно **Block Icon Editor**

После выхода из редактора рисунок в блоке подсистемы обновляется (рис. 3.13), а в поле **Icon Drawing commands** и в командной строке MATLAB выводится команда, обеспечивающая построение рисунка. Например, для фигуры, изображенной на рис. 3.12.

ans =

```
>>plot(0,0,100,100,[20,20,80,80,20],...
[80,20,20,80,80],[40,20],[80,60],[80,60],...
[60,80],[60,80],[20,40],[20,40],[40,20],...
[30,30,70,70,30],[70,30,30,70,70])
```

В левой части вкладки **Icon&Ports** расположена группа параметров **Options** (Опции изображения). Эта группа содержит четыре раскрывающихся списка.

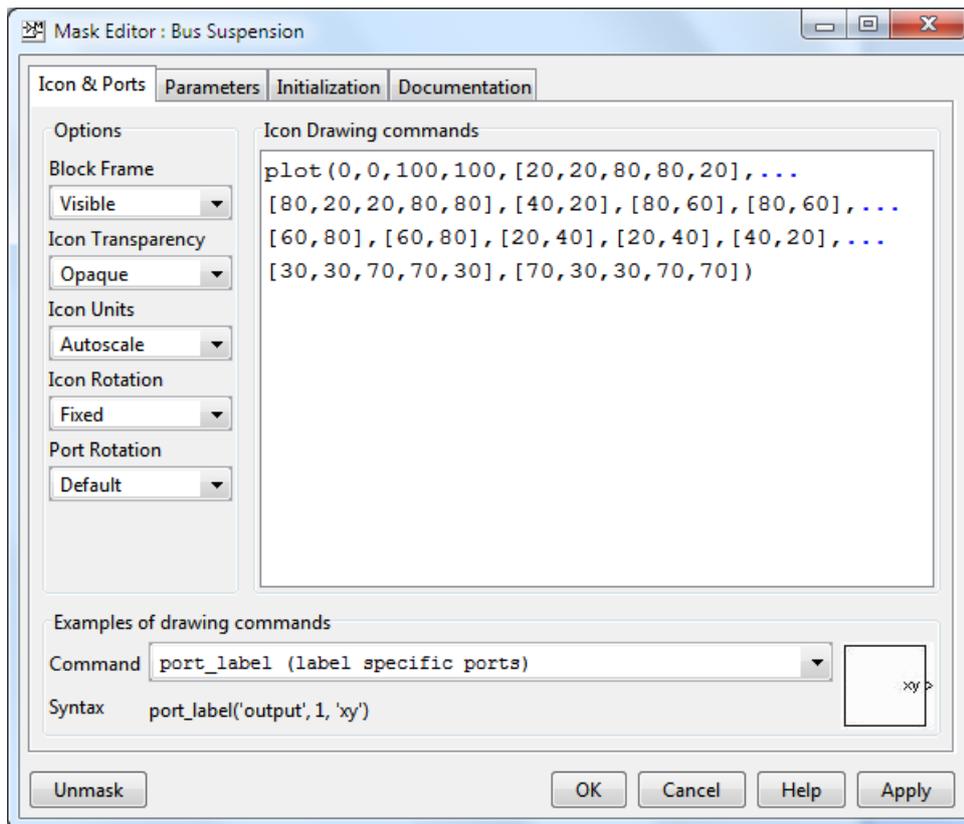


Рис. 3.13. Вид блока с созданным рисунком и поле **Drawing commands** с соответствующей командой

**Block Frame** – выбор способа отображения рамки блока: **Visible** – рамка видна; **Invisible** - рамка не видна.

**Icon Transparency** – настройка прозрачности изображения блока: **Opaque** – изображение не прозрачное; **Transparent** - изображение прозрачное.

**Icon Units** – задание способа масштабирования изображения: **Autoscale** – автоматическое масштабирование, Рис. занимает максимально возможную площадь внутри блока; **Pixels** – размер рисунка задается в пикселях и не изменяется при изменении масштаба блока; **Normalized** – постоянный масштаб рисунка, при изменении размере блока подсистемы пропорционально изменяется и масштаб рисунка.

**Icon Rotation** – настройка возможности вращения рисунка при повороте блока: **Fixed** – при повороте блока ориентация рисунка не изменяется; **Rotates** – Рис. вращается вместе с блоком.

**Port Rotation** – устанавливается способ отображения портов маскированного блока при вращении его по часовой стрелке. Список содержит 2 позиции. **Default** – порты переупорядочиваются; эта

опция, принятая по умолчанию, наиболее удобна при использовании моделей систем управления. **Physical** – порты вращаются вместе с блоком без переупорядочивания. Рис. 3.14 иллюстрирует работу команд из списка **Port Rotation**.

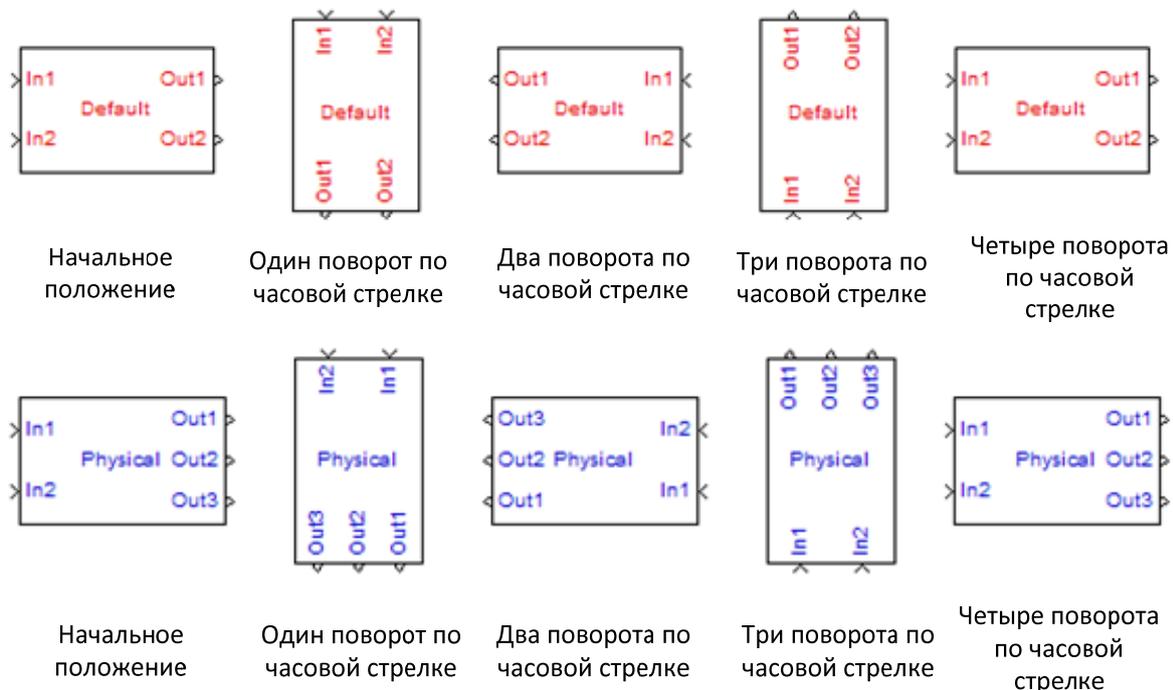


Рис. 3.14. Вращение блоков с помощью команды **Port Rotation**

### 3.2.3. Задание параметров подсистемы

Вкладка **Parameters** позволяет создавать собственное диалоговое окно параметров маскированной подсистемы и задавать переменные, которые соответствуют этим параметрам. Вкладка состоит из следующих элементов (рис. 3.15): панели **Dialog Parameters**, панели **Dialog options for selected parameter** и **Type-specific options**.

На панели **Dialog Parameters** устанавливаются и редактируются параметры маскированной подсистемы. Панель представляет собой таблицу, каждая строка которой отображает основные свойства одного из параметров подсистемы. Таблица имеет шесть столбцов.

**Prompt** – здесь вводится текст, который будет в диалоговом окне описывать данный параметр.

**Variable** – в этом столбце задаются имена локальных переменных, значения которым будут потом присвоены в диалоговом окне маскированной подсистемы. Имя переменной

должно совпадать с переменной, указанной в элементах подсистемы, если нет, то тогда эти переменные необходимо приравнять друг к другу на вкладке **Initialization**.

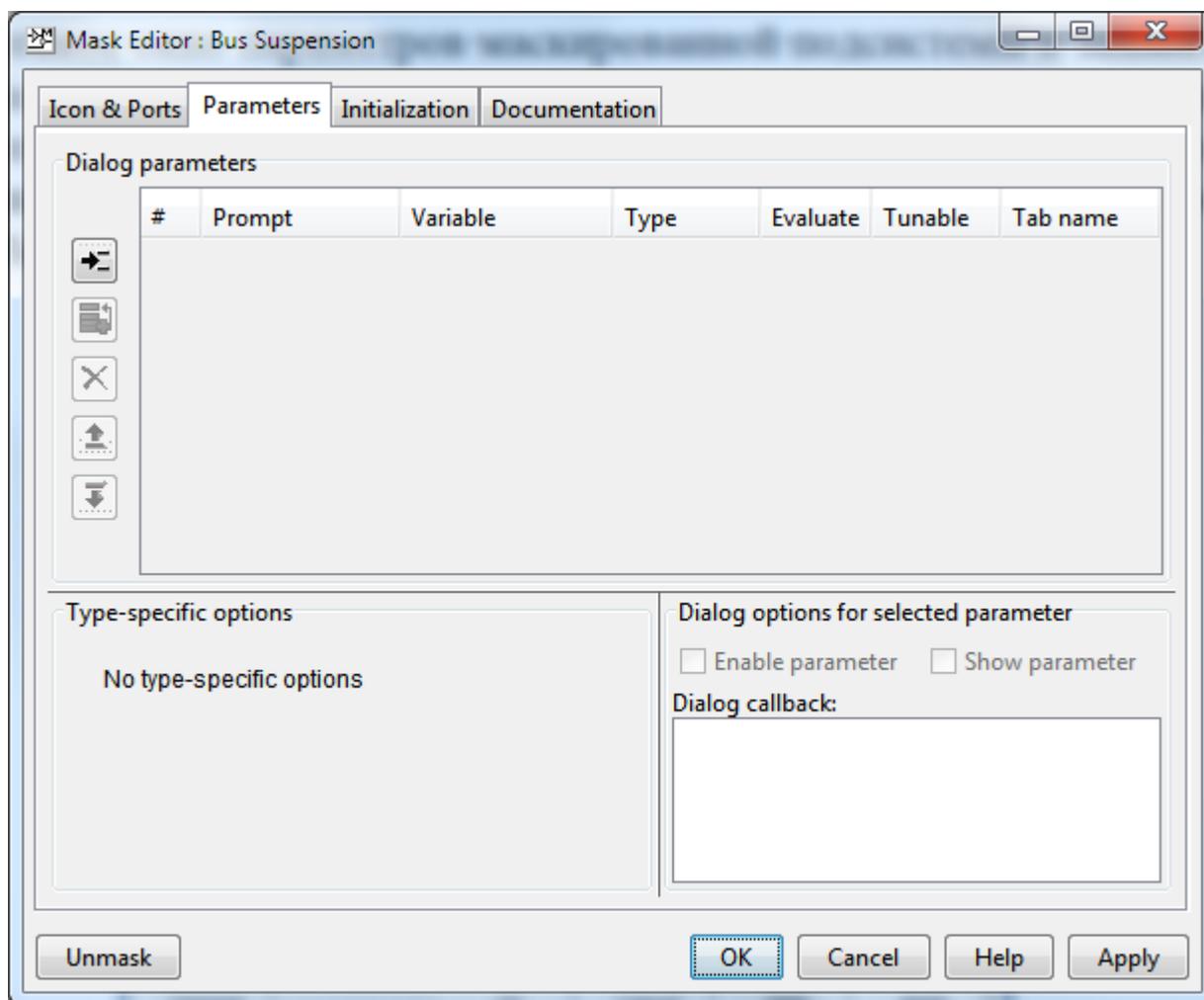


Рис. 3.15. Вкладка **Parameters** диалогового окна **Mask Editor**

Следует отметить, что блоки, в маскированной подсистеме не имеют доступа к переменным, размещенным в рабочей области MATLAB. Маскированная подсистема имеет свою собственную область памяти, независимую от рабочей области MATLAB и других маскированных подсистем в модели. Это устраняет возможность конфликтов имен переменных. Кроме того, Simulink не делает различий между прописными и строчными буквами в имени локальной переменной маскированной подсистемы, поэтому, например Gain, GAIN и gain трактуется как одно и то же имя.

В столбце **Type** в раскрывающемся списке выбирается способ задания значения данного параметра. Наиболее удобно использовать следующие позиции: **edit**, **checkbox** и **popup**.

Выбор позиции **edit**, принятой по умолчанию, позволяет вводить значения параметра в специальном текстовом поле диалогового окна параметров подсистемы.

При выборе позиции **checkbox** в диалоговом окне параметров маскированной подсистемы будет сформирован флажок, позволяющий выбрать значение переменной.

Позиция **popup** позволяет пользователю выбрать требуемое значение параметра из раскрывающегося списка. Сам список возможных значений задается в окне **Popups** в поле **Option for selected parameter**.

Опция **Evaluate** определяет тип переменной данного параметра. В зависимости от того, установлен флажок или нет, значение переменной может принимать числовое или символьное значение (табл. 3.1).

**Tunable** – выбор этой опции позволяет пользователю изменять значение данного параметра непосредственно в течение процесса моделирования.

**Tub name**: здесь устанавливается имя вкладки в таблице, в которой должен появиться соответствующий параметр.

В левой части вкладки **Parameters** окна **Mask Editor** расположены кнопки, служащие для управления списком параметров подсистемы.

 (**Add parameter**) – добавляет новый параметр в список параметров маскированной подсистемы. При нажатии этой кнопки на панели **Dialog Parameters** становится активной новая строка.

 (**Promote underlying block parameter(s) to this mask parameter**) – открывает диалоговое окно, в котором можно выбрать параметр для активации.

 (**Delete parameter**) – эта кнопка позволяет удалить текущий параметр.

 (**Move up**) – перемещает выделенный параметр на одну строку вверх.

 (**Move down**) – перемещает выделенный параметр на одну строку вниз.

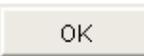
Значение локальной переменной при различных установках опций  
**Type** и **Evaluate**

Type	Evaluate	
	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<b>edit</b>	В поле вводится числовое значение или выражение	Переменной присваивается символьное значение
<b>checkbox</b>	Флажок установлен - переменной присваивается значение 1. Флажок снят - переменной присваивается значение 0	Флажок установлен - переменной присваивается значение 'on'. Флажок снят - переменной присваивается значение 'off'
<b>popup</b>	Переменной, священной со списком, присваивается значение, равное порядковому номеру пункта	Переменной присваивается значение символьной строки, соответствующей выбранному пункту

Рассмотрим сначала процесс задания параметров для маскированной подсистемы *Bus Suspension* из рис. 3.15.

– Для того чтобы вызвать редактор маски **Mask Editor** уже маскированной подсистемы, необходимо выделить подсистему и выбрать команду **Edit mask** из меню **Edit** в окне модели или из контекстного меню.

– В появившемся окне **Mask Editor** выберем вкладку **Parameters**. При помощи кнопки  (**Add parameter**) создаем локальные переменные (рис. 3.16), при этом в поле **Prompt** указываем описание переменных, а в поле **Variable** – их имена. Эти же имена должны быть указаны в блоках, входящих в состав подсистемы. Остальные параметры оставим принятыми по умолчанию.

Создание локальных переменных завершено. Нажимаем кнопку  и в окне модели дважды щелкаем левой кнопкой мыши на маскированной подсистеме. Подсистема не открывается, зато появляется диалоговое окно (рис.3.17), в котором предлагается указать значения локальных переменных маскированной подсистемы.

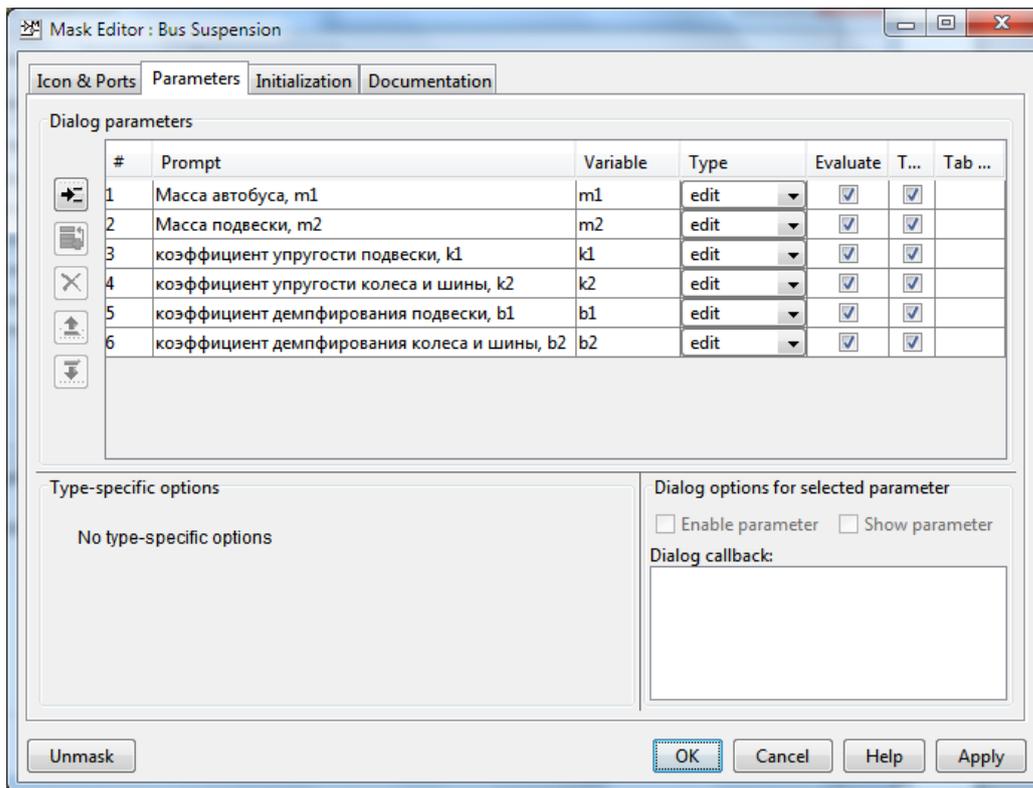


Рис. 3.16. Пример задания локальных переменных

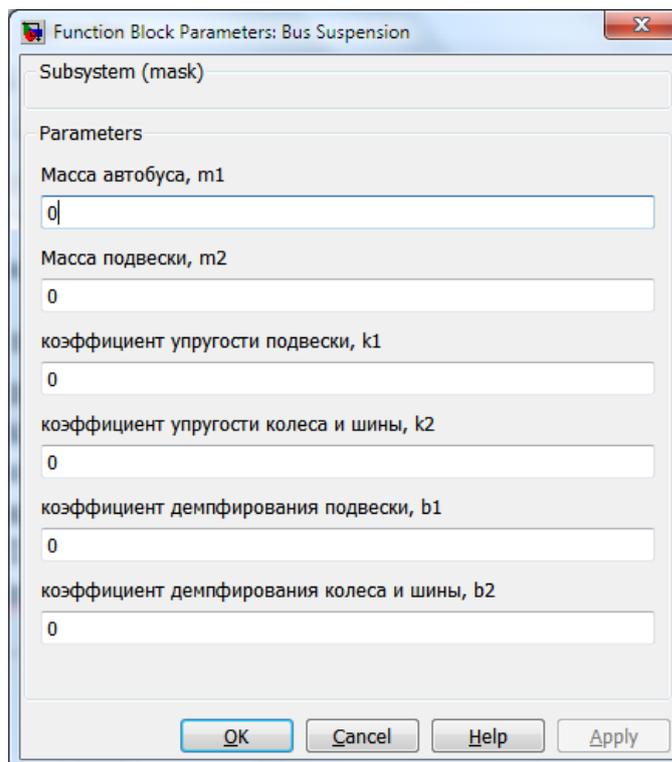


Рис. 3.17. Диалоговое окно локальных переменных маскированной подсистемы

Присвоим каждому из параметров системы соответствующие значения:  $m_1=2500$ ;  $m_2=320$ ;  $k_1=80000$ ;  $k_2=500000$ ;  $b_1 = 350$ ;  $b_2 = 15020$  (рис. 3.18). Теперь с подсистемой можно работать как с обычным блоком Simulink.

Открыть маскированную подсистему можно при помощи команды **Look under mask** из меню **Edit** или из контекстного меню необходимо, а также нажав комбинацию клавиш **<Ctrl + U>**.

Для изменения параметров маскированной подсистемы, необходимо выделить подсистему и выбрать команду **Edit mask** из меню **Edit** в окне модели или из контекстного меню.

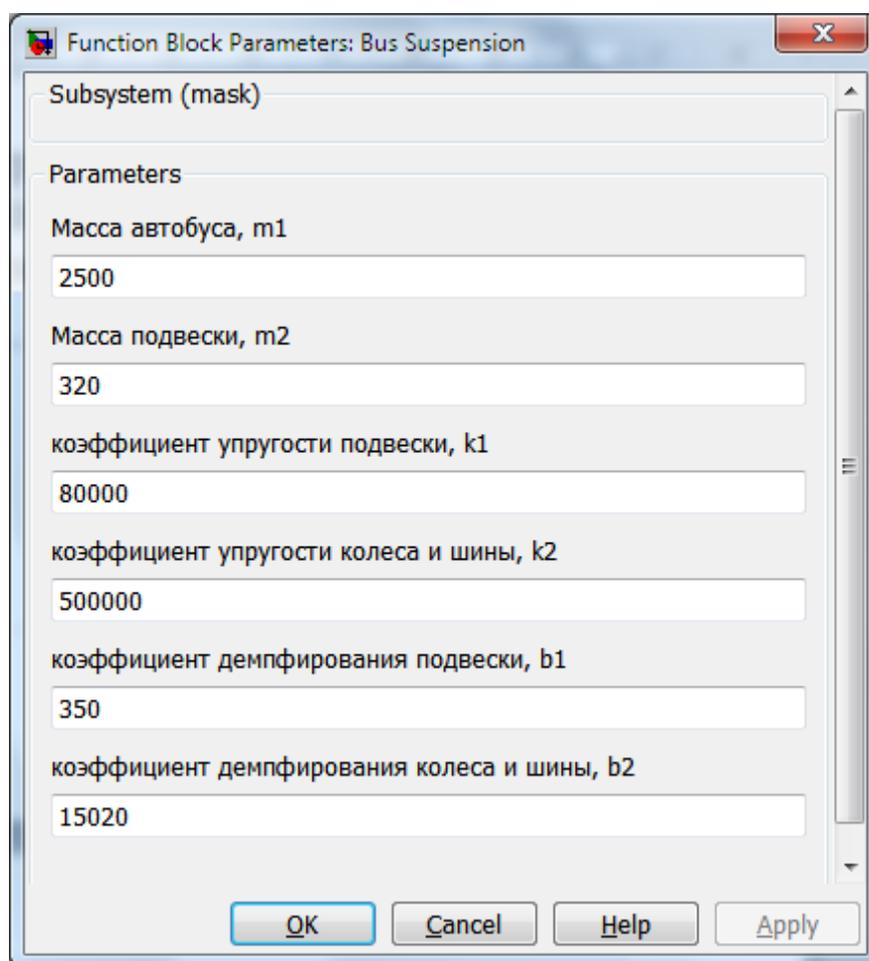


Рис. 3.18. Диалоговое окно параметров подсистемы

Чтобы отменить маскирование подсистемы, нужно в окне **Mask Editor** нажать кнопку **Unmask**.

### *Вкладка Initialization*

Вкладка **Initialization** (рис. 3.19) позволяет пользователю применять команды MATLAB для дополнительной настройки маскированной подсистемы. Вкладка содержит три поля.

Поле **Dialog variables** содержит список переменных, которые были заданы на вкладке **Parameters**. При необходимости здесь эти имена можно изменить, чтобы не возвращаться на предыдущую вкладку.

В поле **Initializations commands** вводятся команды для дополнительной настройки параметров маскированной подсистемы. Здесь, например, можно задать начальные значения параметров, установить связь между параметрами подсистемы, заданными на вкладке **Parameters** с переменными, указанными в блоках подсистемы, записать команды, выводящие в командном окне MATLAB сообщение о выполнении (невыполнении) какого-то условия в процессе моделирования и т.д.

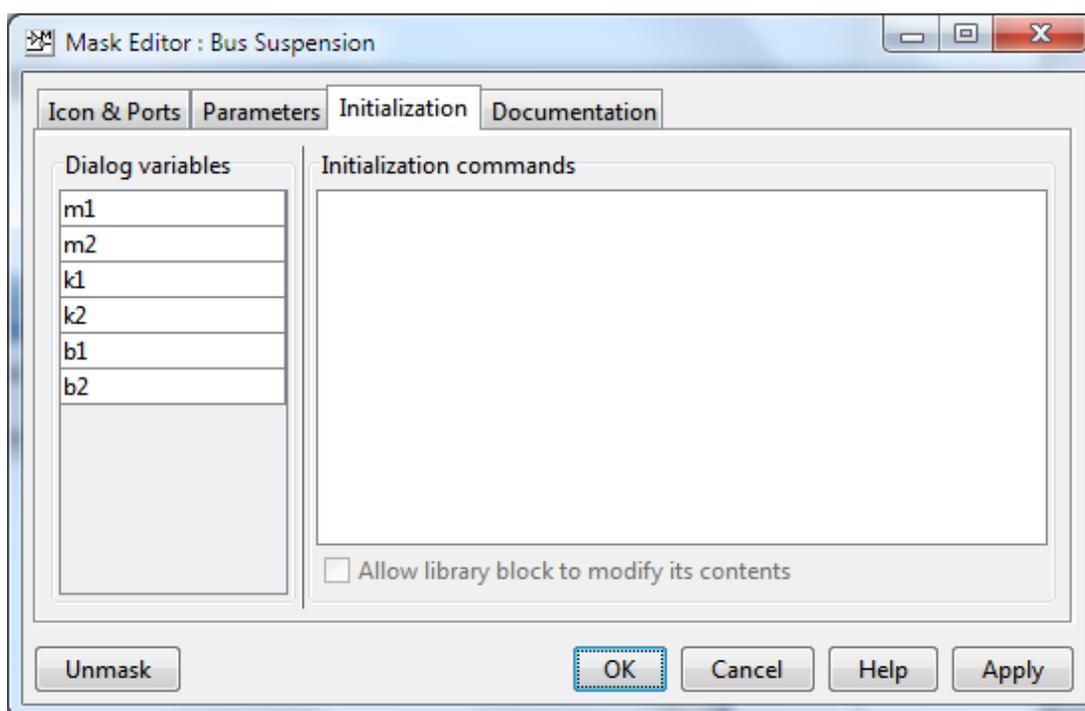


Рис. 3.19. Вкладка **Initialization** окна **Mask Editor**

В поле **Initializations commands** можно использовать любые команды и функции на языке MATLAB. Чтобы результаты выполнения команд не отображались в рабочем окне MATLAB, их нужно закрывать точкой с запятой (;).

Флаг **Allow library block to modify its components** становится доступным лишь в том случае, если маскированная подсистема помещена в библиотеку блоков. При установленном флаге разрешается изменять содержимое маскированной подсистемы: добавлять или удалять блоки, изменять их параметры.

### 3.2.4. Создание справочной информации

Вкладка **Documentation** (рис. 3.20) служит для создания справочной информации о маскированной подсистеме. Вкладка имеет три текстовых поля.

В поле **Mask type** рекомендуется указать имя, характеризующее тип блока подсистемы, например, «Пневмоподвеска». Это имя будет отображаться в заголовке диалогового окна параметров маскированной подсистемы (рис. 3.20). Чтобы пользователь мог отличить маскированную подсистему от обычного встроенного блока, Simulink в окне параметров добавляет «(mask)».

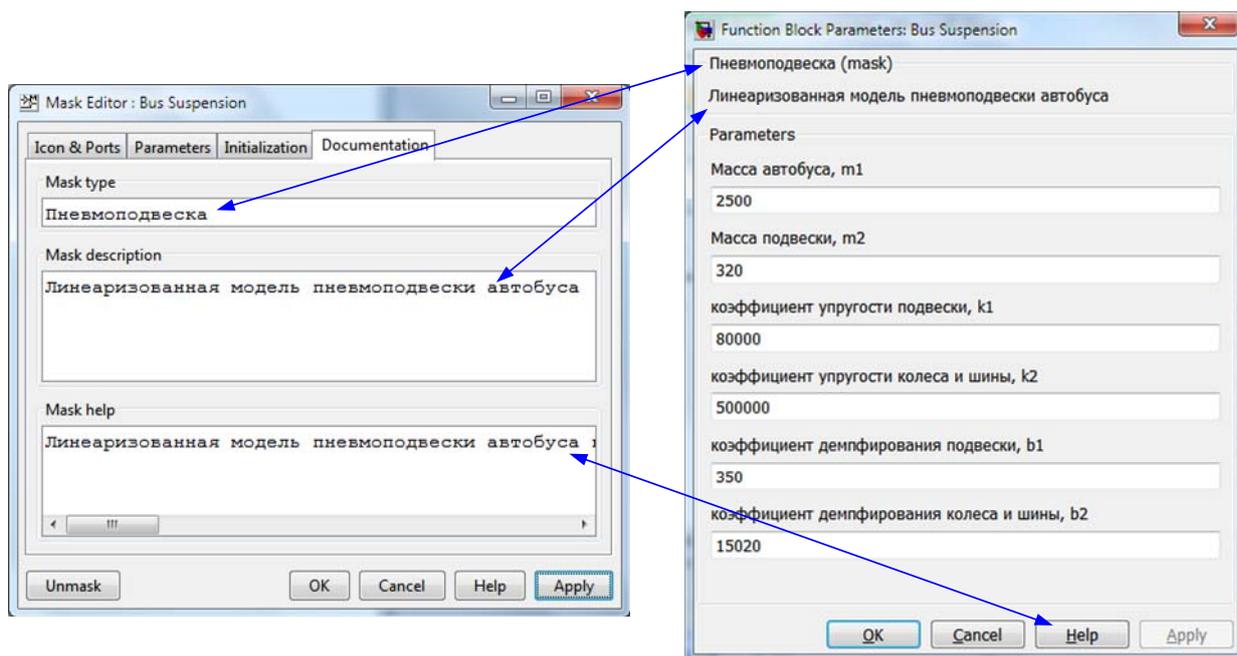


Рис. 3.20. Вкладка **Documentation** окна **Mask Editor** и ее связь с окном параметров маскированной подсистемы

**Mask description** - описание маски. В этом поле вводится краткая информация о назначении подсистемы. Эта информация также будет отображаться в окне параметров подсистемы.

В поле **Mask help** рекомендуется ввести подробную информацию о назначении подсистемы и о том, как с ней работать. Эта информация будет размещена в справочной системе Simulink (рис. 3.21); ее можно будет просмотреть, нажав на кнопку  в диалоговом окне параметров маскированной подсистемы или выбрав команду **Help** из ее контекстного меню.

Теперь создадим теперь маскированную подсистему для двигателя постоянного тока (рис. 2.67).

- Выделим щелчком мыши подсистему DC Motor (рис. 3.7) и нажмем клавиши **<Ctrl+M>**.

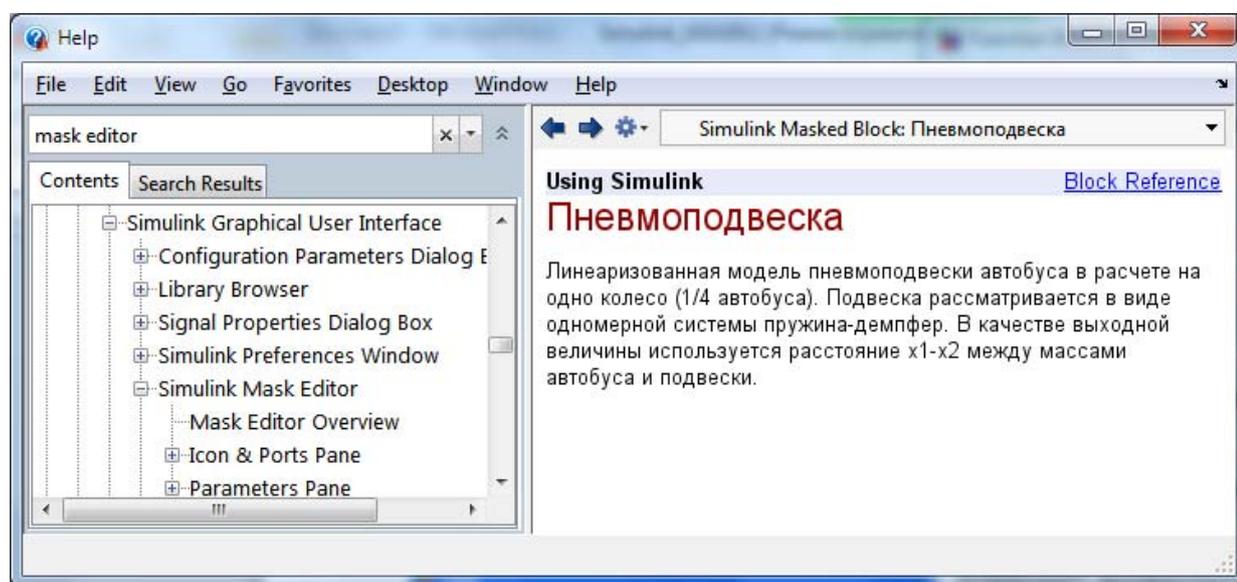


Рис. 3.21. Вывод справки о маскированной подсистеме

- В появившемся окне **Mask Editor: DC Motor** на вкладке **Icon&Ports** при помощи команды `image(imread('имя_файла.jpg'))` зададим путь к файлу с изображением двигателя.

- Перейдем на вкладку **Parameters** и нажмем кнопку  (**Add parameter**), при этом станет активной первая строка в таблице **Dialog parameters**.

- Щелкнем левой кнопкой мыши в ячейке **Prompt** и запишем в ней «Соппротивление якоря, Ом» и нажмем клавишу **<Enter>**.

- Поставим курсор в следующую ячейку **Variable** и укажем в ней переменную «R», соответствующую сопротивлению цепи якоря. Подтвердим ввод клавишей **<Enter>**.

- Щелкнем левой кнопкой мыши в ячейке **Tab name** и запишем в ней «Электрическая часть».
- Вновь нажмем кнопку  (**Add parameter**), чтобы добавить в таблице **Dialog parameters** новую строку.
- В ячейке **Prompt** запишем «Индуктивность якоря, Гн».
- В ячейке **Variable** запишем переменную «L», соответствующую индуктивности цепи якоря.
- Выделим во второй строке ячейку **Tab name** и запишем в ней «Электрическая часть».
- Добавим новую строку к таблице **Dialog parameters** и в ячейке **Prompt** запишем «Постоянная якоря, Нм/А», в ячейке **Variable** укажем имя этой переменной «Km», а в ячейке **Tab name** запишем «Электрическая часть».
- Аналогичным образом введем информацию об электромеханической постоянной двигателя «Ke», которая имеет размерность в·с/рад.
- Применим введенные изменения в подсистеме нажатием кнопки **Apply** в правом нижнем углу окна **Mask Editor**.
- Перейдем в окно модели и дважды щелкнем мышью на блоке подсистемы DC Motor. При этом на экране должно появиться окно, показанное на рис. 3.22 справа внизу.

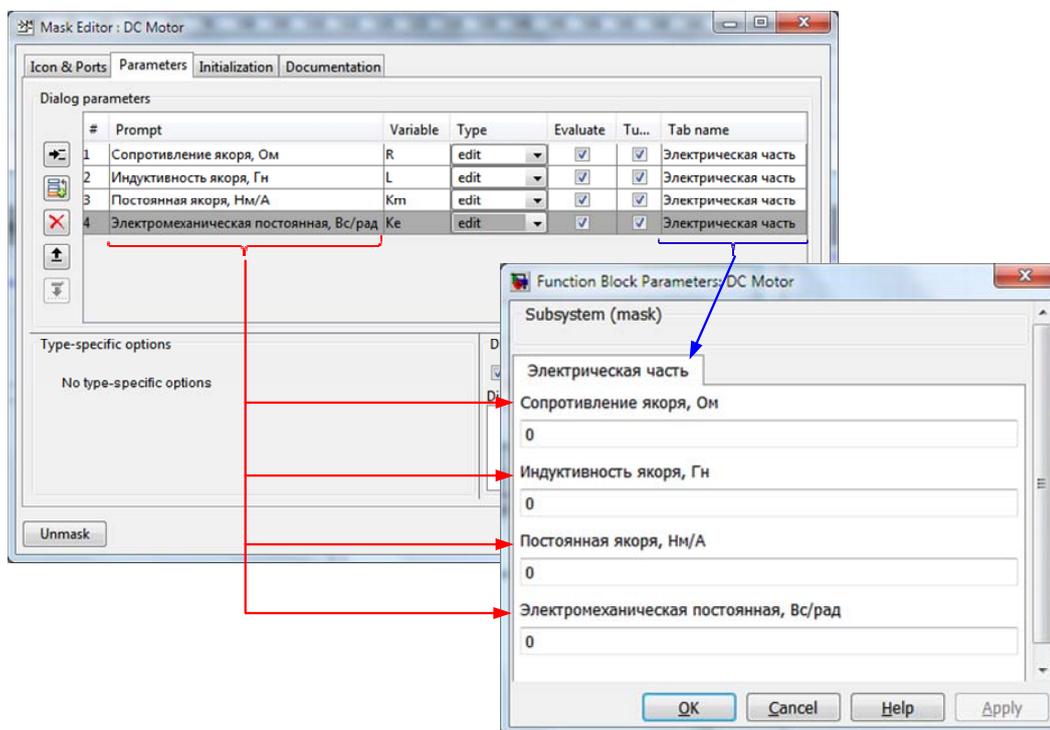


Рис. 3.22. Готовое диалоговое окно маскированной подсистемы

Теперь добавим параметры механической части двигателя: момент инерции и коэффициент демпфирования.

- Выделим подсистему DC Motor и вызовем окно **Mask Editor** одновременным нажатием клавиш <Ctrl+M>. Напомним, что это же окно можно вызвать, выбрав команду **Edit Mask** из контекстного меню блока подсистемы или из меню **Edit** панели инструментов окна модели.

- Перейдем на вкладку **Parameters** и нажмем кнопку .

- В ячейке **Prompt** запишем «Момент инерции, кгм<sup>2</sup>/с<sup>2</sup>» .

- В ячейке **Variable** укажем соответствующую переменную «J».

- Перейдем к ячейке **Tab name** и запишем в ней «Механическая часть».

- Добавим в таблице **Dialog parameters** еще одну строку, нажав кнопку .

- В ячейке **Prompt** укажем «Коэффициент демпфирования, Нмс».

- В ячейке **Variable** зададим переменную «b».

- В ячейке **Tab name** снова запишем «Механическая часть» и нажмем кнопку **OK**.

Вернемся к нашей модели и дважды щелкнем мышью на блоке DC Motor. На экране появится окно **Function Block Parameters** с двумя вкладками: **Электрическая часть** и **Механическая часть**. На этих вкладках в соответствующих полях указываются численные значения электродвигателя, например, следующие (рис. 3.23):

- сопротивление цепи якоря  $R = 1$  Ом;

- индуктивность цепи якоря  $L = 0,5$  Гн;

- постоянная якоря  $K_m = 0,01$  Нм/А;

- электромеханическая постоянная двигателя  $K_e = 0,01$  В·с/рад;

- момент инерции ротора  $J = 0,01$  кгм<sup>2</sup>/с<sup>2</sup>;

- коэффициент демпфирования  $b = 0,1$  Нмс.

Нам осталось ввести справочную информацию о подсистеме DC Motor.

- Откроем окно Mask Editor, выделив блок подсистемы и нажав одновременно клавиши <Ctrl+M>.

- В окне Mask Editor перейдем на вкладку **Documentation**.

- В поле Mask type запишем «Двигатель постоянного тока».

- В поле Mask description введем краткое описание нашей модели, например: «Линеаризованная модель двигателя постоянного тока, управляемого по цепи якоря».

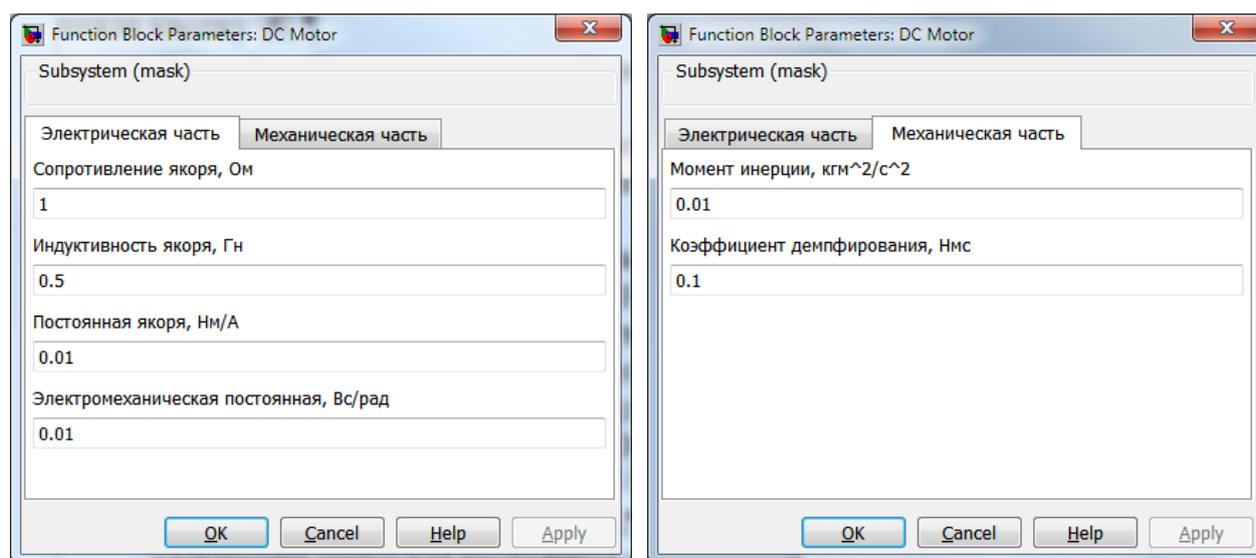
- В поле Mask help запишем более подробную информацию о назначении подсистемы (рис. 3.24), например: «Модель двигателя постоянного тока с независимым возбуждением, управляемого по цепи якоря. При построении модели сделаны следующие допущения:

- пренебрегали эффектами гистерезиса в обмотке;
- пренебрегали падением напряжения на щетках;
- магнитное поле считалось постоянным.

Для работы с моделью двойным щелчком мыши вызовите окно Function Block Parameters и задайте значения параметров двигателя».

- Нажмем кнопку ОК.

Введенный текст можно наблюдать в окне Help блока DC Motor (рис. 3.25).



а

б

Рис. 3.23. Вкладки окна **Function Block Parameters** двигателя с заданными значениями параметров: **а** – вкладка **Электрическая часть**; **б** – вкладка **Механическая часть**

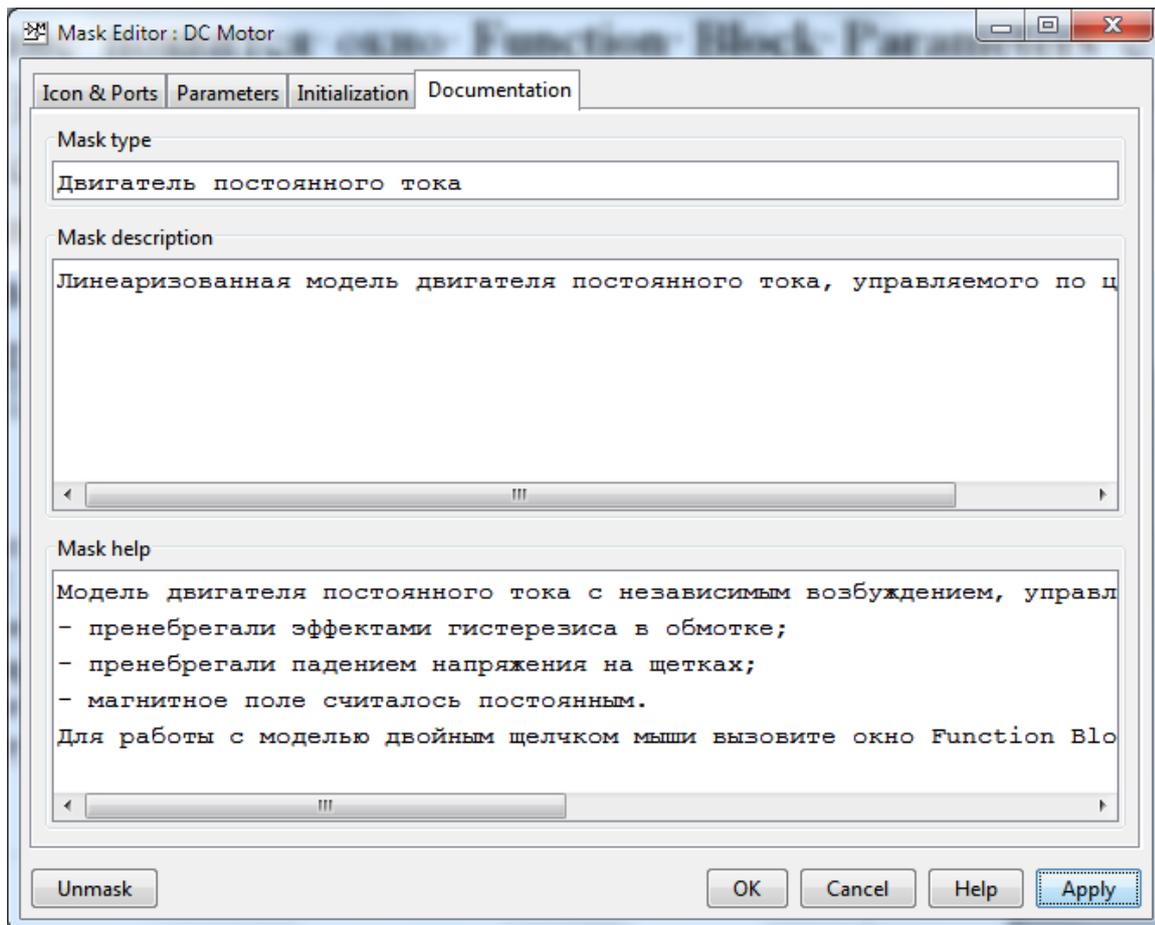


Рис. 3.24. Вкладка Documentation окна Mask Editor блока DC Motor

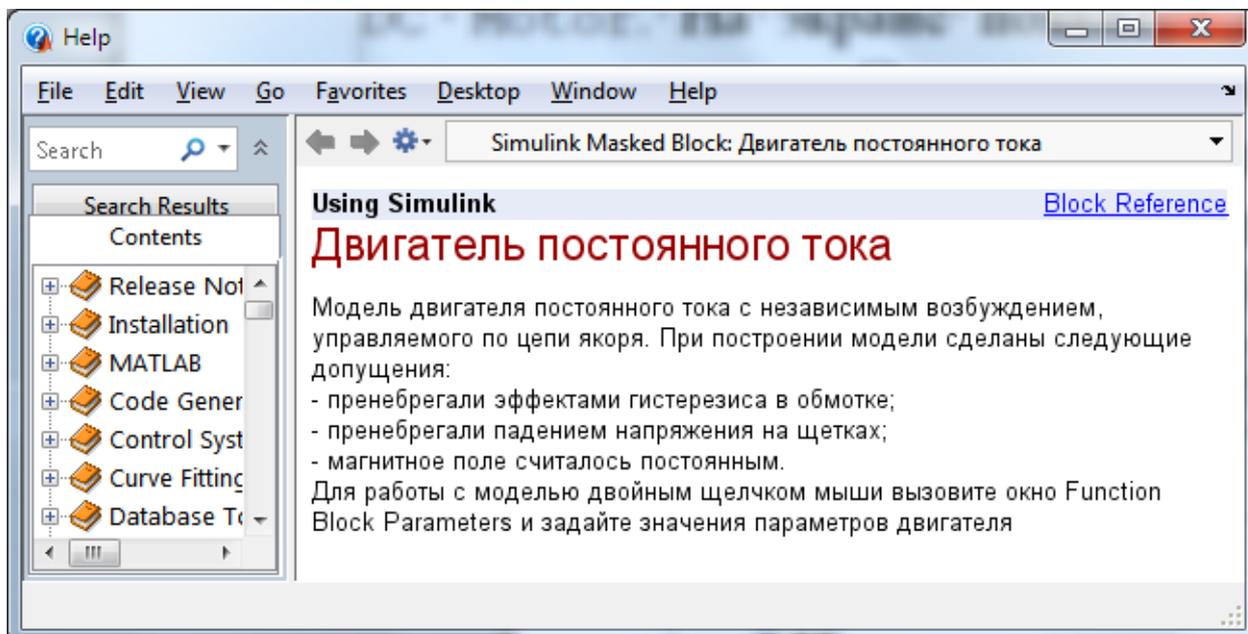


Рис. 3.25. Готовое окно Help блока DC Motor

## 4. УСТОЙЧИВОСТЬ ЛИНЕЙНЫХ СИСТЕМ

### 4.1 Общие сведения

В главах 2 и 3 были рассмотрены проблемы математического моделирования неизменяемой части и внешней среды системы управления. Казалось бы, теперь можно приступить к разработке закона управления, то есть алгоритма работы управляющего устройства. Но необходимо ещё определиться с требованиями, которые предъявляются к функционированию системы. Это необходимо сделать, поскольку для удовлетворения различных требований к системе необходимы и различные управляющие устройства.

Для определения требований к работе системы управления, вновь обратимся к ее функциональной схеме (рис. 4.1).

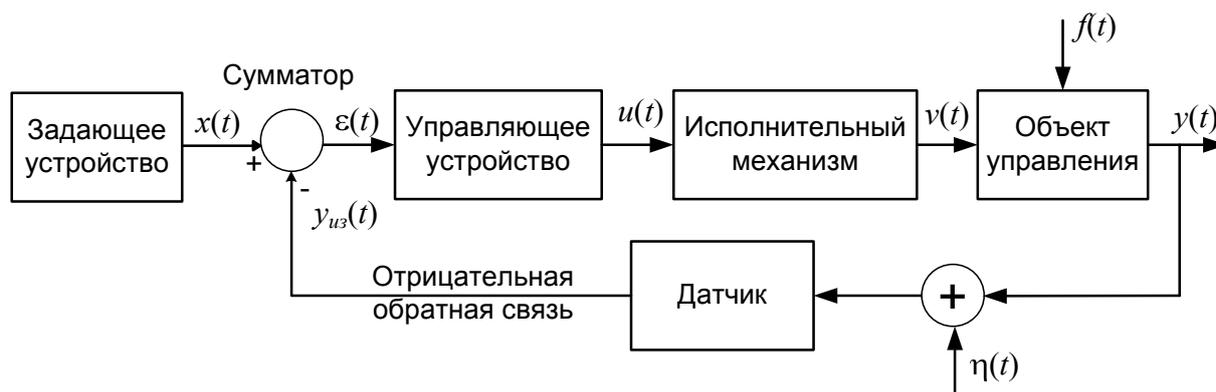


Рис. 4.1. Типовая функциональная схема замкнутой системы управления

Исходно представляется, что требования к системе можно сформулировать следующим простым образом: система должна «хорошо» обрабатывать все характерные задающие воздействия  $x(t)$  при всех характерных возмущениях  $f(t)$  и помехах измерений  $\eta(t)$ .

На интуитивном уровне «хорошей» можно считать обработку задающего воздействия  $x(t)$ , при которой выходная величина  $y(t)$  располагается достаточно близко к  $x(t)$  на всём временном интервале нормальной эксплуатации системы.

Пример «хорошей» и «плохой» обработки задающего воздействия приведён на рис. 4.2.

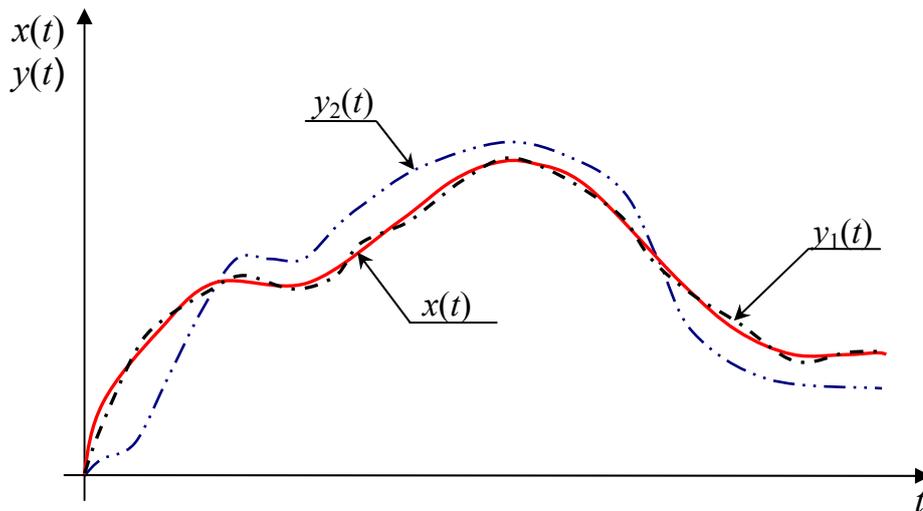


Рис. 4.2. Примеры «хорошей»  $y_1(t)$  и «плохой»  $y_2(t)$  отработки задающего воздействия  $x(t)$ .

Однако такая форма требований к свойствам системы управления неконструктивна, то есть не даёт математического аппарата разработки её закона управления. Требования «хорошей» отработки задающего воздействия  $x(t)$  целесообразно расчленить на три более простых конкретных требования. И эти требования предъявлять в формализованном виде, то есть в виде требований к численным показателям, однозначно определяемым процессами в системе.

Такое расчленение общего требования объясняется тем, что «хорошее» движение системы обеспечивается несколькими её свойствами, и достижение этих свойств требует специфических исследований и решений.

Перейдем к ознакомлению с формализуемыми требованиями к системам управления.

Чаще всего в качестве детерминированных характерных внешних воздействий  $x(t)$ ,  $f(t)$  и  $\eta(t)$  инженеры рассматривают временные полиномы  $A \cdot 1(t)$ ,  $A + B \cdot t$ ,  $A + B \cdot t + D \cdot t^2$ , гармонические воздействия  $A \cdot \sin \omega t$  или случайные функции времени с математическими ожиданиями  $m_x(t)$ ,  $m_f(t)$ ,  $m_\eta(t)$ , опять же представляющими собой одну из функций  $A \cdot 1(t)$ ,  $A + B \cdot t$ ,  $A + B \cdot t + D \cdot t^2$ ,  $A \cdot \sin \omega t$ .

При приложении таких внешних воздействий система управления, как и всякая динамическая система, приходит в движение, характер которого зависит от а) свойств внешних воздействий,

б) свойств системы и в) её исходного состояния.

Тот этап движения, на котором сказывается исходное состояние (проявляет себя собственная энергия неизменяемой части системы), называется *переходным процессом*.

Если внешние воздействия меняются по указанным выше законам, то система по истечении некоторого времени может втянуться в *установившееся движение*, характер которого уже зависит только от свойств внешних воздействий и свойств самой системы. Но может и не втянуться ни в какое установившееся движение, а попасть, например, в *режим колебаний с увеличивающейся амплитудой* и прекратить выполнять свои функции.

Соответственно этому для обеспечения «хорошей» отработки задающего воздействие  $x(t)$  целесообразно предъявлять требования:

- к способности системы втягиваться в установившиеся режимы, соответствующие рассматриваемым  $x(t)$ ,  $f(t)$  и  $\eta(t)$ , т.е. требование к устойчивости переходного процесса;
- к качествам переходных процессов системы;
- к свойствам возможных установившихся режимов системы.

## **4.2 Устойчивость установившихся режимов**

### *4.2.1 Понятие устойчивости переходного процесса.*

Способность или неспособность динамической системы втягиваться в установившийся режим, соответствующий рассматриваемым  $x(t)$ ,  $f(t)$  и  $\eta(t)$  – это достаточно неожиданное и неочевидное её свойство. По крайней мере, для инженеров второй половины XIXв., проектировавших паровые двигатели и двигатели внутреннего сгорания, неспособность систем управления некоторыми из вновь созданных агрегатов стабилизировать число оборотов их валов, оказалась непонятным явлением. Именно эта ситуация дала мощный толчок к ускоренному развитию теории устойчивости движения.

Казалось бы, если система управления устроена так, что при некоторых  $x(t)$ ,  $f(t)$  и  $\eta(t)$  в ней возможен только один установившийся режим  $y_{уст}(t)$ , то почему бы системе не стремиться в этот режим?

Проблема возникает не из-за того, что система может не стремиться войти в возможный установившийся режим, а из-за того, что может стремиться войти в него слишком энергично. На рис. 4.4 приведены процессы в двух системах управления, возможные установившиеся режимы  $y_{1уст}(t)$  и  $y_{2уст}(t)$  которых равны задающему воздействию  $x(t) = A$ , при  $x(t) = A \cdot 1(t)$ ,  $f(t) = 0$  и  $\eta(t) = 0$ .

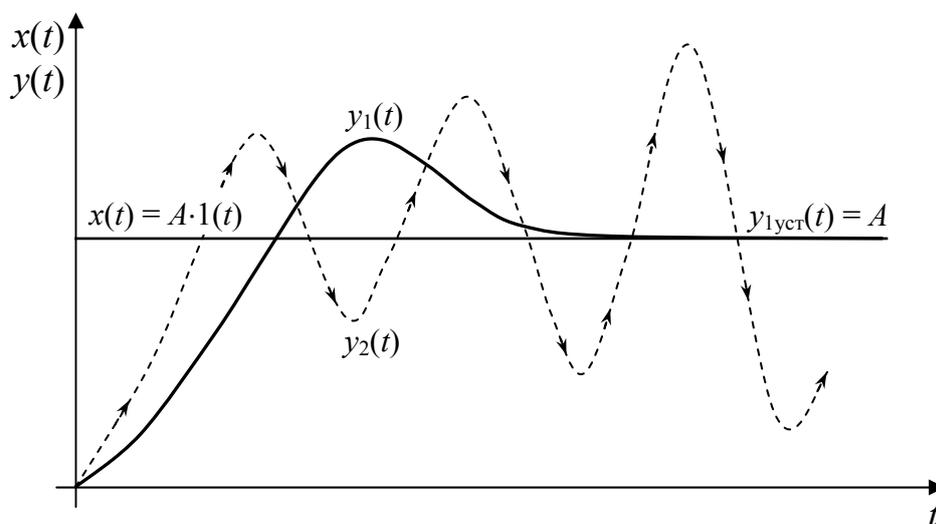


Рис. 4.4. Устойчивый  $y_1(t)$  и неустойчивый  $y_2(t)$  процессы в двух системах управления

Первая система стремится войти в  $y_{1уст}(t)$  умеренным образом и остаётся в этом режиме. В теории движения динамических систем такое движение системы к  $y_{1уст}(t)$  называется *устойчивым*.

Вторая система тоже стремится войти в  $y_{2уст}(t) = A$ . Стрелки на кривой переходного процесса подчёркивают это её стремление. Но стремится слишком энергично, каждый раз «проскакивая» режим  $y_2 = A$ . Такое движение системы к  $y_{2уст}(t)$  называется *неустойчивым*.

Подобное свойство наблюдается в системах с «жесткими<sup>1</sup>» регуляторами, резко реагирующими на появление даже малых отклонений  $\varepsilon(t) = x(t) - y(t)$ , и достаточно инерционными объектами

<sup>1</sup> Под «жестким» будем понимать регулятор, который способен за короткое время  $t$ , меньшее, чем время  $t_{ин}$  переходного процесса в исполнительном механизме и объекте, обеспечивать большое значение  $u(t)$  при  $\varepsilon(t) \neq 0$ .

управления. В этих системах при появлении  $\varepsilon(t) \neq 0$  быстро появляется большое управление  $u(t)$  (рис. 4.1) и большое воздействие  $v(t)$  исполнительного механизма на объект. Но в силу инерционности объекта управления большое  $v(t)$  некоторое время практически не вызывает изменений  $y(t)$  и  $\varepsilon(t) = x(t) - y(t)$ , а значит  $u(t)$  сохраняется дольше, чем следовало бы.

При  $\varepsilon(t) > 0$  это приводит к накоплению системой излишней энергии, при  $\varepsilon(t) < 0$  – к излишней потере энергии и, как результат, к самораскачиванию системы с увеличивающейся амплитудой.

Всё сказанное позволяет сформулировать первое требование к системам управления:

***Система управления должна стремиться войти во все представляющие интерес возможные установившиеся режимы устойчивым образом.***

Или, несколько иначе:

***Система управления должна быть устроена так, чтобы все её представляющие интерес установившиеся режимы были устойчивыми.***

#### 4.2.2 Исследование устойчивости

Из сказанного выше можно сделать следующий вывод:

***Причиной неустойчивого поведения систем управления являются динамические задержки их информационных и управляющих сигналов.***

Исследованием устойчивости поведения динамических систем (в том числе и систем автоматического управления) занимается теория устойчивости движения. Теория устойчивости движения берет свое начало от работ Дж. К. Масквелла (1868 г.), И.А. Вышнеградского (1876 г., 1877 г.), Э. Дж. Рауса (1877 г., 1884 г.) и Н.Е. Жуковского (1882 г.).

В 1892 году в издательстве Харьковского математического общества была опубликована докторская диссертация А.М. Ляпунова «Общая задача об устойчивости движения». Эта работа содержит так много плодотворных идей и результатов, что всю историю теории устойчивости не без основания делят на «доляпуновский» и «послеляпуновский» периоды.

А.М. Ляпунов первым указал на то, что *понятие устойчивости относится к движению, а не к системе* и дал строгое определение устойчивости движения.

В данном пособии не будем давать строгие определения Ляпунова об устойчивости движения, а ограничимся лишь интуитивно понятными определениями, которые вытекают из приведенных в предыдущем параграфе рассуждений.

*Определение 1:*

**Система устойчива, если при ограниченном входном сигнале, выходной сигнал также ограничен.**

Это определение хорошо иллюстрирует рис. 4.4. Оно отображает условие устойчивости системы, когда входной сигнал не равен нулю.

Для *свободной системы*, т.е. системы на которую не действуют входные сигналы, можно дать следующее определение.

*Определение 2:*

**Система устойчива, если после прекращения внешнего воздействия она возвращается в исходное состояние.**

Последнее определение иллюстрирует рис. 4.5.

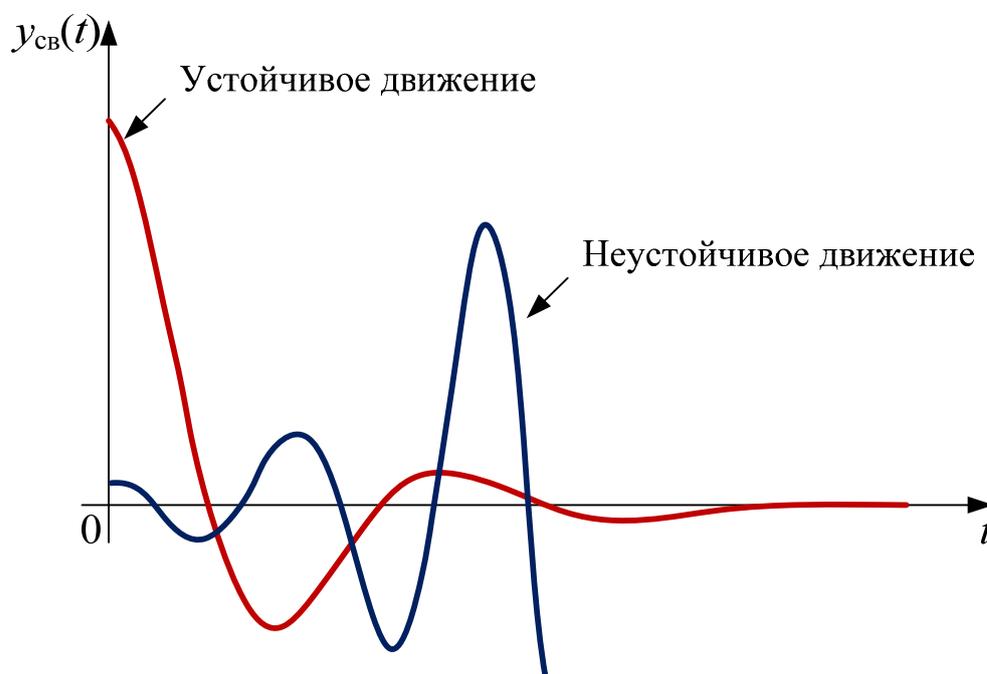


Рис. 4.5. Устойчивое и неустойчивое свободное движение системы

Оба эти определения эквивалентны лишь для линейных стационарных систем. В нелинейных системах такой простой связи

между этими двумя типами устойчивости нет, т.к. свободное движение системы может существенно отличаться от вынужденного.

Как было показано ранее, решение задачи анализа системы управления при помощи ЭВМ заключается в численном интегрировании дифференциальных уравнений, описывающих динамику системы, удовлетворяющих заданным начальным условиям и воздействиям. По результатам этих расчетов сразу можно сделать вывод об устойчивости или неустойчивости системы. Поэтому, моделирование является наиболее универсальным фактором исследования устойчивости систем автоматического управления [11]. Однако исследования устойчивости специальными методами, например, предложенными А.М. Ляпуновым, позволяют более рационально организовать процесс проектирования регуляторов.

Определим, что же в математической модели системы определяет её устойчивость или неустойчивость.

В общем случае динамика линеаризованной системы описывается дифференциальным уравнением  $n$ -го порядка:

$$\begin{aligned} a_n \frac{d^n y(t)}{dt^n} + a_{n-1} \frac{d^{n-1} y(t)}{dt^{n-1}} + \dots + a_1 \frac{dy(t)}{dt} + a_0 y(t) = \\ = b_m \frac{d^m x(t)}{dt^m} + b_{m-1} \frac{d^{m-1} x(t)}{dt^{m-1}} + \dots + b_1 \frac{dx(t)}{dt} + b_0 x(t). \end{aligned} \quad (4.1)$$

При изменении входного сигнала  $x(t)$  выходную величину  $y(t)$  можно записать:

$$y(t) = y_{\text{св}}(t) + y_{\text{вын}}(t), \quad (4.2)$$

где  $y_{\text{св}}(t)$  - общее решение однородного уравнения

$$a_n \frac{d^n y(t)}{dt^n} + a_{n-1} \frac{d^{n-1} y(t)}{dt^{n-1}} + \dots + a_1 \frac{dy(t)}{dt} + a_0 y(t) = 0. \quad (4.3)$$

описывающего процесс изменения выходной величины  $y(t)$  в свободном движении (при отсутствии внешних сил).

$y_{\text{вын}}(t)$  - частное решение дифференциального уравнения (4.1) и описывает тот режим, в который *вынуждает* войти систему входное воздействие  $x(t)$ .

Рассмотрим свободное движение линеаризованной системы, т.е. движение из ненулевых начальных условий при отсутствии внешних сил. Это соответствует случаю, когда на систему в момент  $t = 0$  подать возмущающее импульсное воздействие, и сразу же его снять. В этом случае правая часть уравнения (4.1) равна 0, и поведение системы описывается однородным уравнением (4.3).

Согласно приведенному выше определению 2 система устойчива, если при отсутствии внешних сил выходная величина  $y(t) = y_{св}(t)$  системы с течением времени стремиться к нулю (см. рис. 4.5), т.е.

$$\lim_{t \rightarrow \infty} y_{св}(t) = 0. \quad (4.4)$$

Уравнение (4.3) имеет  $n$  корней  $y_{св}$ , которые, в общем случае, могут быть комплексными, т.е. иметь как вещественную  $p$ , так и мнимую  $q$  составляющие:

$$y_{свk} = p_k \pm jq_k, \quad (4.5)$$

где  $k$  – номер корня уравнения (4.2),  $j = \sqrt{-1}$ .

Комплексные корни всегда сопряжены между собой: если есть корень с положительной мнимой частью, то существует с такой же по модулю, но отрицательной мнимой частью.

Согласно А.М. Ляпунову:

**система будет устойчива, если вещественные части всех корней  $p_k$  уравнения (4.3) будут отрицательными.**

Поскольку уравнение (4.3) характеризует свойства системы, то его называют *характеристическим уравнением*.

Итак, если все корни характеристического уравнения (4.3) имеют отрицательные вещественные части, то соответствующая система будет устойчивой. Наличие мнимых частей в корнях (4.3) характеристического уравнения свидетельствует о том, что переходный процесс будет иметь колебательный характер (рис. 4.6, *г-е*). Если же все корни чисто вещественные, то колебаний выходной величины не будет, т.е. переходный процесс будет *апериодическим* (рис. 4.6, *а-в*).

Если среди всех корней характеристического уравнения (4.3) имеется хотя бы один корень с положительной вещественной частью, то система будет неустойчивой.

Если среди всех корней уравнения (4.3) имеются корни, вещественная часть которых равна нулю, а остальные имеют отрицательную вещественную часть, то такой случай является *критическим*. В этом случае для определения устойчивости необходим дополнительный анализ.

Отдельно можно отметить случай, когда вещественные части всех корней, кроме одного, отрицательны, а *один* корень – имеет нулевую вещественную часть. Такие системы находятся на *границе устойчивости*. При этом если система имеет один нулевой корень, то она находится на *апериодической границе устойчивости* (рис. 4.6, в). Если система имеет пару чисто мнимых корней, то она находится на *колебательной границе устойчивости* (рис. 4.6, е).

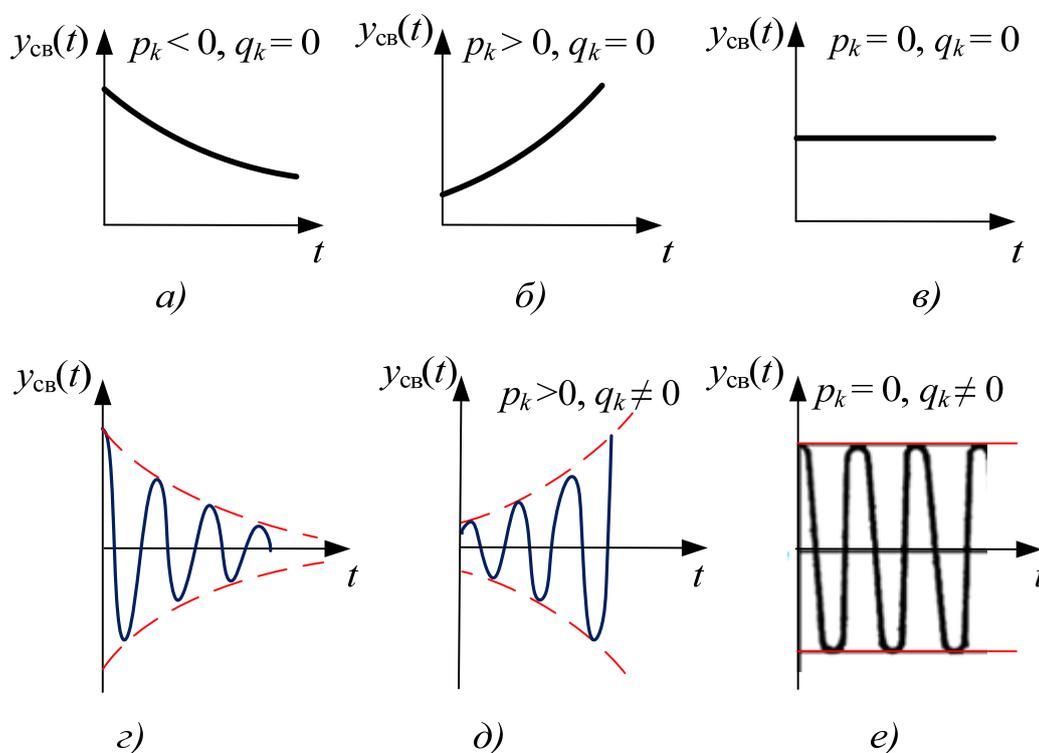


Рис. 4.6. Переходные характеристики при различных знаках  $p_k$  и  $q_k$

Корни характеристического уравнения (4.3) можно разместить на комплексной плоскости. Тогда для устойчивости линейной системы необходимо и достаточно, чтобы все корни характеристического уравнения находились слева от мнимой оси, т.е. мнимая ось является границей устойчивости системы.

Картину расположения корней характеристического уравнения системы можно получить в MATLAB. Рассмотрим в качестве примера двигатель постоянного тока.

1. Откроем файл DCmotor.mdl с моделью двигателя.

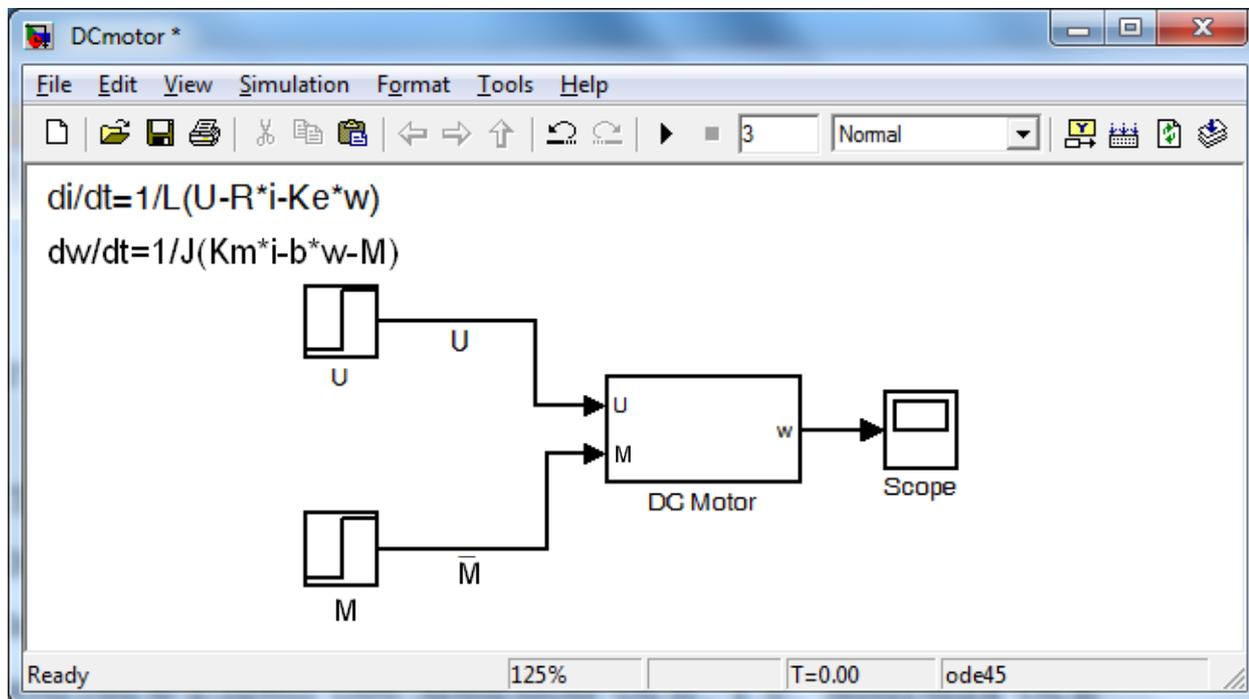


Рис. 4.7. Модель двигателя постоянного тока

В качестве входной величины по-прежнему будем рассматривать напряжение  $U$ , подаваемое на обмотку якоря, а в качестве выходной – скорость вращения  $\omega$  вала двигателя. Зададим соответствующие сигналы в качестве входных и выходных переменных для линеаризации системы.

- Щелкнем правой кнопкой мыши по выходной линии блока ступенчатой функции  $U$ .

- В появившемся контекстном меню выберем **Linearization Points** → **Input Point** (Точки для линеаризации → Входная точка), при этом возле линии связи появится соответствующий значок (кружок со стрелкой, направленной вниз).

- Щелкнем правой кнопкой мыши на выходной линии подсистемы DC Motor.

- В появившемся контекстном меню выберем **Linearization Points** → **Output Point** (Точки для линеаризации → Выходная точка),

при этом возле линии связи появится кружок со стрелкой, направленной вверх (рис. 4.8).

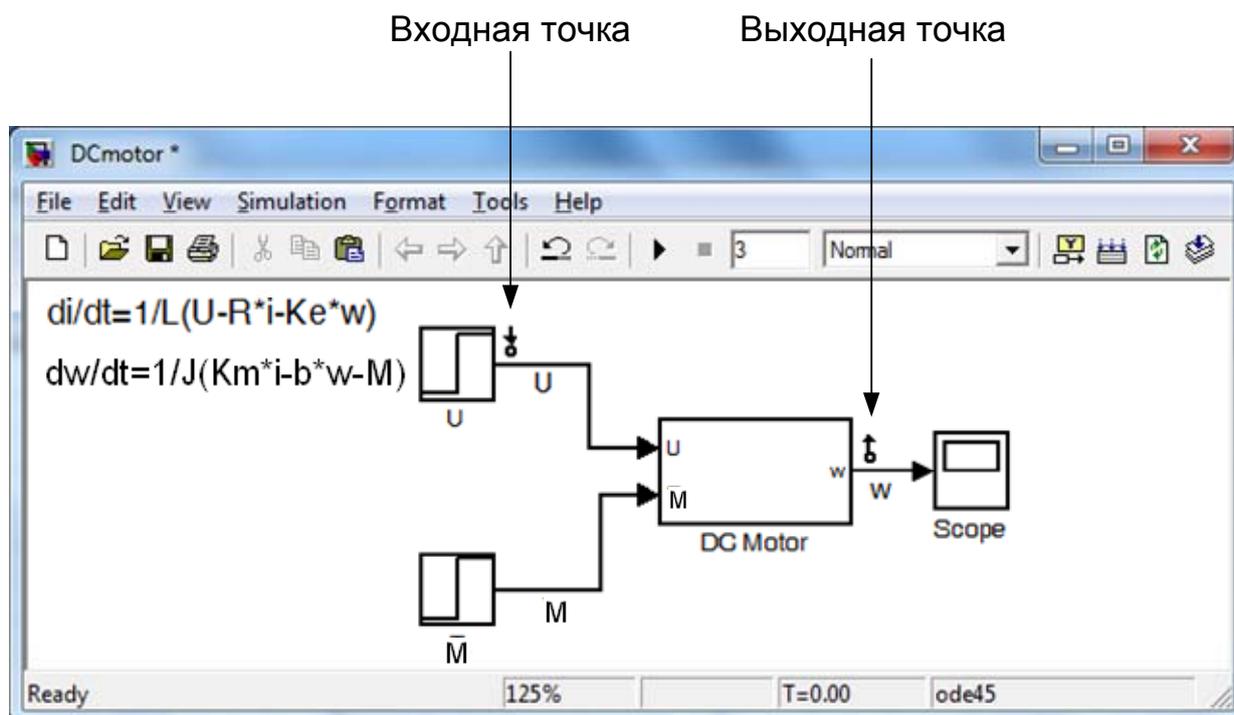


Рис. 4.8. Модель установки входной и выходной точек в модели двигателя постоянного тока

Перейдем теперь в библиотеку Simulink Control Design → Linear Analysis Plots. В этой библиотеке расположены блоки, позволяющие строить различные характеристики, используемые при анализе и проектировании линейных систем управления.

Перетянем в окно нашей модели блок Pole-Zero Plot (рис. 4.9). Этот блок позволяет построить картину расположения корней левой (полюсов) и правой (нулей) частей уравнения (4.1).

Двойным щелчком левой кнопки мыши откроем диалоговое окно параметров блока Pole-Zero Plot (рис. 4.10). Это окно имеет следующие вкладки: линеаризации **Linearizations**, установки ограничений **Bounds**, сохранения результатов **Logging** и сигнализации выхода за рамки ограничений **Assertion**. На вкладке **Linearizations** в таблице **Linearization inputs/outputs** мы можем наблюдать, какие сигналы являются входными, а какие – выходными и, при необходимости, внести изменения.

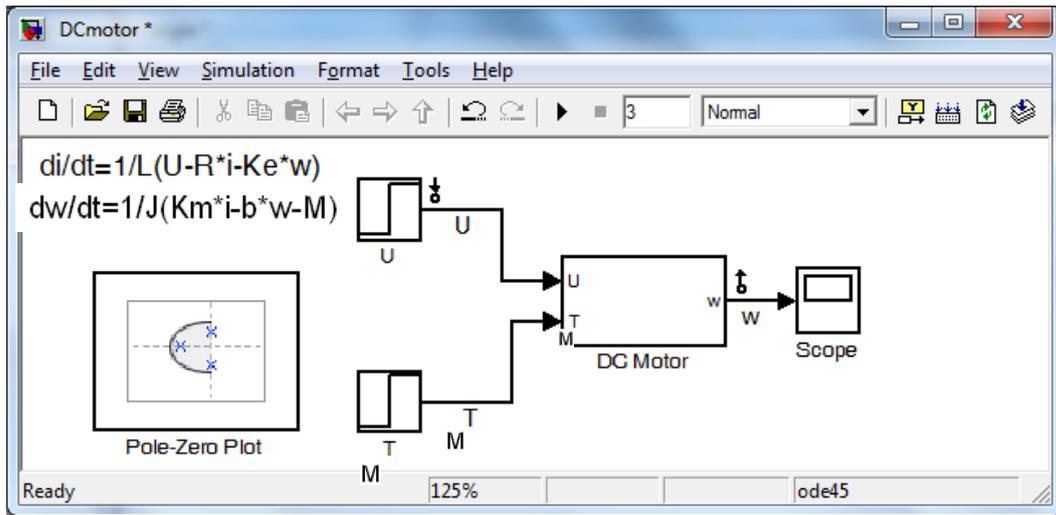


Рис. 4.9. Модель установка входной и выходной точек в модели двигателя постоянного тока

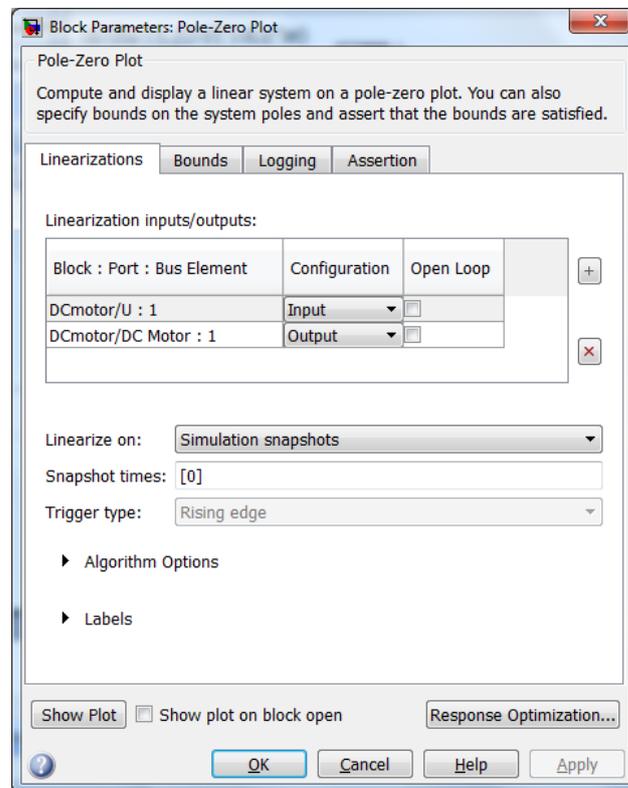


Рис. 4.10. Окно параметров блока Pole-Zero Plot

Нажмем кнопку **Show Plot** в левом нижнем углу окна параметров, чтобы открыть окно с комплексной плоскостью (рис. 4.11).

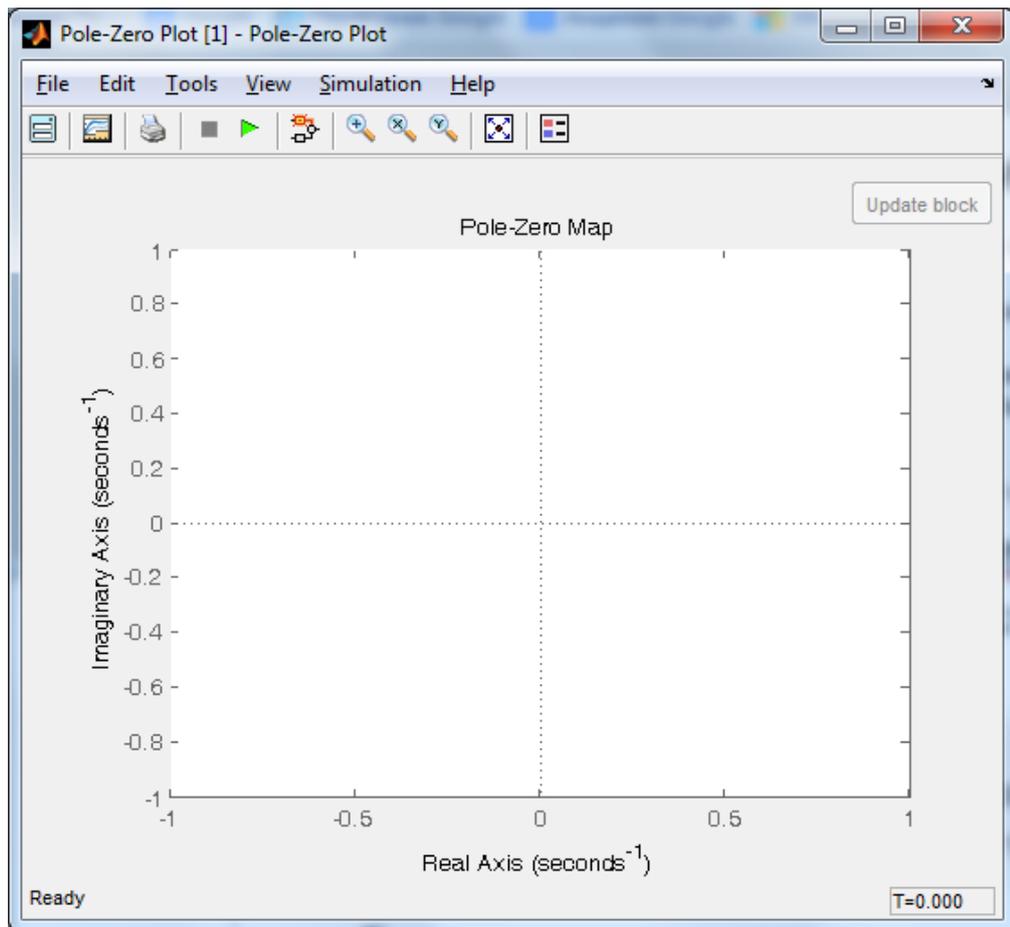


Рис. 4.11. Окно результатов блока Pole-Zero Plot

Для построения картины расположения корней запустим моделирование нажатием кнопки  на панели инструментов графического окна, или выбрав Simulation → Start. Это действие также при необходимости выполняет линейризацию части модели между указанными входной и выходной точками.

В графическом окне появится картина расположения корней характеристического уравнения модели нашего двигателя (рис. 4.12). Корни этого уравнения обозначены крестиками. Как видим, они вещественные и отрицательные (их значения равны -2 и -10), следовательно, переходный процесс будет аperiodическим и устойчивым.

Рассмотрим теперь тот же двигатель постоянного тока, однако в качестве выходной величины будем рассматривать не скорость вращения  $\omega$ , а угол поворота  $\theta$  вала. Для этого внесем изменения в нашу модель.

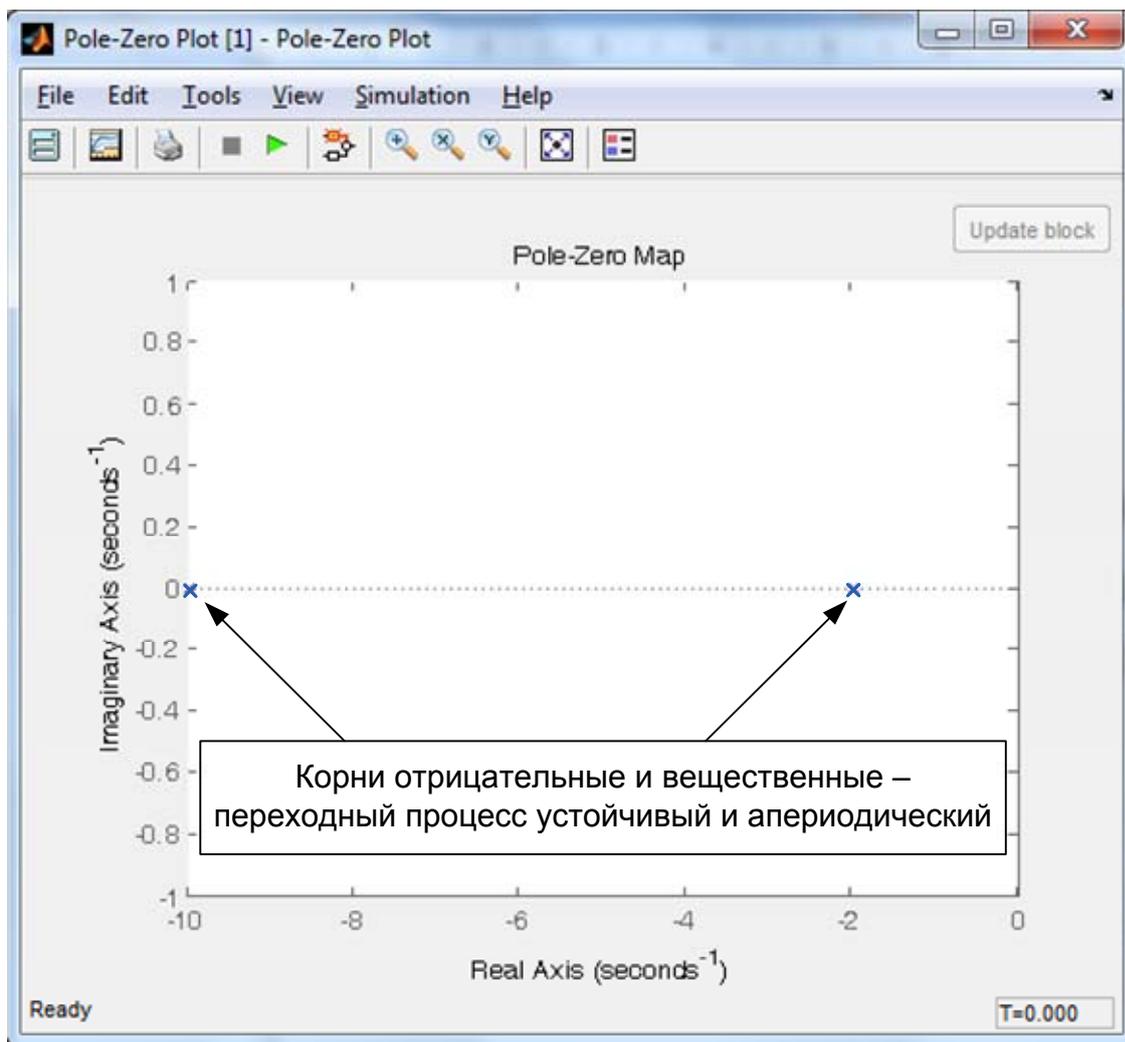


Рис. 4.12. Результаты определения корней характеристического уравнения модели двигателя постоянного тока

- Поскольку  $\omega(t) = d\theta(t)/dt$ , то добавим на выходе модели двигателя блок интегратора Integrator.
- Присвоим имя «theta» выходной линии блока Integrator.
- Щелкнем правой кнопкой мыши по линии «w», выберем **Linearization Points** и снимем флажок напротив позиции **Output Point**.
- Щелкнем правой кнопкой мыши по линии «theta», выберем **Linearization Points** и поставим флажок напротив позиции **Output Point** обозначив, тем самым, переменную theta в качестве выходной переменной при линейризации системы (рис. 4.13).
- Сохраним модель с новым именем, например, DCmotor\_angle.mdl.

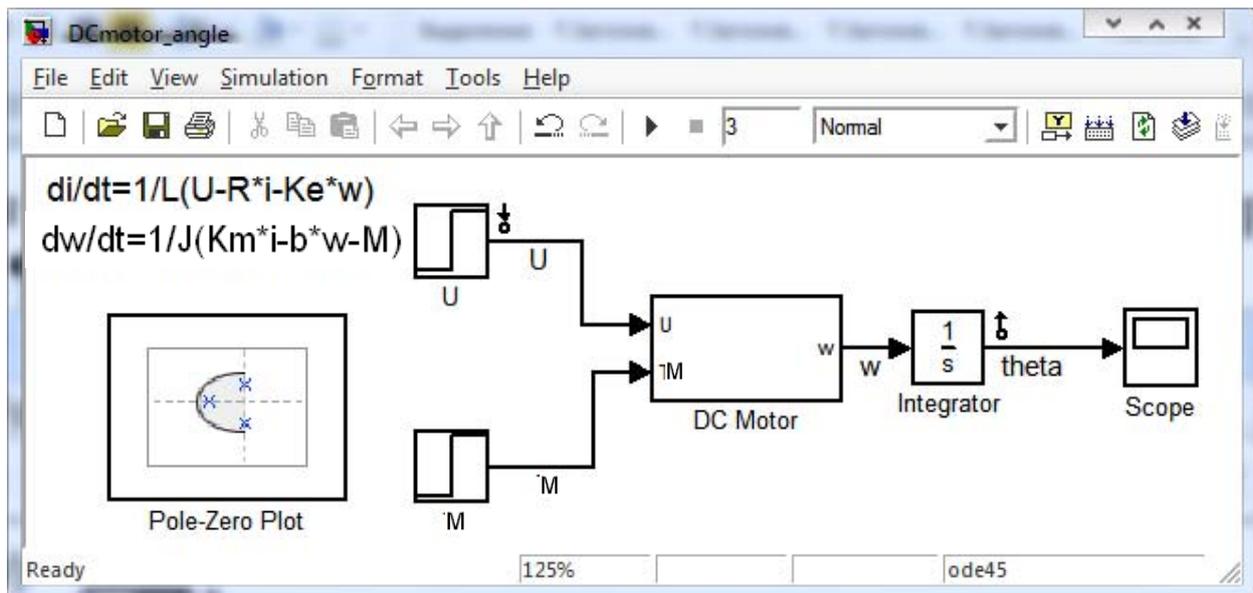


Рис. 4.13. Вид изменённой модели двигателя постоянного тока

- Двойным щелчком левой кнопки мыши откроем диалоговое окно параметров блока Pole-Zero Plot и нажмем кнопку **Show Plot** в левом нижнем углу окна параметров блока.

- В появившемся графическом окне нажмем кнопку  построив, тем самым, картину расположения корней характеристического уравнения (рис. 4.14) двигателя постоянного тока, выходом в котором является угол theta поворота вала.

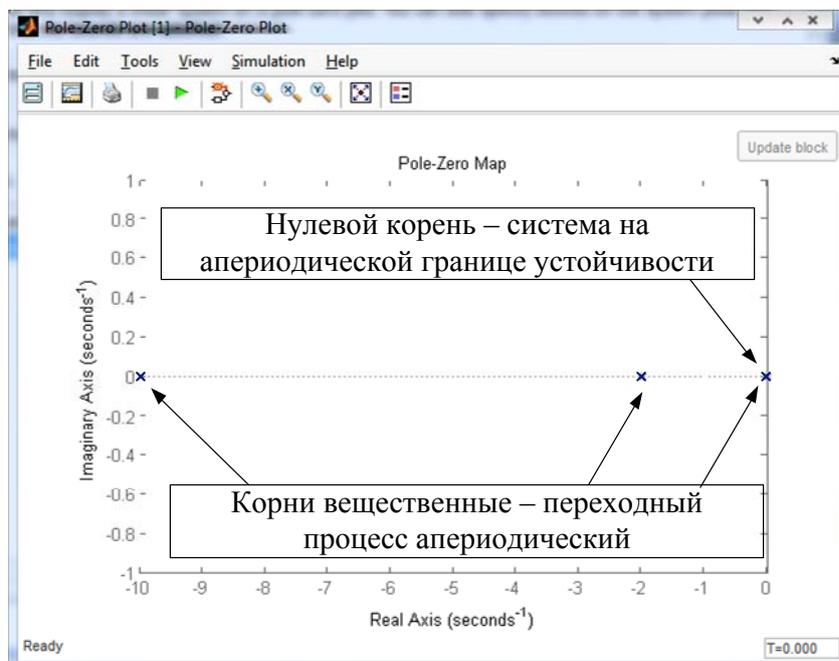


Рис. 4.14. Результаты определения корней характеристического уравнения изменённой модели двигателя постоянного тока

Как видим, уравнение этой системы имеет уже три вещественных корня, два из которых отрицательны, а один – нулевой (рис. 4.14). Следовательно, данная система находится на аperiодической границе устойчивости (рис. 4.15). Правильность характеристики на рис. 4.15 подтверждается простыми логичными рассуждениями: при подаче напряжения на якорную обмотку двигателя, угол поворота его вала будет линейно возрастать с течением времени.

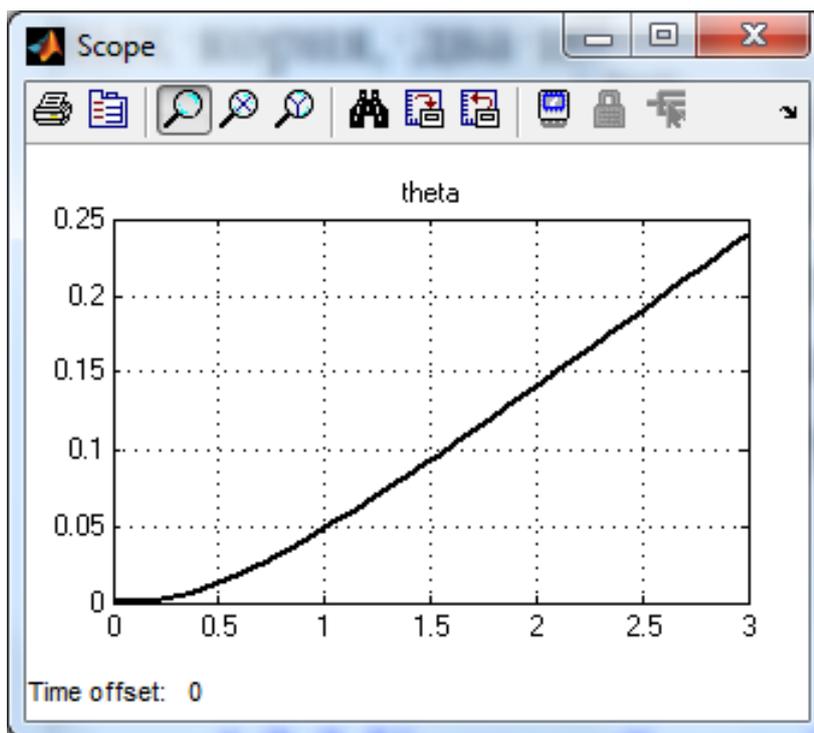


Рис. 4.15. Переходная характеристика двигателя постоянного тока, если выходная величина – угол  $\theta$  поворота вала

Как правило, на практике необходимо обеспечить поворот вала двигателя на заданный угол за определенное время. Этого можно добиться, введя обратную связь по углу поворота (рис. 4.16).

Здесь, чтобы не затенять суть дела, будем считать, что датчик угла является безынерционным элементом с коэффициентом передачи, равным 1. Сохраним модель с новым именем, например, DCmotor\_angle\_feedback.mdl. Далее, введением соответствующего регулятора мы сможем обеспечить выполнение требований, предъявляемых к данной системе.

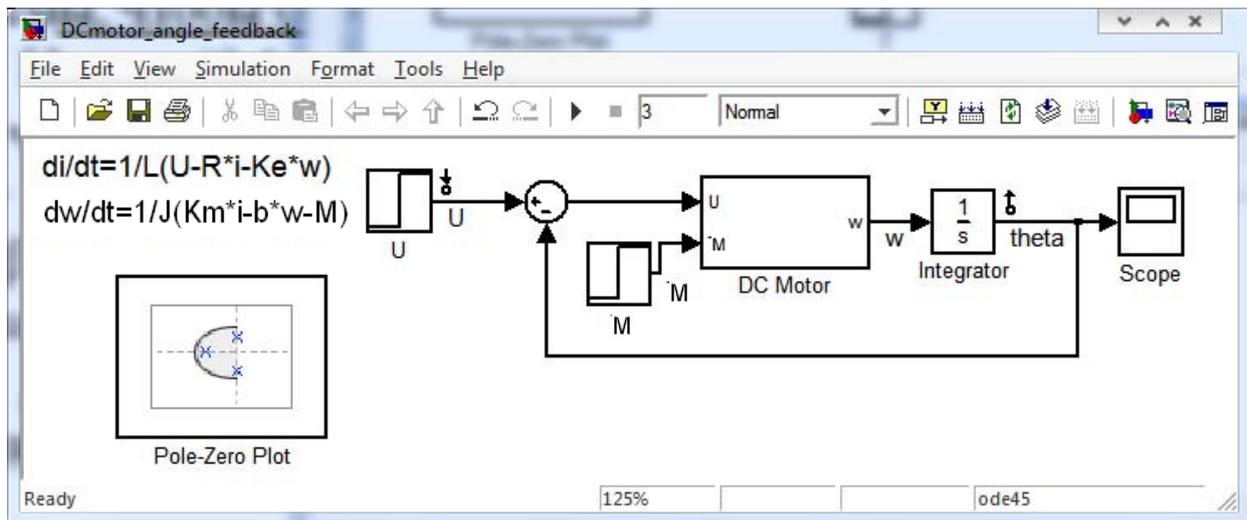


Рис. 4.16. Вид модели двигателя постоянного тока с обратной связью

### 4.2.3 Критерий устойчивости Найквиста

В предыдущем разделе было показано, что для исследования устойчивости линейной системы необходимо решить характеристическое уравнение и проанализировать знаки  $p_k$  вещественных частей его корней. До широкого использования возможностей вычислительной техники, решение характеристического уравнения сложной системы вызывало определенные вычислительные трудности из-за наличия производных высокого порядка. Поэтому были разработаны критерии, которые позволяют, не решая характеристического уравнения, определить, находятся ли все его корни в левой полуплоскости.

Одним из классических критериев устойчивости является частотный критерий Найквиста. Этот критерий был предложен в 1932 г. американским ученым Гарри Найквистом для исследования устойчивости электронных усилителей с отрицательной обратной связью. Достоинство этого критерия заключается в том, что он дает дополнительную полезную информацию о системе, например, о запасах устойчивости и качестве её работы. Рассмотрим кратко суть этого критерия, а затем, попытаемся использовать его при анализе систем управления в MATLAB.

При подаче на вход линейной системы гармонического сигнала определенной частоты  $\omega$ , выходной сигнал будет также гармоническим, будет иметь ту же частоту  $\omega$ , что и входной сигнал,

однако может отличаться от входного амплитудой и фазовым сдвигом. При различных частотах  $\omega$ , отношение  $A(\omega)$  амплитуды выходного сигнала к амплитуде входного сигнала, а также величина фазового сдвига может существенно изменяться в зависимости от свойств системы, например, её инерционности.

Можно построить зависимость изменения  $A(\omega)$  при изменении частоты входного гармонического сигнала от 0 до  $+\infty$ . Такая зависимость называется амплитудной частотной характеристикой (АЧХ) и показывает, во сколько раз амплитуда выходного сигнала отличается от амплитуды входного сигнала с изменением частоты входного гармонического сигнала.

Зависимость  $\varphi(\omega)$  называется фазово-частотной характеристикой (ФЧХ) и показывает, как изменяется фаза выходного сигнала относительно фазы входного сигнала с изменением частоты входного гармонического сигнала.

Найквист предложил рассматривать амплитудно-фазовую частотную характеристику (АФЧХ или диаграмма Найквиста). По сути, диаграмма Найквиста представляет собой АЧХ и ФЧХ, построенные в одной системе координат.

Строится диаграмма Найквиста следующим образом. При фиксированной частоте  $\omega$  входного сигнала отношение амплитуд выходного и входного сигналов можно рассматривать как вектор на комплексной плоскости длиной  $A(\omega)$  и углом к вещественной оси, равным  $\varphi(\omega)$ . С изменением частоты  $\omega$  входного сигнала длина вектора  $A(\omega)$  и угол  $\varphi(\omega)$  меняются. В результате конец вектора  $A(\omega)$  опишет на комплексной плоскости кривую - годограф (от греч. hodos – путь, движение, направление и grapho – пишу). Пример АФЧХ приведен на рис. 4.17.

Вернемся, теперь к критерию устойчивости Найквиста. Он позволяет судить об устойчивости замкнутой системы по диаграмме Найквиста *условно разомкнутой системы*.

Если система в разомкнутом состоянии является неустойчивой, то её характеристическое уравнение содержит  $l$  корней в правой полуплоскости и  $n - l$  корней – в левой и критерий Найквиста имеет следующую формулировку.

**Для устойчивости замкнутой системы необходимо и достаточно, чтобы АФЧХ разомкнутой системы  $l/2$  раз охватывала против часовой стрелки критическую точку  $(-1; j_0)$ ,**

где  $l$  – количество корней характеристического уравнения разомкнутой системы, лежащих в правой полуплоскости.

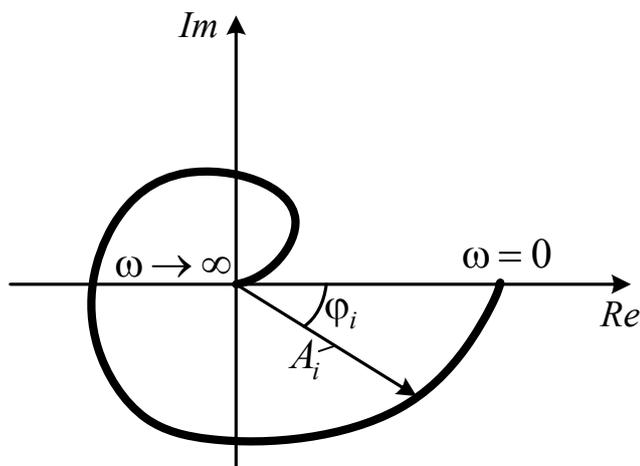
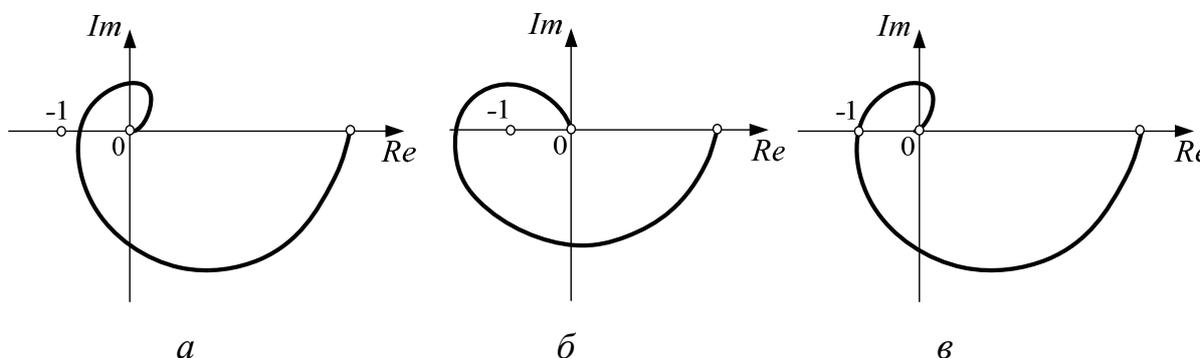


Рис. 4.17. Пример амплитудно-фазовой частотной характеристики

Если разомкнутая система устойчива, то  $l = 0$  и критерий можно сформулировать в следующем виде:

**Если система устойчива в разомкнутом состоянии, то для устойчивости замкнутой системы необходимо и достаточно, чтобы АФЧХ разомкнутой системы не охватывала критическую точку с координатами  $(-1; j_0)$ .**

На рис. 4.18 приведены характеристики системы, устойчивой в замкнутом состоянии. Рис. 4.18, а соответствует устойчивой замкнутой системе, рис. 4.18, б - неустойчивой системе, рис. 4.18, в – системе, находящейся на границе устойчивости.



Рису 4.18. Примеры амплитудно-фазовых частотных характеристик систем  
а) устойчивая система; б) неустойчивая система, в) система на границе устойчивости

Критерий устойчивости Найквиста удобно использовать для определения запасов устойчивости системы. Дело в том, что любая математическая модель лишь приближенно описывает поведение реальной системы. Следовательно, работа с моделью может показать, что система устойчива, а реальная система окажется неустойчивой. Поэтому при проектировании систем стремятся обеспечить их устойчивость с некоторой гарантией, так чтобы изменение параметров в некоторых пределах не могло привести к неустойчивости. Для этой цели используются понятия *запасов устойчивости*.

Исходя из критерия Найквиста, запасы устойчивости показывают, как далеко находится АФЧХ разомкнутой системы от критической точки с координатами  $(-1, j_0)$ . Обычно используется два показателя запаса устойчивости.

*Запас устойчивости по модулю  $a$*  (говорят также запас по амплитуде или по усилению). Он обеспечивает сохранение устойчивости при увеличении коэффициента усиления системы.

Запас устойчивости по модулю определяется как число, на которое должен быть умножен коэффициент усиления разомкнутой системы, чтобы она оказалась на границе устойчивости. Иными словами, запас устойчивости по модулю равен  $a=1/|\beta|$ , где  $\beta$  - координата ближайшей к критической точке места пересечения АФЧХ действительной оси (рис. 4.19). Если коэффициент усиления разомкнутой системы умножить на значение  $a$ , то АФЧХ пройдет через точку  $(-1, j_0)$ , и замкнутая система окажется на границе устойчивости.

Для устойчивых систем значение  $a$  больше единицы, для неустойчивых – меньше единицы.

*Запас устойчивости по фазе  $\gamma$*  определяется наименьшей величиной угла, на который надо повернуть АФЧХ, чтобы она прошла через точку  $(-1, j_0)$ . Он определяется на той частоте  $\omega_c$ , при которой  $A(\omega_c) = 1$ . Эта частота  $\omega_c$  называется *частотой среза*. Частоту среза можно найти, если определить точку пересечения АФЧХ и окружности единичного радиуса с центром в начале координат (рис. 4.19).

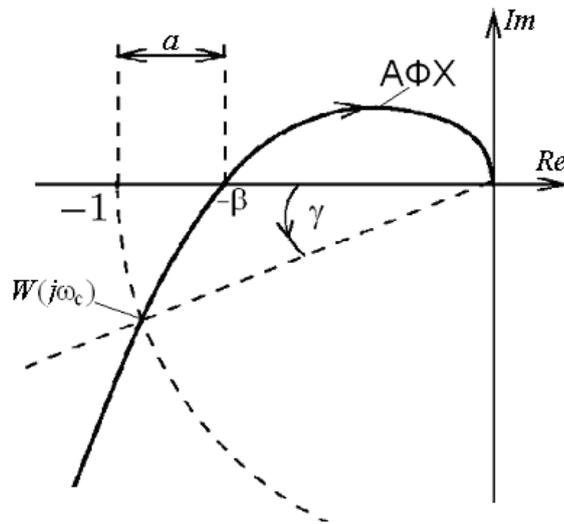


Рис. 4.19. Определение запасов устойчивости по АФЧХ [5]

Для неустойчивой системы запас по фазе считается отрицательным.

Встречается еще *запас по задержке* - это минимальная задержка по времени, при добавлении которой в контур система теряет устойчивость.

Рассмотрим, как определять запасы устойчивости замкнутой системы средствами Simulink. В качестве примера рассмотрим замкнутую систему управления углом поворота вала двигателя постоянного тока (рис. 4.20).

- Перейдем в библиотеку Simulink Control Design → Linear Analysis Plots и перенесем в окно модели блок Gain and Phase Margin Plot (рис. 4.21).

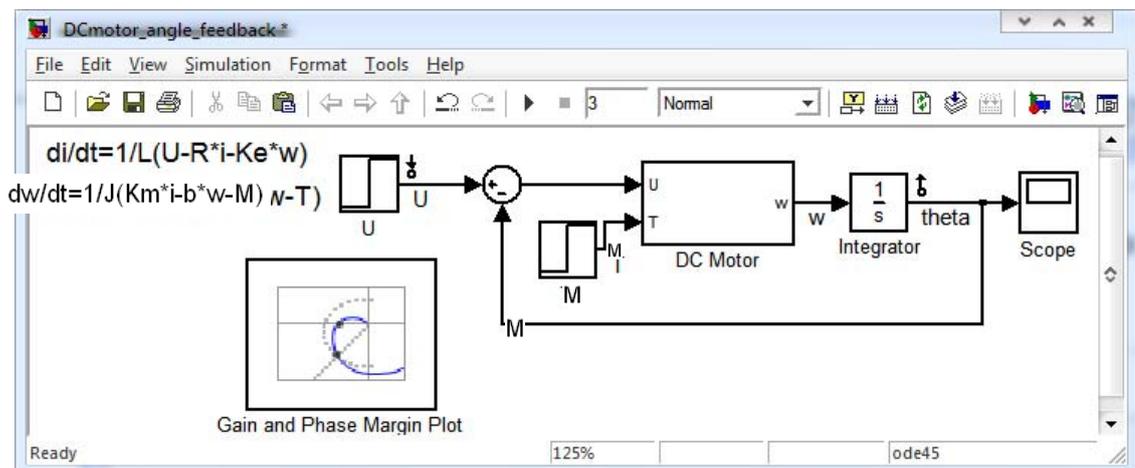


Рис. 4.20. Модель замкнутой системы управления углом поворота вала двигателя с блоком Gain and Phase Margin Plot

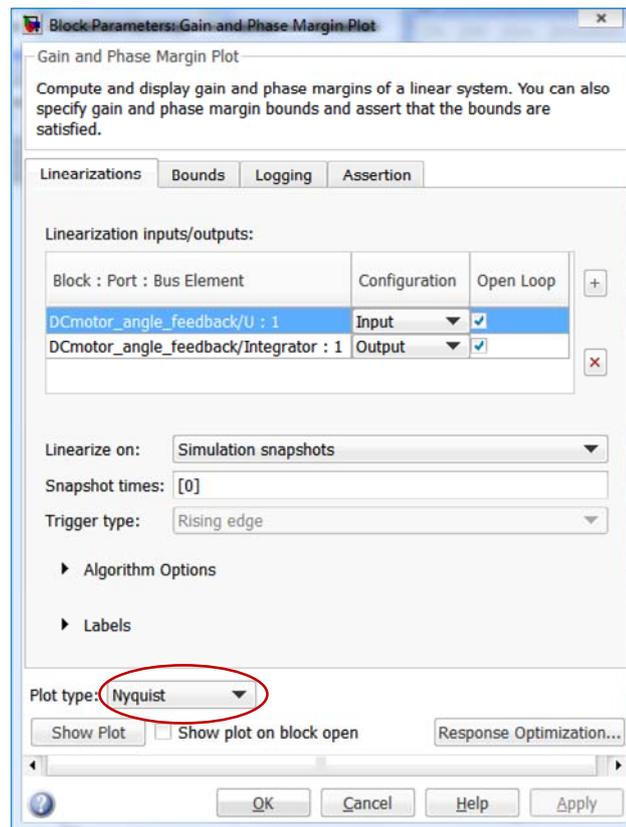


Рис. 4.21. Окно параметров блока Gain and Phase Margin Plot с выбранной диаграммой Найквиста

- Дважды щелкнем по блоку Gain and Phase Margin Plot для вызова окна его параметров. Это окно (рис. 4.21) подобно окну параметров блока Pole-Zero Plot (рис. 4.10), однако, имеет еще одно окно в левом нижнем углу – окно Plot Type (тип графика).

- Выберем в этом окне из списка позицию Nyquist, чтобы построить диаграмму Найквиста и нажмем кнопку Show Plot (рис. 4.22).

- В появившемся графическом окне нажмем кнопку . В результате на экране появится окно с диаграммой Найквиста разомкнутой системы (рис. 4.22). Стрелки на линиях указывают на направление перемещения конца вектора  $A(\omega)$  при изменении частоты входного сигнала, а точка  $(-1, j_0)$  отмечена красным знаком «+».

Как видим, диаграмма Найквиста не охватывает критическую точку  $(-1, j_0)$  и расположена от нее достаточно далеко. Следовательно, замкнутая система должна обладать большими запасами устойчивости. Проверим это.

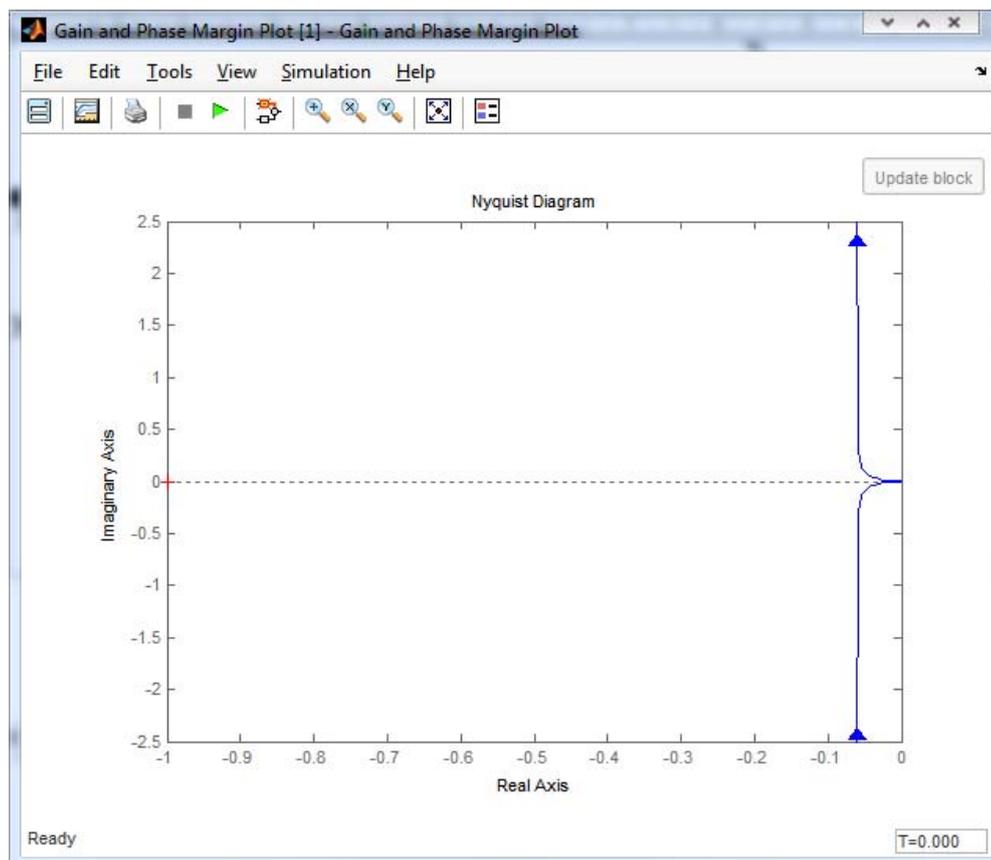


Рис. 4.22. Диаграмма Найквиста разомкнутой системы управления углом поворота вала двигателя

- Щелкнем в свободной части графического окна правой кнопкой мыши, чтобы вызвать контекстное меню.

- В появившемся меню выберем **Characteristics** → **Minimum Stability Margins** (минимальные запасы устойчивости), как показано на рис. 4.23.

В результате на графике появятся точки, по которым можно определить запасы устойчивости. Для вывода на экран численных значений этих запасов, достаточно щелкнуть по соответствующим точкам левой кнопкой мыши. При этом появятся дополнительные информационные окна (рис. 4.24). Эти окна можно при помощи мыши перемещать в более удобное место.

Из рис. 4.24 видно, что запас устойчивости по модулю (Gain Marge) равен 41,6 Дб при частоте 4,47 рад/с; запас устойчивости по фазе (Phase Margin) 86,6° и запас по задержке (Delay Margin) 15,1 с. при частоте 0,0998 рад/с. Замкнутая система является устойчивой (Closed loop stable! Yes).

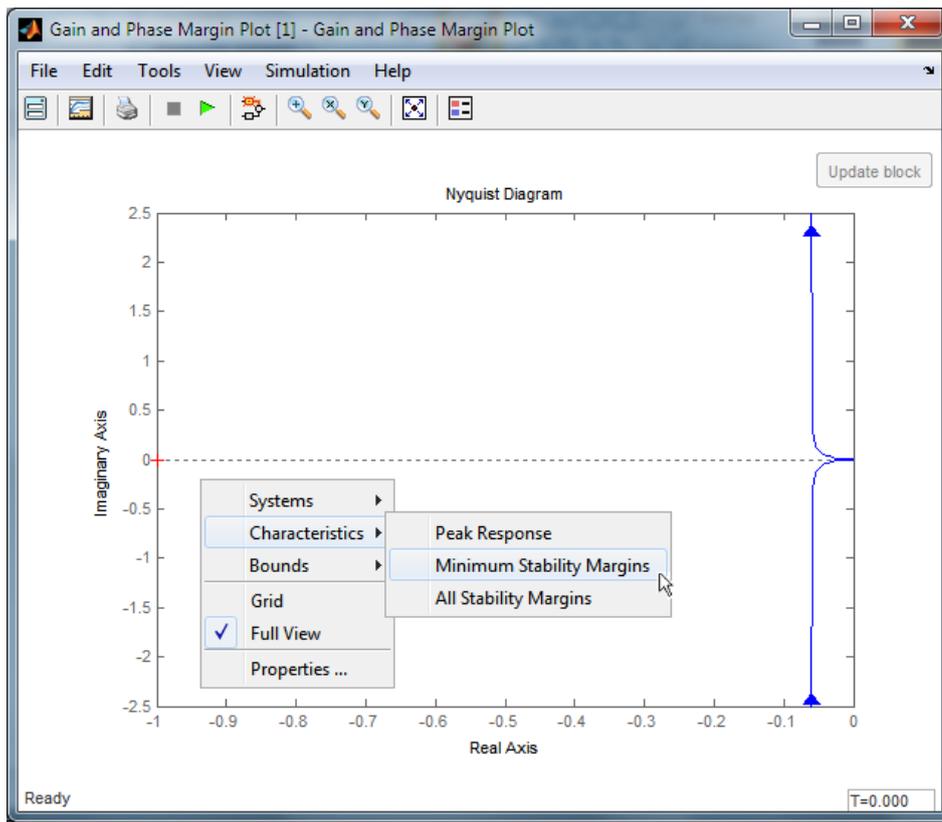


Рис. 4.23. Отображение минимальных запасов устойчивости

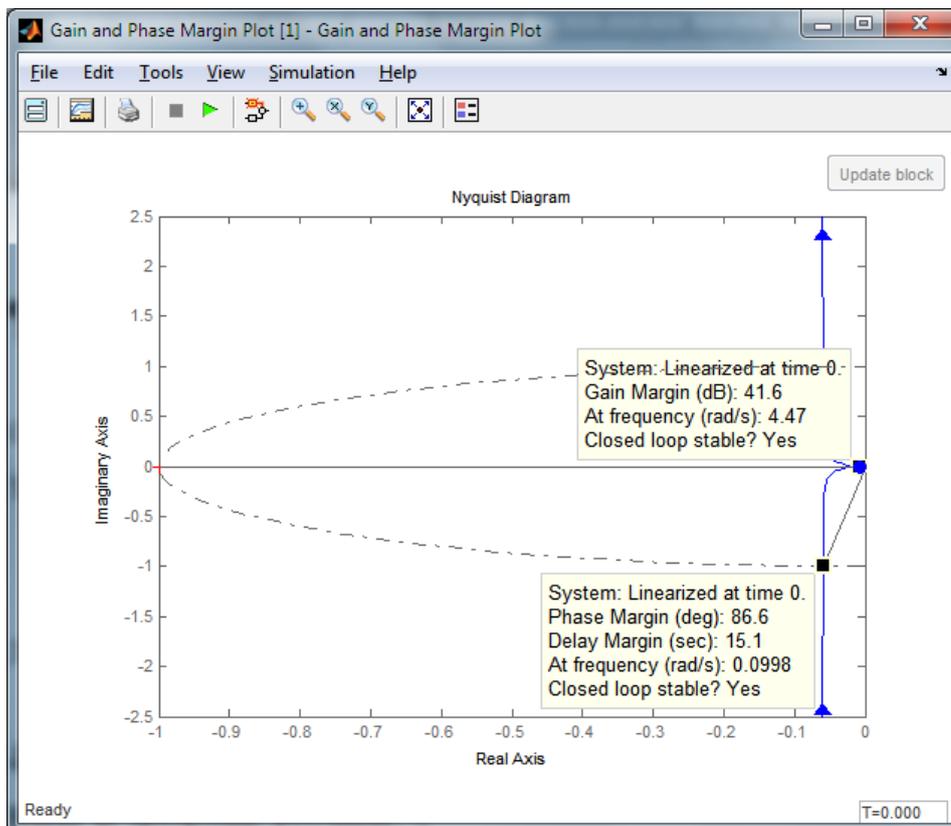


Рис. 4.24. Отображение численных значений минимальных запасов устойчивости

Слишком большие значения запасов устойчивости свидетельствуют о том, что использованы не все возможности системы и она имеет резерв по улучшению качества работы в переходных режимах. Однако запас устойчивости в следящей системе, в которой задающее воздействие меняется непрерывно или довольно часто должен быть значительно больше, чем в системах стабилизации, в которых задающее воздействие меняется редко.

### 4.3 Точность системы в установившемся режиме

Возможное установившееся движение  $y_{уст}(t)$  системы, соответствующее рассматриваемым  $x(t)$ ,  $f(t)$  и  $\eta(t)$ , должно отличаться от  $x(t)$  при всех  $t > t_p$  (где  $t_p$  – время практического окончания переходного процесса) не более, чем на заданную величину  $\varepsilon_{пред}$ . В большинстве практических случаев  $\varepsilon_{пред} = 0$ , то есть предъявляется требование:  $y_{уст}(t) = x(t)$ .

На рис. 4.25 для примера показаны возможные установившиеся режимы  $y_{1уст}(t)$  и  $y_{2уст}(t)$  в двух системах управления с характерными задающими воздействиями  $x(t) = A \cdot 1(t)$ ,  $A = \text{const}$ ,  $f(t) = 0$ ,  $\eta(t) = 0$ .

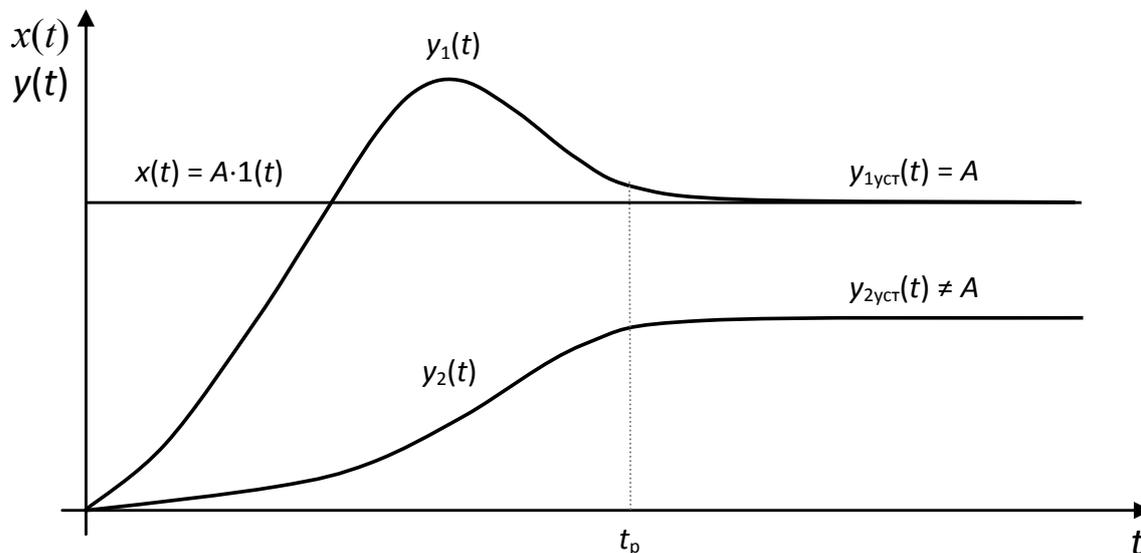


Рис. 4.25. Возможные установившиеся режимы в двух системах управления

Возможный установившийся режим  $y_{1уст}(t)$  первой системы характеризуется отклонением (ошибкой)  $\varepsilon_{1уст}$ , равной нулю. Во второй системе  $\varepsilon_{2уст} \neq 0$ .

#### 4.4 Требования к качествам переходных процессов

На начальном этапе развития теории автоматического управления качество системы регулирования совершенно естественно связывалось с качеством поведения её выходной величины  $y(t)$  при движении из состояния покоя под влиянием достаточно простых в описании, но тяжёлых для системы воздействий  $x(t)$  или  $f(t)$  типа  $A \cdot 1(t)$ . Так на рис. 4.26 показаны переходные функции (т.е. реакции на  $x(t) = 1(t)$ ) двух систем управления с одинаковыми исполнительными механизмами и объектами, но разными управляющими устройствами. И одна и другая системы обеспечивают устойчивый выход  $u_{уст} = x$ . С этой точки зрения они эквивалентны. Но переходная функция  $h_1(t)$  явно предпочтительней переходной функции  $h_2(t)$ : она плавнее и заканчивается быстрее. Показатели качества переходных процессов можно разбить, таким образом, на две группы: показатели быстродействия и показатели колебательности.

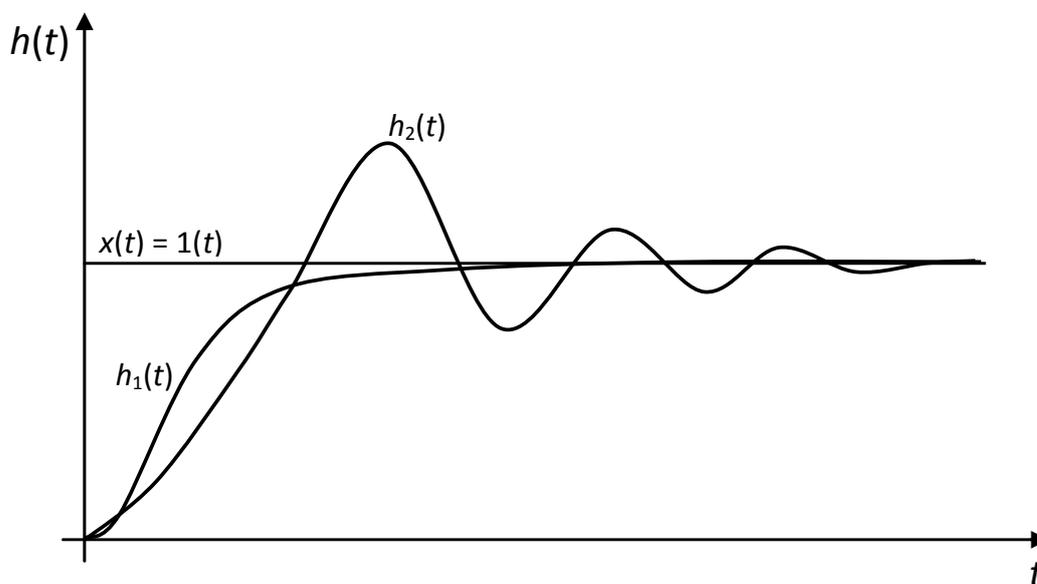


Рис. 4.26. Переходные функции двух систем управления

Показатели быстродействия системы следующие (рис. 4.27).

- Время регулирования (установления)  $t_p$  — время, за которое выходная переменная достигнет установившегося значения с точностью  $\Delta = (2-3)\%$  от установившегося значения  $h_{уст}$  (в MATLAB по умолчанию  $\Delta = 2\%$ ).

- Время нарастания  $t_H$  - промежуток времени, за который выходная величина изменяется от 10% до обычно 90% от  $h_{уст}$ .
- Время  $t_{max}$  наступления первого максимума.

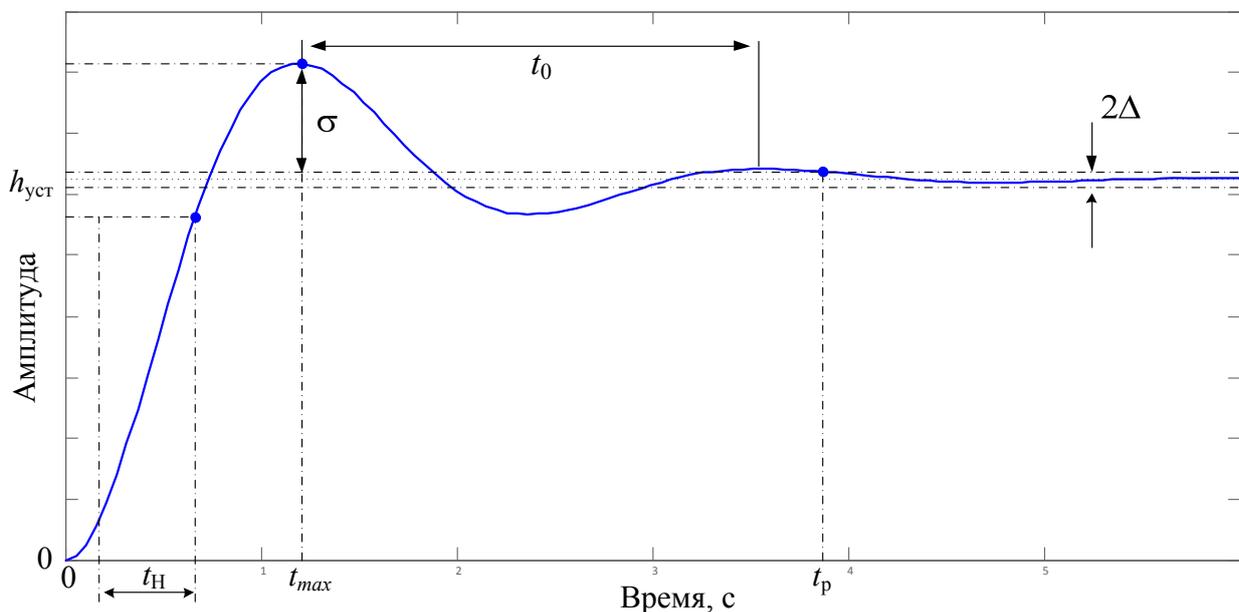


Рис. 4.27. Показатели качества системы управления

Колебательность системы характеризуют следующие показатели (рис. 4.27).

- Максимальное перерегулирование  $\sigma$  - наибольшее отклонение переходной функции от  $h_{уст}$ , обычно выражается в процентах:

$$\sigma = \frac{h_{max} - h_{уст}}{h_{уст}} \cdot 100\% . \quad (4.6)$$

Максимальное перерегулирование характеризует степень удаления системы от колебательной границы устойчивости, т.е. запас устойчивости. Поэтому большое перерегулирование нежелательно. Теоретически считается, что  $\sigma$  не должно превышать 20%, однако часто на практике необходимо добиться перерегулирования 2÷5%, а иногда перерегулирование вообще недопустимо. Тем не менее, в некоторых случаях максимальное перерегулирование может достигать 70%.

- Число колебаний  $N$  выходной величины в течение времени переходного процесса. Обычно  $N$  не должно превышать 2-3.

- Период  $t_0$  и частота  $\omega_0$  колебаний

$$\omega_0 = \frac{2\pi}{t_0}. \quad (4.7)$$

- Коэффициент затухания  $\xi$ , характеризующий быстроту затухания колебаний переходного процесса и равный отношению модулей двух смежных перерегулирований:

$$\xi = \frac{|h_{\max_i} - h_{\text{уст}}|}{|h_{\max_{i+1}} - h_{\text{уст}}|}, \quad (4.8)$$

где  $h_{\max_i}$  и  $h_{\max_{i+1}}$  – расположенные рядом экстремумы кривой переходной функции.

На практике чаще применяется логарифмический декремент затухания:

$$d_c = \ln \frac{h_{\max_i}}{h_{\max_{i+1}}}. \quad (4.9)$$

Чем больше логарифмический декремент затухания, тем быстрее происходит затухание переходного процесса.

Существуют и другие способы введения формализованных оценок качества переходных процессов.

Итак, третье требование к системам автоматического управления формулируется следующим образом.

**Система управления должна быть устроена так, чтобы представляющие интерес переходные процессы характеризовались минимальными значениями используемых показателей качества.**

Рассмотрим теперь, как определить показатель качества при помощи MATLAB.

Конечно, численные значения показателей качества системы можно определить приблизительно, «на глаз», подав на вход единичный ступенчатый сигнал при помощи блока Step. Однако можно использовать специальный блок - Linear Step Response Plot из библиотеки Simulink Control Design → Linear Analysis Plots. Использование этого блока тем более удобно, что

он выполняет линеаризацию нелинейной модели между указанными входными и выходными точками, и строить реакцию на ступенчатый сигнал уже линеаризованной системы. Это может быть очень полезно при проектировании регуляторов.

Рассмотрим модель двигателя постоянного тока с обратной связью, в которой выходной величиной является угол поворота вала двигателя (файл DCmotor\_angle\_feedback.mdl).

- Перейдем в библиотеку Simulink Control Design → Linear Analysis Plots и переместим блок Linear Step Response Plot в окно нашей модели (рис. 4.28).

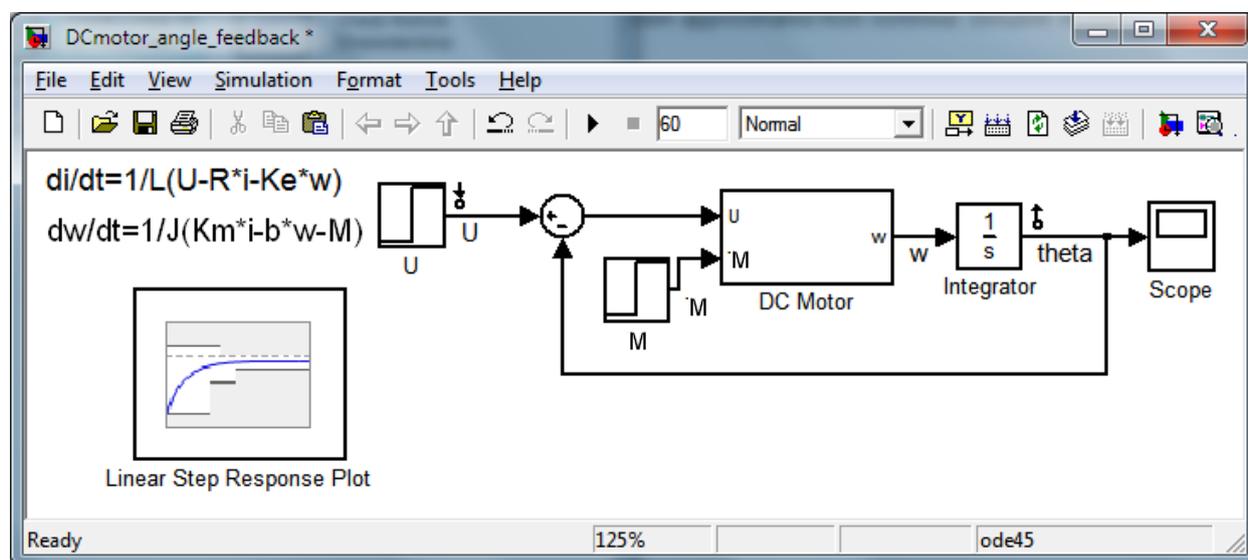


Рис. 4.26. Модель замкнутой системы управления углом поворота вала двигателя постоянного тока с блоком Linear Analysis Plots

- Дважды щелкнем левой кнопкой мыши на блоке Linear Step Response Plot для вызова окна параметров блока (рис. 4.27) Block Parameters. Это окно, как и для блоков Pole-Zero Plot и Gain and Phase Margin Plot имеет следующие панели: линеаризации **Linearizations**, установки ограничений **Bounds**, сохранения результатов **Logging** и сигнализации выхода за рамки ограничений **Assertion**.

- Если в окнах Linearization inputs/outputs заданы правильные входные и выходные переменные (в нашем случае выходы блоков U и Integrator), то параметры блока можно оставить без изменений.

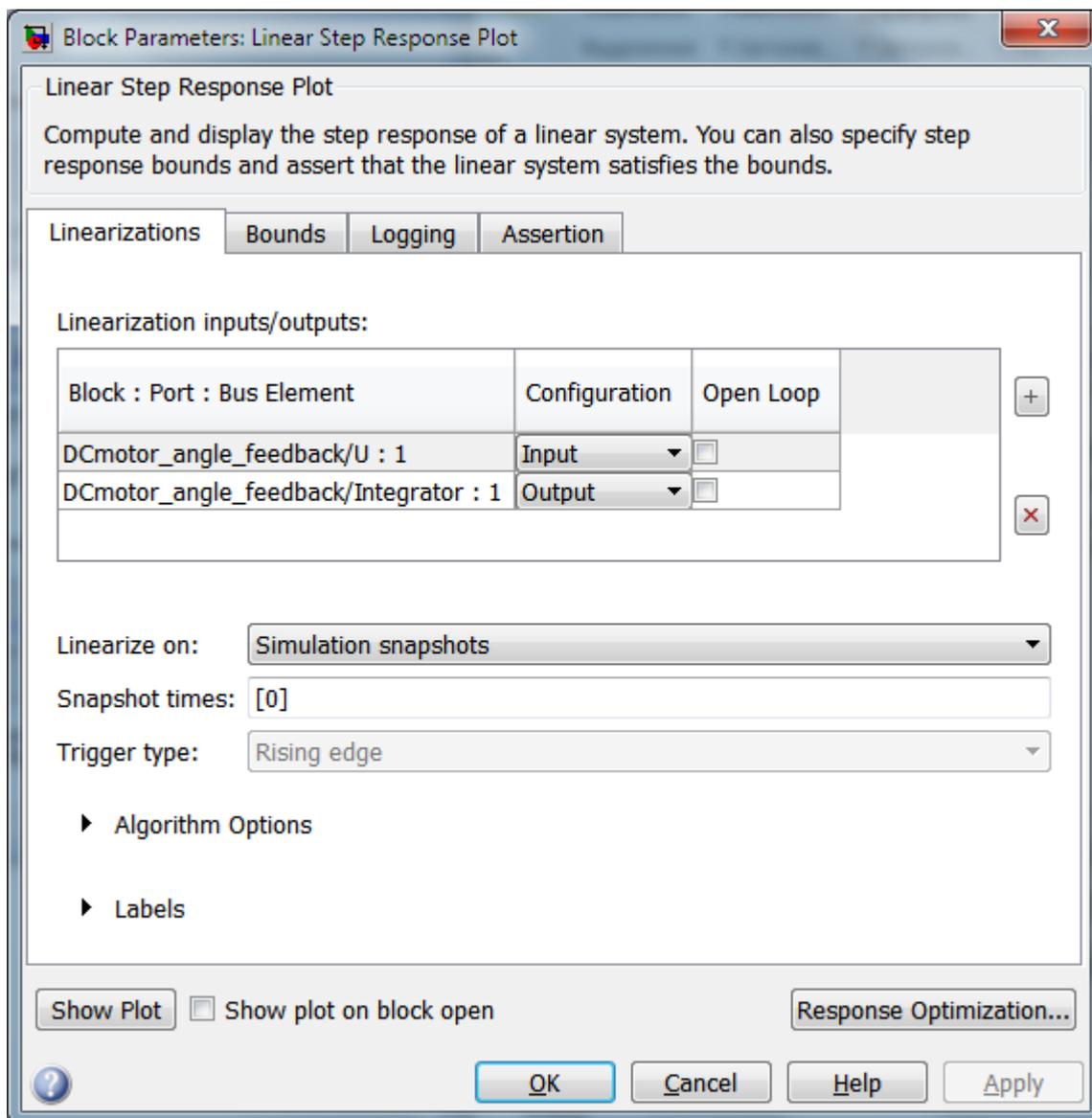


Рис. 4.27. Окно параметров блока Linear Analysis Plots

- В противном случае нажатием кнопки  справа от таблицы Linearization inputs/outputs откроем дополнительное окошко Model signal.
- Щелчком левой кнопкой мыши в окне модели на линию, которая является входом в систему.
- Нажмем кнопку  чтобы передать сведения в таблицу.
- Аналогичным образом задать точку выхода и в колонке Configuration выбираем позицию Output (рис. 4.28). Кнопка  удаляет ошибочно выбранную точку.

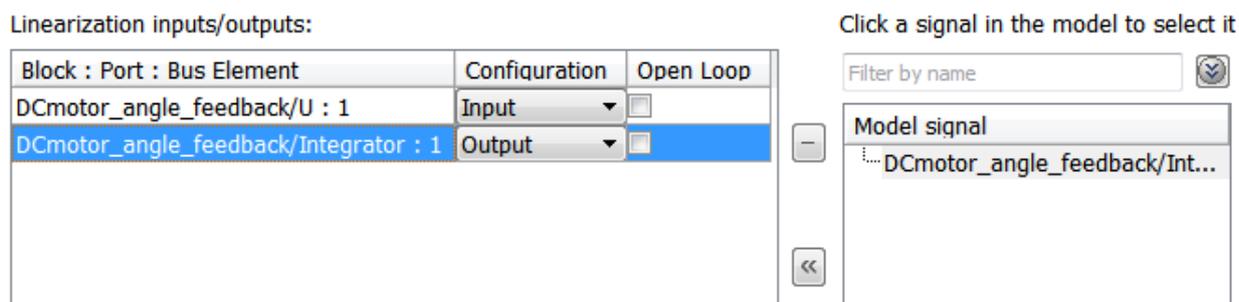


Рис. 4.28. Выбор точки выхода блока Linear Analysis Plots

- После изменений обязательно подтвердим их нажатием кнопки ОК или Apply.

- Нажимаем кнопку Show plot для вывода на экран графического окна Linear Step Response Plot (опция Show plot on block open обеспечивает вызов окна Linear Step Response Plot вместо окна Block Parameters при щелчке на изображении блока).

- В окне Linear Step Response Plot нажмем на кнопку  для построения графика переходной функции (рис. 4.29).

Вызвав контекстное меню и поставив в подменю **Characteristics** флажки напротив соответствующих показателей качества, можно получить наглядное представление о качестве системы. Если подвести курсор мыши к какой-либо характерной точке, то на экране отображается числовое значение соответствующего показателя качества (рис. 4.29). Полученное окно можно зафиксировать щелчком мыши и, затем перемещать в произвольном направлении или редактировать при помощи контекстного меню.

Проанализировав рис. 4.29 можно сделать вывод, что вал двигателя поворачивается на требуемый угол непозволительно долго (время регулирования  $t_p = 37,3$  с), поэтому необходимо улучшение работы этой системы, в частности, повышать её быстродействие.

Рассмотрим, теперь, систему управления скоростью вращения валом двигателя. Перенесем в окно её модели блок Linear Step Response Plot из библиотеки Simulink Control Design → Linear Analysis Plot (рис. 4.30).

Выполнив приведенные выше операции, получим график переходной функции данной системы (рис. 4.31) со значениями показателей качества.

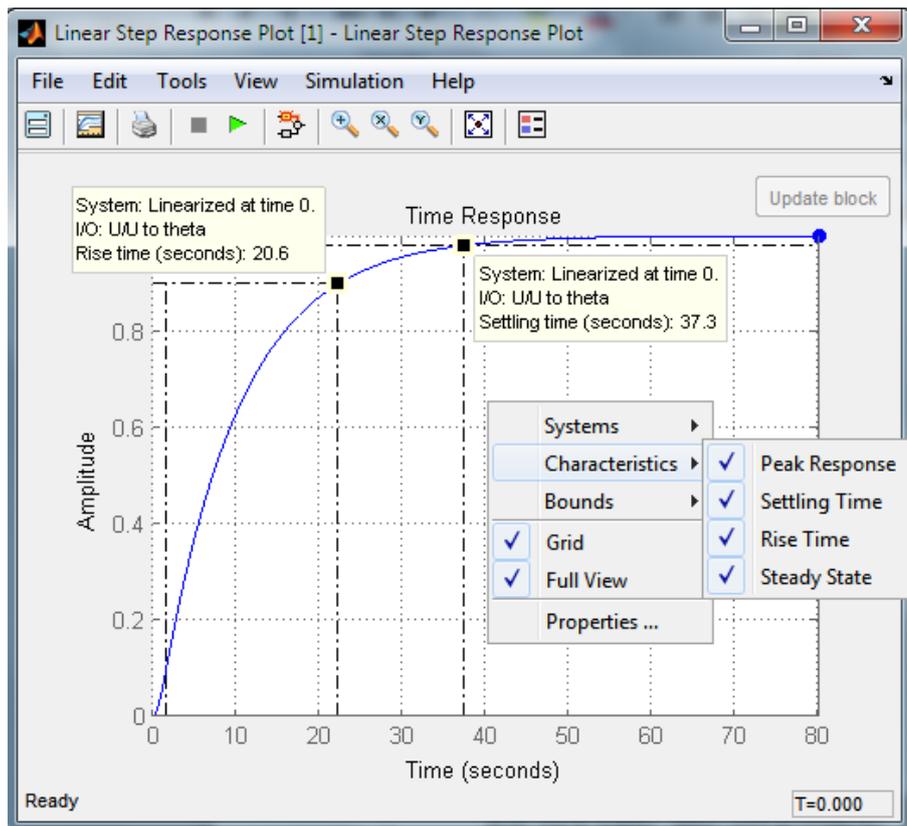


Рис. 4.29. Определение показателей качества по переходной функции

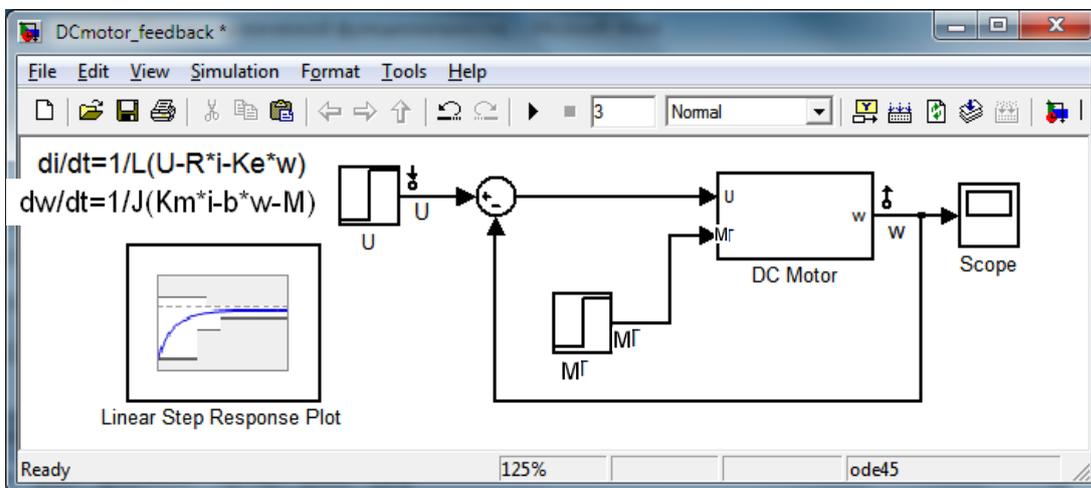


Рис. 4.30. Модель управления угловой скоростью вращения вала двигателя постоянного тока с блоком Linear Analysis Plots

Быстродействие этой системы удовлетворительное ( $t_p = 1,85$  с), однако, установившаяся ошибка составляет 9,08% (Final value: 0.0908). Следовательно, и система управления скоростью вращения вала двигателя нуждается в улучшении.

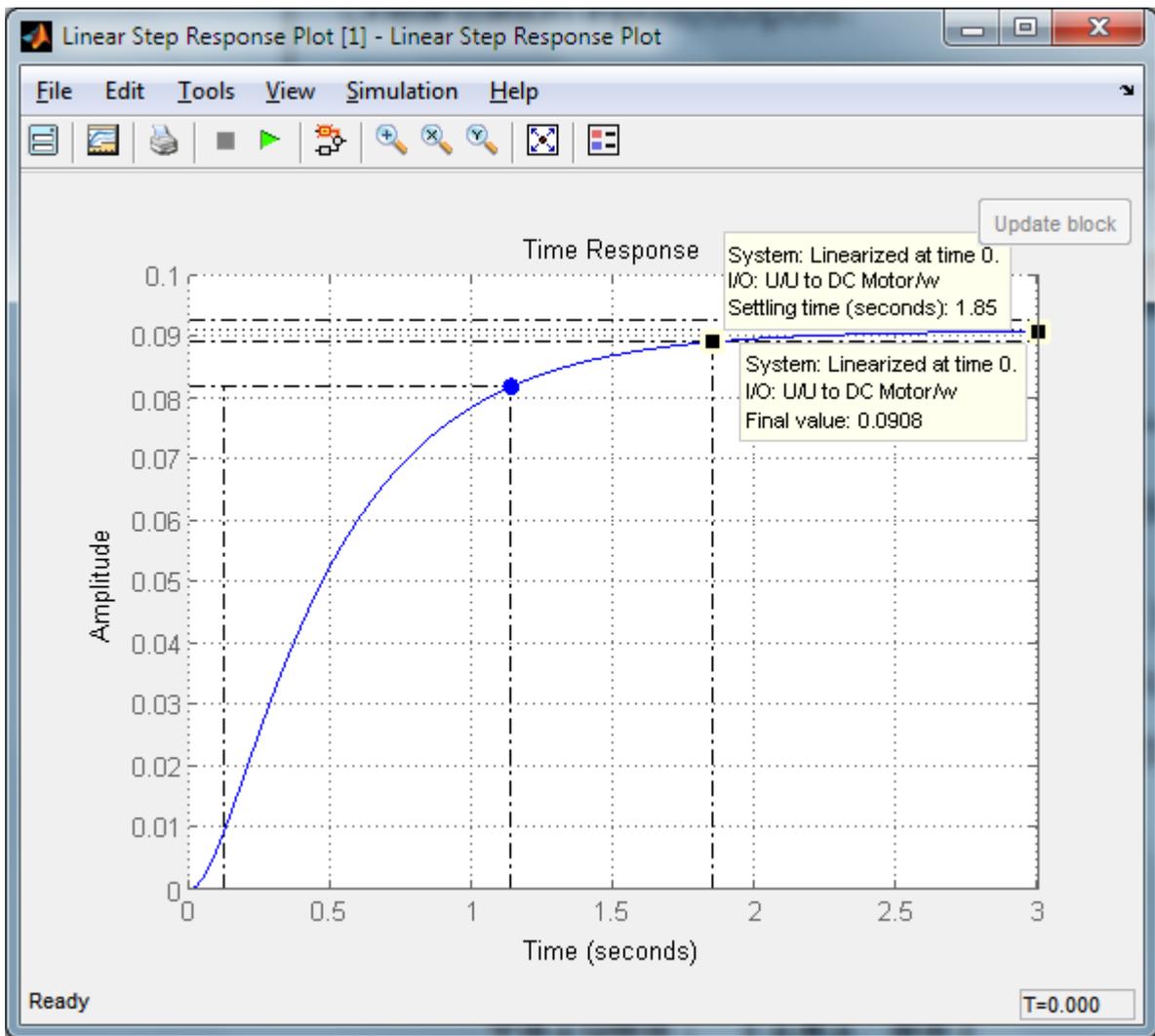


Рис. 4.31. Результаты определение показателей качества по переходной функции

Основные способы улучшения качества работы систем управления будут рассмотрены в следующей главе.

## **5. СИНТЕЗ ЛИНЕЙНЫХ СИСТЕМ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ**

### **5.1. Постановка задачи синтеза**

Как уже упоминалось, в теории автоматического управления выделяется две основные задачи:

1) задача анализа, которая заключается в определении устойчивости и оценке качества работы заданной системы;

2) задача разработки наиболее простой структуры системы автоматического управления и расчета ее параметров, обеспечивающих заданные показатели качества и точности – это задача синтеза.

Вторая задача сложнее в виду своей неоднозначности, многое определяется творческими способностями проектировщика. Например, требования к обеспечению точности системы и запасов её устойчивости являются противоречивыми.

В общем случае при синтезе системы управления необходимо определять как алгоритмическую, так и функциональную структуру системы [12]. Алгоритмическую структуру находят с помощью математических методов и на основе требований, записанных в строгой математической форме. Определение функциональной структуры системы заключается в выборе её конкретных элементов с учетом их физической природы, типов, мощностей входных и выходных сигналов смежных элементов. Эта процедура синтеза пока не имеет строгой математической основы и относится к области инженерного искусства [12].

В данном пособии мы будем рассматривать лишь вопросы синтеза алгоритмической структуры системы управления.

### **5.2. Основные этапы решения задачи синтеза**

Укрупненно процедура синтеза системы управления должна состоять из следующих этапов [12, 13].

1-й этап. Постановка задачи. Здесь определяется назначение системы, выходные переменные, которые подлежат управлению и требования к ним. Например, назначением системы управления

подвеской автобуса является обеспечение комфортной езды по неровностям дороги, выходной переменной является расстояние  $x_1 - x_2$ , которое необходимо поддерживать на постоянном уровне (требование к выходной переменной).

На этом этапе также рассматриваются режимы работы будущей системы, энергетические вопросы; строится функциональная схема системы и выбирается тип исполнительных механизмов, датчиков, усилителей и пр.

2-й этап. Математическое описание задачи синтеза. На этом этапе строятся математические модели всех элементов системы управления; формализуются требования к качеству динамических и статических режимов; выбирается тип регулятора и место его включения.

Типовые требования к качеству динамического и статического режимов можно графически представить в виде «коробочки Солодовникова» (рис. 5.1), которая отображает следующие требования [9]:

1) нулевая установившаяся ошибка при подаче на вход системы единичного ступенчатого сигнала  $x(t) = 1(t)$ , т.е.  $\varepsilon_{уст} = 0$ ;

2) максимальное перерегулирование  $\sigma_{max}$  в системе не должно превышать допустимого  $\sigma_{доп}$  ( $\sigma_{max} \leq \sigma_{доп}$ );

3) время регулирования (время установления)  $t_p$  не должно превышать допустимого  $t_{p доп}$ ;

4) кроме того, максимальное ускорение выходной переменной при заданных условиях не должно превышать допустимого значения.

3-й этап. Параметрический синтез регулятора, т.е. определение его параметров (настроек), например коэффициентов усиления и/или постоянных времени, которые бы обеспечивали желаемое качество.

4-й этап. Анализ качества работы системы и проверка его соответствия предъявляемым требованиям. Если система не удовлетворяет этим требованиям, то необходимо вернуться ко второму и третьему этапам. Возможно, что дело не ограничится изменением параметров регулятора и придется заменить конфигурацию системы, выбрать другое исполнительное устройство или датчик с улучшенными характеристиками.

5-й этап. Аппаратная реализация регулятора при помощи электрических, механических, гидравлических или других элементов. Если предполагается реализация регулятора при помощи

ЭВМ, то формируются требования к этой ЭВМ, строится алгоритмическое и программное обеспечение.

6-й этап. Испытания системы.

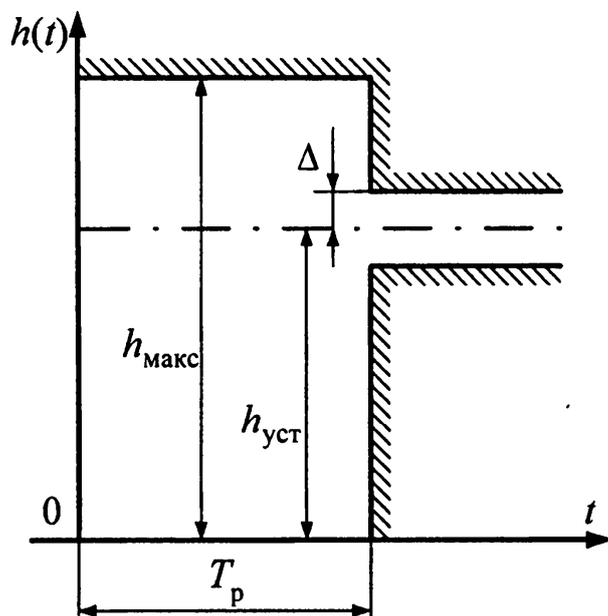


Рис. 5.1. Область допустимых значений переходной функции

Эта последовательность этапов синтеза системы автоматического управления является идеальной и, как правило, по ряду причин оказывается невозможной. Например, часто приходится «улучшать» уже созданную систему, основная часть которой (объект, исполнительный механизм, а иногда и датчики) уже задана и её нельзя изменить, поэтому ее так и называют – *неизменяемой*. Если неизменяемая часть системы управления не отвечает заданным требованиям, то необходимо вводить дополнительные элементы – корректирующие устройства или регуляторы. Эти элементы должны обеспечить требуемые динамические свойства системы.

На всех этапах синтеза целесообразно использовать возможности специальных пакетов прикладных программ, например, Simulink. Это позволяет относительно быстро исследовать большое количество вариантов структур системы и параметров её элементов и тем самым существенно ускорить решение задачи синтеза [5, 10, 11].

### 5.3. Классификация регуляторов

Для улучшения свойств системы управления в ее структуру вводятся корректирующие устройства или регуляторы. Регуляторы можно классифицировать по целому ряду признаков: по назначению, способу включения в структуру, принципу действия, по виду используемой энергии, по роду действия, виду закона регулирования и т.д.

По назначению регуляторы подразделяются на специализированные (например, регуляторы температуры, скорости, давления, уровня, напряжения и т.д.) и универсальные с нормированными входными и выходными сигналами и пригодные для управления различными параметрами.

По способу включения в структуру системы регуляторы бывают последовательные, параллельные, обратной связи и комбинированные. Преимущества и недостатки каждого из способов включения можно найти в специальной литературе, например, в [12 – 14]. Отметим лишь, что в большинстве существующих промышленных систем используется последовательная коррекция, когда регулятор преобразует сигнал ошибки  $\varepsilon(t)$  в управляющее воздействие  $U(t)$ , подаваемое на объект для уменьшения этого отклонения (рис. 5.2). Достаточно часто к регулятору относят задающее и сравнивающее устройства, а также усилитель мощности.

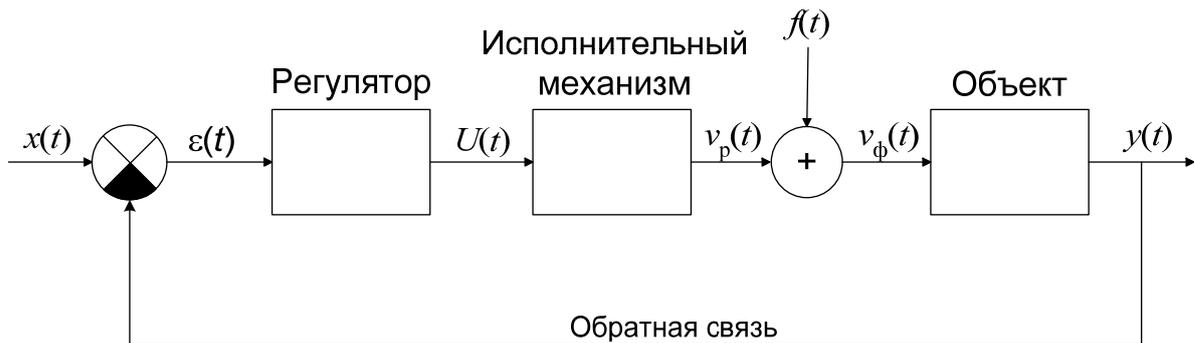


Рис. 5.2. Упрощенная структура системы управления

$x(t)$  – задающее воздействие;  $\varepsilon(t)$  – сигнал ошибки;  $U(t)$  – управление;  
 $v_p(t)$  – расчетное силовое воздействие на объект;  $f(t)$  – силовое возмущение;  
 $v_\phi(t) = v_p(t) + f(t)$  – фактическое силовое воздействие на объект;  
 $y(t)$  – выходная величина

По принципу действия регуляторы делят на регуляторы прямого и непрямого действия. Регуляторы прямого действия для выработки управляющего сигнала не используют внешнюю энергию, а используют энергию объекта управления или регулируемой среды (например, регуляторы давления). Регулятором непрямого действия для работы необходим внешний источник энергии.

По виду используемой энергии регуляторы подразделяются на электрические, пневматические, гидравлические, механические и комбинированные (например, электромеханические). Выбор регулятора по виду используемой энергии определяется характером самой системы.

По роду действия выделяют непрерывные и дискретные регуляторы. В последнее время наибольшее распространение получают дискретные регуляторы, построенные с помощью средств вычислительной техники.

По виду закона регулирования регуляторы бывают линейные и нелинейные.

К линейным, например, относятся регуляторы, реализующие простейшие, так называемые типовые законы регулирования: пропорциональный, интегрирующий, дифференцирующий и их комбинации. Среди нелинейных законов регулирования наиболее распространены релейные законы: двухпозиционный и трехпозиционный. Аналитически двухпозиционный закон регулирования описывается следующим образом:

$$U(t) = \begin{cases} U_{\max}, & \text{если } \varepsilon \geq \varepsilon_{\text{доп}}, \\ -U_{\max}, & \text{если } \varepsilon < \varepsilon_{\text{доп}}. \end{cases} \quad (5.1).$$

Трехпозиционный закон регулирования имеет следующий вид:

$$U(t) = \begin{cases} U_{\max}, & \text{если } \varepsilon > \varepsilon_{\text{н}}, \\ 0, & \text{если } \varepsilon_{\text{н}} \leq 0 \leq \varepsilon_{\text{н}}, \\ -U_{\max}, & \text{если } \varepsilon < \varepsilon_{\text{н}}. \end{cases} \quad (5.2)$$

Здесь величина  $\varepsilon_{\text{н}}$  определяет зону нечувствительности регулятора.

Примерами применения релейных регуляторов являются бытовые холодильники, термостаты, первые системы круиз-контроля.

Релейные законы позволяют достаточно просто достичь высокого быстродействия системы, хотя, как правило, в ущерб точности.

К рассматриваемому классификационному признаку можно отнести адаптивные и оптимальные регуляторы.

#### 5.4. Типовые линейные законы регулирования

Как уже отмечалось, наиболее распространенной в промышленных приложениях является структура, приведенная на рис. 5.1. Основной характеристикой регулятора является формируемый им закон регулирования:

$$U = f(\varepsilon), \quad (5.3)$$

который определяет основные качественные характеристики системы.

Существуют, так называемые, типовые законы регулирования, использование которых позволяет обеспечить выполнение тех или иных показателей качества.

##### 5.3.1. П-регулятор

Простейшим является *пропорциональный* закон и регулятор в этом случае называют *П-регулятором*. При этом

$$U_{\text{п}}(t) = K_{\text{п}}\varepsilon(t), \quad (5.4)$$

где  $K_{\text{п}}$  – коэффициент пропорциональности, который может иметь любой знак и любое значение.

Основным достоинством П-регулятора является простота. По существу, он представляет собой обычный усилитель с коэффициентом усиления  $K_{\text{п}}$ .

При выборе значения коэффициента  $K_{\Pi}$  регулятора приходится находить компромисс между двумя противоречивыми требованиями к системе управления. С одной стороны, время регулирования, т.е. время переходного процесса, должно быть как можно меньше, ведь система должна быть быстродействующей.

С другой стороны, необходимо обеспечить достаточный запас устойчивости, чтобы изменение параметров системы или условий ее работы в некоторых пределах не привело к неустойчивости. С этим требованием тесно связан такой показатель качества как максимальное перерегулирование  $\sigma$ . Напомним, что теоретически  $\sigma$  не должно превышать 20% (а как показывает практика, для нормальной работы многих систем управления перерегулирование должно быть не больше 2 ÷ 5%).

Влияние коэффициента усиления  $K_{\Pi}$  на показатели качества системы отображено в табл. 5.1. Как видим, повышение  $K_{\Pi}$  увеличивает быстродействие системы, поскольку П-регулятор более энергично стремится уменьшить ошибку регулирования  $\varepsilon(t)$  (см. 5.3). Однако при этом исполнительный механизм и объект быстрее накапливают энергию и система по инерции «перескакивает» желаемое значение выходной величины. Это приводит к повышению колебательности переходного процесса (рис. 5.3) и даже к потере устойчивости (см. п.п. 4.2.3). Поэтому и необходимо подходить к выбору значения  $K_{\Pi}$  с особой тщательностью, балансируя между быстродействием и максимальным перерегулированием.

*Таблица 5.1*

Эффект от повышения  $K_{\Pi}$

Время нарастания, $t_n$	Время регулирования, $t_p$	Перерегулирование, $\sigma$	Запас устойчивости	Установившаяся ошибка, $\varepsilon_{уст}$
Уменьшается	Немного уменьшается	Увеличивает	Уменьшает	Уменьшает

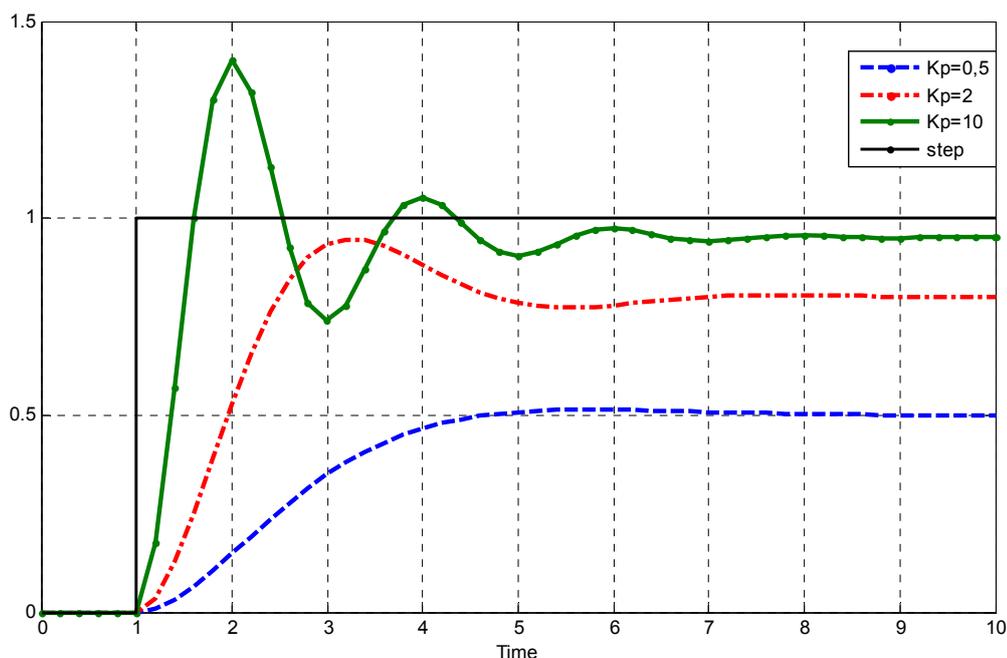


Рис. 5.3. Переходные характеристики системы при разных значениях  $K_p$

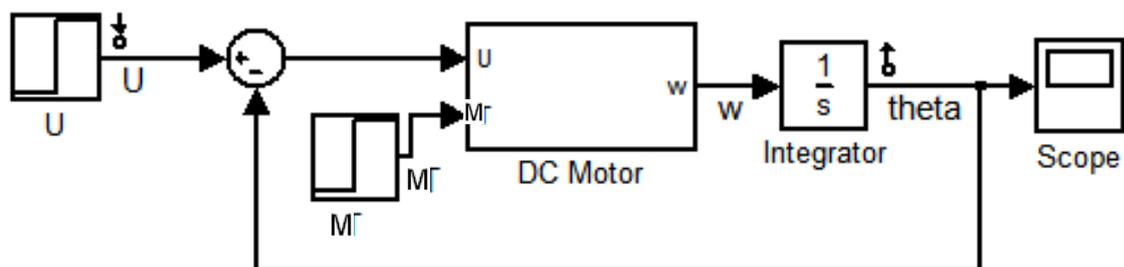
Еще одним недостатком П-регулятора является то, что в установившемся режиме *всегда* остается некоторое отклонение  $\varepsilon_{уст}$  регулируемой величины от заданного значения (см. рис.5.3).

Благодаря своей простоте П-регуляторы достаточно часто применяются на практике. Рассмотрим, как можно использовать возможности Simulink для оптимизации значения коэффициента  $K_p$ .

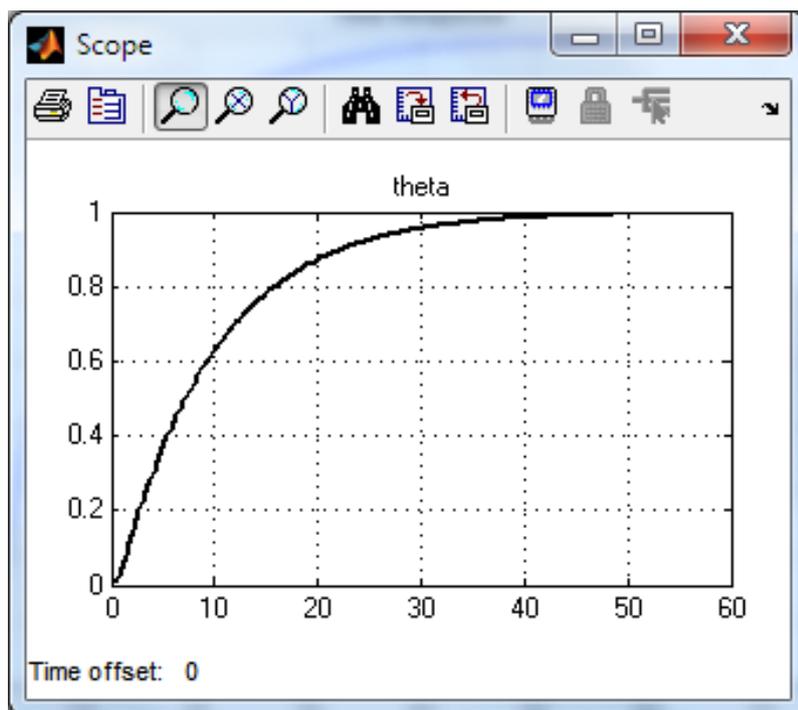
Возьмем в качестве примера систему управления углом  $\theta$  поворота вала двигателя постоянного тока (рис. 4.16). Для удобства приведём эту модель ещё раз (рис. 5.4, а). Переходная характеристика этой системы представлена на рис. 5.4, б.

Как видим, обратная связь обеспечивает нулевую установившуюся ошибку (т.к. система обладает интегрирующими свойствами), однако, хотелось-бы увеличить быстродействие, т.к. сейчас время регулирования составляет около 40 с. Для этих целей попробуем использовать П-регулятор.

- Установим в качестве входа системы напряжение  $U$ , вырабатываемое блоком Step, а в качестве выхода – выходной сигнал  $\theta$  блока Integrator. Напомним, что для этого надо щелкнуть правой кнопкой мыши на соответствующей линии связи и выбрать **Linearization Points** → **Inpun** (или **Output**) **Point**.



а



б

Рис. 5.4. Модель системы управления углом поворота вала двигателя постоянного тока (а) и её переходная характеристика (б)

- Вставим блок усилителя Gain между сумматором и подсистемой двигателя DC Motor.

- На панели меню модели Simulink выберем **Tools → Control Design → Compensator Design...** В результате появится окно **Control and Estimation Tools Manager**<sup>1</sup> (рис.5.5).

В правой части этого окна имеется три вкладки: **Tunable Blocks**, **Closed-Loop Signals** и **Operating Points**.

<sup>1</sup> Требуется установка пакета Simulink Control Design

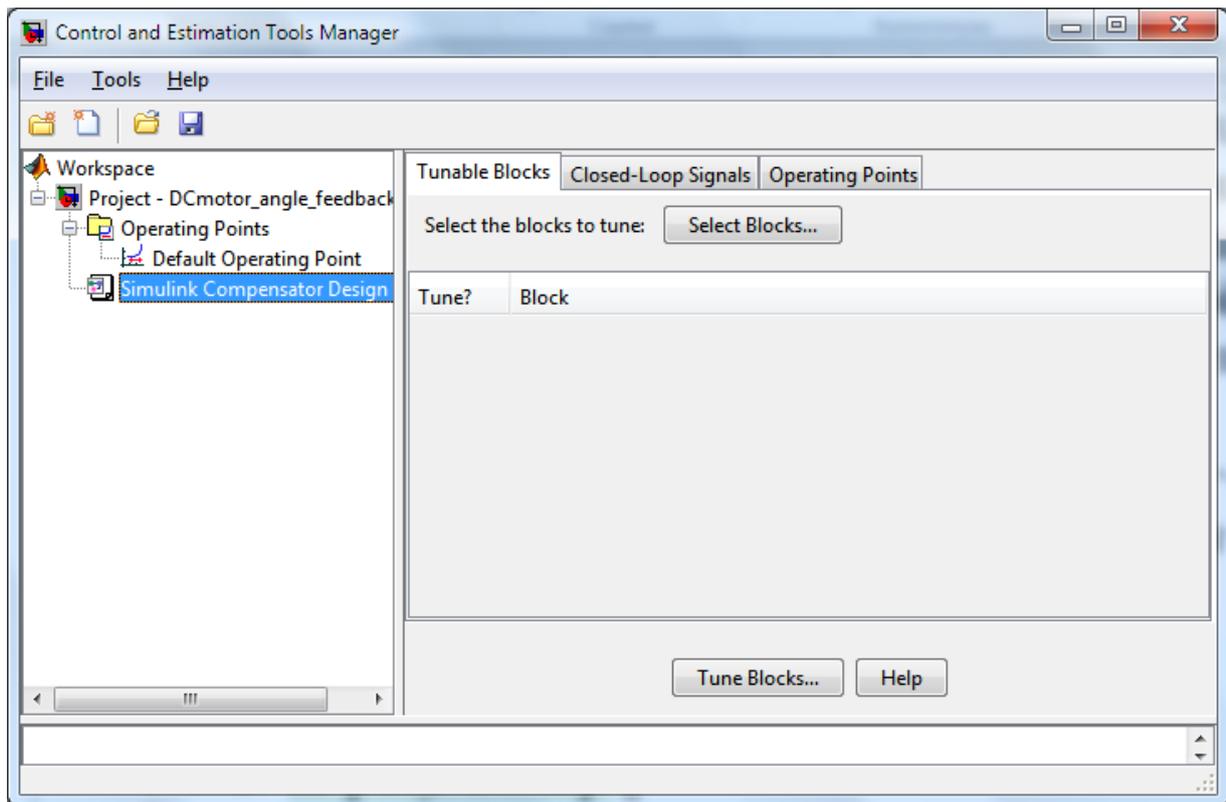


Рис. 5.5. Окно блока **Control and Estimation Tools Manager**

На открытой по умолчанию вкладке **Tunable Blocks** выберем блок, параметры которого надо настроить. В нашем случае это – блок **Gain**.

– Нажимаем на кнопку **Select Blocks**.

– В появившемся диалоговом окне **Select Blocks to Tune** дана таблица со списком блоков в модели, доступных для настройки. В нашем случае блок один – **Gain**.

– **Отмечаем блок Gain** (рис. 5.6). Для проверки правильности выбора можно «подсветить» выбранный блок, нажав кнопку **Highlight Selected Block**, расположенную в нижней части окна **Select Blocks to Tune**.

– Подтверждаем сделанный выбор нажатием кнопки **OK**, при этом мы вновь вернемся к окну **Control and Estimation Tools Manager**.

– Теперь откроем вкладку **Closed-Loop Signals** и проверим правильность задания входных и выходных сигналов (рис. 5.7).

– Если все в порядке, то нажимаем кнопку **Tune Blocks...**

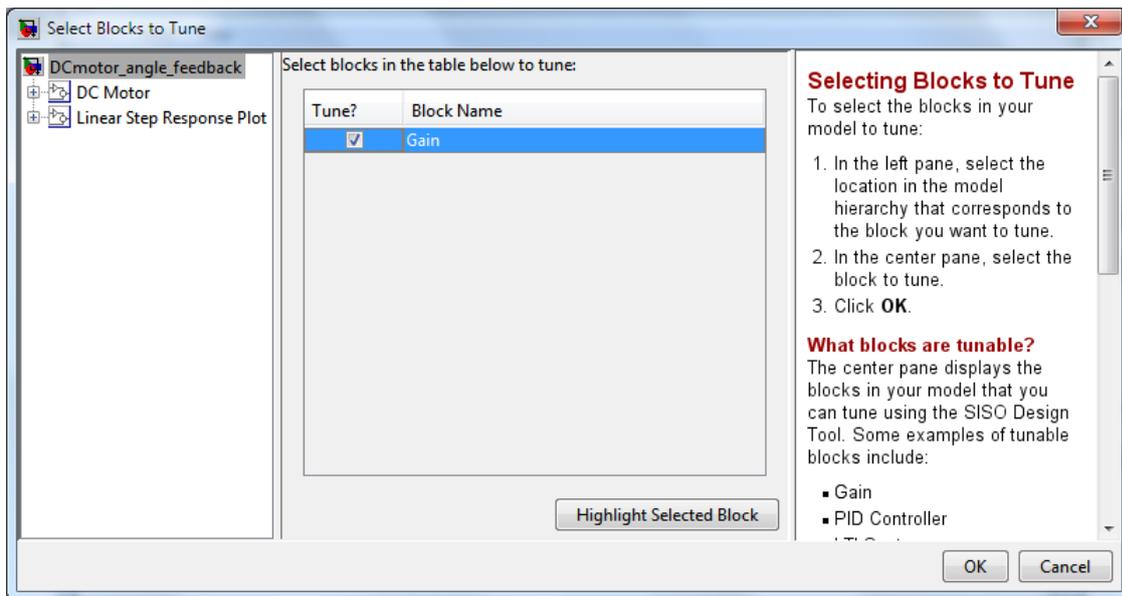


Рис. 5.6. Окно параметров **Select Blocks to Tune**

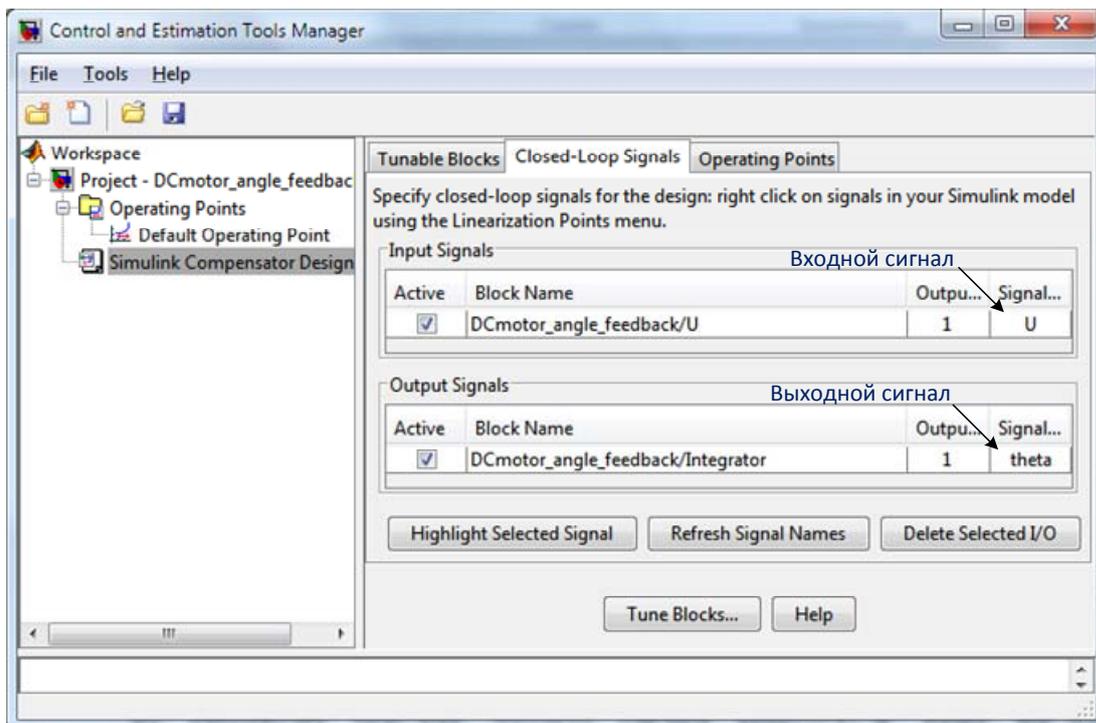


Рис. 5.7. Окно вкладки **Closed-Loop Signals**

После выполнения процесса анализа модели отрывается приветственное окно мастера конфигурации проектирования Design Configuration Wizard. В этом окне описываются возможности мастера. Нажимаем кнопку **Next**.

– В следующем окне (рис.5.8) предлагается таблица, в которой предлагается казать, при помощи какой характеристики будет проводится настройка выбранного блока Gain. Остановимся на этом пункте более подробно.

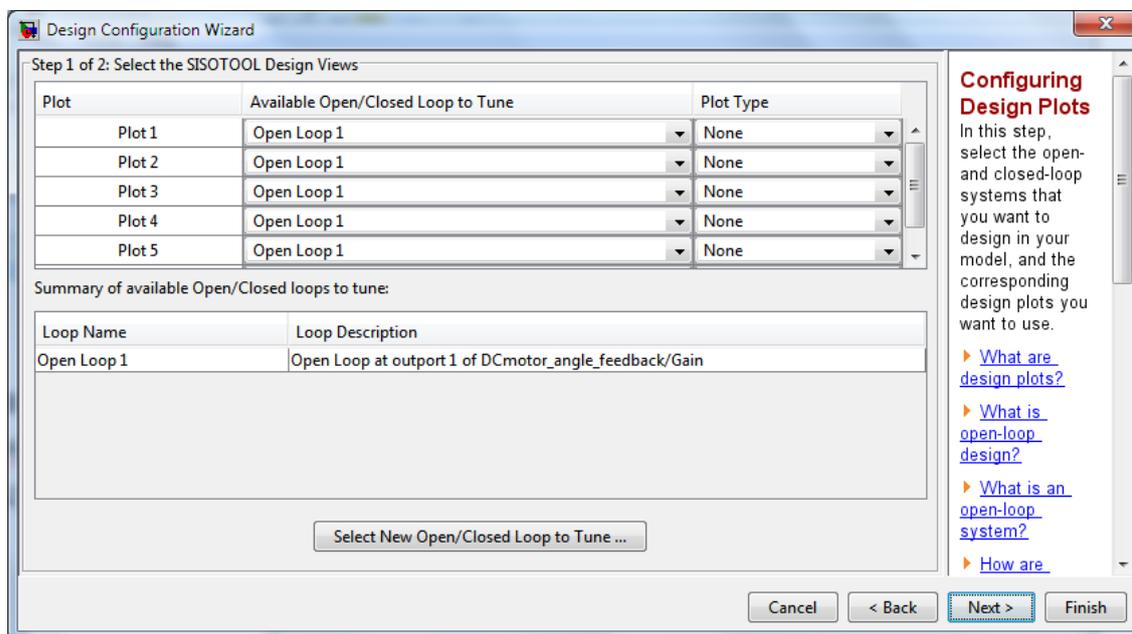


Рис. 5.8. Окно **Design Configuration Wizard**

В п. 4.2 мы говорили, что корни характеристического уравнения (4.3) определяют вид переходного процесса. Таким образом, обеспечив определенное расположение этих корней, можно добиться желаемого качества работы системы. Поскольку коэффициент усиления разомкнутой системы входит в состав характеристического уравнения, то изменяя значение этого коэффициента, мы меняем расположение корней на комплексной плоскости. Можно, однако, решить и обратную задачу: расположить корни на комплексной плоскости желаемым образом, и найти значение коэффициента усиления, который бы обеспечил требуемые значения корней.

Траекторию, отображающую расположение корней уравнения движения системы при изменении какого-либо его параметра (например, коэффициента усиления  $K$ ), называется *корневым годографом (по-английски - Root Locus)*.

Вернемся теперь к процедуре синтеза П-регулятора в Simulink.

- Выбираем в таблице (рис. 5.8) в качестве характеристики, при которой будет производится настройка корневой годограф - *Root Locus* (рис. 5.9).



Рис. 5.9. Выбор корневого годографа *Root Locus*

- Нажимаем кнопку Next. В результате появляется окно, в котором предлагается выбрать характеристики замкнутой системы, по которым будет контролироваться результат синтеза регулятора (рис 5.10).

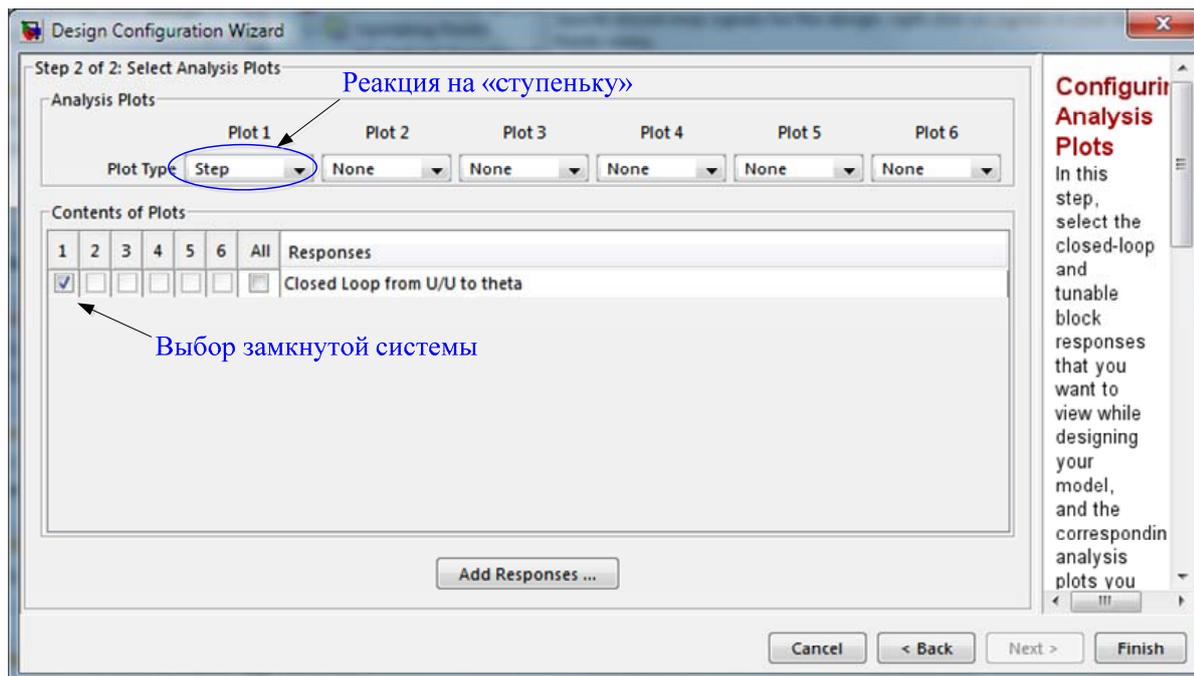


Рис. 5.10. Окно настройки П-регулятора

Для анализа можно использовать следующие характеристики:

- реакцию на единичный ступенчатый сигнал (Step response);
- реакцию на единичный импульс (Impulse response);
- диаграммы Боде (Bode and Bode magnitude);
- диаграмма Найквиста (Nyquist);
- диаграмма Никольса (Nichols);
- картина расположения полюсов и нулей системы (Pole/Zero).

В этом примере для анализа влияния изменения коэффициента  $K$  на динамику системы, выберем график реакции замкнутой системы на единичный ступенчатый сигнал.

- Выберем позицию Step из меню Plot 1.

- Укажем, что мы хотим рассматривать замкнутую систему (closed-loop system), для этого поставим флажок в соответствующей ячейке поля Content of Plots (рис. 5.10).

- Нажимаем кнопку Finish.

После нажатия кнопки Finish появляются два окна: с корневым годографом (рис. 5.11, а) и с реакцией замкнутой системы на ступенчатый сигнал (рис. 5.11, б).

Розовые квадратики на корневом годографе отображают значения корней характеристического уравнения (4.3) при принятом по умолчанию коэффициенте усиления блока Gain  $K = 1$ . Переходная характеристика на рис. 5.11, б, аналогична характеристике, приведенной на рис. 5.4, б.

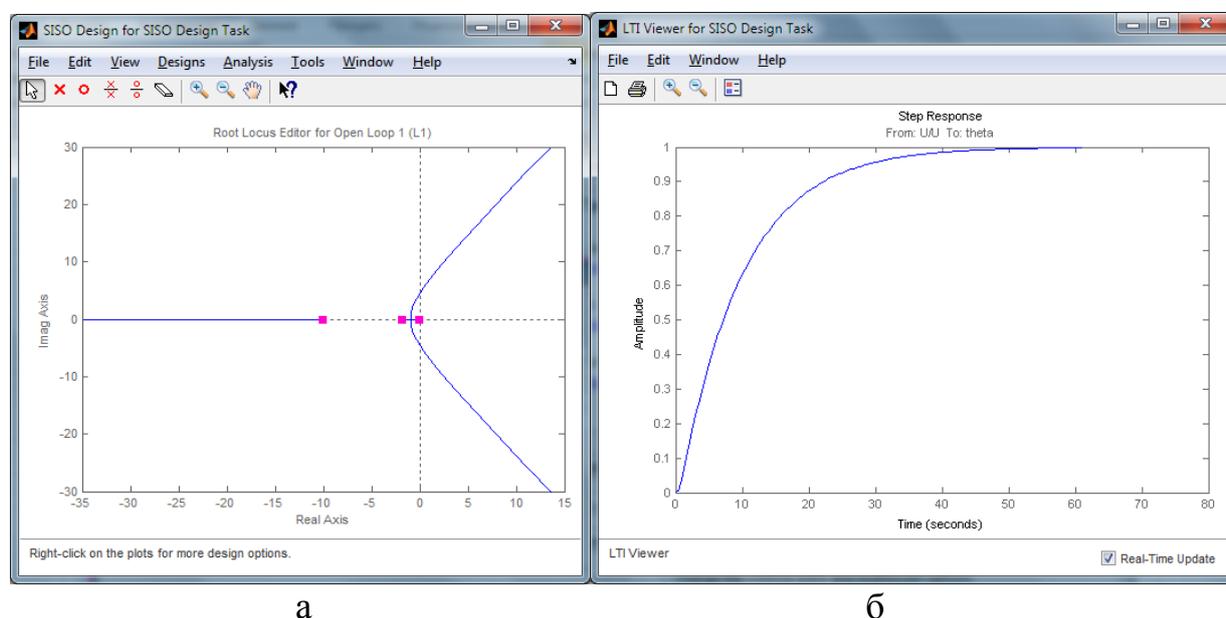


Рис. 5.11. Результаты установки корней характеристического уравнения при коэффициенте усиления  $K = 1$

Щелкнем левой кнопкой мыши на одном из квадратиков на корневом годографе, и начнем их перемещать вдоль возможной траектории, при этом одновременно меняется и вид переходной характеристики. Выберем такое расположение корней, которое бы соответствовало удовлетворительным показателям качества, например, как показано на рис. 5.12. Соответствующая переходная характеристика представлена на рис. 5.13.

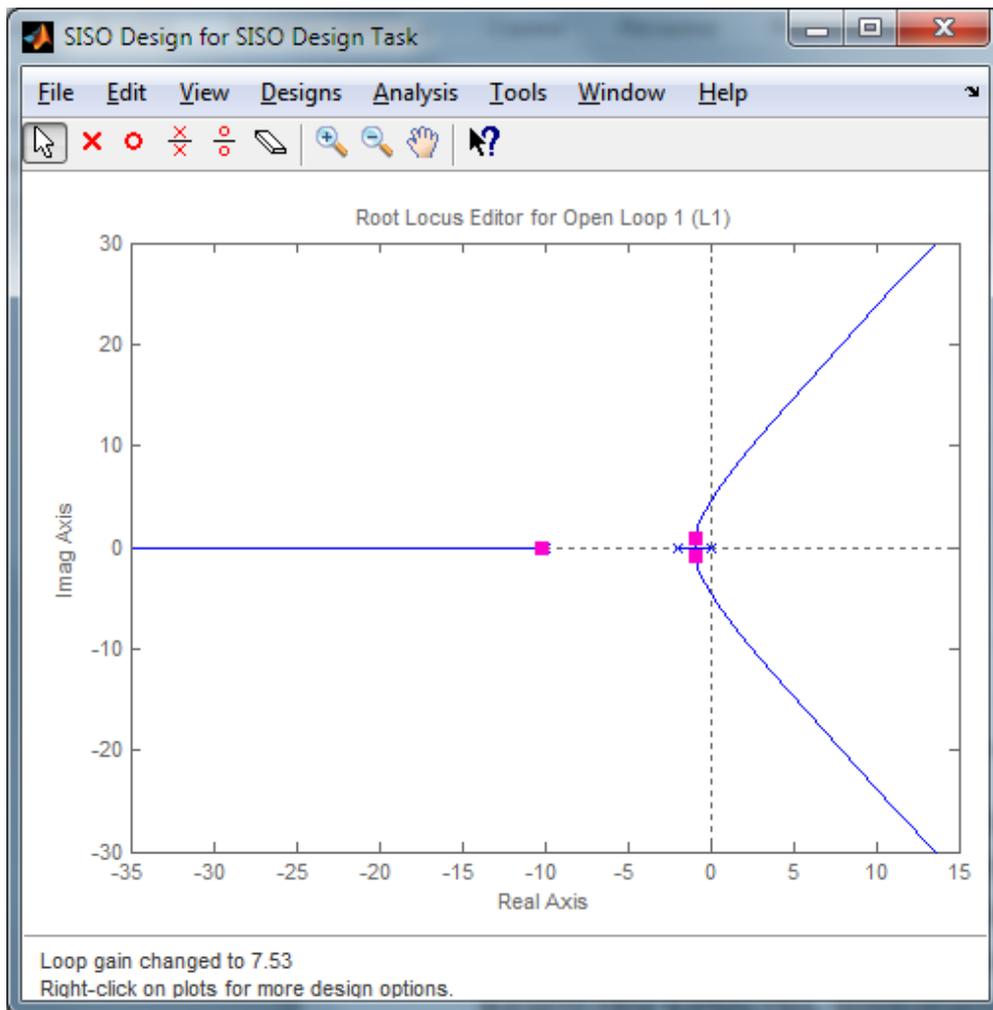


Рис. 5.12. Результаты установки корней характеристического уравнения

Как видно из рис. 5.13, расположение полюсов системы, приведенное на рис. 5.12, соответствует времени регулирования 4,83 с (вместо исходных 40 с.) и перерегулированию 2,86 %.

Посмотрим, какой коэффициент усиления  $K$  блока Gain обеспечивает такие показатели качества. Для этого перейдем на вкладку **Compensator Editor** окна **Control and Estimation Tools Manager** (рис. 5.14). На этой вкладке в ячейке **Value** отображается предлагаемое значение коэффициента усиления  $K$ . Это значение можно изменить, перемещая бегунок в ячейке **Slider**. В ячейках **Min Value** и **Max Value** можно задавать минимальное и максимальное значение диапазона изменения  $K$ . Применить новое значение коэффициент  $K$  можно, нажав кнопку **Update Simulink Block Parameters** (рис. 5.14). Чтобы значение параметра блока Gain автоматически обновлялось в окне модели, следует поставить флажок напротив позиции **Automatically update block parameters**.

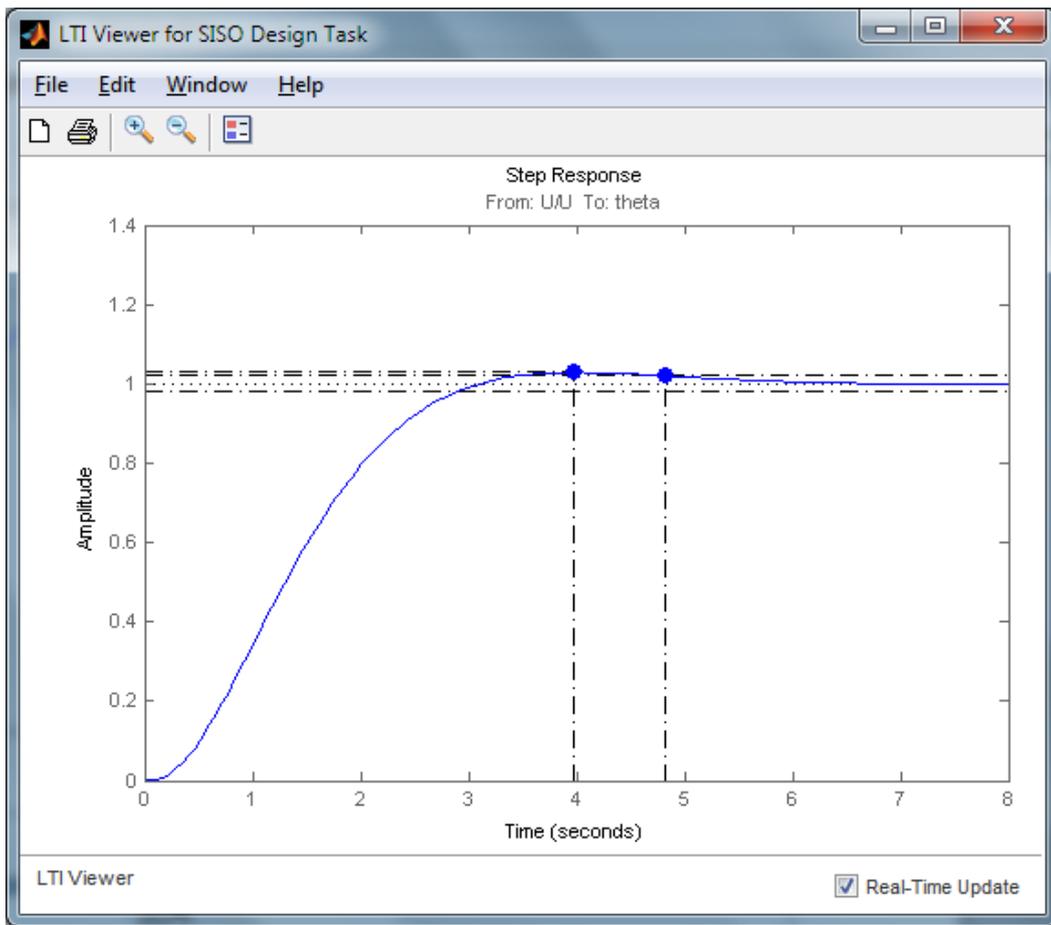


Рис. 5.13. Результаты определения быстродействия и величины перерегулирования.

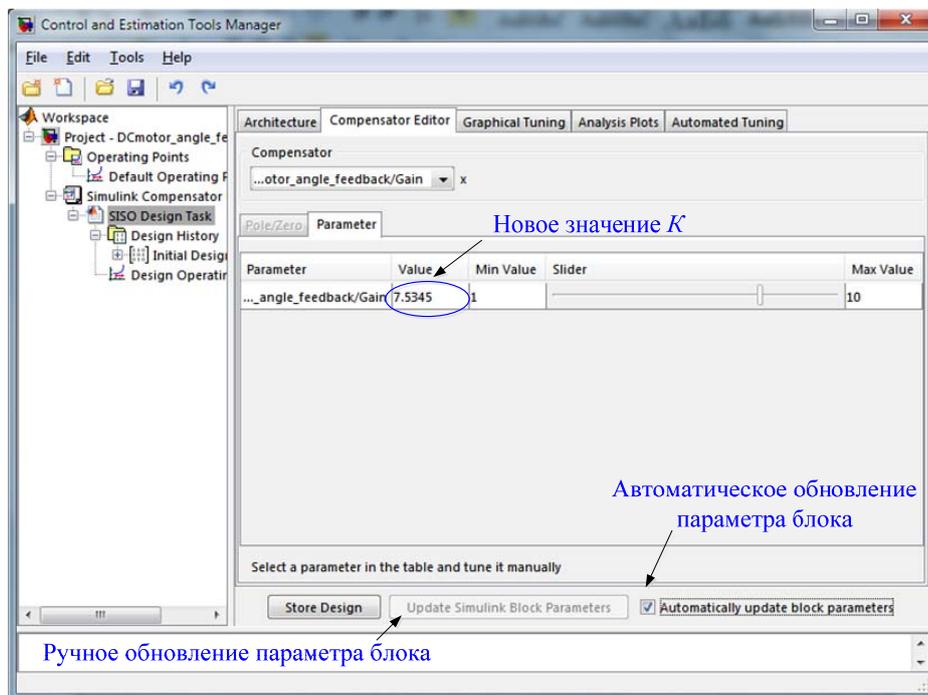


Рис. 5.14. Окно Control and Estimation Tools Manager

Значение параметра блока Gain можно подобрать и в автоматическом режиме при помощи встроенных в Simulink методов оптимизации.

Для автоматической настройки параметра блока перейдем в окне **Control and Estimation Tools Manager** на вкладку **Automated Tuning**. На этой вкладке в раскрывающемся списке Design method надо выбрать метод проектирования (рис. 5.15):

В данном случае нас интересует позиция **Optimization-Based Tuning** – оптимизация параметров корректирующего устройства. Отметим, также, что здесь имеется возможность настройки параметров ПИД-регулятора.

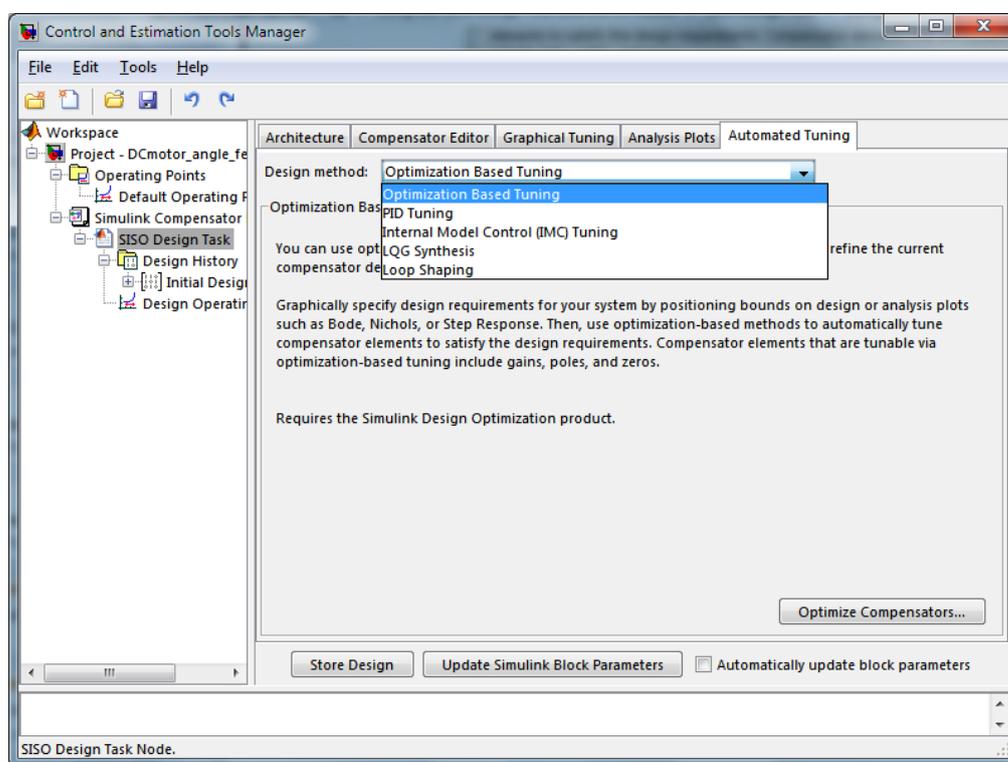


Рис. 5.15. Вкладка **Automated Tuning** окна **Control and Estimation Tools Manager**

Выберем метод **Optimization-Based Tuning** и нажмем кнопку **Optimize Compensators** (рис. 5.16), после чего становятся доступными 4 вкладки: **Overview** (Обзор), **Compensators** (Корректирующие устройства), **Design Requirments** (Конструктивные требования) и **Optimization** (Оптимизация).

На вкладке **Overview** отображены этапы оптимизации (рис. 5.17).

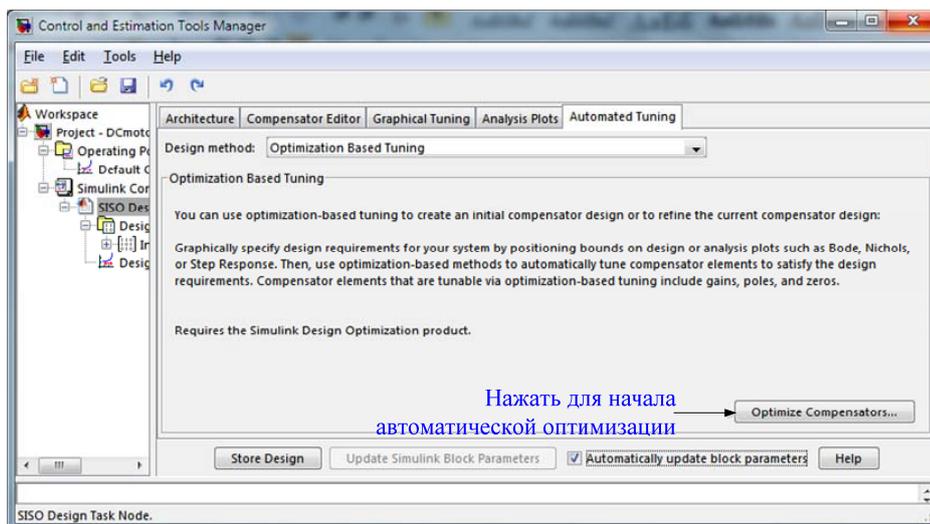


Рис. 5.16. Вкладка **Optimization-Based Tuning** окна **Control and Estimation Tools Manager**

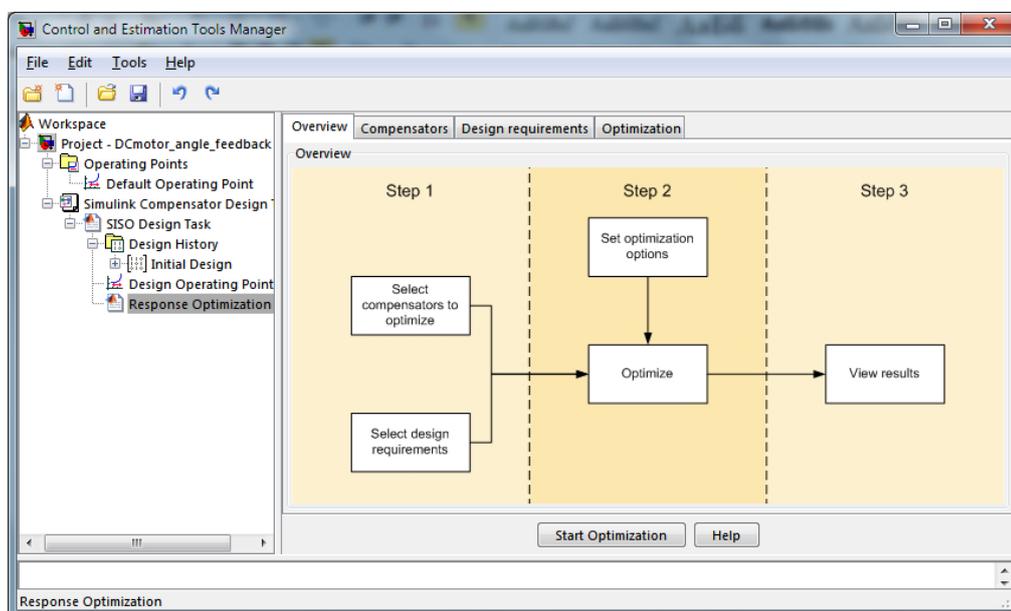


Рис. 5.17. Вкладка **Overview** окна **Control and Estimation Tools Manager** при автоматической настройке параметров блока

На вкладке **Compensators** выбираются блоки, параметры которых подлежат оптимизации. Установим здесь флажок напротив позиции Gain (рис. 5.18).

На вкладке **Design Requirements** устанавливаются требования к показателям качества системы, причем эти требования можно задать как аналитически, так и графически, при помощи «коробочки Солодовникова».

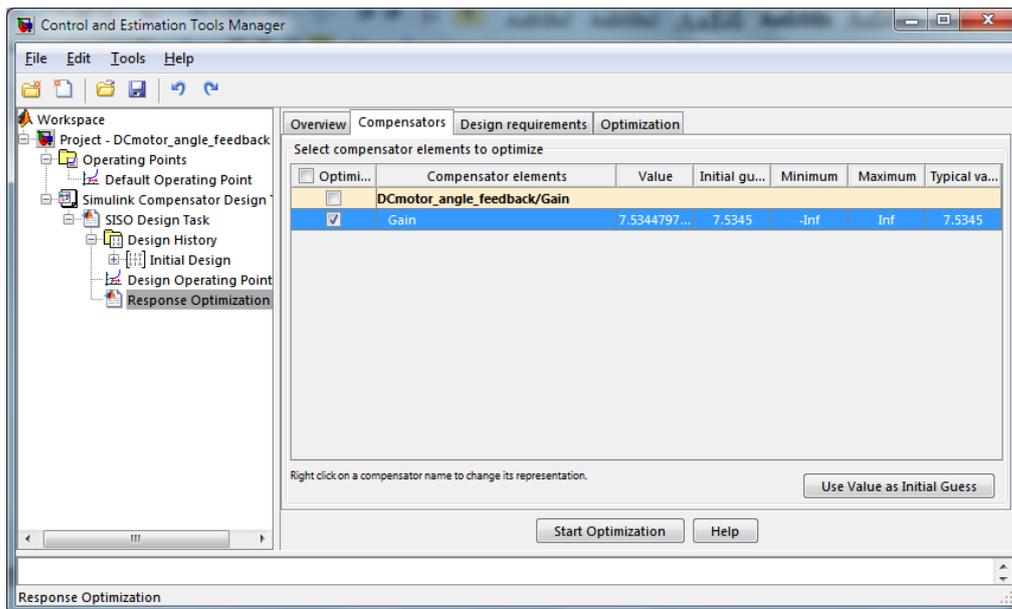


Рис. 5.18. Вкладка **Compensators** окна **Control and Estimation Tools Manager**

Для аналитического задания требований к качеству системы на вкладке **Design Requirements** нажмем кнопку **Add new design requirement** (рис. 5.19).

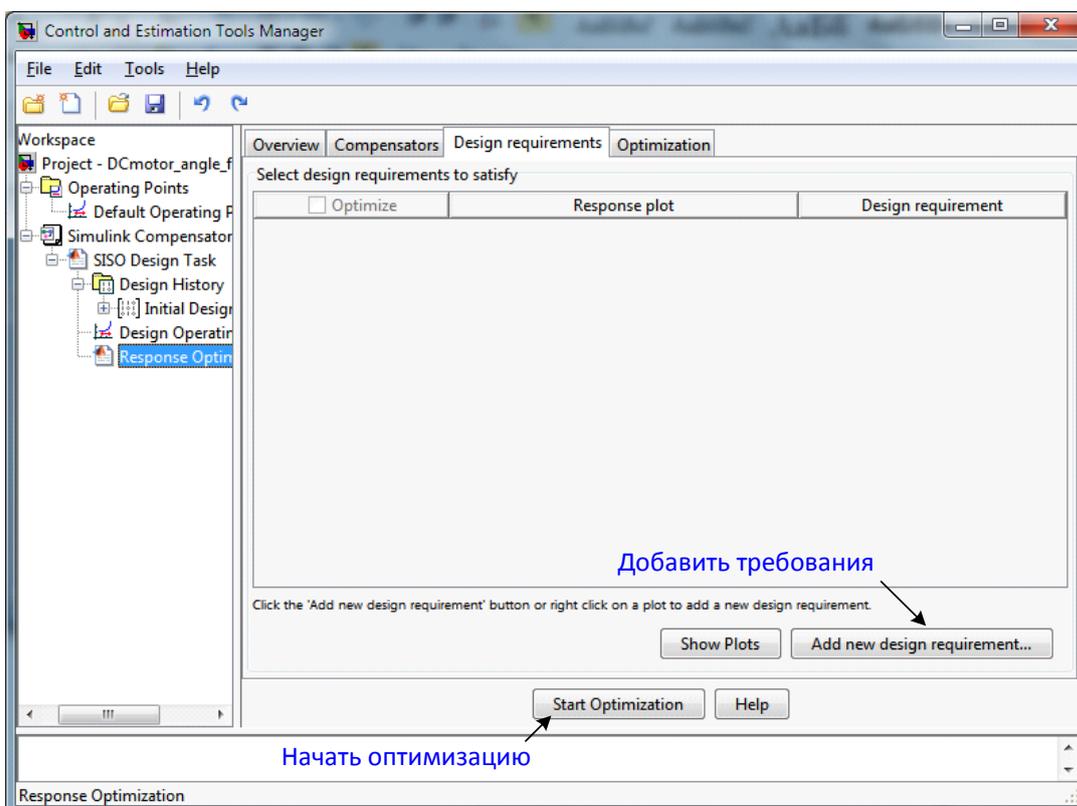
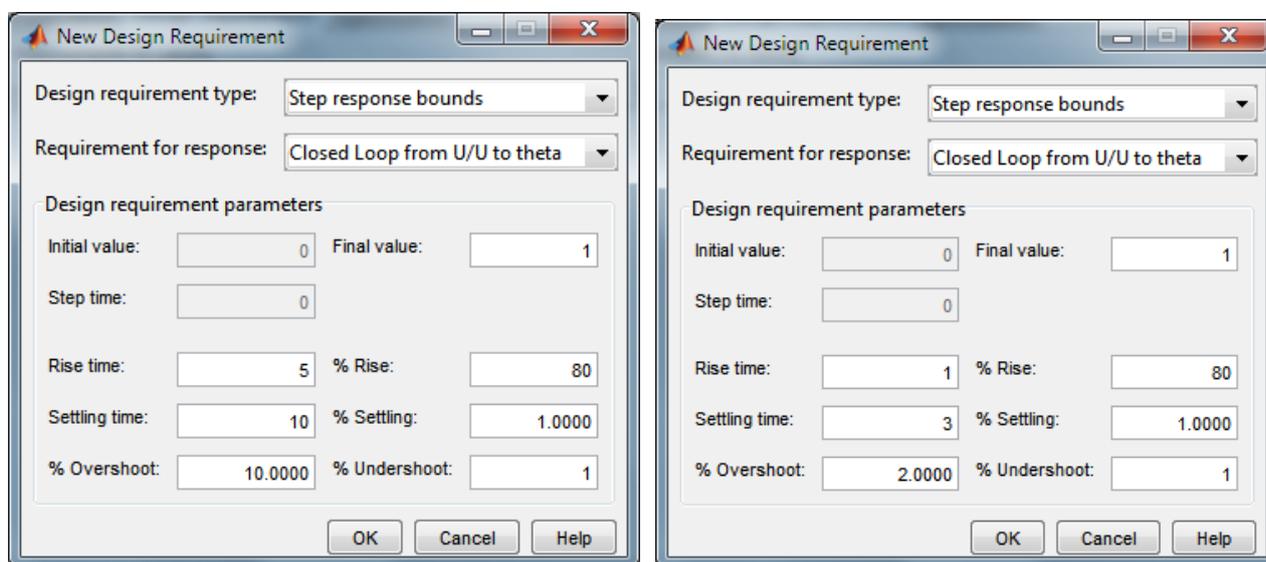


Рис. 5.19. Вкладка **Design Requirements**, позволяющая устанавливать требования к показателям качества системы

Это действие вызывает на экран окно **New Design Requirement** (рис. 5.20, а), в котором можно аналитически задать показатели качества переходного процесса. Поскольку ранее мы выбрали П-регулятор, который обеспечивает время регулирования 4,83 с при перерегулировании 2,86 %, ужесточим эти требования: попробуем достичь следующих показателей:

- время нарастания (**Rise time**) 1 с;
- время регулирования (**Setting time**) 3 с;
- перерегулирование (**Overshoot**) 2%.

Введем эти значения в соответствующие текстовые поля окна **New Design Requirement** (рис. 5.20, б) и нажмем кнопку **OK**. На экране появится окно **LTI Viewer for SISO Design Task** (рис. 5.21).



а

б

Рис. 5.20. Окна **New Design Requirement**

Это окно отображает заданные нами требования к системе в виде «коробочки Солодовникова». При необходимости, эти требования можно изменить, перемещая при помощи мыши границы «коробочки». Кроме того, в окне отображена переходная характеристика системы с текущим значением коэффициента усиления блока Gain.

Возвращаемся на вкладку **Design Requirments** окна **Control and Estimation Tools Manager** и нажимаем кнопку **Start Optimization** для запуска оптимизации. После этого появляется вкладка **Optimization**, на которой отображается ход процесса оптимизации (рис. 5.22). Одновременно в окне **LTI Viewer for SISO Design Task**

можно наблюдать изменение переходной функции. Но вот процесс оптимизации закончен и в окне вкладки **Optimization** выводится сообщение:

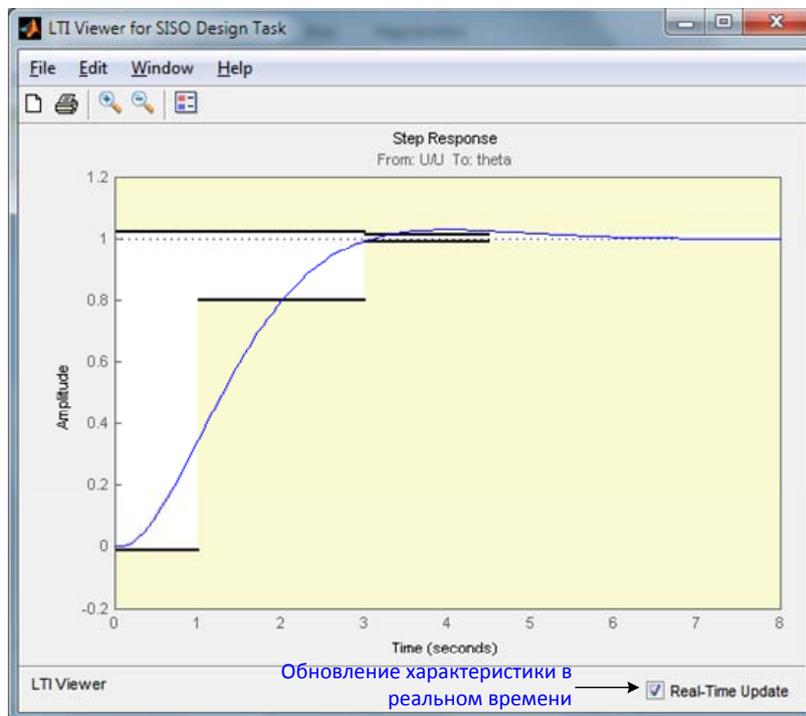


Рис. 5.21. Отображение результатов настройки П-регулятора

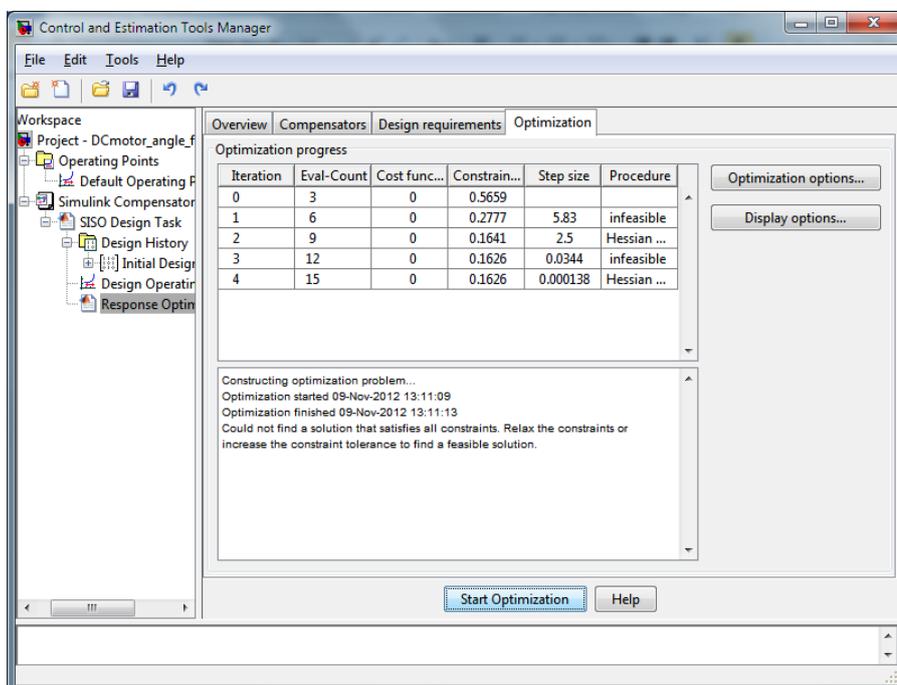


Рис. 5.22. Вкладка **Optimization** с сообщением об ошибке оптимизации

«Could not find a solution that satisfies all constraint. Relax the constraints or increase the constraint tolerance to find a feasible solution» (Не удалось найти решение, удовлетворяющее всем ограничениям. Ослабьте ограничения или увеличьте допуск для нахождения подходящего решения).

Неудачность процесса оптимизации иллюстрирует и график в окне **LTI Viewer for SISO Design Task** (рис. 5.23).

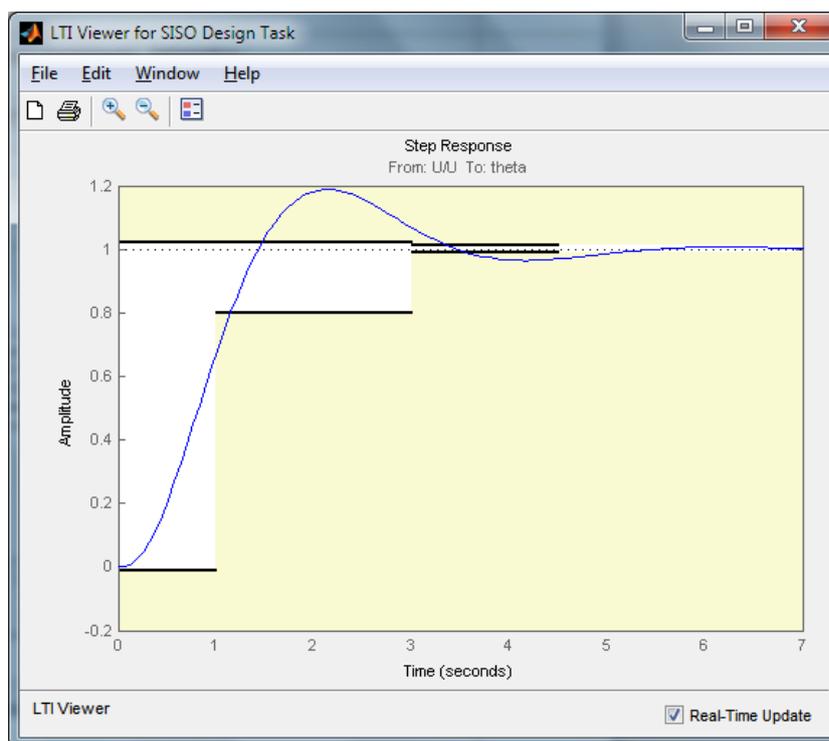


Рис. 5.23. Окно **LTI Viewer for SISO Design Task** после неудачного процесса оптимизации

Дело в том, что мы задали такие требования к качеству системы, которые невозможно достичь использованием одного лишь П-регулятора. Попробуем ослабить эти требования. Перемещая при помощи левой кнопки мыши границы «коробочки Солодовникова» добьемся следующих требований (рис. 5.24):

- время нарастания (**Rise time**) 2,5 с;
- время регулирования (**Setting time**) 4 с;
- перерегулирование (**Overshoot**) 2%.

Вновь запустим оптимизацию, нажав в окне **Control and Estimation Tools Manager** кнопку **ОК**. Теперь оптимизация прошла успешно (рис. 5.25 и рис. 5.26).

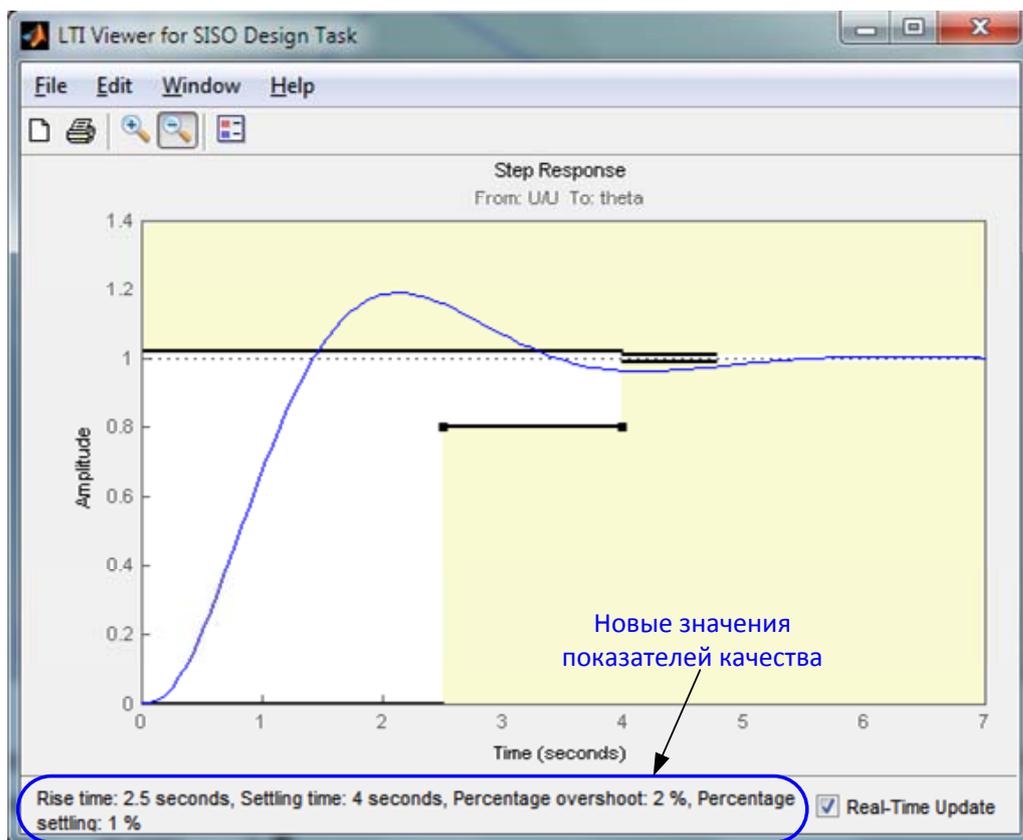


Рис. 5.24. Окно LTI Viewer for SISO Design Task после второй попытки ОПТИМИЗАЦИИ

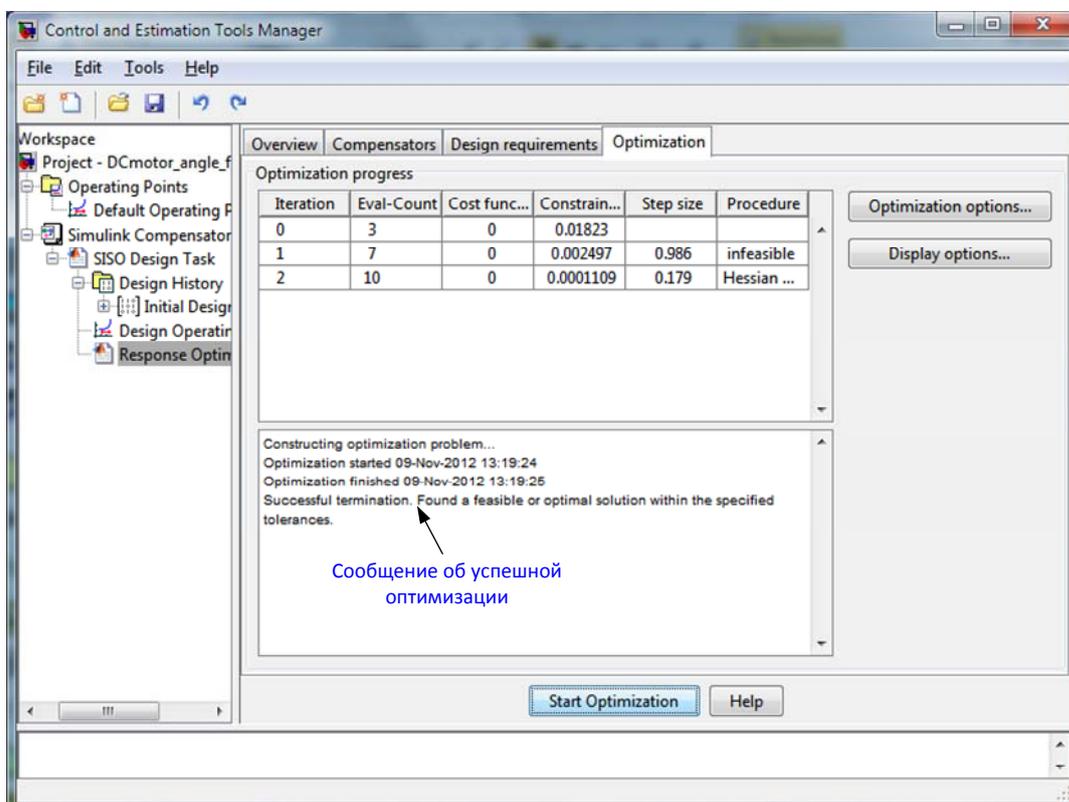


Рис. 5.25. Вкладка **Optimization** с сообщением о успешной оптимизации

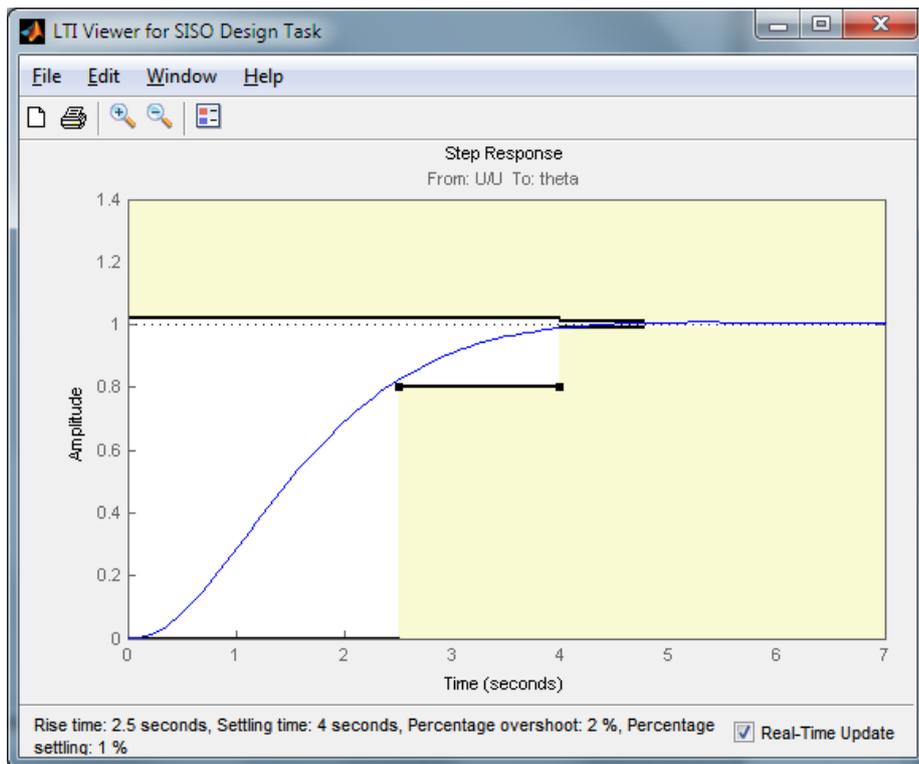


Рис. 5.26. Окно **LTI Viewer for SISO Design Task** после второй попытки оптимизации

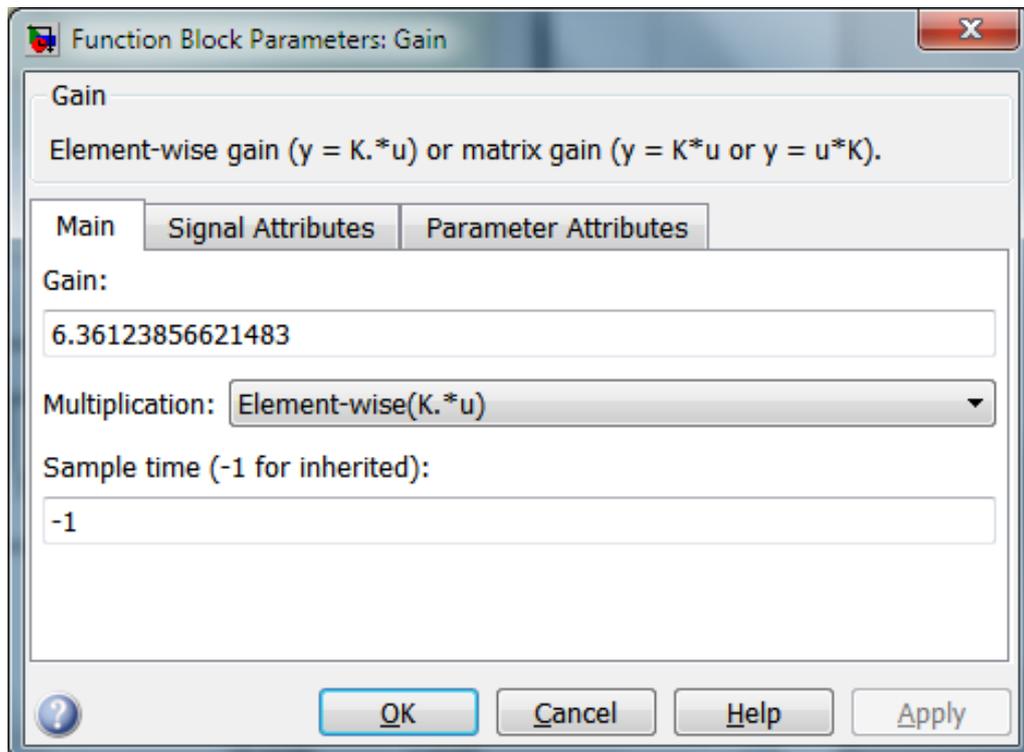


Рис. 5.27. Окно параметров блока **Gain**, с изменённым значением коэффициента усиления  $K$

Если в окне **Control and Estimation Tools Manager** стоит флажок напротив позиции **Automatically update block parameters**, то новое значение коэффициента усиления  $K$  автоматически передалось в блок Gain (рис. 5.27);  $K \approx 6,36$ . В противном случае это значение можно посмотреть на вкладке **Compensators** (рис. 5.28).

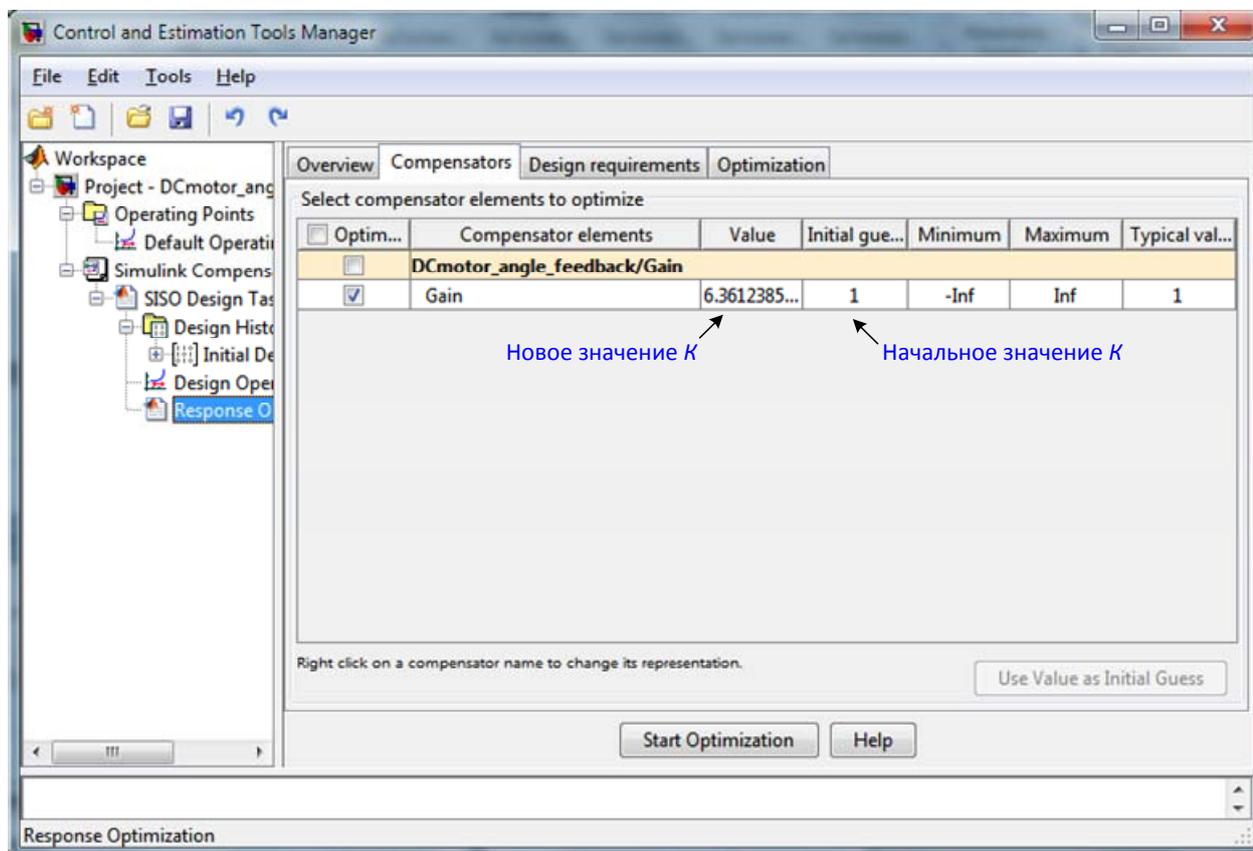


Рис. 5.28. Выкладка **Compensators**, со значением коэффициента усиления

На рис. 5.29 для сравнения приведены переходные характеристики системы управления углом поворота вала двигателя постоянного тока без регулятора и с П-регулятором при  $K \approx 6,36$ .

Читателю предоставляется возможность самостоятельно синтезировать П-регулятор для системы управления скоростью поворота вала двигателя. Обратите внимание, что, в этом случае, в системе будет присутствовать установившаяся ошибка.

В заключение отметим, что описанным выше способом можно оптимизировать значения и других блоков, а не только блока Gain.

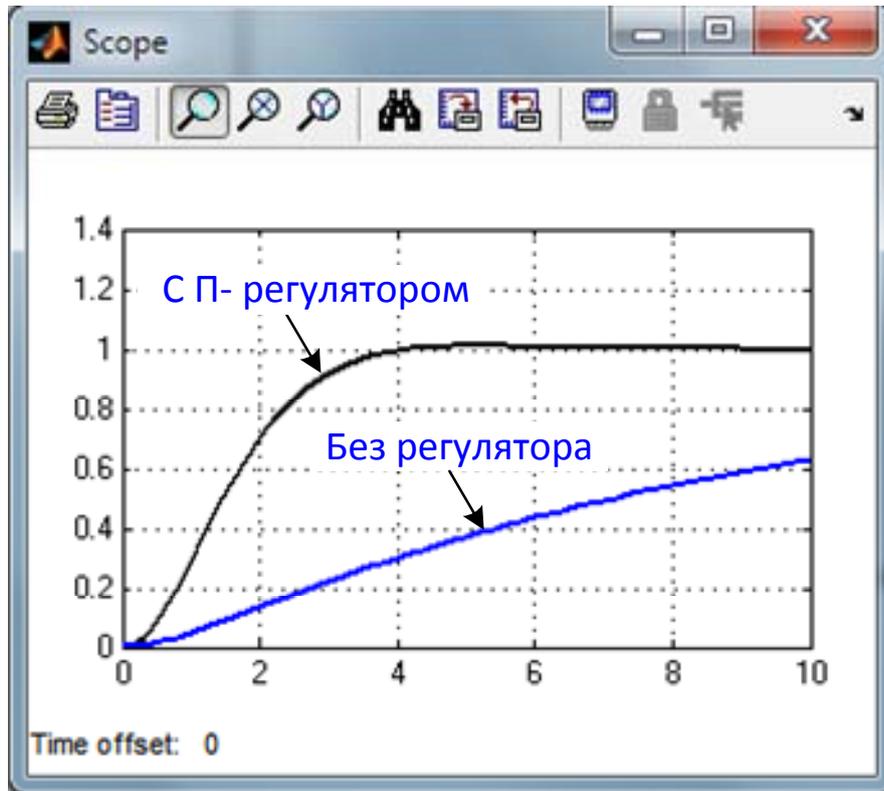


Рис. 5.29. Сравнительные результаты расчёта угла поворота вала двигателя постоянного тока с П-регулятором и без него

### 5.3.2. Интегрирующий регулятор

Интегрирующий закон регулирования реализует И – регулятор:

$$U_{и}(t) = U_{и}(0) + K_{и} \int_0^t \varepsilon(\tau) d\tau, \quad (5.5)$$

где  $U_{и}(0)$  – исходное значение выходной величины интегрирующего канала,  $K_{и}$  – коэффициент усиления, которому тоже может быть придано любое значение.

Остановимся на объяснении работы интегрирующего канала подробнее, поскольку опыт показывает, что операция интегрирования по времени часто остается для инженеров до конца непонятной.

Будем полагать, что движение системы при перемене желаемого режима  $x(t)$  начинается из состояния покоя и присвоим всем координатам в этом состоянии нулевые значения, в том числе и величине  $U_{и}(0)$ . Примем  $K_{и} = 1$ , чтобы не затенять суть дела.

Итак, пусть

$$U_{\text{и}}(t) = \int_0^t \varepsilon(\tau) d\tau. \quad (5.6)$$

Знак интеграла  $\int$  – это стилизованное изображение буквы **S** – начальной буквы английского слова *sum* (сумма). Интегрирование – это суммирование, накопление. Интегратор, таким образом, – это «копилка».

Пусть процесс  $\varepsilon(t)$  имеет вид, показанный на рис. 5.30. Для того чтобы определить значение  $U_{\text{и}}(t)$  к моменту  $t_1$ , разобьем интервал  $(0; t_1)$  на  $m$  малых интервалов длительностью  $\Delta t$ . Будем на каждом таком интервале считать ошибку  $\varepsilon(t)$  величиной постоянной и равной, например, ее среднему значению на интервале  $\Delta t$ . Тогда

$$U_{\text{и}}(t_1) \approx \sum_{i=1}^m \varepsilon(t_i) \cdot \Delta t. \quad (5.7)$$

Формула (5.7) разъясняет смысл формулы (5.5). Но, если угодно, то именно она является первородной формулой в понятии интегрирования, а (5.6) – это предельное значение формулы (5.7) при  $\Delta t \rightarrow 0$ .

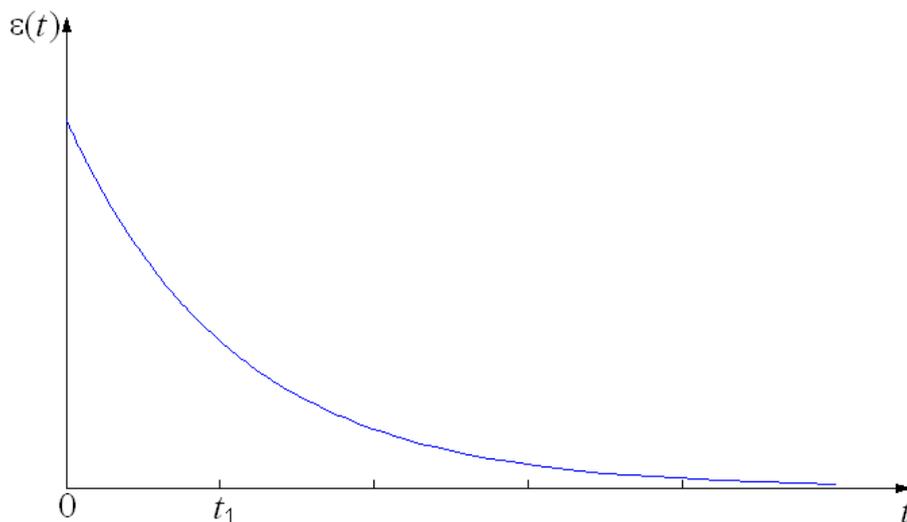


Рис. 5.30. Процесс изменения ошибки  $\varepsilon(t)$

Из (5.7) следует, что  $U_{\text{и}}(t)$  к моменту  $t_1$  численно будет равно площади, заштрихованной на рис. 5.31.

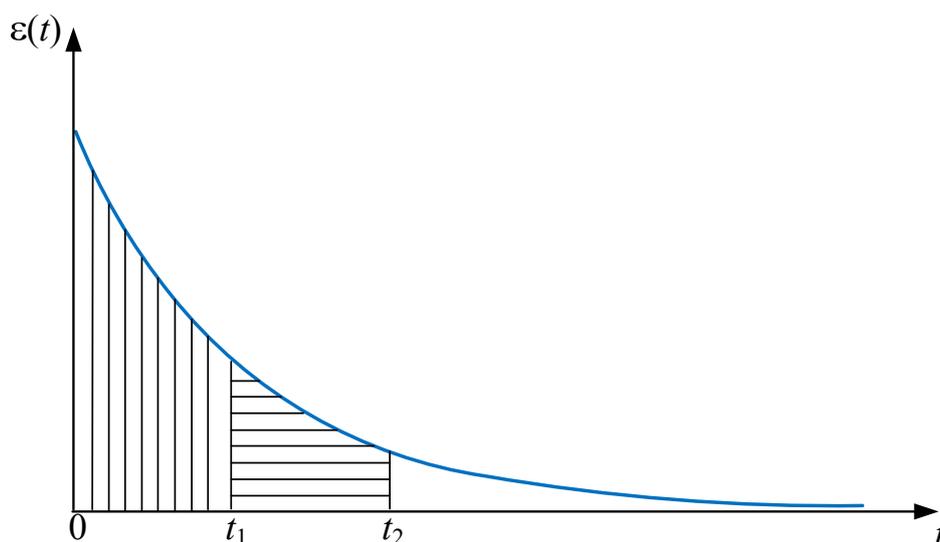


Рис.. 5.31. Площадь, численно определяющая значение управления  $U_{и}(t)$

Хотелось бы обратить внимание на использованную выше терминологию: «Для того чтобы определить  $U_{и}(t)$  к моменту  $t_1$ , разобьем...». Можно было бы сказать и так: «Для того чтобы определить  $U_{и}(t)$  в момент  $t_1$ , разобьем...». Примененная терминология должна привить понимание постепенности накопления  $U_{и}(t_1)$ . Значение  $U_{и}(t_1)$  не определяется значением  $\varepsilon(t)$  в какой-либо конкретный момент времени;  $U_{и}(t_1)$  – это суммарная, интегральная оценка всей предыстории процесса  $\varepsilon(t)$  при  $t < t_1$ .

В момент  $t_2 > t_1$  управления  $U_{и}(t_2)$ , будет равно  $U_{и}(t_1)$  плюс численное значение площади, заштрихованной на рис. 5.31 горизонтальными линиями.

Все сказанное позволяет построить график  $U_{и}(t)$ , соответствующий графику  $\varepsilon(t)$ , приведенному на рис. 5.30, 5.31,. Этот график приведен на рис. 5.32.

Важнейшим для процесса регулирования является свойство интегрирующего регулятора, заключающееся в том, что **управление  $U_{и}(t)$  перестает менять свое значение только при  $\varepsilon(t) = 0$** . В противном случае  $U_{и}(t)$  постепенно меняет свое значение до минимального или конструктивного максимального возможного значения в зависимости от знаков  $\varepsilon(t)$  и коэффициента усиления  $K_{и}$ .

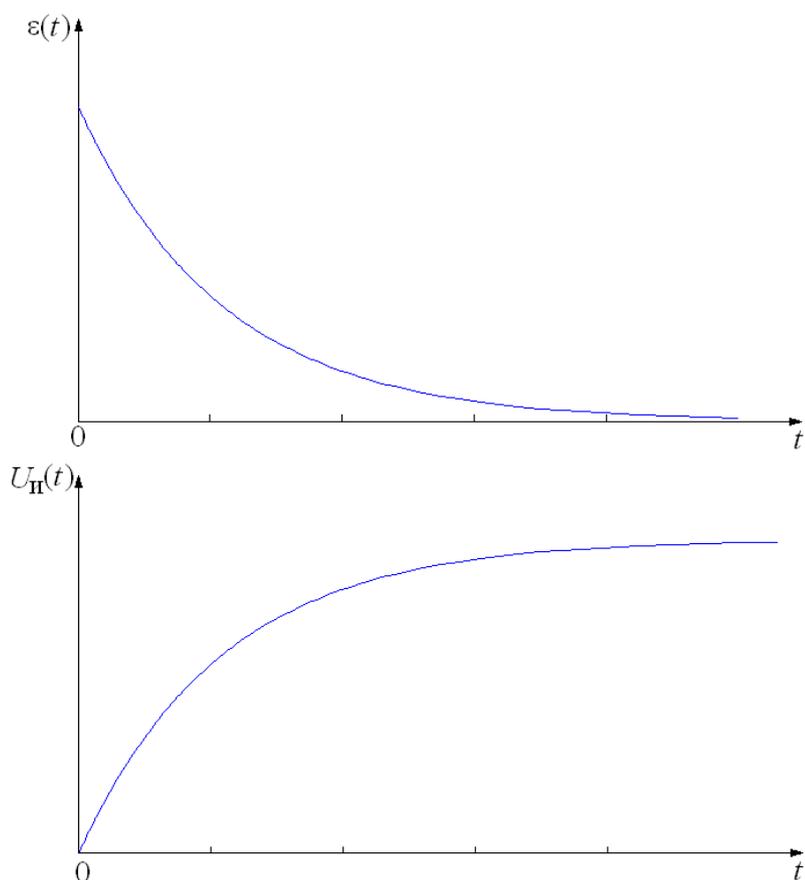


Рис. 5.32. Соответствующие графики процессов  $\varepsilon(t)$  и  $U_{II}(t)$

Таким образом, И-регуляторы устраняют установившуюся ошибку. Вместе с тем И-регулятор вызывает уменьшение запаса устойчивости системы (см. табл. 5.2). На рис. 5.33 приведены переходные характеристики одной и той же системы при разных значениях  $K_{II}$ .

Таблица 5.2

Эффект от повышения  $K_{II}$

Время нарастания, $t_n$	Время регулирования, $t_p$	Перерегулирование, $\sigma$	Запас устойчивости	Установившаяся ошибка, $\varepsilon_{уст}$
Уменьшается	Увеличивает	Увеличивает	Уменьшает	Устраняет

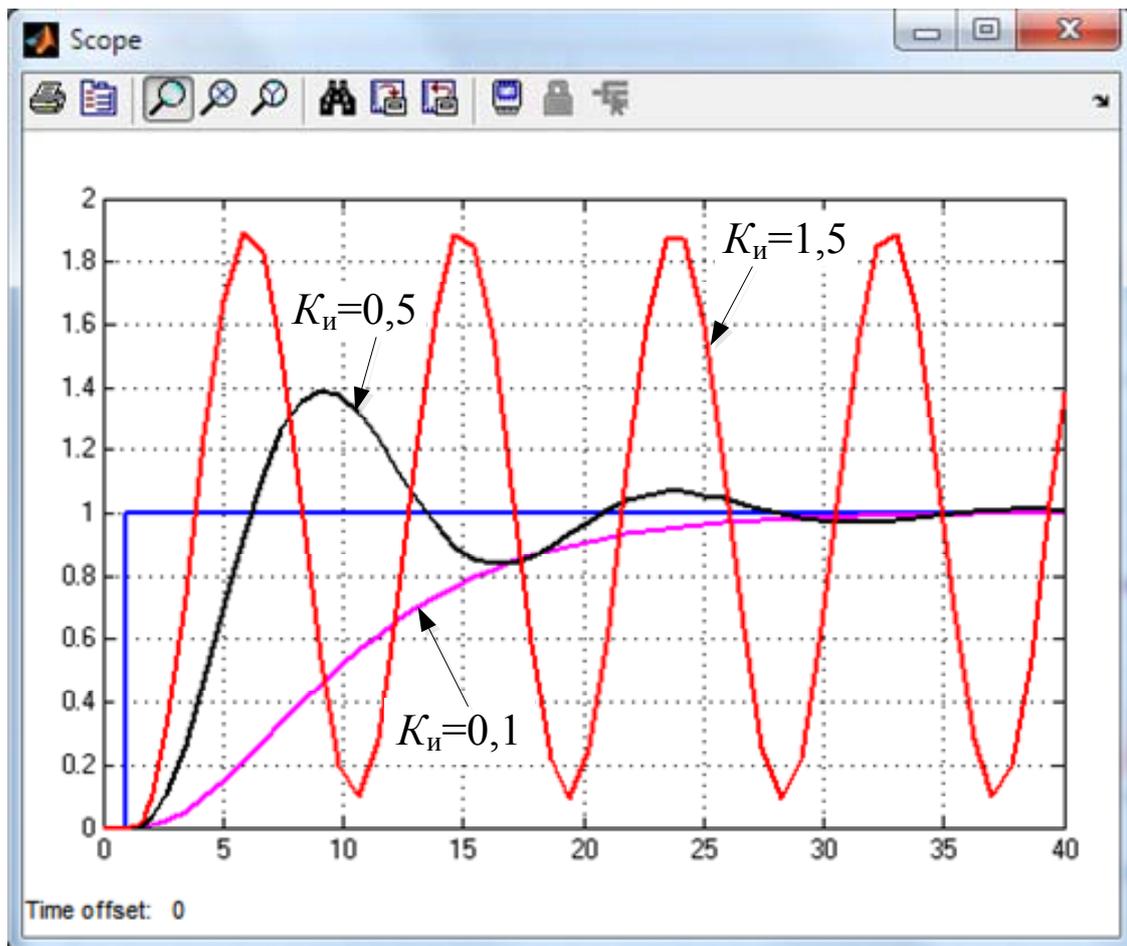


Рис. 5.33. Переходные характеристики системы при разных значениях  $K_I$

### 5.3.3. Дифференцирующий регулятор

Часто для улучшения поведения системы в переходном режиме в закон регулирования вводят производную от сигнала ошибки (5.6):

$$U_d(t) = K_d \cdot \frac{d\varepsilon(t)}{dt}. \quad (5.8)$$

Алгоритм (5.6) реализует так называемый дифференцирующий, или Д-регулятор, на выходе которого образуется управление  $U_d(t)$  пропорциональное скорости изменения ошибки  $\varepsilon(t)$ . Введение в состав системы Д-регулятора может значительно повысить ее быстродействие. Однако он имеет весьма существенный недостаток (рис. 5.34).

Дело в том, что Д-регулятор очень усиливает помеху  $\eta(t)$  измерения выходной величины  $y(t)$ , в результате чего управление  $U_d(t)$  будет сильно флуктуировать, вызывая флуктуации управляющего воздействия  $v_p(t)$  (а, следовательно, и  $v_\phi(t)$ ) и выходной величины  $y(t)$ . В результате система будет «трястись».

Объясним это явление. Пусть выходная: величина объекта на некотором временном интервале процесса регулирования возрастает с практически постоянной скоростью. Тогда ошибка  $\varepsilon(t)$  будет убывать практически с постоянной скоростью, и на выходе Д-регулятора должна была бы наблюдаться постоянная («спокойная») отрицательная величина.

Реально же  $y(t)$  всегда измеряется с помехой  $\eta(t)$ . Эта помеха может представлять собой случайный процесс с небольшой дисперсией (небольшими амплитудами) и практически не ухудшать точность измерения самой выходной величины  $y(t)$ . Но если она «высокочастотна» и меняет свои значения с большими скоростями, то на выходе Д-регулятора появится управление  $U_d(t)$  с большой дисперсией. Все сказанное иллюстрирует рис. 5.34.

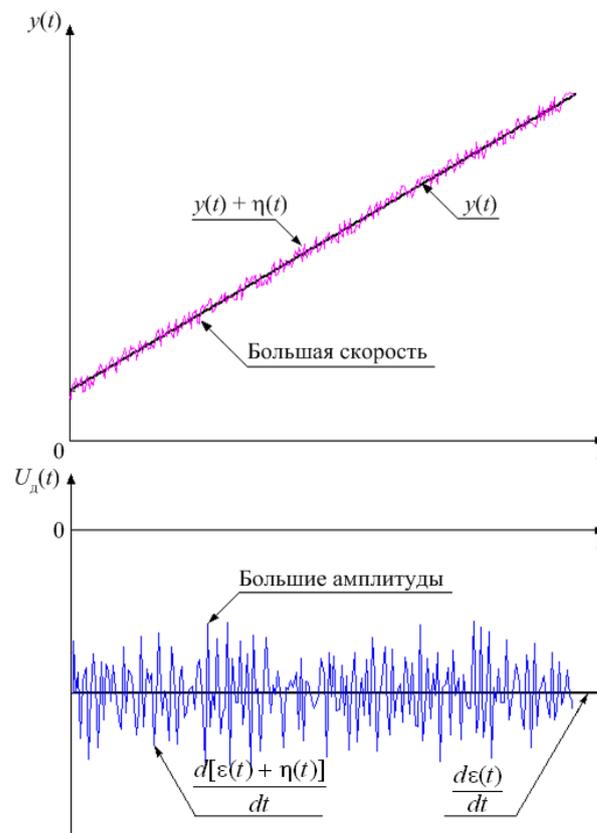


Рис. 5.34. Иллюстрация появления мощных флуктуаций на выходе идеального Д-регулятора

Отметим, что Д-регулятор отдельно не используется. Более эффективным является применение комбинированных регуляторов, которые позволяют объединить положительные свойства каждого из законов регулирования.

#### 5.4.4 ПИД-регуляторы

Самыми распространенными типами регуляторов в промышленной практике являются ПИД-регуляторы, изобретенные ещё в 1910 году [15]. Порядка 90-95% регуляторов, находящихся в настоящее время в эксплуатации, используют ПИД- и ПИ-алгоритмы.

Аббревиатура ПИД объясняется тем, что эти регуляторы содержат каналы, параллельно реализующие пропорциональный, интегрирующий и дифференцирующий законы регулирования. Структура системы с классическим ПИД-регулятором приведена на рис. 5.35. На входы всех каналов ПИД-регулятора подается ошибка  $\varepsilon(t) = x(t) - y(t)$ .

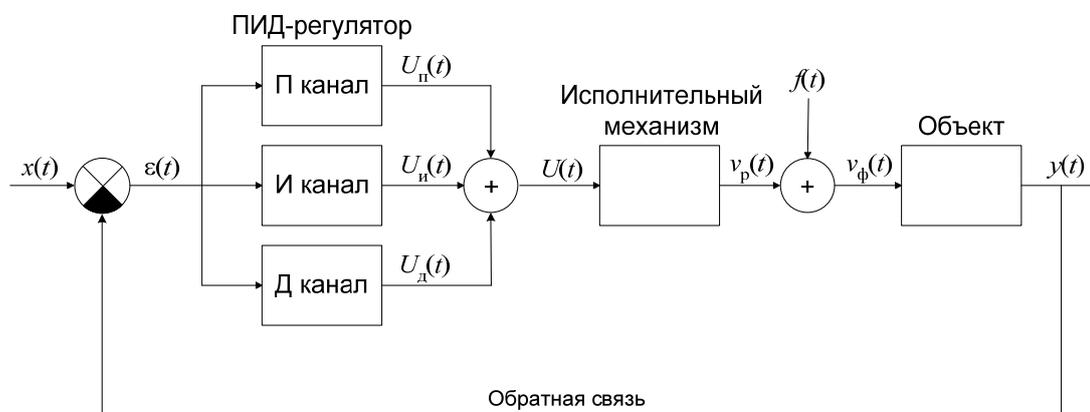


Рис. 5.35. Структура системы с классическим ПИД-регулятором

На рис. 5.35  $U_{п}(t)$ ,  $U_{и}(t)$  и  $U_{д}(t)$  – выходные величины пропорционального, интегрирующего и дифференцирующего каналов соответственно.

Таким образом, ПИД-регулятор реализует следующий закон регулирования:

$$U_{\text{ПИД}}(t) = K_{\text{П}}\varepsilon(t) + K_{\text{И}}\int_0^t \varepsilon(\tau)d\tau + K_{\text{Д}}\frac{d\varepsilon(t)}{dt}, \quad (5.9)$$

где  $K_{\Pi}$ ,  $K_{\text{И}}$  и  $K_{\text{Д}}$  – коэффициенты пропорциональности пропорционального, интегрирующего и дифференцирующего каналов.

Часто также встречаются и другие варианты записи уравнения (5.9):

$$U_{\text{ПИД}}(t) = K_{\Pi}\varepsilon(t) + \frac{1}{T_{\text{И}}} \int_0^t \varepsilon(\tau) d\tau + K_{\text{Д}} \frac{d\varepsilon(t)}{dt}, \quad (5.10)$$

где  $T_{\text{И}} = 1/K_{\text{И}}$  – постоянная времени интегрирующего канала.

$$U_{\text{ПИД}}(t) = K_{\Pi} \left( \varepsilon(t) + \frac{1}{T_{\text{И}}} \int_0^t \varepsilon(\tau) d\tau + \frac{1}{T_{\text{Д}}} \frac{d\varepsilon(t)}{dt} \right), \quad (5.11)$$

где  $T_{\text{И}} = K_{\Pi}/K_{\text{И}}$  – постоянная времени интегрирующего канала;  $T_{\text{Д}} = K_{\Pi}/K_{\text{Д}}$  – постоянная времени дифференцирующего канала.

Рассмотрим работу системы с ПИД-регулятором [4]. На рис. 5.36 приведены графики изменения всех координат некоторой системы управления при  $x(t) = 1(t)$ ,  $f(t) = 0$  и движении из состояния покоя.

Допустим, система регулирования находилась до некоторого момента времени в состоянии покоя. Примем этот момент за  $t = 0$ , а состояние системы за ноль отсчета по всем координатам. Пусть в момент  $t = 0$  произведена скачкообразная переменная желаемого режима:  $x(t) = A \cdot 1(t)$ ,  $A = \text{const}$ ,  $A > 0$ . При этом мгновенно возникает ошибка  $\varepsilon(0) = x(0) - y(0) = A - 0 = A$ . На выходе П-канала тут же появится управление  $U_{\text{П}}(0) = K_{\text{П}} \cdot \varepsilon(t) = K_{\text{П}} \cdot A$ , которое даст команду исполнительному механизму на приведение объекта в движение в направлении желаемого режима.

Управление  $U_{\text{И}}(0)$  на выходе интегрирующего канала будет равно нулю, так как интегратор за очень малое время еще «не успеет» что-либо наинтегрировать.

Управление  $U_{\text{Д}}(0)$  на выходе дифференцирующего канала мгновенно примет максимально возможное значение, поскольку  $d1(t)/dt = \delta(t)$ , и сразу же опять станет равным нулю. Такое ограниченное по величине и по времени воздействие не окажет на движение системы существенного влияния.

Некоторое время в силу своей инерционности ни исполнительный механизм, ни объект управления не будут проявлять

видимого движения, накапливая энергию. Пропорциональный канал все это начальное время будет определять управление  $U_{\text{п}}(t) \approx K_{\text{п}} \cdot A$ . На выходе дифференцирующего канала управления  $U_{\text{д}}(t)$  будет практически равно нулю.

Но вот исполнительный механизм, а за ним и объект управления придут в движение. Выходная величина  $y(t)$  объекта начнет меняться в направлении  $x(t) = A \cdot 1(t)$ . Ошибка  $\varepsilon(t) = x(t) - y(t)$  начнет уменьшаться. Управление  $U_{\text{п}}(t)$  тоже начнет уменьшаться. Управление  $U_{\text{и}}(t)$ , наоборот, будет увеличиваться, так как оно по-прежнему приблизительно определяется формулой  $U_{\text{и}}(t) = K_{\text{и}} \cdot A t$  (см. рис.5.36).

Управление  $U_{\text{д}}(t)$  станет отрицательным, так как  $\varepsilon(t) = x(t) - y(t)$  и при росте  $y(t)$  уменьшается, то есть имеет отрицательную производную.

Так продолжится и далее. Управление  $U_{\text{п}}(t)$  будет уменьшаться, так как уменьшается  $\varepsilon(t)$ . Управление  $U_{\text{и}}(t)$  будет возрастать с течением времени (см. рис. 5.36). Управление  $U_{\text{д}}(t)$  останется отрицательным и некоторое время будет увеличиваться по абсолютному значению.

Наличие отрицательного значения  $U_{\text{д}}(t)$  может вызвать недоумение. Затем оно нужно, ведь  $y(t)$  необходимо увеличивать до  $x(t) = A \cdot 1(t)$ . Здесь необходимо вспомнить о том, что исполнительный механизм и объект это динамическая группа, способная накапливать энергию и совершать собственное движение под влиянием этой энергии. Необходимо вспомнить и о том, что одна только выходная величина  $y(t)$  не определяет эту энергию. Что из того, что сейчас  $y(t)$  еще меньше, чем  $x(t) = A \cdot 1(t)$ ? Возможно, исполнительный механизм и объект уже накопили такую энергию движения, что  $y(t)$  скоро превысит значение  $A$  и придется заниматься обратной задачей – снижать  $y(t)$  до  $A$ ? Информация о  $\frac{d\varepsilon(t)}{dt}$ ,  $\left( \frac{d\varepsilon(t)}{dt} = -\frac{dy(t)}{dt} \right)$  и приоткрывает завесу над этой энергией. Отрицательное управление  $U_{\text{д}}(t)$  – это «разумный тормоз» движения неизменяемой части.

Таким образом, Д-регулятор следит за скоростью приближения ошибки управления (а, следовательно, и выходной величины) к установившемуся значению. Если эта скорость слишком большая, а ошибка маленькая, то в системе будет перерегулирование и для его

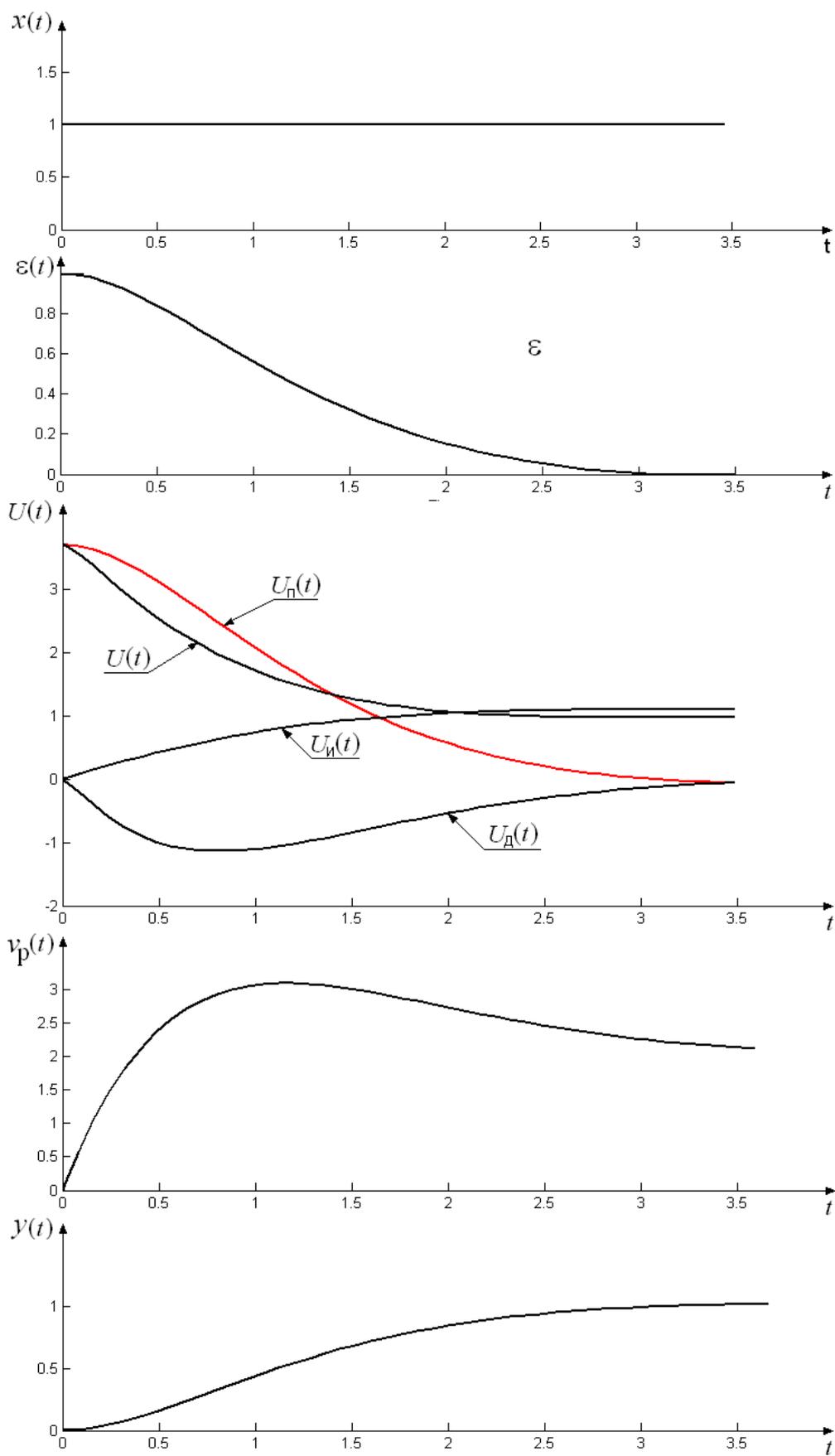


Рис. 5.36. Переходные процессы в системе с ПИД-регулятором

предотвращения уменьшается управляющее воздействие. Если скорость мала, а ошибка все еще достаточно велика, то выходная величина, возможно, никогда не достигнет установившегося значения, поэтому сигнал управления увеличивается.

При правильно выбранных коэффициентах  $K_p$ ,  $K_i$  и  $K_d$  кривая  $y(t)$  плавно подойдет к желаемому режиму и объект останется в нем. При этом  $U_p(t)$  будет равно нулю,  $U_d(t)$  тоже будет равно нулю, а  $U_i(t)$  будет таким, которое способно поддерживать режим  $y_{уст} = A$ . Это единственно возможный установившийся в системе режим, поскольку именно в нем  $\varepsilon(t) = 0$  и  $U_i(t)$  не понуждается к изменению.

#### 5.4.5. ПИ-регуляторы

ПИ-регулятор не имеет дифференцирующего канала (рис.5.37). Его уравнение описывается выражением:

$$U_{ПИ}(t) = K_P \varepsilon(t) + K_I \int_0^t \varepsilon(\tau) d\tau, \quad (5.12)$$

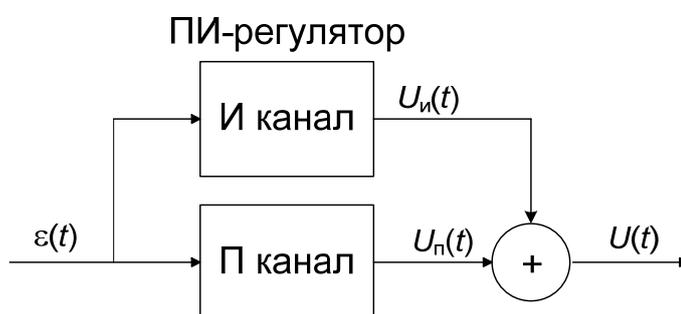


Рис. 5.37 Структура ПИ-регулятора

Теоретически возможности ПИ-регулятора ниже возможностей ПИД-регулятора. Для примера на рис. 5.38 показаны переходные функции систем с ПИ- и ПИД-регуляторами и одной и той же неизменяемой частью. Настройки ПИ- и ПИД-регуляторов выбраны таким образом, чтобы одновременно обеспечить минимальную длительность и монотонность переходных процессов.

Рис. 5.38 подтверждает теоретические преимущества ПИД-регулятора – он обеспечивает большее быстроедействие системы. Однако довольно часто встречаются случаи, когда появление даже минимального перерегулирования недопустимо.

Системы с ПИД-регулятором практически всегда будут иметь перерегулирование, поэтому в таких случаях необходимо использовать ПИ-регулятор. Имеется, к тому же, еще одно обстоятельство, которое препятствует безусловному использованию только ПИД-регуляторов.

Выше был описан эффект усиления Д-регулятором высокочастотного шума  $\eta(t)$ , сопровождающего процесс измерения выходной величины. Для устранения этой проблемы на практике применяются различные схемные решения, некоторые из которых будут описаны в п. 5.8. Однако, при значительной помехе  $\eta(t)$  измерений может оказаться так, что лучше всего перейти на использование ПИ-регулятора, если это позволяют условия устойчивости замкнутой системы. По многим оценкам, именно ПИ-регуляторы чаще всего используются на практике.

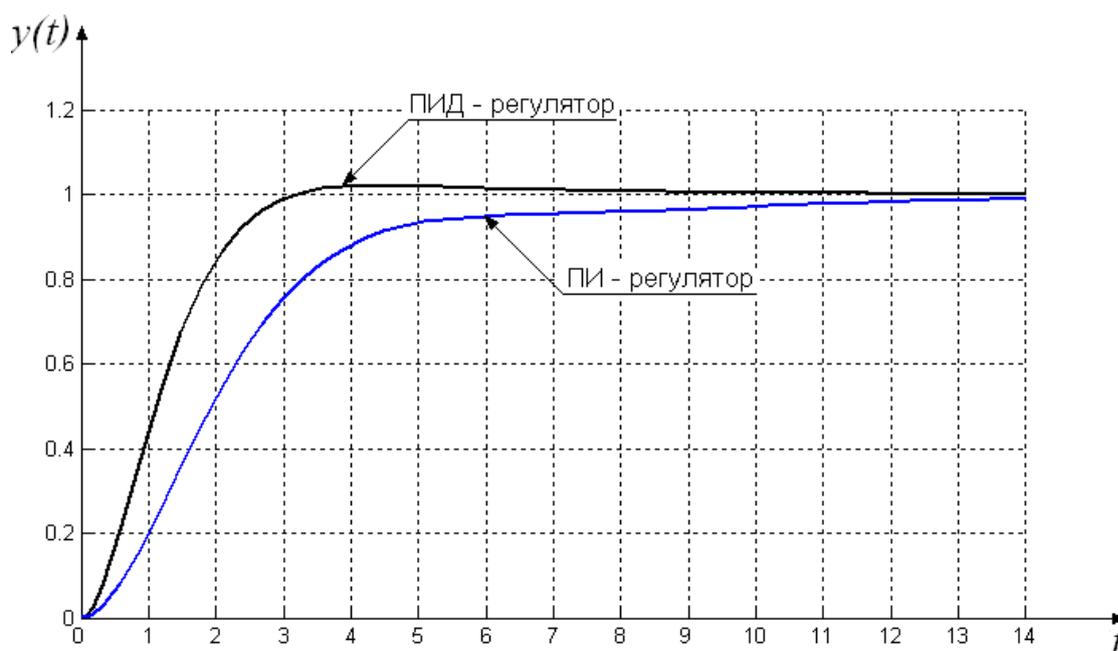


Рис. 5.38. Переходные функции систем с ПИ- и ПИД-регуляторами

#### 5.4.6. ПД-регуляторы

ПД-регулятор представляет собой параллельное соединение пропорционального и дифференцирующего каналов (рис. 5.39), следовательно, уравнение ПД-регулятора имеет вид:

$$U_{\text{ПД}}(t) = K_{\text{П}}\varepsilon(t) + K_{\text{Д}} \frac{d\varepsilon(t)}{dt}, \quad (5.13)$$

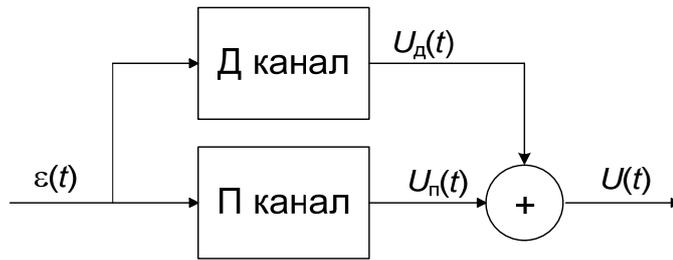


Рис. 5.39 Структура ПД-регулятора

В сравнении с П-регуляторами, ПД-регуляторы обеспечивают большее быстродействие и меньшее перерегулирование. Однако ПД-регуляторы не подходят для управления изначально неустойчивыми объектами. Кроме того, остается проблема с усилением высокочастотного шума.

Эти регуляторы целесообразно применять с объектами, обладающими интегрирующими свойствами, в которых обеспечивается нулевая установившаяся ошибка (например, при управлении углом поворота двигателя постоянного тока).

В большинстве случаев процедура синтеза системы управления заключается в выборе типа регулятора и определении (настройке) его коэффициентов. Выбор типа регулятора осуществляется на основе требований, предъявляемых к системе и исходя из их свойств, рассмотренных выше.

## 5.7. Достоинства и недостатки ПИ- и ПИД-регуляторов

Почему же ПИ- и ПИД-регуляторы так распространены на практике?

Во-первых, эти регуляторы достаточно легко реализуются как программно в контроллере, так и аппаратно, если они являются электрическими, механическими, пневматическими или гидравлическими устройствами.

Во-вторых, они очень часто обеспечивают хорошие свойства системы управления. Но это утверждение требует более детального обсуждения.

Прежде всего, вспомним требования, которые обычно

предъявляются к системе управления.

1. В системе должны быть возможными только такие установившиеся процессы, в которых рассогласование  $\varepsilon_{уст} = x_{уст} - y_{уст}$  не превышает заданной предельной величины  $\varepsilon_{пред}$ , чаще всего равной нулю.

2. Возможные установившиеся режимы должны быть устойчивы. В линейных системах это требование может формулироваться как требование устойчивости системы.

3. Переходные процессы должны иметь по возможности минимальную длительность и быть достаточно монотонными.

В соответствии с этими требованиями и проанализируем свойства ПИ- и ПИД-регуляторов.

1. Для большинства (конечно, не для всех) промышленных установок воздействие  $x(t)$  меняется скачками через достаточно длительные интервалы времени, а  $f(t)$  представляет собой неопределенный процесс.

Реальное требование к системе управления в такой ситуации заключается в том, чтобы она стремилась *в среднем* (!) без ошибки воспроизводить желаемое движение  $x(t)$ . Это требование будет выполняться, если система будет стремиться вывести объект в режим  $y_{уст} = A$  при любом  $A$ , подавляя при этом любое среднее значение  $B$  действующего возмущения.

Формализовано это означает, что система должна быть устроена так, чтобы при  $x(t) = A \cdot 1(t)$  и  $f(t) = B \cdot 1(t)$  в ней был возможен только один установившийся режим с постоянными значениями координат:  $\varepsilon_{уст} = 0$ ,  $y_{уст} = A$ . Говорят, что система должна обладать *астатизмом* 1-го порядка по отношению к  $x(t)$  и  $f(t)$ .

ПИ- и ПИД-регуляторы обеспечивают выполнение этого важнейшего в промышленной практике требования. Действительно, до тех пор, пока  $\varepsilon(t)$  не станет равной нулю, интегрирующий канал будет менять (увеличивать или уменьшать) управление  $U_{и}(t)$  и приводить в движение объект, не давая ему стабилизировать значения выходной величины. Система с ПИ- или ПИД-регулятором может находиться в стабильном установившемся состоянии только при  $\varepsilon_{уст} = 0$ , то есть при  $y_{уст} = A$  при любом  $B$ .

2. Большинство промышленных установок и технологических аппаратов хорошо моделируются дифференциальными уравнениями

не выше 2-го порядка, а исполнительные механизмы – дифференциальными уравнениями 1-го порядка. Для таких неизменяемых частей правильным выбором коэффициентов ПИД – регулятора  $K_p$ ,  $K_i$  и  $K_d$  почти всегда можно обеспечить устойчивость возможных установившихся режимов. ПИ-регулятор имеет в этом смысле более ограниченную сферу применения.

3. Правильным выбором коэффициентов ПИ- и ПИД-регуляторов можно получить достаточно неплохие результаты и по качеству переходных процессов.

4. ПИ и ПИД-регуляторы позволяют производить настройку своих коэффициентов непосредственно в процессе эксплуатации системы без использования надежных моделей исполнительного механизма и объекта, разработка которых всегда представляет собой непростую задачу.

5. Кроме того, эти регуляторы имеют низкую стоимость.

Указанные свойства ПИ и ПИД-регуляторов и обусловили их широкое распространение в промышленности.

К принципиальным недостаткам рассматриваемых регуляторов следует отнести то, что они борются с возмущением  $f(t)$  «вслепую», не имея информации о его значении или поведении во времени. Поэтому эти регуляторы не инвариантны к внешним условиям работы. Регулятор, настроенный (путем выбора коэффициентов  $K_p$ ,  $K_i$  и  $K_d$ ) на хорошую отработку задающего воздействия  $x(t) = A \cdot 1(t)$  при  $f(t) = 0$  не обеспечивает оптимального подавления возмущения  $f(t) = B \cdot 1(t)$  при  $x(t) = 0$ . И вообще, ПИ- или ПИД – регулятор, настроенный на одно сочетание  $(x_1(t), f_1(t))$ , не является лучшим при другой паре  $(x_2(t), f_2(t))$ . Система с ПИ- или ПИД- регулятором почти все время работает неплохо, но и не лучшим образом.

Поскольку  $f(t)$  – это практически всегда неопределенный процесс, то проблема выбора лучших в среднем (только в среднем) коэффициентов ПИ или ПИД-регулятора так и остается обычно до конца нерешенной.

К существенным практическим недостаткам ПИД – регулятора необходимо отнести и то, что его Д-канал очень усиливает помеху  $\eta(t)$  измерения выходной величины  $y(t)$ , что вызывает сильную флуктуацию общего управления  $U(t)$  и система начинает «трястись» (см. рис. 5.34).

Еще одна проблема, связанная с использованием Д-канала по

схеме рис. 5.35. При ступенчатом изменении задающего воздействия  $x(t)$ , управление  $U_d(0)$  на выходе дифференцирующего канала будет иметь достаточно большое значение (т.к.  $d1(t)/dt = \delta(t)$ ). В результате исполнительный механизм подвергается «удару» (рис.5.40).

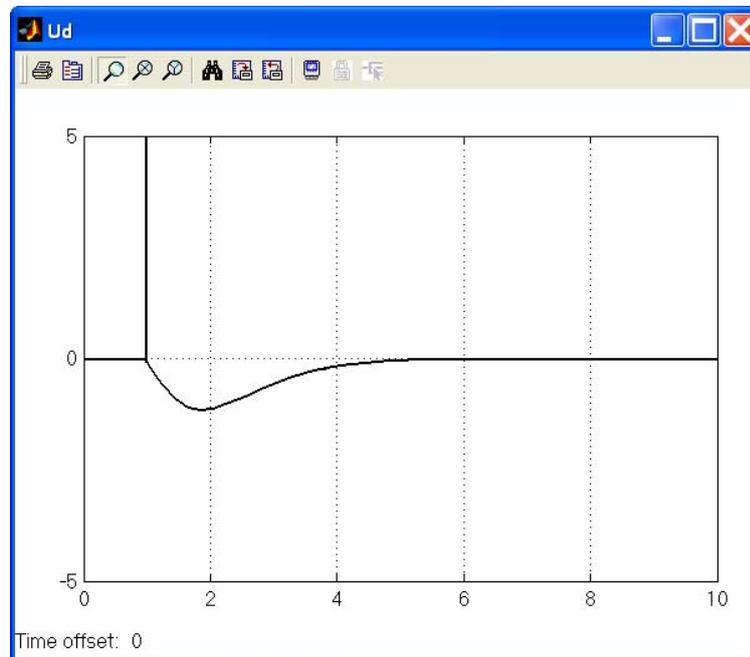


Рис. 5.40. Выходной сигнал  $U_d(t)$  дифференцирующего канала ПИД-регулятора, собранного по схеме рис. 5.35. Хорошо виден скачок в момент ступенчатого изменения задающего воздействия (при  $t = 1$  с)

В связи с этим на практике используют несколько иные схемы ПИД-регуляторов, отличные от представленной на рис. 5.35. Рассмотрим их.

## 5.8. Практическая реализация ПИД-регулятора

Ранее мы предполагали, что ПИД-регулятор имеет структуру, изображенную на рис. 5.35. Однако, в силу проблем, связанных с реализацией канала производной, в таком виде этот регулятор применяется очень редко. Вместо этого используются другие схемные решения, которые, тем не менее, должны быть математически эквивалентны классической структуре ПИД-регулятора.

Наиболее часто для фильтрации высокочастотных шумов измерения  $\eta(t)$ , усиленных операцией дифференцирования, в Д-канале ставится фильтр низких частот (устройство, пропускающее только колебания низких частот). В качестве такого фильтра чаще всего используется апериодическое звено 1-го порядка и тогда Д-канал ПИД-регулятора описывается уравнением:

$$U_d(s) = \frac{N}{1 + N \frac{1}{s}} E(s) = \frac{Ns}{s + N} E(s), \quad (5.14)$$

где  $N$  – коэффициент фильтра, задающий его граничную частоту;  $s$  – комплексная переменная в преобразовании Лапласа;  $E(s)$  и  $U_d(s)$  – преобразованные по Лапласу ошибка управления и выход Д-канала соответственно.

Чем меньше значение коэффициента фильтра  $N$  в (5.14), тем более узок диапазон пропускаемых частот, но тем сильнее систематически искажается истинная производная  $d\varepsilon(t)/dt$ . При значительной помехе  $\eta(t)$  может оказаться так, что лучше всего перейти на использование ПИ-регулятора, если это позволяют условия устойчивости замкнутой системы.

Для реализации ПИД-регулятора часто используется структура, изображенная на рис. 5.41, в которой дифференцируется только сигнал обратной связи. Поскольку этот сигнал обычно изменяется относительно медленно, то на вход объекта будет поступать не такое большое воздействие, как в системе на рис. 5.35 и при ступенчатом изменении входного сигнала объект не будет подвергнут «удару».

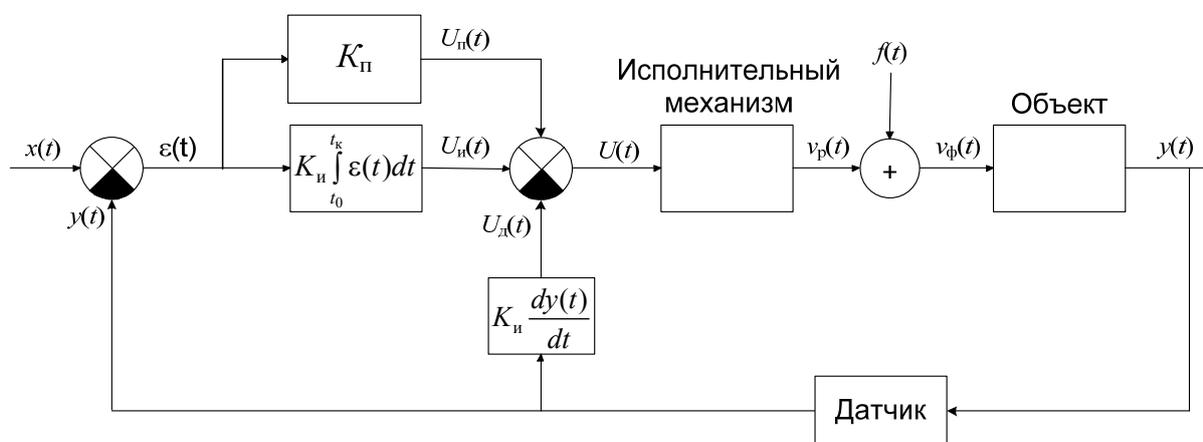


Рис. 5.41. ПИД – регулятор с дифференцированием выхода системы

Если кроме измерения выходного сигнала системы можно измерить и скорость его изменения, то регулятор лучше всего реализовать с помощью схемы на рис. 5.42. При данном способе реализации дифференцирование вообще отсутствует, хотя общее управление  $U(t)$  будет тем же, что и при схеме на рис. 5.35. При этом не происходит дифференцирования сигнала и, следовательно, не возникают проблемы, связанные с наличием высокочастотного шума.

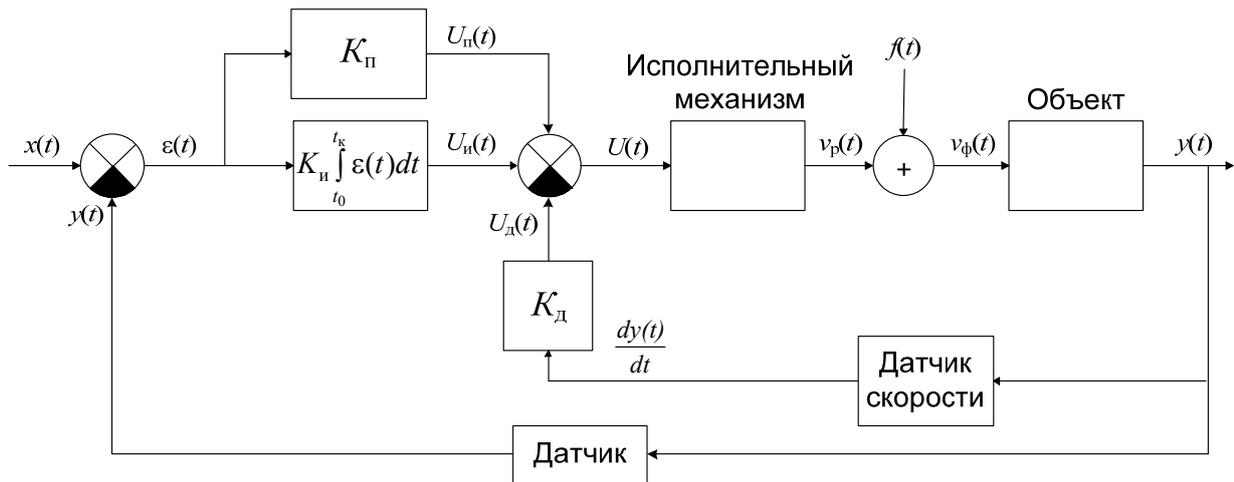


Рис. 5.42. ПИД – регулятор с обратной связью по скорости

## 5.9. Пример синтеза ПИД-регулятора в Simulink

В качестве объекта при синтезе ПИД-регулятора будем рассматривать подвеску автобуса, модель которой была рассмотрена в главе 2. Как было показано, рассматриваемая подвеска не очень хорошо реагирует на внешние возмущения, в качестве которых могут служить, например, неровности дороги (рис. 2.52, 2.53).

Мы хотим разработать такой регулятор, который бы при внешних возмущениях ( $W$ ), моделируемых ступенчатой функцией, обеспечивал бы время регулирования выходной величины ( $X_1-X_2$ ) менее 5 секунд и перерегулирование не более 5%. Например, при наезде автобуса на бордюр высотой в 10 см, кузов должен колебаться с амплитудой не более  $\pm 5$  мм и колебания должны прекратиться в течение 5 секунд [9].

В качестве регулятора будем использовать ПИД-регулятор. В принципе, мы можем собрать модель ПИД-регулятора по структуре

рис. 5.35 при помощи стандартных блоков Simulink: Gain, Integrator и Derivative. Модель такого ПИД-регулятора представлена на рис. 5.43. Оптимизацию параметров регулятора можно выполнить способом, описанным при рассмотрении П-регулятора.

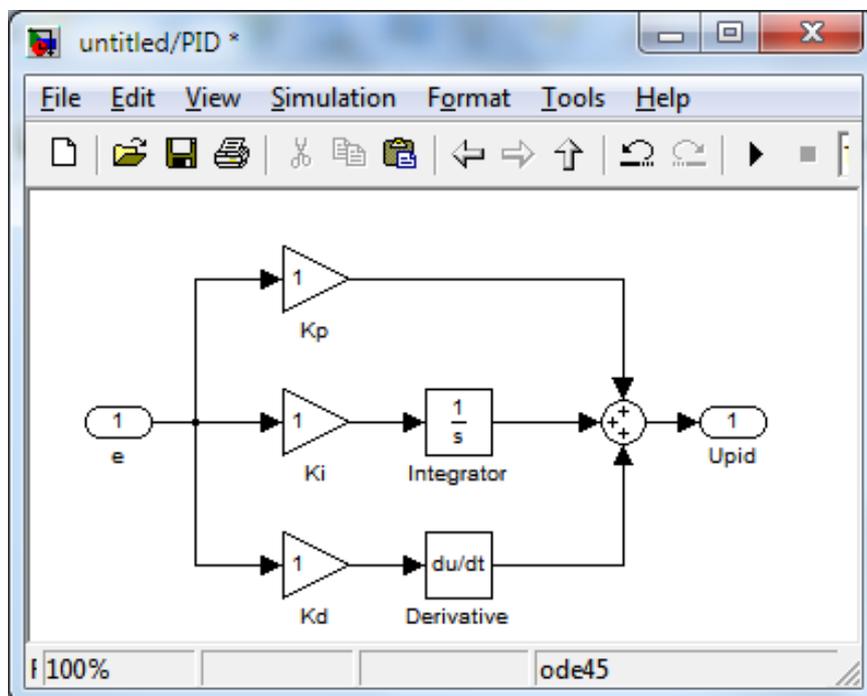


Рис. 5.43. ПИД – регулятор составленный из стандартных блоков Simulink

Однако в Simulink 7.8 (R2011b) имеется отдельный блок PID Controller (рис. 5.44), который расположен в библиотеке Continuous.

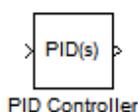


Рис. 5.44. Блок PID Controller

При помощи этого блока пользователь имеет возможность автоматического определения оптимальных значений коэффициентов ПИД-регулятора при помощи удобного графического пользовательского интерфейса (GUI)<sup>1</sup>.

<sup>1</sup> Требуется установка пакета Simulink Control Design

Алгоритм автоматической настройки ПИД-регулятора позволяет определить такие значения его коэффициентов, которые обеспечивают баланс между производительностью (временем регулирования, шириной полосы пропускания) и запасами устойчивости (по умолчанию алгоритм обеспечивает запас по фазе в 60 градусов).

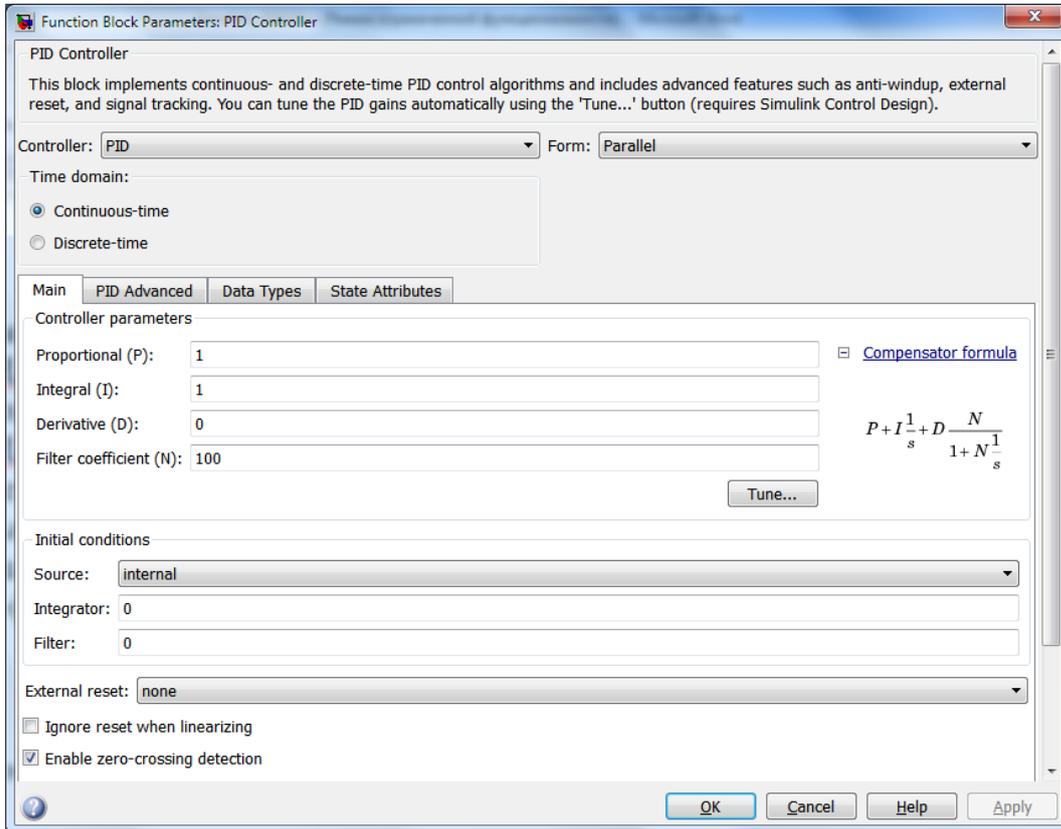


Рис. 5.45. Окно параметров блока PID Controller

Блок PID Controller имеет следующие параметры (рис. 5.45).

**Controller** – в раскрывающемся списке выбирается тип регулятора: **PID**, **PI**, **PD**, **P** или **I** (рис. 5.46).

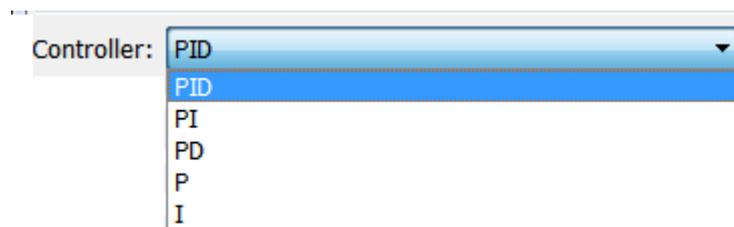


Рис. 5.46. Выбор типа регулятора

**Form** - форма модели регулятора (параллельная или идеальная).

**Time domain** – здесь указывается то, что является регулятор непрерывным (**continuous**) или дискретным (**discrete**).

На основной вкладке **Main** доступны следующие параметры.

**Controller parameters** – параметры регулятора, т.е. коэффициенты пропорционального (**P**), интегрирующего (**I**) и дифференцирующего (**D**) каналов, а также коэффициент фильтра (**N**). При выборе другого типа регулятора, список параметров соответствующим образом меняется.

**Initial conditions** – в этой группе параметров блока задается информация о начальных условиях.

- В списке **Source** (источник) указывается источник сведений о начальных условиях интегратора и фильтра: **internal** (внутренний) или **external** (внешний).

Если выбран внутренний (**internal**) источник, то значения начальных условий интегратора и фильтра задаются в соответствующих полях **Integrator** и **Filter**. При выборе внешнего источника (**external**) вид блока PID Controller изменяется, к нему добавляется два дополнительных входных порта для задания внешних условий.

**External reset** (Внешний сброс) – в этом меню задается событие, при котором сбрасываются выходы интегратора и фильтра к начальным условиям, указанным полях **Integrator Initial condition** и **Filter Initial condition**. Раскрывающийся список содержит пять пунктов, при выборе которых (кроме пункта **none**) у блока появляется дополнительный вход для сигнала сброса:

– **none** (нет) - сброс не выполняется (установлен по умолчанию);

– **rising** (возрастание) – выходной сигнал интегратора сбрасывается в начальное значение, когда при увеличении управляющего сигнала (сигнала сброса) происходит пересечение нулевого значения;

– **falling** (убывание) - выходной сигнал интегратора сбрасывается в начальное значение, когда при уменьшении управляющего сигнала происходит пересечение нулевого значения;

– **either** (любой) – сочетание последних двух пунктов: выход интегратора сбрасывается до начального значения, если управляющий сигнал пересекает нуль как сверху, так и снизу;

– **level** (уровень) - сброс выполняется если сигнал на управляющем входе становится неравным нулю.

При установленном флажке напротив позиции **Ignore reset when linearizing** Simulink игнорирует состояния, соответствующие механизму сброса.

Если установлен флажок напротив позиции **Enable zero crossing detection** (Разрешить определение пересечения нуля), то определяются моменты пересечения выходным сигналом регулятора нулевого значения при любом из следующих событий: сброс, вход или выход из состояния верхнего насыщения, вход или выход из состояния нижнего насыщения.

Основным параметром вкладки **PID Advanced** является параметр **Limit output** (ограничение выходного сигнала). При установке флажка напротив этой позиции, выходной сигнал регулятора будет ограничен сверху и снизу значениями, указанными соответственно в полях **Upper saturation limit** и **Lower saturation limit**. Кроме того, на изображении блока PID Controller появляется изображение статической характеристики типа «насыщение» (рис.5.47).

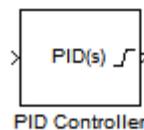


Рис. 5.47. Изображение блока PID Controller при активированном параметре **Limit output**

На вкладке **Data Types** указывается тип данных выходного сигнала, например, **double** - число двойной точности, **uint8** - целые неотрицательные числа в диапазоне [0, 255] и т.д.

На вкладке **State Attributes** можно задать имена состояниям интегратора и фильтра.

Рассмотрим теперь процедуру автоматического определения оптимальных значений коэффициентов ПИД-регулятора для управления подвеской автобуса.

- Откроем созданную ранее модель подвески (рис.3.8) и добавим в неё блок Sum.

- Обозначим входные порты блока Sum как «+».

- Соединим первый блок Step с положительным входом блока Sum.

- Соединим выходной порт блока Sum с входом подсистемы Bus Suspension.

- Сделаем ответвление от выходной линии подсистемы Bus Suspension и соединим его со вторым (отрицательным) входом блока Sum создав, тем самым, обратную связь.

- Переименуем указанный блок Step на «r» и присвоим всем параметрам этого блока значения «0» (мы будем поддерживать значение  $x_1 - x_2$  на постоянном исходном уровне).

- Установим значение параметра **Step time** блока Step с именем «w» на «0», а значение **Final Value** на «-0.1» (будем моделировать ситуацию, когда автобус попадает в яму глубиной 10 см).

- Добавим блок PID Controller из библиотеки Continuous между блоками Sum и подсистемой Bus Suspension.

- Обозначим линию связи между блоками Sum и PID Controller как «e» (ошибка), а линию между блоками PID Controller и Bus Suspension как «Upid» (управление ПИД-регулятора).

Если все сделано правильно, то Ваша модель должна выглядеть так, как показано на рис. 5.48.

Перейдем к настройке блока PID Controller. Откроем окно его параметров двойным щелчком мыши.

Параметры на вкладке **Main** оставим без изменений, т.е. будем использовать непрерывный (**Continuous-time**) ПИД-регулятор с параллельной формой модели.

Перейдем на вкладку **PID Advanced**. Вспомним процессы, протекающие в пневмоподвеске автобуса. Если колёса автобуса проваливаются в ямку, то пневмобаллоны его подвески растягиваются, и давление воздуха в них уменьшается. Для стабилизации давления воздух из ресивера подается в баллоны. Поскольку давление воздуха в ресивере ограничено (около 0,8 МПа), то необходимо ограничить сверху и выходной сигнал ПИД-регулятора.

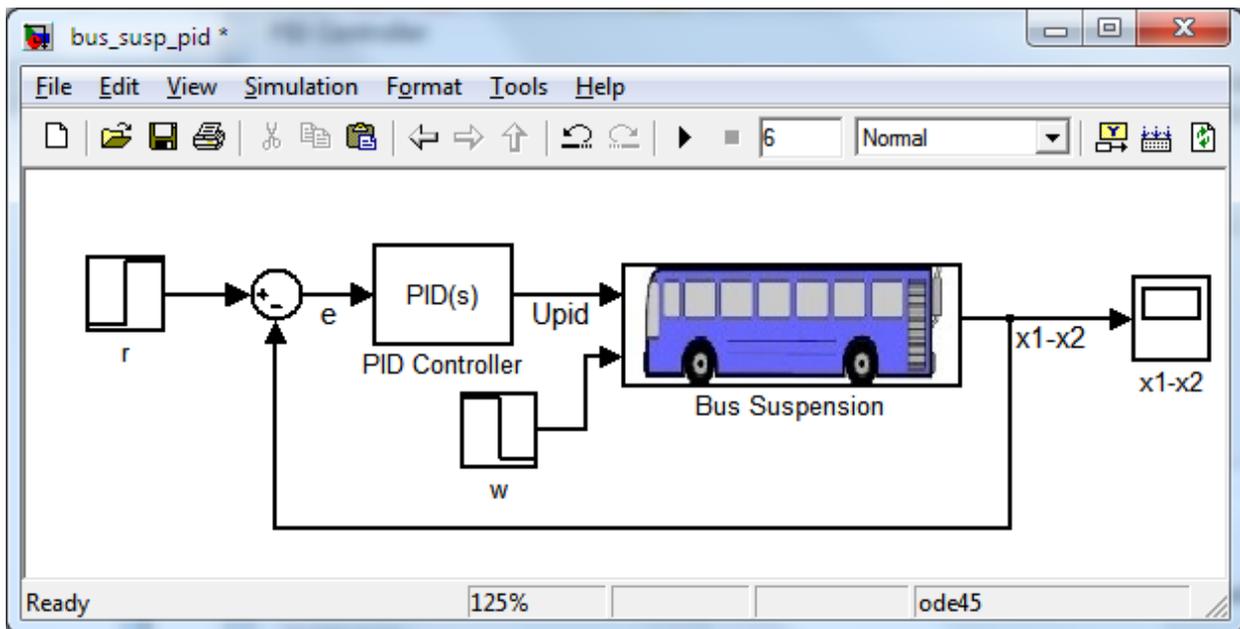


Рис. 5.48. Вид модели пневмоподвески автобуса с блоком PID Controller

При сжатии баллона «лишний» воздух выпускается из него в атмосферу. Разумеется, весь воздух выйти не может (как минимум, в баллоне установится атмосферное давление около 0,1 МПа), поэтому необходимо ограничить выходной сигнал регулятора и снизу.

Для задания ограничений выходного сигнала регулятора на вкладке **PID Advanced**:

- установим флажок напротив позиции **Limit output**;
- в поле **Upper saturation limit** укажем  $0.8 \cdot 10^6$ ;
- в поле **Lower saturation limit** укажем  $0.1 \cdot 10^6$  (рис. 5.49).

После применения этих изменений нажмем кнопки **Apply**, на пиктограмме блока PID Controller появится изображение статической характеристики типа «насыщение» (рис.5.47).

Вернемся теперь на вкладку **Main** и нажмем кнопку **Tune** для начала подбора настроек ПИД-регулятора. После выполнения линеаризации системы на экране появится окно **GUI PID Tuner** (рис. 5.50). В этом окне видны две реакции системы на единичную ступенчатую функцию: серая – с настройками ПИД-регулятора, принятым и по умолчанию ( $P = 1$ ,  $I = 1$ ,  $D = 0$  и  $N = 100$ ), а синяя – с подобранными настройками регулятора. Чтобы применить эти настройки, нажмем кнопку **Apply** (рис.5.50).

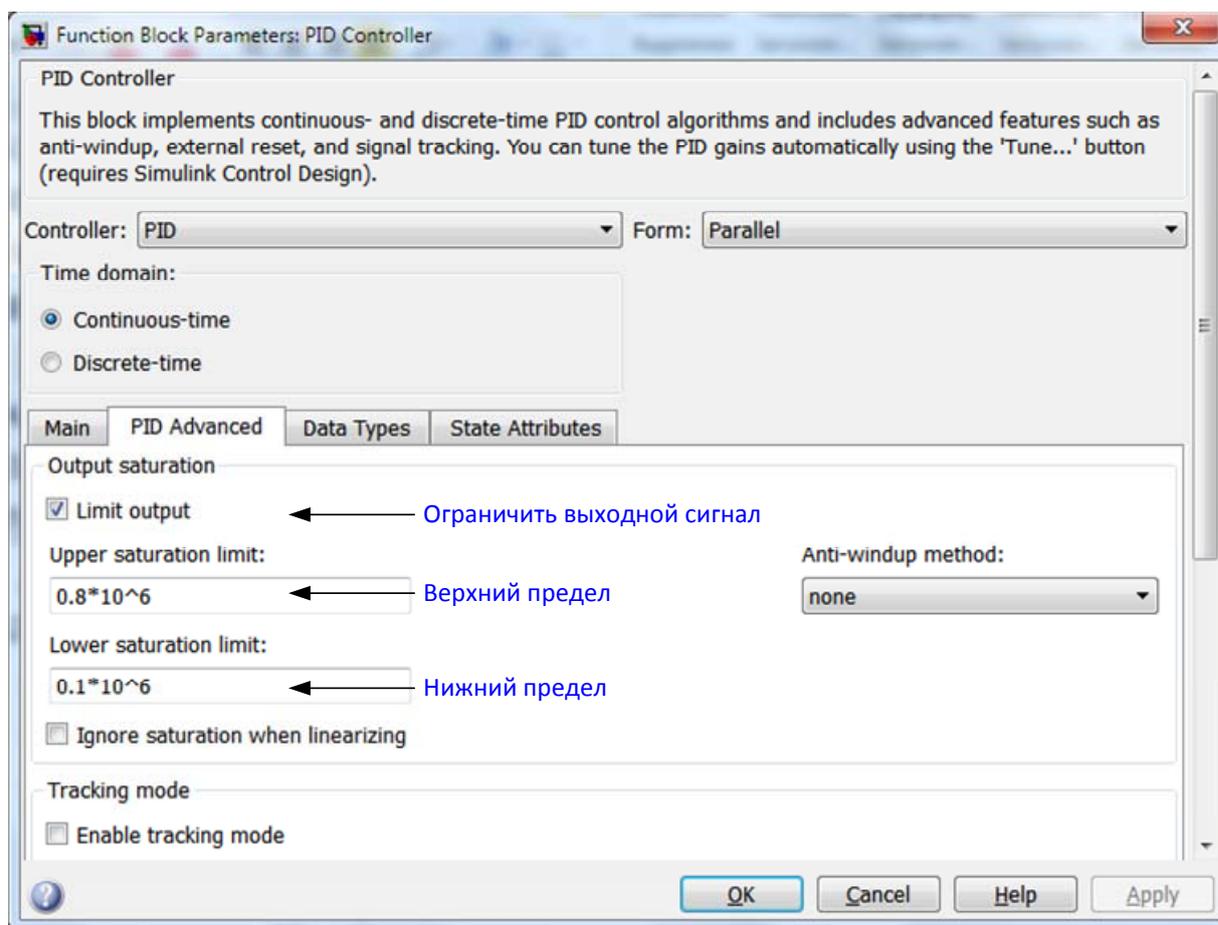


Рис. 5.49. Окно настройки блока PID Controller

Просмотреть предлагаемые Simulink параметры регулятора, а также численные значения показателей качества можно, нажав стрелку напротив позиции **Show parameters** (рис. 5.51).

Как видим, (рис. 5.52) предлагаемые параметры обеспечивают чрезвычайно высокое быстродействие, которое вряд ли удастся достичь в реальной системе, при достаточно большом перерегулировании в 10,6%.

Однако, не следует забывать, что мы рассматриваем *линеаризованную*, т.е. упрощенную модель, кроме того, реальному автобусу не придется преодолевать препятствия высотой (или глубиной) в 1 м. Единичная ступенчатая функция является лишь тестовым сигналом при исследовании линейных систем.

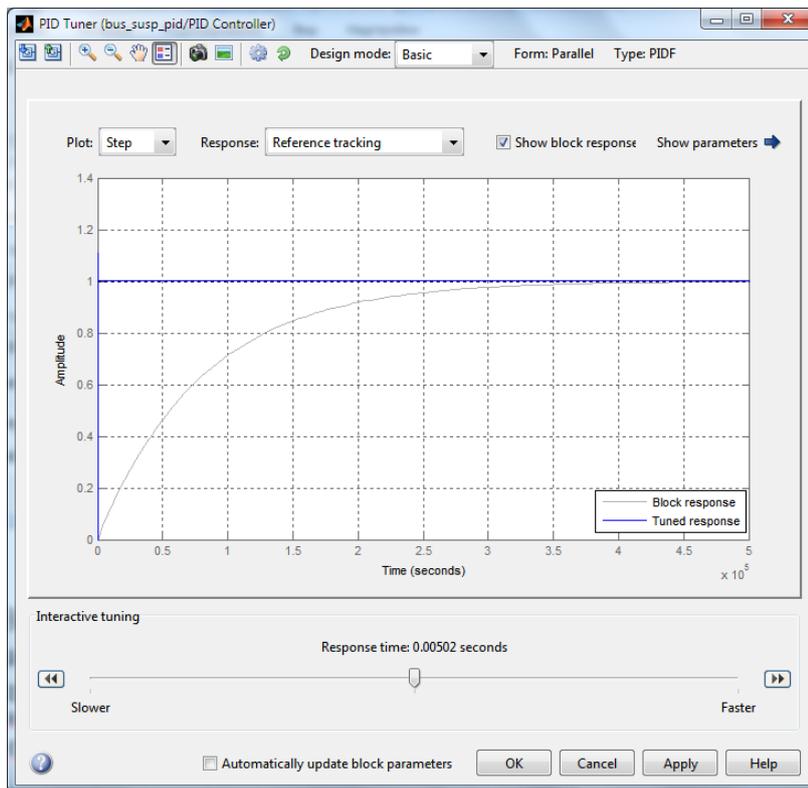


Рис. 5.50. Окно GUI **PID Tuner**, демонстрирующую реакции системы на единичную ступенчатую функцию

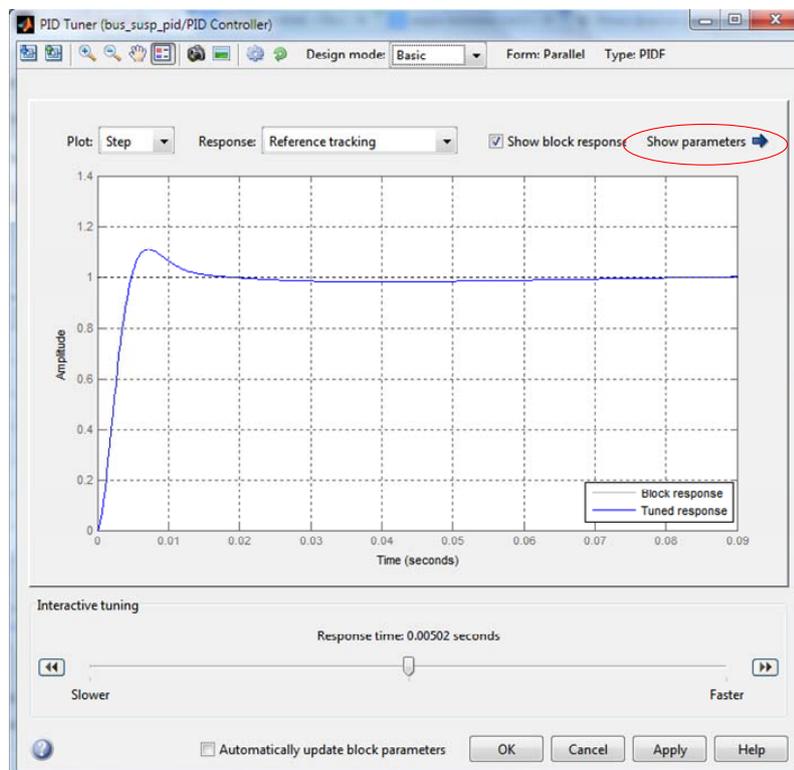


Рис. 5.51. Окно **PID Tuner** после применения новых параметров регулятора

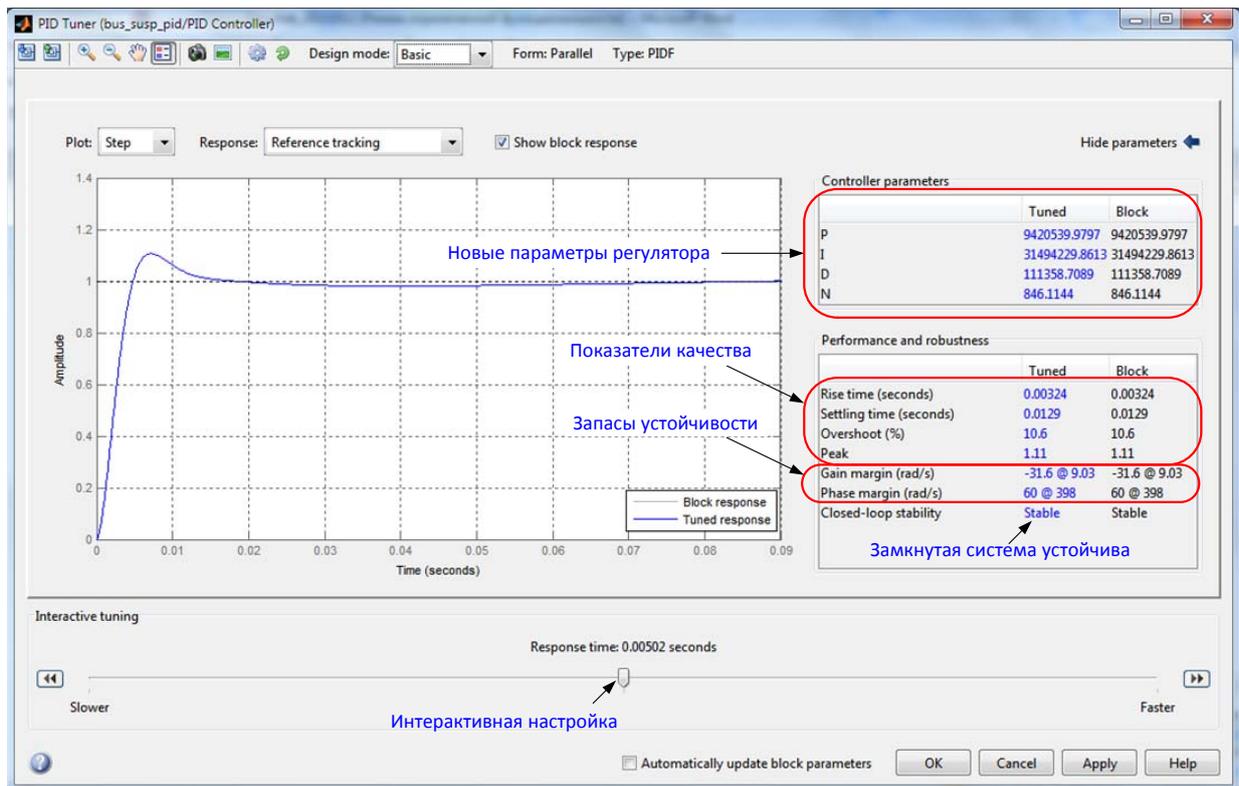


Рис. 5.52. Окно **PID Tuner** с численными данными показателей качества

Посмотрим, как наша подвеска с автоматически настроенным ПИД регулятором справиться с ямой, глубиной 10 см.

- Нажмем в окне **PID Tuner** кнопку **OK** и перейдем в окно модели.

- Проверим, применились ли новые значения параметров ПИД регулятора. Для этого откроем окно параметров блока PID Controller (рис. 5.53).

- Если все в порядке, то изменим время моделирования в окне **Simulation stop time** панели инструментов на 3 с и запустим моделирование, нажав кнопку **Start simulation** или используя сочетание клавиш **<Ctrl+T>**.

- Когда моделирование закончится, дважды щелкнем на блоке Score и нажмем кнопку **Autoscale** в окне графика. Реакция автобуса на возмущающее воздействие в -0,1 м показана на рис. 5.54.

Реакция, приведенная на рис. 5.54 соответствует предъявленным выше требованиям к системе управления: максимальное перерегулирование составляет 3,5 %, а время регулирования около 1,5 с. (при требуемых 5% и 5 с).

Таким образом, можно считать, что ПИД-регулятор обеспечивает хорошее качество переходного процесса.

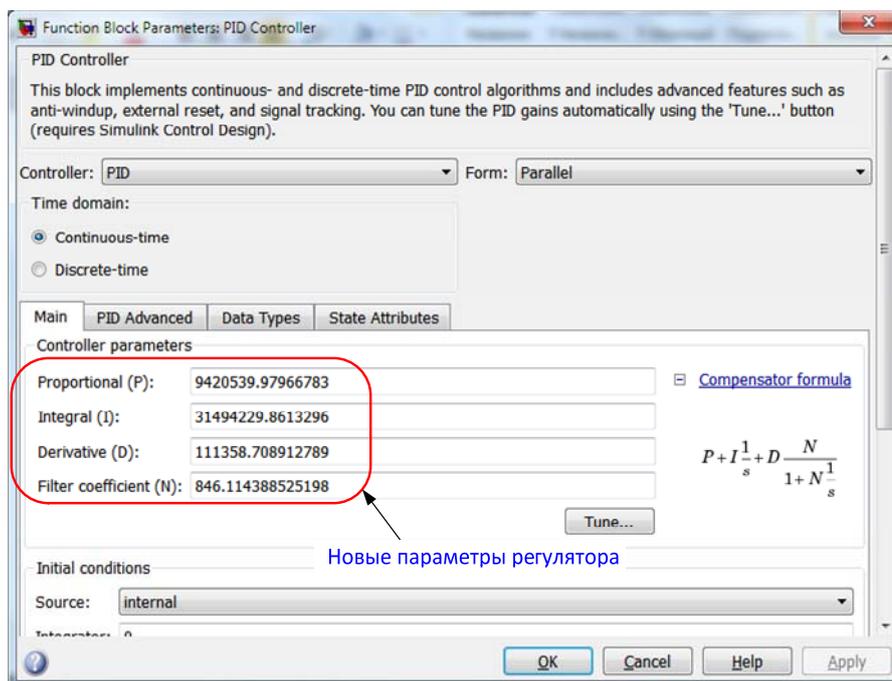


Рис. 5.53. Окно **PID Tuner** с новыми параметрами ПИД-регулятора

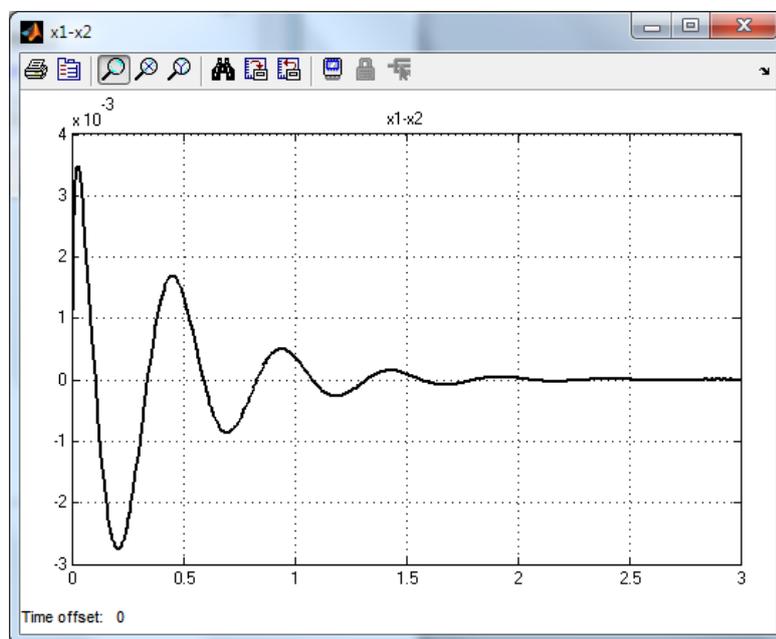


Рис. 5.54. Результаты расчёта реакции автобуса на возмущающее воздействие в -0,1 с настроенным ПИД-регулятором

## 6 МОДЕЛИРОВАНИЕ НЕЛИНЕЙНЫХ СИСТЕМ

### 6.1 Особенности нелинейных систем

Большинство реальных систем автоматического управления описываются нелинейными уравнениями. Их линеаризация и рассмотрение как линейных упрощает расчеты, но делает их приближенными. Иногда для достижения требуемой точности нельзя пренебрегать нелинейными свойствами системы. В этих случаях в структурные схемы наряду с линейными моделями включают модели различных нелинейных элементов, что позволяет повысить точность моделирования реальных систем.

Все нелинейности можно разделить на два основных типа: статические и динамические. Наиболее простыми являются статические нелинейности, в которых выходной сигнал зависит от текущего значения входного сигнала, но только нелинейным образом. Статические нелинейности можно описать при помощи алгебраических или трансцендентных (имеющих показательные, логарифмические или тригонометрические функции) уравнений

$$y(t) = f(x(t)), \quad (6.1)$$

где  $f$  – нелинейная функция. Например

$$y(t) = ax(t) + bx^2(t). \quad (6.2)$$

В Simulink статические нелинейности можно моделировать при помощи блоков библиотеки Discontinuities, а также некоторых блоков из библиотек User-Defined Functions и Math Operations (например, Abs, Sign, Sqrt и др.). В данном пособии мы в основном будем рассматривать именно статические нелинейности.

Во многих случаях зависимость (6.1) очень сложная, тогда её приходится упрощать, например, аппроксимировать степенными рядами высокого порядка или выполнять кусочно-линейную аппроксимацию.

Динамические нелинейности описываются нелинейными дифференциальными уравнениями, например,

$$\frac{d^2 y(t)}{dt^2} + c \left( \frac{dy(t)}{dt} \right)^3 + y(t) = x(t). \quad (6.3)$$

Такие уравнения удобно представлять в виде структурных схем (рис. 6.1), состоящих из линейных динамических элементов и статических нелинейностей [16]. К сожалению, часто нелинейную зависимость нельзя выразить в математическом виде или эта зависимость справедлива только приближенно, поэтому такой подход не является общим, в отличие от линейных дифференциальных уравнений.

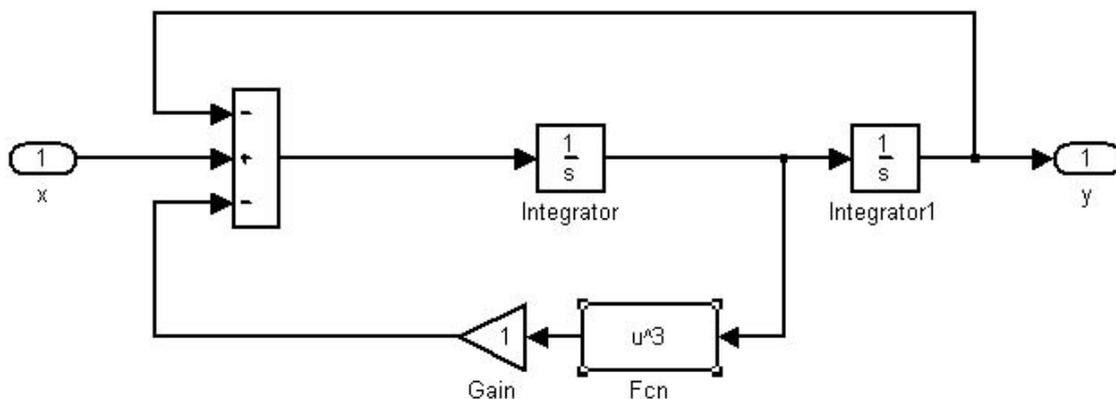


Рис. 6.1. Реализации уравнения в Simulink

Нелинейные системы автоматического управления обладают рядом принципиальных особенностей по сравнению с линейными системами. Главные особенности вытекают из того, что *они не подчиняются принципу суперпозиции*.

1. В линейных системах при подаче на вход гармонического сигнала на выходе получаем также гармонический сигнал, но другой амплитуды и сдвинутый по фазе. Колебания переходного процесса в нелинейных системах могут отличаться от входного гармонического сигнала, как по форме, так и по частоте. Например, для нелинейного элемента со статической характеристикой  $y(x) = |x|$  при подаче на него входного сигнала  $x(t) = A \cdot \sin \omega t$  выходные колебания не являются гармоническими, они имеют совершенно другую форму и период вдвое меньший, чем период входных колебаний (рис. 6.2).

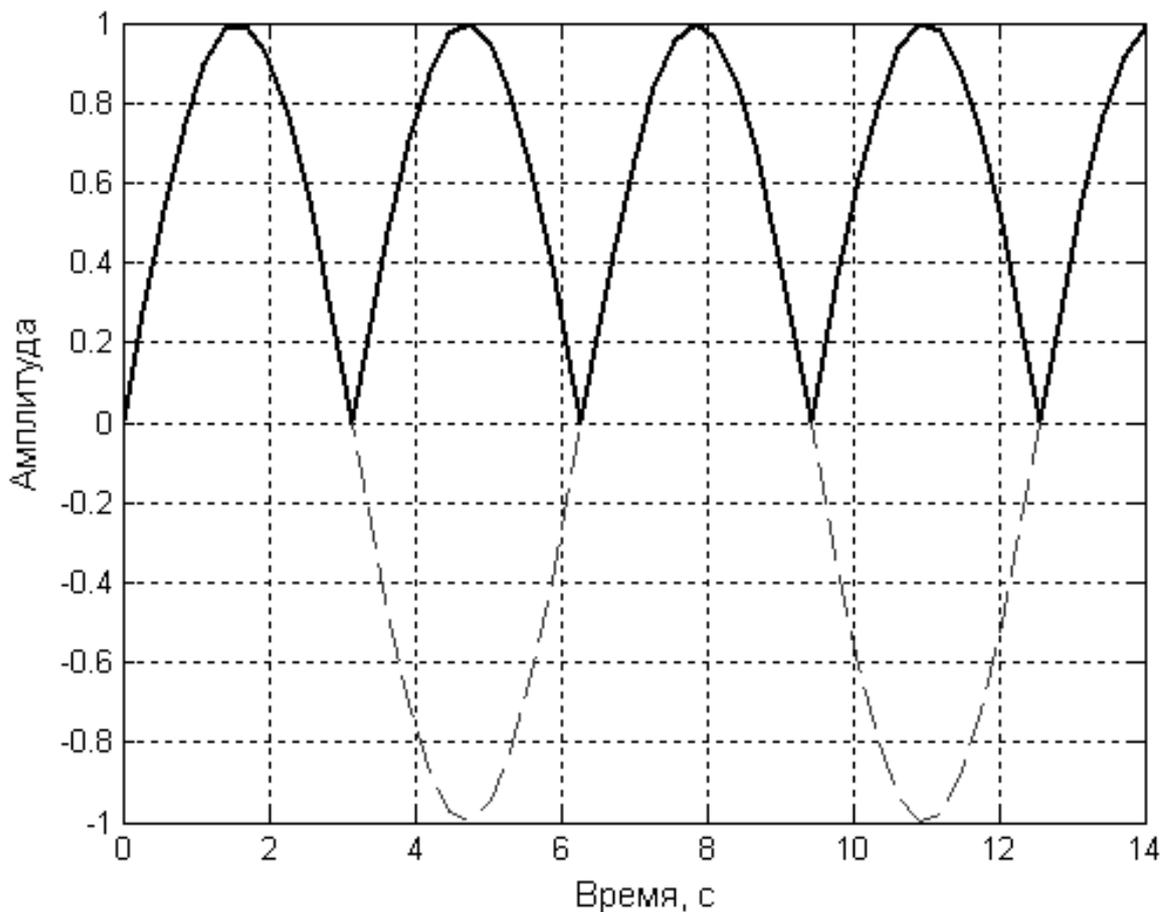
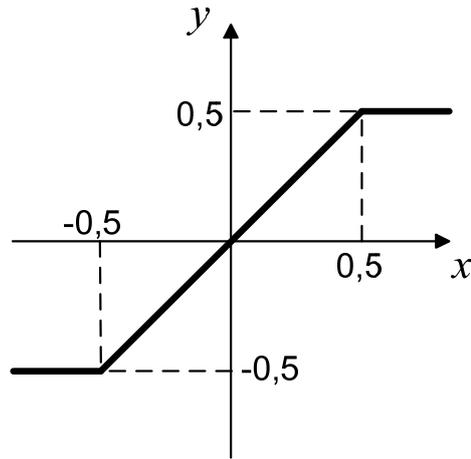


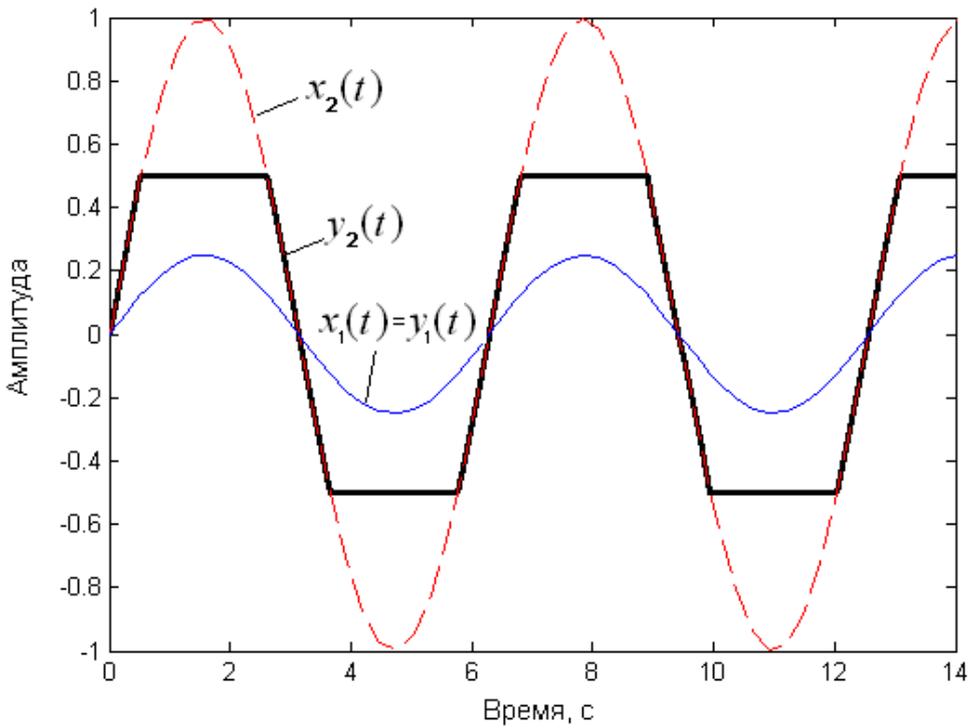
Рис. 6.2. Вид выходных колебаний в звене  $y(x) = |x|$  при входном синусоидальном сигнале

2. В линейных системах вид и качество переходного процесса полностью определяются свойствами самой системы и не зависят от величины входного сигнала и начальных условий.

В нелинейных системах поведение выходного сигнала существенно зависит как от величины входного воздействия, так и от начальных условий. Например, нелинейное звено со статической характеристикой, представленной на рис. 6.3 а, при различных амплитудах  $A$  входного сигнала  $A\sin(\omega t)$  ведет себя по-разному: при значениях амплитуды  $A \leq 5$  выходной сигнал пропорционален входному, т.е. звено ведет себя как линейное, а с увеличением  $A$  выходной сигнал уже существенно отличается от входного (рис. 6.3, б).



а



б

Рис. 6.3. Зависимость выходного сигнала нелинейного звена от амплитуды входного сигнала: а – статическая характеристика нелинейного звена; б – входные и выходные колебания

Или другой пример [16]. Если на вход нелинейности (6.3) подать входной сигнал вида  $A\sin(\omega t)$ , то при частотах  $\omega_1 = 1$  рад/с и  $\omega_2 = 3$  рад/с выходные сигналы имеют совершенно разную форму (рис. 6.4).

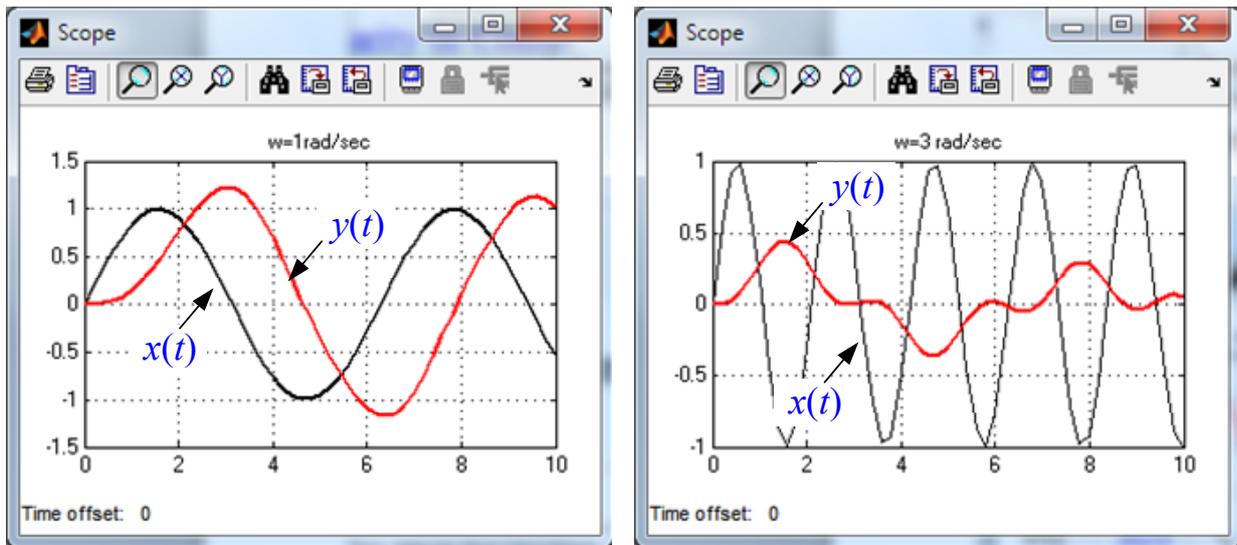


Рис. 6.4. Зависимость выходного сигнала нелинейного звена при различных частотах

Рассмотрим модель подвески (рис. 6.5) с нелинейной пружиной. Динамика такой системы описывается следующим нелинейным уравнением:

$$\frac{d^2 x(t)}{dt^2} + b \frac{dx(t)}{dt} + kx^3(t) = f(t). \quad (6.4)$$

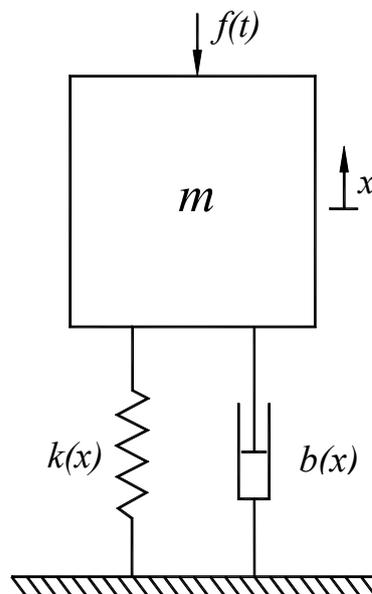


Рис. 6.5. Модель подвески с нелинейной пружиной

На рис. 6.6 отображена зависимость перемещения  $x$  от времени при ступенчатом изменении действующей силы  $f$ .

Очевидно, что реакция на положительное ступенчатое воздействие имеет большую колебательность, чем на отрицательное ступенчатое изменение  $f$ . Это уменьшение коэффициента затухания, очевидно, связано с увеличением жесткости нелинейной пружины при больших значениях  $|x|$ . Отметим также, что время регулирования  $t_p$  при  $f = 5$  отличается от  $t_p$  при  $f = 50$ , как можно было бы ожидать в линейном случае.

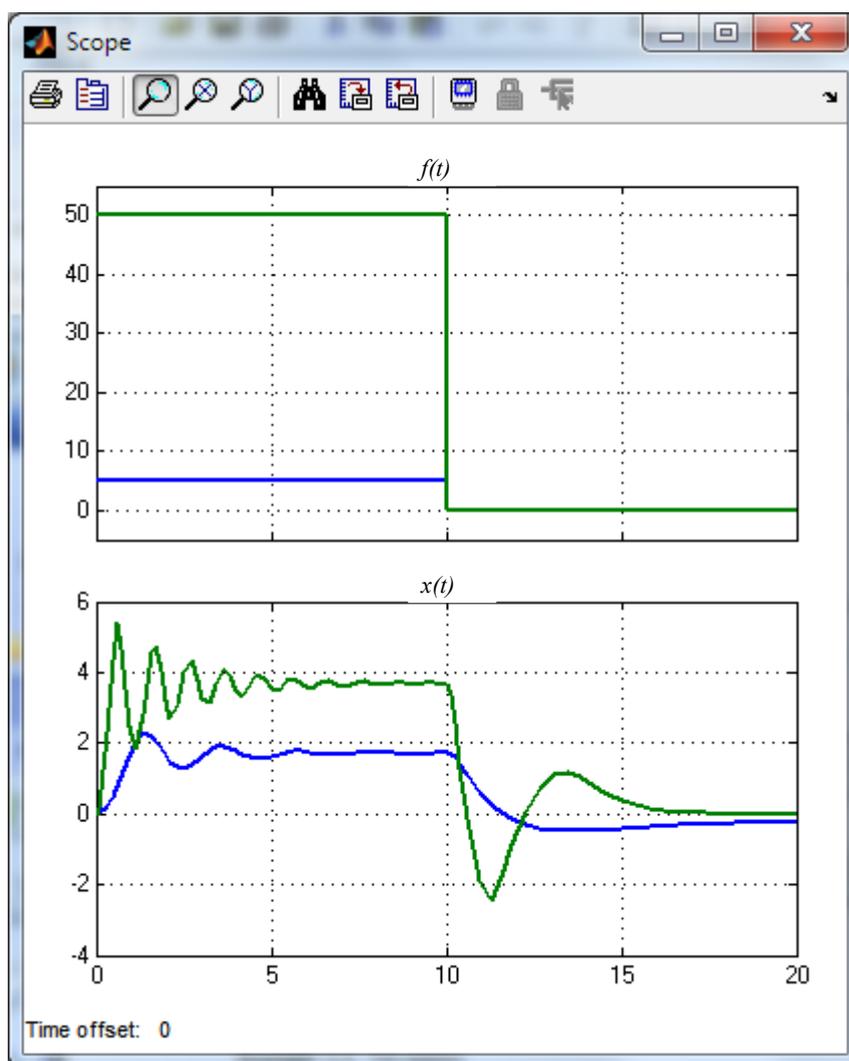


Рис. 6.6. Зависимость перемещения  $x$  от времени при ступенчатом изменении действующей силы  $f$

3. Нелинейная система может быть устойчивой при одних значениях воздействий и начальных условий и неустойчива при других значениях. Здесь нельзя говорить однозначно, устойчива

система или нет. Поэтому в случае нелинейных систем говорят не об устойчивости системы, а об *устойчивости режимов движения*.

4. В нелинейных системах могут существовать особые режимы движения – *автоколебания*.

Автоколебания – это устойчивые собственные колебания с постоянной амплитудой и частотой, возникающие из-за нелинейных свойств системы при особых условиях (рис. 6.7, 6.8).

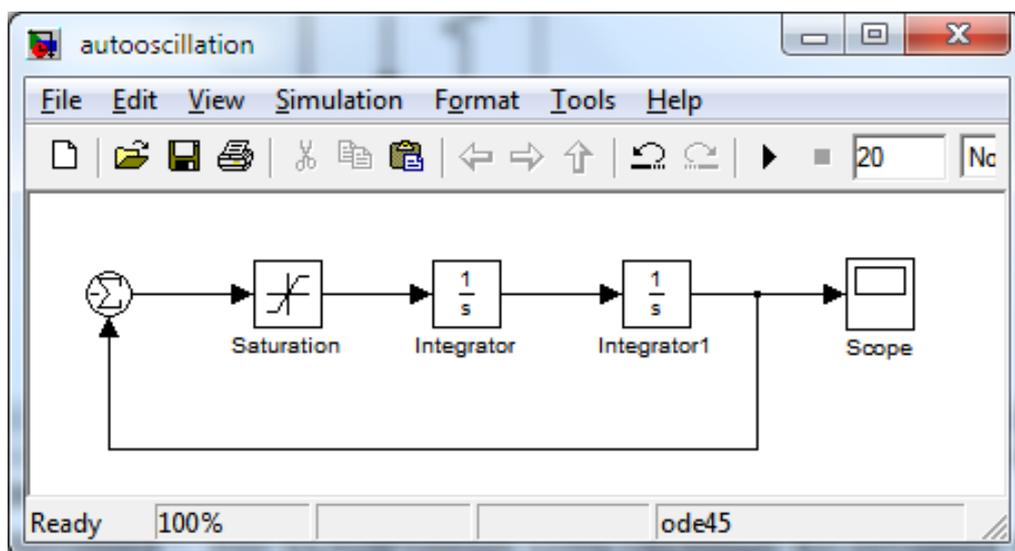


Рис. 6.7. Простейшая модель автоколебательной системы

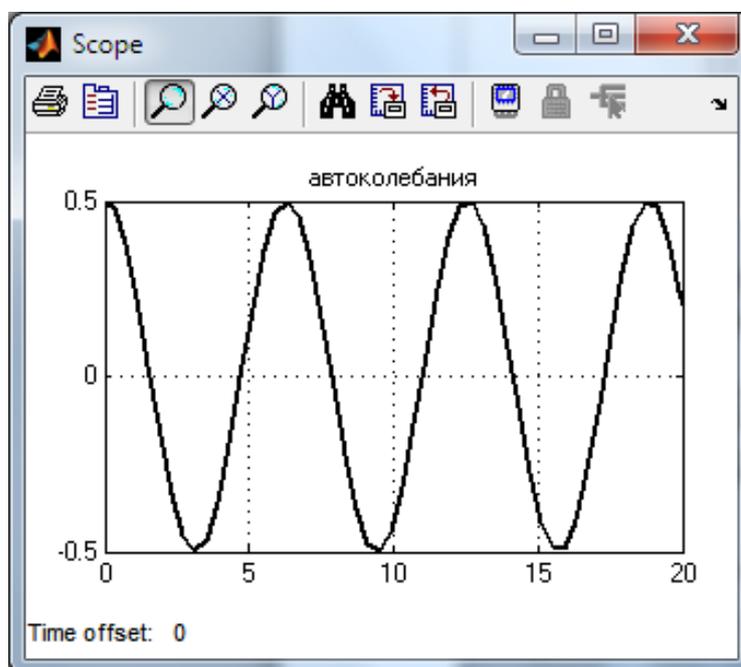


Рис. 6.8. Автоколебания

Автоколебания происходят потому, что энергия, которая расходуется при колебаниях выходной величины, восполняется от внутреннего источника энергии. При этом от источника берется ровно столько энергии, сколько необходимо для поддержания постоянной амплитуды и частоты, не больше и не меньше.

Режим автоколебаний принципиально отличается от колебаний линейной системы, находящейся на границе устойчивости. В отличие от автоколебаний, эти колебания неустойчивые, т.е. они не могут существовать длительное время, т.к. малейшие изменения параметров системы приводят к изменению колебательного процесса и он становится либо сходящимся, либо расходящимся. Поскольку в реальной системе ее параметры не могут быть длительное время постоянными (они изменяются под действием внешних условий, в результате старения элементов и т.д.), в линейной системе автоколебания существовать не могут, в отличие от нелинейной системы, в которой параметры колебаний зависят не только от параметров системы, но и от амплитуды колебаний.

Как правило, автоколебания в нелинейных системах нежелательны, а иногда и недопустимы. Следует, однако, отметить, что в некоторых нелинейных системах автоколебания являются основным рабочим режимом. К таким системам относятся, например, электрические генераторы колебаний. Автоколебательную систему представляют собой часы: в них внутренним источником энергии является заведенная пружина или батарейка. Примером автоколебательной системы может служить и сердце человека.

Рассмотрим примеры моделирования основных нелинейностей в Simulink.

## **6.2 Нелинейные блоки библиотеки *Discontinuities***

Пакет Simulink имеет ряд блоков, моделирующих типовые нелинейности (рис. 6.9). Эти блоки расположены в библиотеке *Discontinuities*. Как принято в теории управления, большинство нелинейных блоков рассматриваются как идеальные, без учета инерционных свойств. Для учета инерционности необходимо использовать блоки из раздела *Continuous*.

Свойства нелинейностей лучше всего исследовать, подавая на их вход синусоидальный сигнал (рис. 6.10).

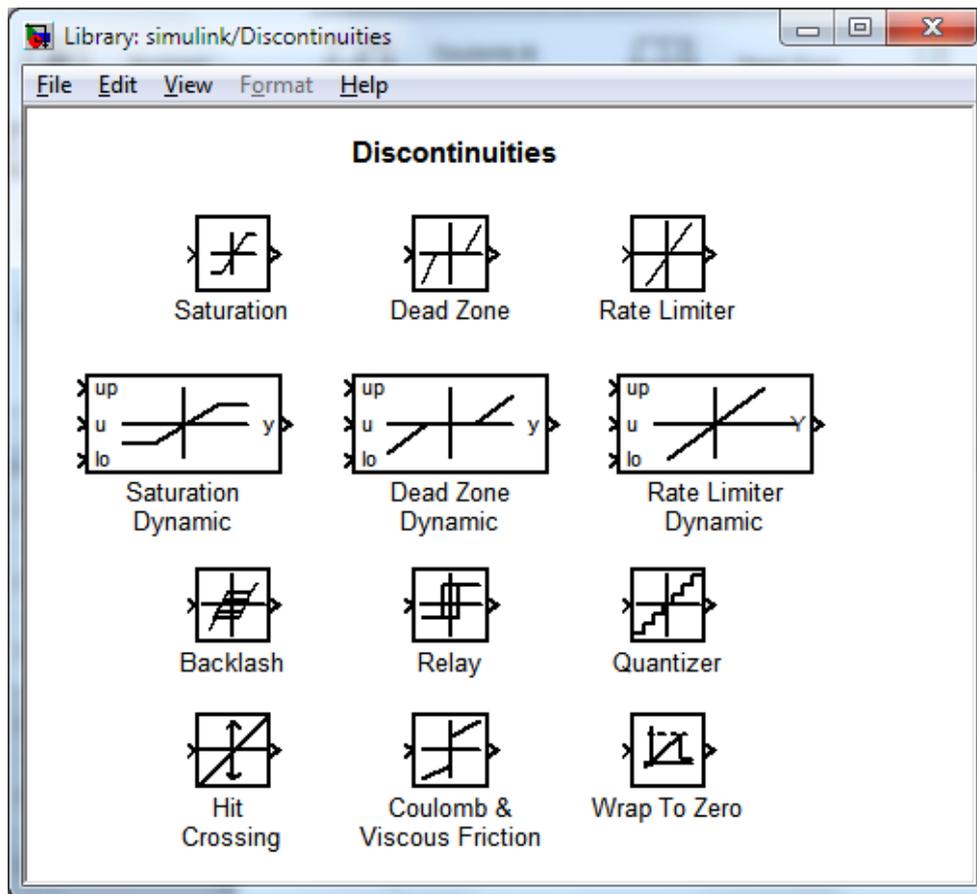


Рис. 6.9. Нелинейные элементы библиотеки Discontinuities



Рис. 6.10. Схема для исследования свойств нелинейного звена

### Блок **Saturation**

Блок **saturation** представляет собой модель одной из наиболее распространенных нелинейностей – «насыщение» или «ограничение».

Сигнал на выходе блока равен входному сигналу до тех пор, пока не достигнет заданных порогов ограничения: верхнего или нижнего. После этого сигнал перестает изменяться (рис. 6.11). Подобной характеристикой обладают многие реальные устройства,

например, усилители, ограниченные по мощности в области больших входных сигналов.

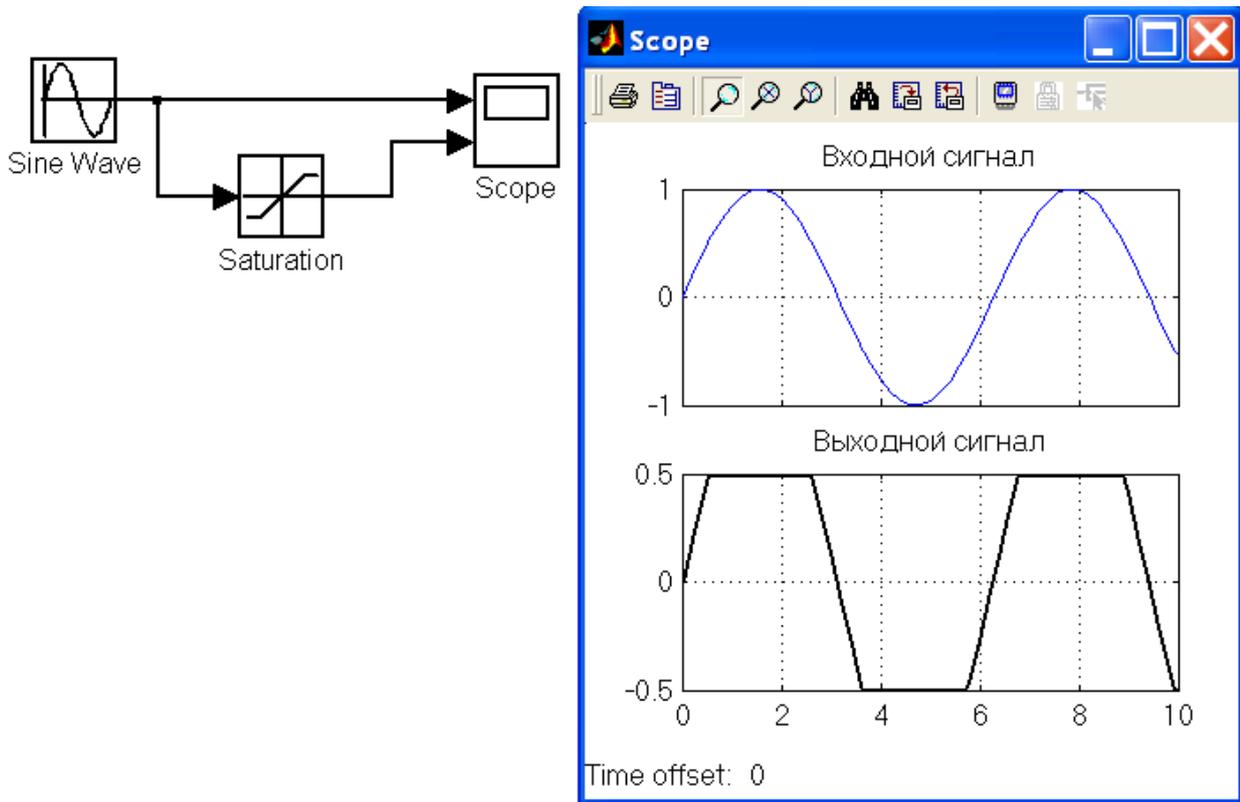


Рис. 6.11. Работа блока **Saturation**

Основная вкладка **Main** окна параметров блока **Saturation** представлено на рис. 6.12. Она содержит следующие поля:

- **Upper limit:** - верхний предел насыщения;
- **Lower limit:** - нижний предел насыщения;
- **Treat as gain when linearizing** (флаг) - рассматривать как усилитель при линейаризации;
- **Enable zero crossing detection** (флаг) - разрешить определение пересечения нуля;
- **Sample time (-1 for inherited)** - период квантования (для наследования от периода квантования до предыдущего блока указывается - 1).

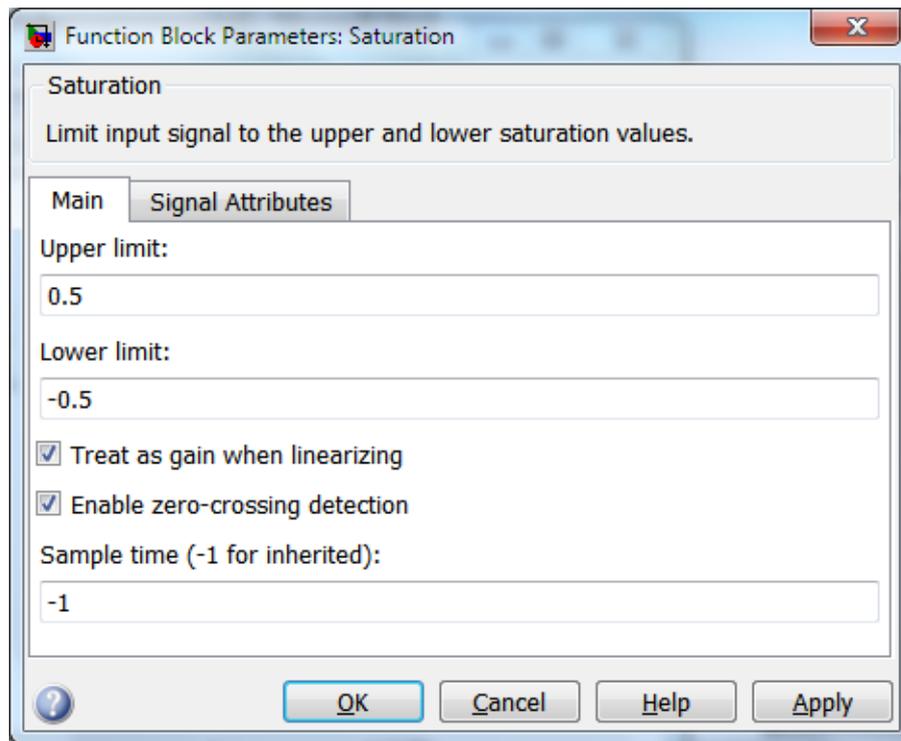


Рис. 6.12. Диалоговое окно параметров элемента **Saturation**

### Блок **Backlash**

**Backlash** или люфт (рис. 6.13, а) - одна из нелинейностей, наиболее часто встречающихся в системах и связанная с наличием зазоров между различными механическими компонентами. Например, этот блок используют при моделировании зубчатых передач, если необходимо учитывать эффект люфта между зубьями шестерен.

На рис. 6.13, б, представлены диалоговое окно параметров нелинейного блока **Backlash** (люфт).

Диалоговое окно содержит три поля ввода:

- **Deadband width** (ширина полосы люфта);
- **Initial output** (начальное значение выходного сигнала),
- **Sample time (-1 for inherited)** - период квантования (-1 для наследования), а также флаг **Enable zero crossing detection** - разрешить определение пересечения нуля.

Сигнал на выходе блока будет равен значению, заданному в поле **Initial output**, пока входной сигнал при возрастании не достигнет половины значения, указанного в поле **Deaband width**, после чего выходной сигнал будет равен  $x-b/2$ , где  $x$  – входной сигнал,  $b$  - ширина полосы люфта.

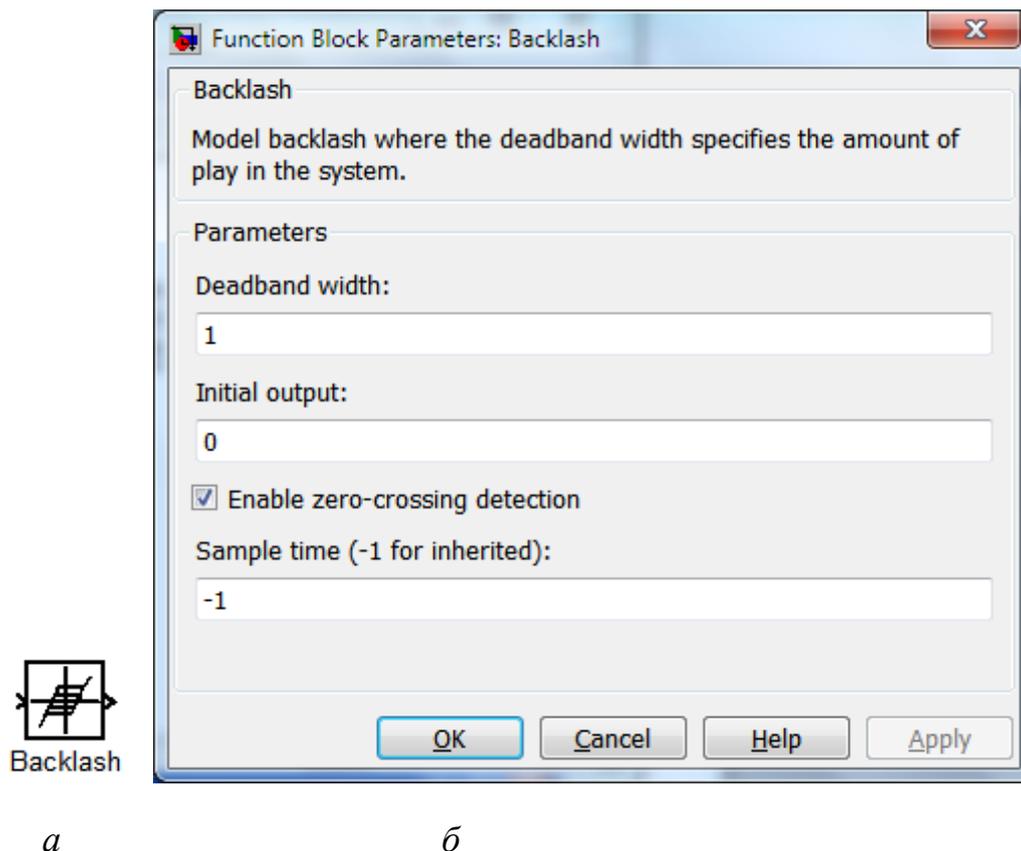


Рис. 6.13. Блок Backlash (а) и диалоговое окно его параметров (б)

После того как произойдет смена направления изменения входного сигнала, выходной сигнал будет оставаться неизменным до тех пор, пока входной сигнал не изменится на величину  $b/2$ , после чего выходной сигнал будет равен  $(x+b)/2$ .

Чтобы проиллюстрировать работу блока **Backlash** разобьем рис. 6.14 на 4 зоны:

**А** – входной сигнал  $x$  изменяется в положительном направлении, однако находится в полосе люфта ( $-b/2 \leq x \leq b/2$ ). Выходной сигнал  $y$  не изменяется.

**В** - входной сигнал  $x$  изменяется в положительном направлении, при этом  $x > b/2$ . Изменение входного сигнала вызывает соответствующее изменение выходного сигнала  $y$ .

**С** – входной сигнал  $x$  меняет направление, однако его изменение не влияет на изменение выходного сигнала  $y$ , т.к. он меньше  $1-0,5$ .

**Д** - входной сигнал пересекает пороговое значение, поэтому его изменение приводит к изменению выходного сигнала.

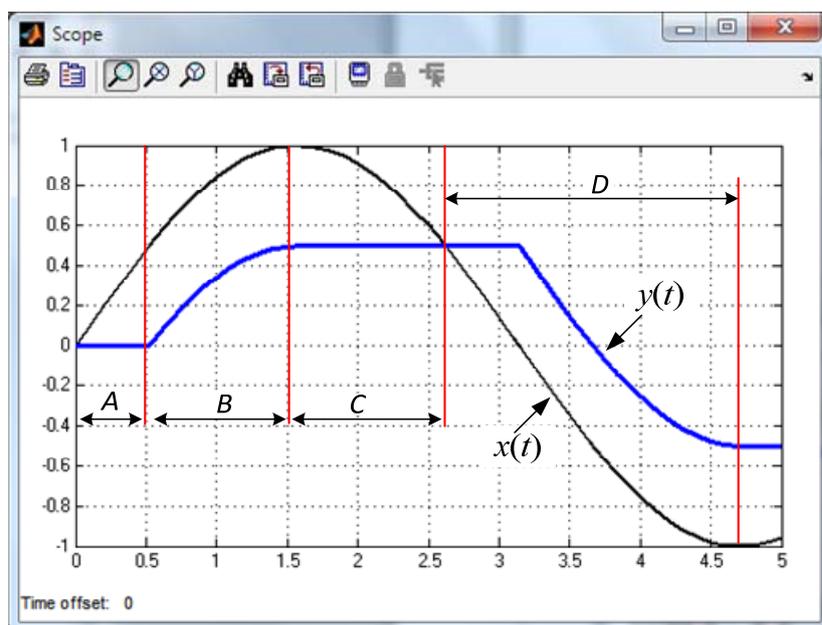


Рис. 6.14. Работа блока **Backlash**

### Блок **Coulomb&Viscous Friction**

Блок **Coulomb&Viscous Friction** (Кулоновское и вязкое трение) служит для моделирования фрикционных эффектов, т.е. учитывает нелинейные эффекты, которые возникают при трении между различными механическими элементами. На рис. 6.15, а представлено окно параметров этого блока. В окне параметров указано уравнение, в соответствии с которым блок **Coulomb&Viscous Friction** формирует выходной сигнал:

$$y = \text{sign}(x) * (\text{Gain} * \text{abs}(x) + \text{Offset}), \quad (6.5)$$

где  $x$  – входной сигнал,

$y$  – выходной сигнал,

$\text{Gain}$  – коэффициент вязкого трения,

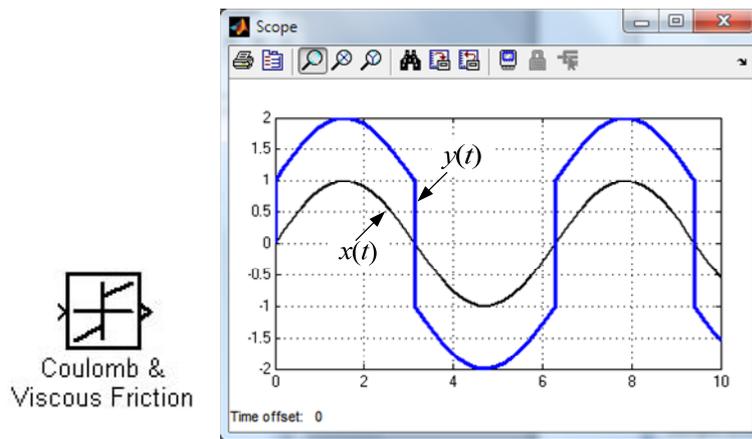
$\text{Offset}$  – величина сухого трения.

Кроме того, окно параметров имеет два текстовых поля:

– **Coulomb friction value (Offset)** – Величина сухого трения;

– **Coefficient of viscous friction (Gain)** – Коэффициент вязкого трения.

На рис. 6.15, б показана реакция блока **Coulomb&Viscous Friction** на синусоидальный входной сигнал с амплитудой  $A = 1$  при значении **Offset** = 2.



*a*

*б*

Рис. 6.15. Окно параметров (*a*) и реакция на входной сигнал (*б*) блока **Coulomb&Viscous Friction**

### Блок **Dead Zone**

**Dead Zone** – мертвая зона (или зона нечувствительности) - одна из нелинейностей, которая представляет собой линейную зависимость выходного сигнала от входного везде, за исключением области мертвой зоны. Подобной характеристикой обладают многие реальные элементы в области малых входных сигналов. Окно параметров блока **Dead Zone** показано на рис.6.16.

Окно параметров содержит три поля ввода:

- **Start of dead zone** - начало мертвой зоны;
- **End of deaf zone** - конец мертвой зоны;
- **Sample time (-1 for inherited)** - период квантования (-1 для наследования), и три параметра – флага:

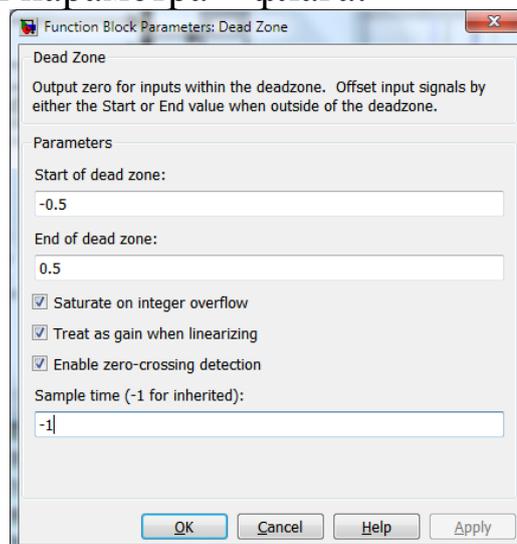


Рис. 6.16. Диалоговое окно параметров элемента **Dead Zone**

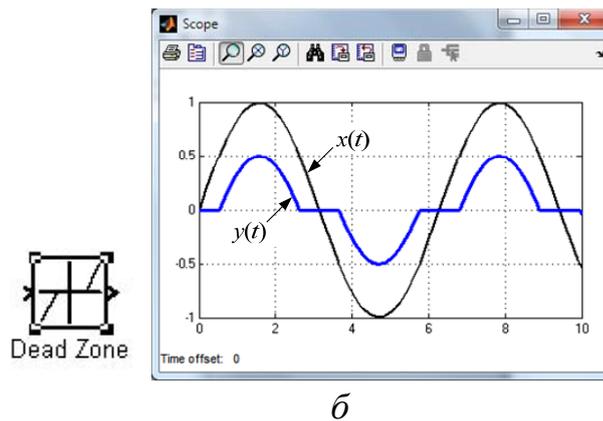


Рис. 6.17. Блок Dead Zone (а) и его выходной сигнал (б)

– **Saturate on integer overflow** - подавлять переполнение целого. При установленном флажке ограничение сигналов целого типа выполняется корректно.

– **Treat as gain when linearizing** - трактовать как усилитель с коэффициентом передачи равным 1 при линейаризации);

– **Enable zero crossing detection** - разрешить определение пересечения нуля.

Выходной сигнал блока (рис. 6.17, б) определяется следующим уравнением:

$$y = \begin{cases} 0, & \text{если } b_1 < x < b_2 \\ x - b_1, & \text{если } x \leq b_1 \\ x - b_2, & \text{если } x \geq b_2 \end{cases}, \quad (6.6)$$

где  $x$  – входной сигнал блока;

$y$  – выходной сигнал блока;

$b_1$  – начало мертвой зоны;

$b_2$  – конец мертвой зоны.

В последних версиях Simulink появился блок Dead Zone Dynamic (рис. 6.18), который фактически является маскированной подсистемой. Отличие блока Dead Zone Dynamic от блока Dead Zone заключается в том, что он имеет два дополнительных входных порта  $up$  (верхняя граница зоны нечувствительности) и  $lo$  (нижняя граница зоны нечувствительности). На эти входы при помощи других блоков подаются изменяющиеся во времени сигналы, устанавливающие мгновенные значения ширины зоны нечувствительности.

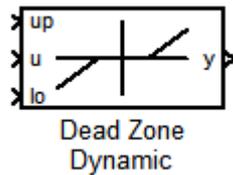


Рис. 6.18. Блок Dead Zone Dynamic

### Блок Hit Crossing

Этот блок осуществляет сравнение входного сигнала с заданным пороговым значением (по умолчанию 0) и выработывает в этот момент единичный импульс.

Диалоговое окно параметров блока (рис. 6.19) содержит текстовое поле **Hit crossing offset**, в нем устанавливается порог, пересечение которого входным сигналом требуется определить.

Следующим параметром является раскрывающийся список **Hit crossing direction** (направление пересечения). Здесь указывается, в каком случае надо фиксировать пересечение порога, при этом пиктограмма блока изменяет свой вид:

- **rising** – при увеличении входного сигнала, т.е. снизу (рис. 6.20);
- **falling** - при уменьшении входного сигнала т.е. сверху (рис. 6.21);
- **either** – если управляющий сигнал пересекает заданный порог как сверху, так и снизу (рис. 6.22).

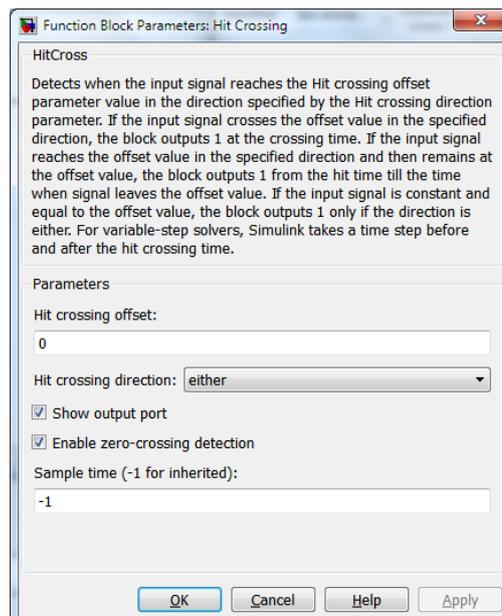


Рис. 6.19. Диалоговое окно параметров блока Hit Crossing

Кроме того, имеются еще два параметра, которые определяются установкой флага:

– **Show output port** (показать выходной порт). В том случае, если этот флаг снят, то точка пересечения сигналом порогового уровня находится, но выходной порт блока скрывается.

– **Enable zero crossing detection** - разрешить определение пересечения нуля.

В поле **Sample time (-1 for inherited)** задается период квантования.

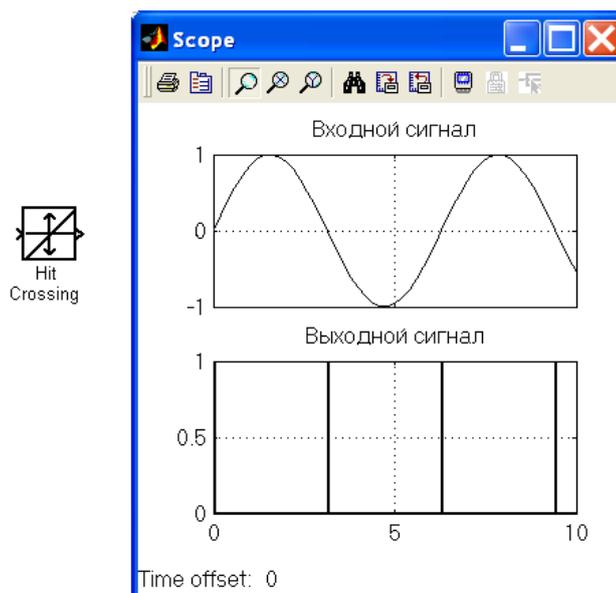


Рис. 6.20. Внешний вид и выходной сигнал блока Hit Crossing при выборе значения **either** параметра **Hit crossing offset**

### Блок **Quantizer**

Блок Quantizer (Квантователь) осуществляет квантование по уровню входного непрерывного сигнала, а между моментами квантования поддерживает последнее значение выходной величины. Такую характеристику имеют, например, аналогово-цифровые преобразователи.

При формировании выходного сигнала блок Quantizer использует округление до ближайшего целого, что при большом шаге квантования даёт заметную погрешность слежения за уровнем входного сигнала:

$$y = q * \text{round}(x/q) \quad (6.7)$$

где  $y$  – выходной сигнал,  $x$  – входной сигнал и  $q$  – шаг квантования по уровню.

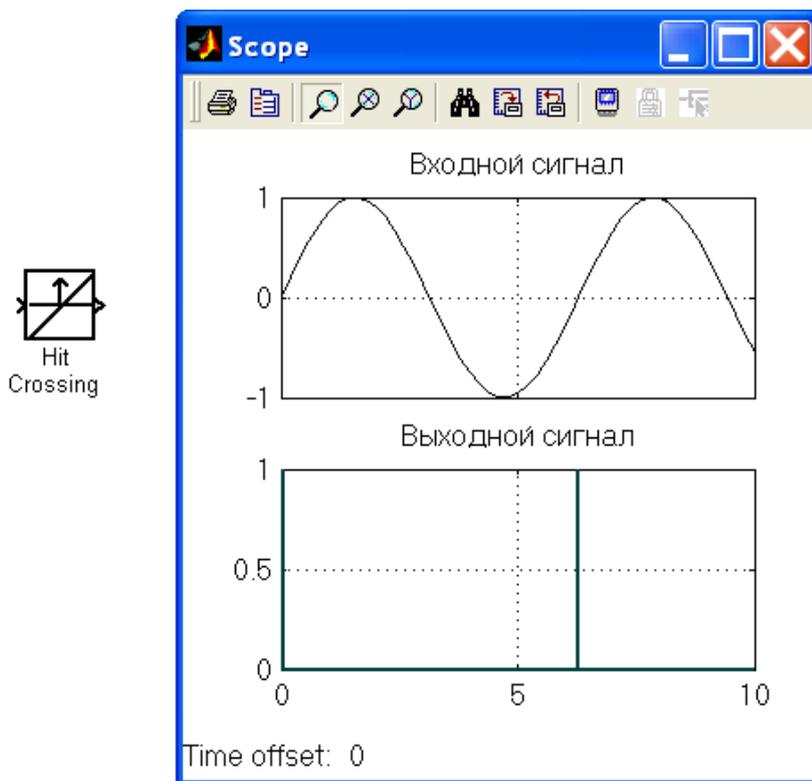


Рис. 6.21. Внешний вид и выходной сигнал блока Hit Crossing при выборе значения **falling** параметра **Hit crossing offset**

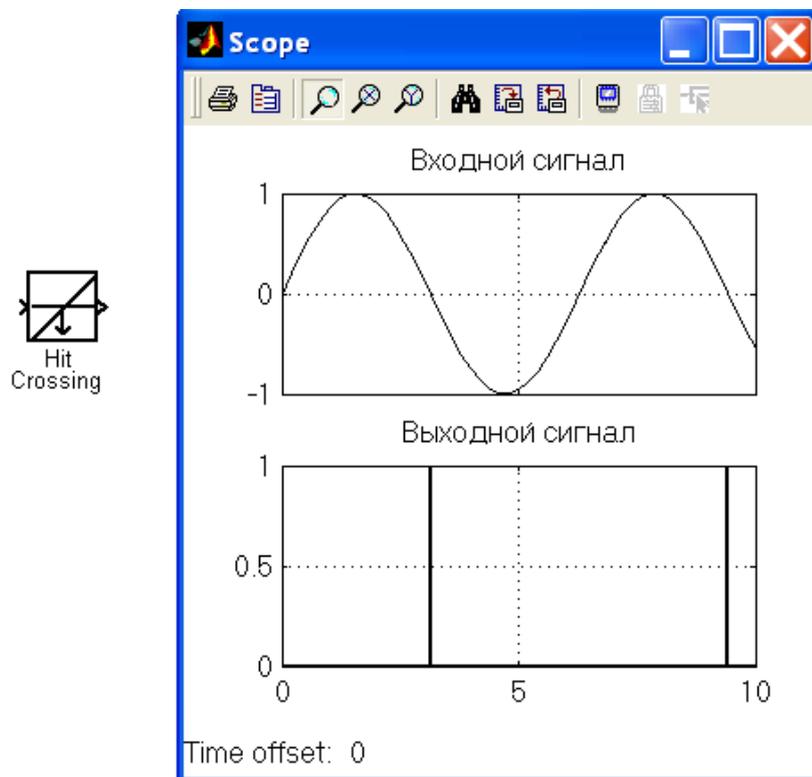


Рис. 6.22. Внешний вид и выходной сигнал блока Hit Crossing при выборе значения **rising** параметра **Hit crossing offset**

На рис. 6.23 представлено диалоговое окно параметров блока Quantizer.

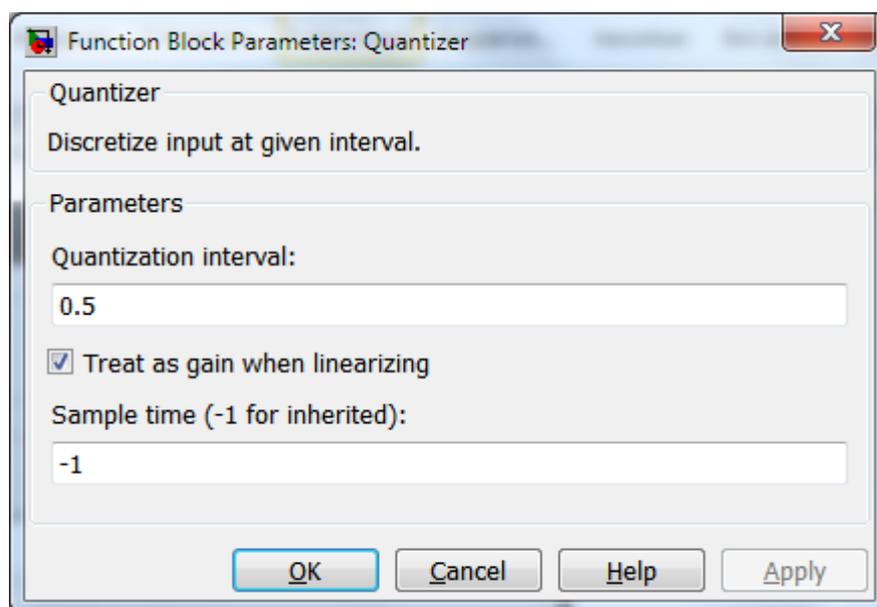


Рис. 6.23. Диалоговое окно параметров элемента Quantizer и его характеристики

Диалоговое окно содержит два поля ввода:

- **Quantization interval** - шаг квантования;
- **Sample time (-1 for inherited)** - период квантования;

и один флаг: **Treat as gain when linearizing** (рассматривать как усилитель при линеаризации).

На рис. 6.24 показан пример использования блока Quantizer, выполняющего квантование по уровню синусоидального сигнала с шагом 0,5.

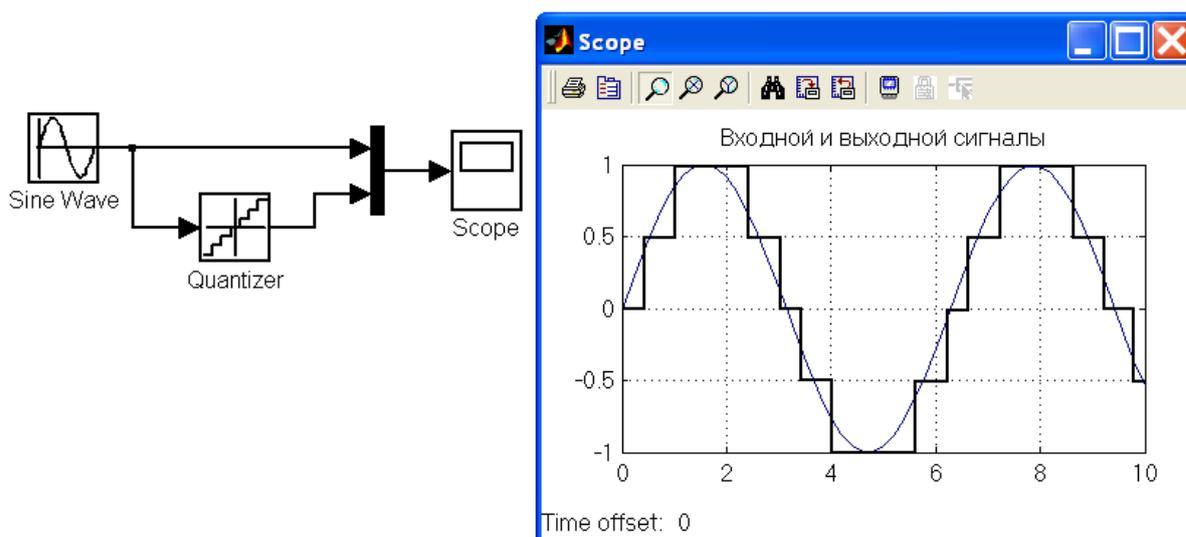


Рис. 6.24. Иллюстрация работы блока Quantizer

## Блок **Rate Limiter**

Блок **Rate Limiter** (ограничитель скорости) обеспечивает ограничение скорости изменения входного сигнала. Скорость определяется при помощи следующих выражений:

$$rate = \frac{x(i) - y(i-1)}{\Delta t}, \quad (6.8)$$

$$\Delta t = t(i) - t(i-1) \quad (6.9)$$

где  $x(i)$  – текущее значение входного сигнала;  
 $y(i-1)$  – выход блока на предыдущем шаге;  
 $t(i)$  и  $t(i-1)$  - время на текущем шаге и предыдущем шаге соответственно.

Окно параметров этого блока и реакция на синусоиду представлены на рис. 6.25. Оно содержит такие параметры:

- **Rising slew rate** – порог, ограничивающий скорость при увеличении входного сигнала;
- **Falling slew rate** - порог, ограничивающий скорость при уменьшении входного сигнала;
- **Sample time mode** – список, в котором выбирается режим квантования по времени: **inherited** (наследование от предыдущего блока) и **continuous** (блок рассматривается как непрерывный).

При выборе позиции **inherited** появляется еще одно поле – **Initial condition**, в котором задаются начальные условия.

Последним параметром блока **Rate Limiter** является один флаг **Treat as gain when linearizing** (рассматривать как усилитель при линеаризации).

Выходной сигнал определяется сравнением значения *rate* скорости изменения входного сигнала со значениями, указанными в полях **Rising slew rate** и **Falling slew rate** окна параметров блока (рис. 6.25).

Если скорость *rate* изменения входного сигнала больше, чем значение  $R$ , указанное в поле **Rising slew rate**, выходной сигнал блока вычисляется как

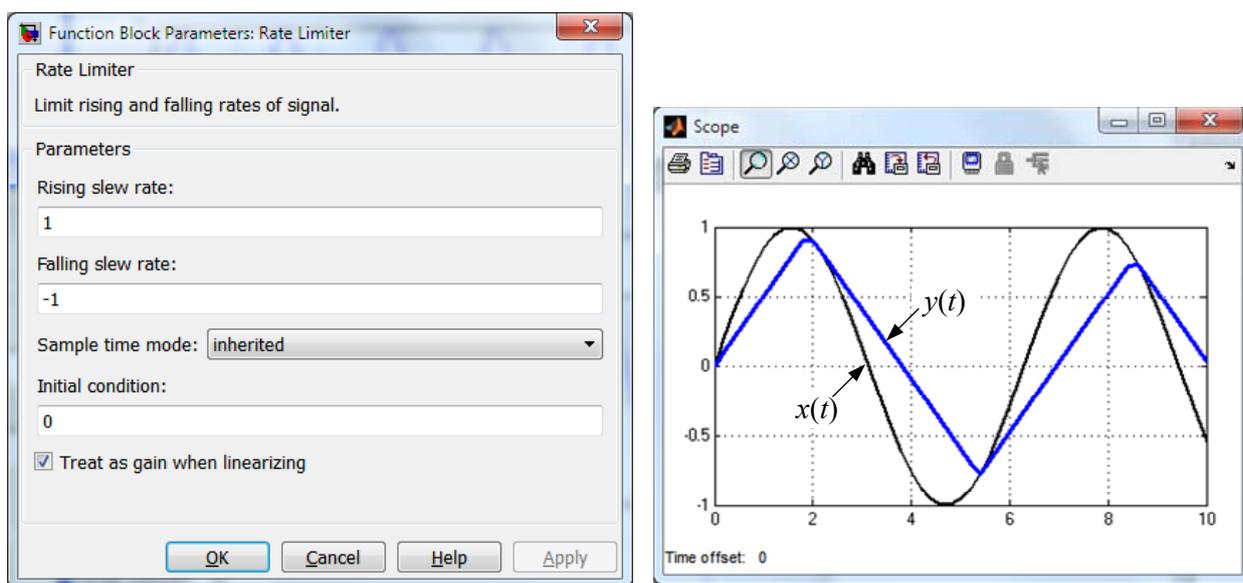
$$y(i) = \Delta t \cdot R + y(i-1). \quad (6.10)$$

Если скорость *rate* изменения входного сигнала меньше, чем значение параметра  $F$ , указанного в поле **Falling slew rate**, выходной сигнал блока вычисляется как

$$y(i) = \Delta t \cdot F + y(i-1). \quad (6.11)$$

Если скорость *rate* изменения входного сигнала лежит в диапазоне между значениями  $R$  и  $F$ , выходной сигнал равен входному:

$$y(i) = x(i). \quad (6.12)$$



*a*

*б*

Рис. 6.25. Окно параметров блока Rate Limiter и его реакция на синусоидальный сигнал

В блоке Rate Limiter Dynamic, как и в блоке Dead Zone Dynamic присутствует два дополнительных входных порта  $up$  и  $lo$  для задания изменяющихся во времени значений  $R$  и  $F$ .

### Блок Relay

Этот блок представляет собой модель двухпозиционного реле. Основная вкладка **Main** окна параметров блока Relay изображена на рис. 6.26.

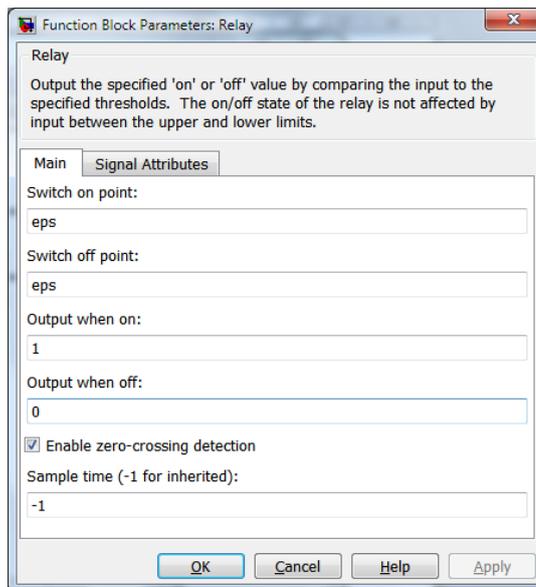


Рис. 6.26. Окно параметров блока Relay, принятое по умолчанию

Если входной сигнал блока больше порогового значения, заданного в поле параметров **Switch on point**, то на выходе появляется постоянный сигнал, значение которого задается в поле **Output when on**. Если же входной сигнал меньше порогового значения, заданного в поле параметров **Switch off point**, то на выходе блока появляется постоянный сигнал, значение которого задается в поле **Output when off**. Значение параметра, указанного в поле **Switch on point** должно быть больше значения, указанного в поле **Switch off point** (если необходимо учитывать явление гистерезиса) или равно ему (если моделируется идеальное реле). По умолчанию в полях **Switch on point** и **Switch off point** установлено бесконечно малое значение порога **eps**.

Устанавливая флажок напротив параметра **Enable zero crossing detection** можно разрешить или запретить определение пересечения нуля. В поле **Sample time (-1 for inherited)**, аналогично предыдущим блокам, устанавливается значение периода квантования.

Реакцию блока Relay на синусоидальный сигнал единичной амплитуды отображает рис. 6.27 при следующих параметрах блока:

**Switch on point:** 0.5;  
**Switch off point:** -0.5;  
**Output when on:** 1;  
**Output when off:** -1.

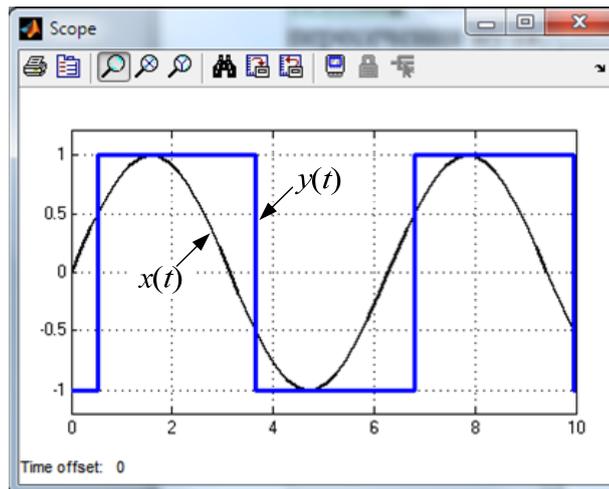


Рис. 6.27. Работа блока Relay

### Блок Wrap To Zero

Блок Wrap To Zero повторяет входной сигнал, когда он меньше или равен пороговому значению **Threshold**, и выдает 0, когда входной сигнал больше порогового значения **Threshold** (рис. 6.28).

Окно параметров блока имеет лишь одно текстовое поле **Threshold**, в котором указывается пороговое значение.

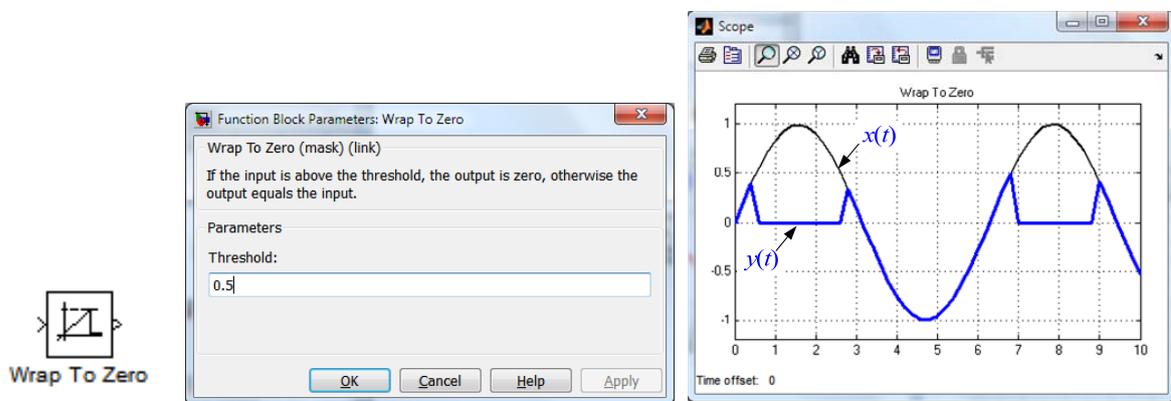


Рис. 6.28. Окно параметров блока Wrap To Zero и его реакция на синусоидальный сигнал

## 6.3 Создание сложных нелинейностей

Конечно, рассмотренные выше блоки не охватывают все нелинейные эффекты, которые могут встретиться в системах управления. Да это и не возможно.

Некоторые нелинейности, которые не представлены отдельными блоками в библиотеке *Discontinuities*, можно задать комбинацией двух или нескольких блоков. Например, на рис. 6.29 показана модель трехпозиционного реле с гистерезисом. Параметры релейных блоков приведены в табл. 6.1.

Таблица 6.1.

Параметры релейных блоков

	Switch on point	Switch off point	Output when on	Output when off
Relay	0.6	0.2	0.8	0
Relay1	-0.2	-0.6	0	-0.8

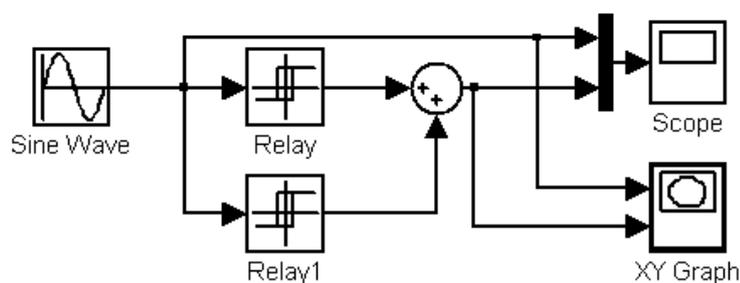


Рис. 6.29. Simulink-модель трехпозиционного реле с гистерезисом и параметры настроек блоков *Relay*

На схеме рис. 6.29 задействован новый блок *XY Graph* (Графопостроитель) из библиотеки блоков регистрации сигналов *Sinks*. Блок *XY Graph* строит график вида  $Y(X)$  и имеет два входа. Верхний вход предназначен для подачи сигнала, который является аргументом ( $X$ ), нижний – для подачи значений функции ( $Y$ ). В диалоговом окне параметров блока (рис. 6.30) задаются предельные значения по осям координат (**x-min**, **x-max**, **y-min**, **y-max**) и шаг квантования (**Sample time**). Результат моделирования приведен на рис. 6.31, а.

Использование блока *XY Graph* позволяет построить статическую характеристику результирующей нелинейности (рис. 6.30, б).

Рассмотрим еще один пример. Многие реальные элементы в области малых входных сигналов имеют зону нечувствительности, а в области больших сигналов обладают эффектом насыщения. Попробуем построить Simulink-модель такого нелинейного звена со следующими параметрами: зона нечувствительности лежит в пределах  $-0,15 < x < 0,15$ , а эффект насыщения наступает при  $\pm 0,8y$ .

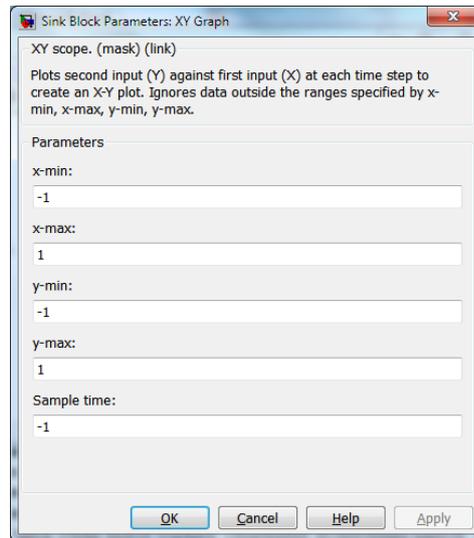
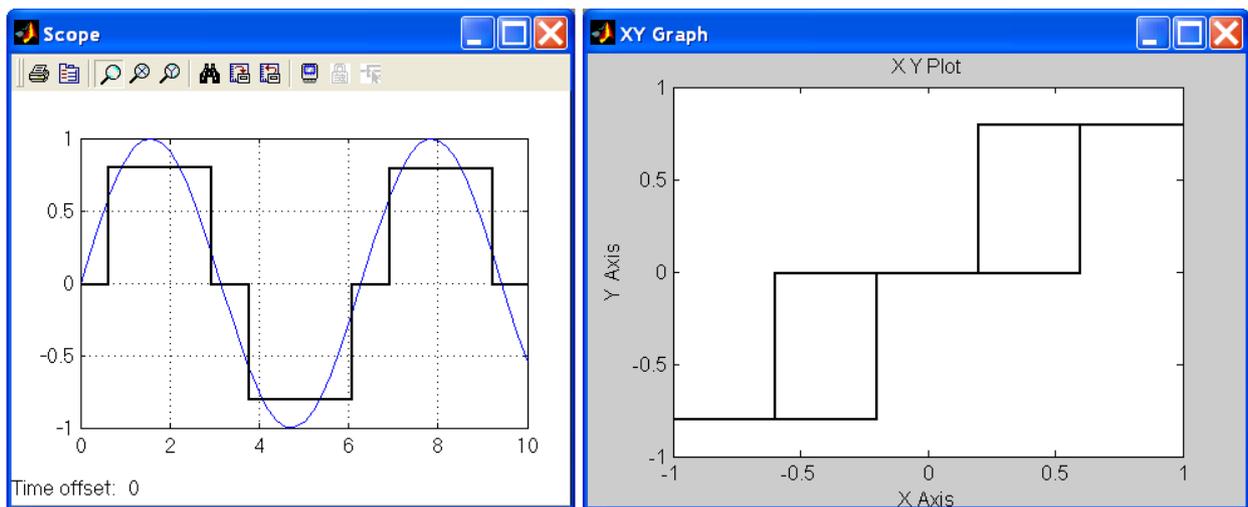


Рис. 6.30. Окно параметров блока XY Graph



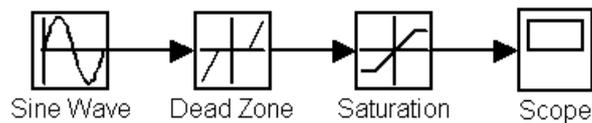
*a*

*б*

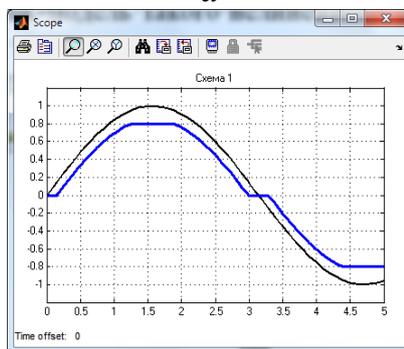
Рис. 6.31. Результаты моделирования трехпозиционного реле с гистерезисом:  
*a* - реакция на синусоиду; *б* - статическая характеристика

Очевидно, что модель будет состоять из двух последовательно соединенных нелинейных блоков: Dead Zone и Saturation. Однако здесь очень важен порядок следования нелинейных блоков.

Сначала рассмотрим способ соединения блоков, представленный на рис. 6.32, *a*. Результат моделирования представлен на рис. 6.32, *б*.



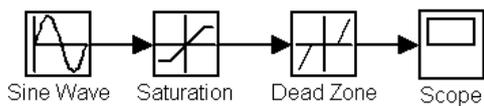
*a*



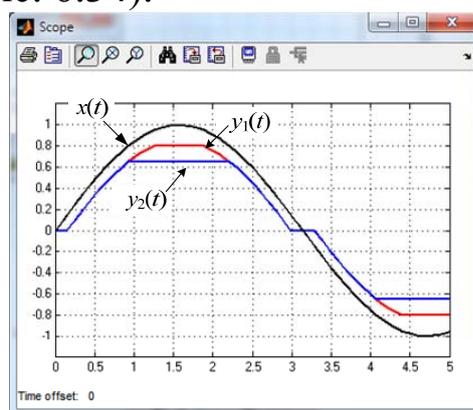
*б*

Рис. 6.32. Первый вариант соединения нелинейных блоков (*a*) и результат моделирования (*б*)

Теперь рассмотрим второй способ соединения блоков (рис. 6.33, *a*). Совместив для удобства полученный результат с выходным сигналом предыдущей модели (рис. 6.33, *б*) хорошо видно, что полученные кривые отличаются. Это связано с тем, что, в отличие от линейных блоков, *нелинейные блоки нельзя менять местами*, т.к. в результате можно получить нелинейность, существенно отличающуюся от желаемой. В нашем случае общий вид статической характеристики в зависимости от последовательности соединения блоков принципиально не меняется. Меняется только ширина полосы пропускания. Это хорошо видно, на статических характеристиках результирующих нелинейностей (рис. 6.34).



*a*



*б*

Рис. 6.33. Второй вариант соединения нелинейных блоков (*a*) и результат моделирования (*б*)

Таким образом, для моделирования нелинейности тапа «зона нечувствительности с ограничением» следует использовать первую модель, показанную на рис.6.33, *а*.

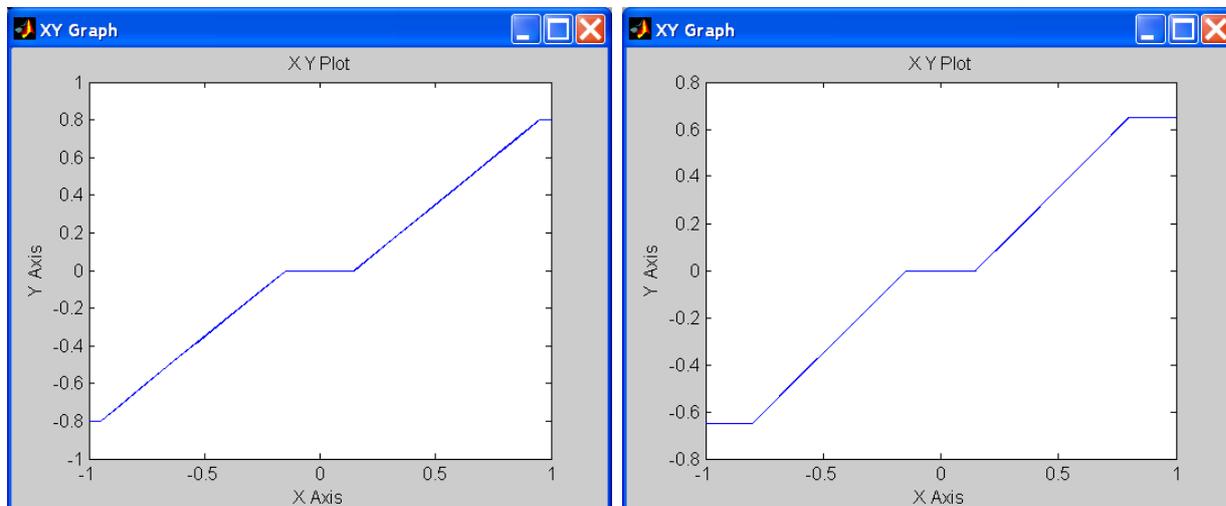


Рис. 6.34. Статические характеристики нелинейности при различной последовательности соединений блоков Dead Zone и Saturation: *а* – Dead Zone → Saturation; *б* – Saturation → Dead Zone

#### 6.4 Блоки библиотеки User-Defined Functions

В том случае, если нелинейность нельзя описать каким-либо блоком из библиотеки Discontinuities или их соединением, можно использовать блоки Fcn (Функция) и MATLAB Fcn (Функция MATLAB) библиотеки User-Defined Functions (Функции, определяемые пользователем).

Блок Fcn формирует скалярный выходной сигнал как функцию от входной переменной  $u$  или вектора входных переменных  $u(n)$  ( $n$  – номер элемента входного вектора). Изображение и диалоговое окно данного блока приведено на рис. 6.35.

Диалоговое окно блока в группе опций **Parameters** содержит поле **Expression** (Выражение), где вводится выражение на языке C, задающее требуемую функцию. В выражении могут использоваться такие функции, как `abs`, `exp`, `sin`, `cos`, `asin`, `acos`, `ln`, `log`, `log10` и др., принятые в языке C знаки арифметических операций (+, -, ^, \*, /), операторы отношения (==, !=, >, >=, <, <=), знаки логических операций (&& (логическое И) и || (логическое ИЛИ)).

В поле **Sample time (-1 for inherited)** задается значение периода квантования, если блок работает в дискретной системе.

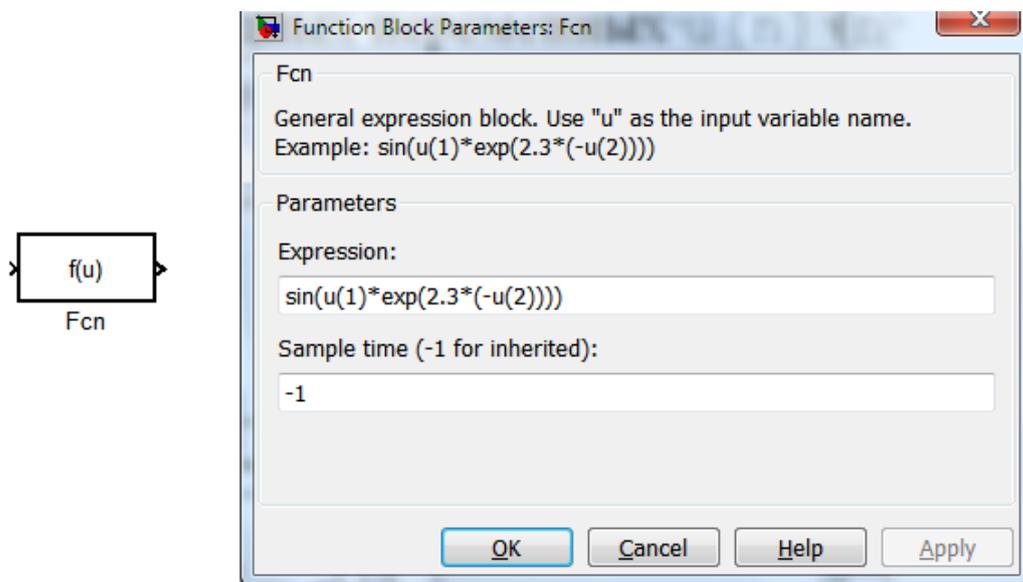


Рис. 6.35. Блок Fcn и диалоговое окно его параметров

Блок MATLAB Fcn (рис. 6.36) позволяет задавать функцию по правилам, принятым в языке программирования MATLAB. В отличие от предыдущего блока, блок MATLAB Fcn позволяет выполнять матричные операции и формировать выходную переменную в виде вектора. Во многих случаях это удобнее, однако уменьшает скорость выполнения операций.

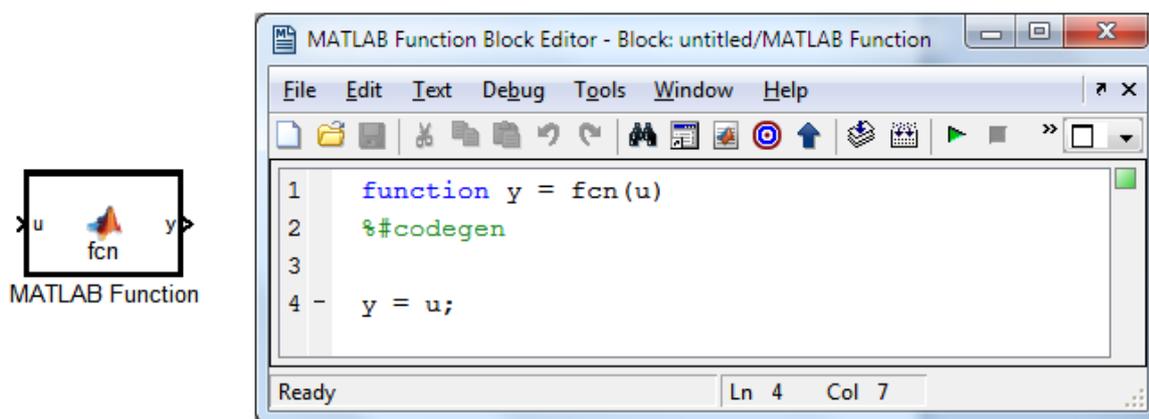


Рис. 6.36. Блок MATLAB Fcn и окно его редактора

Двойной щелчок по блоку MATLAB Fcn вызывает окно редактора **MATLAB Function Block Editor**, в котором на языке MATLAB записывается код вычисления требуемой функции. Например, на рис. 6.37 представлена модель, вычисляющая собственные значения входной матрицы **A**. Соответствующий код блока MATLAB Fcn имеет вид:

```
function y = fcn(u)
y=eig(u); %Команда eig
%вычисляет собственные числа матрицы
```

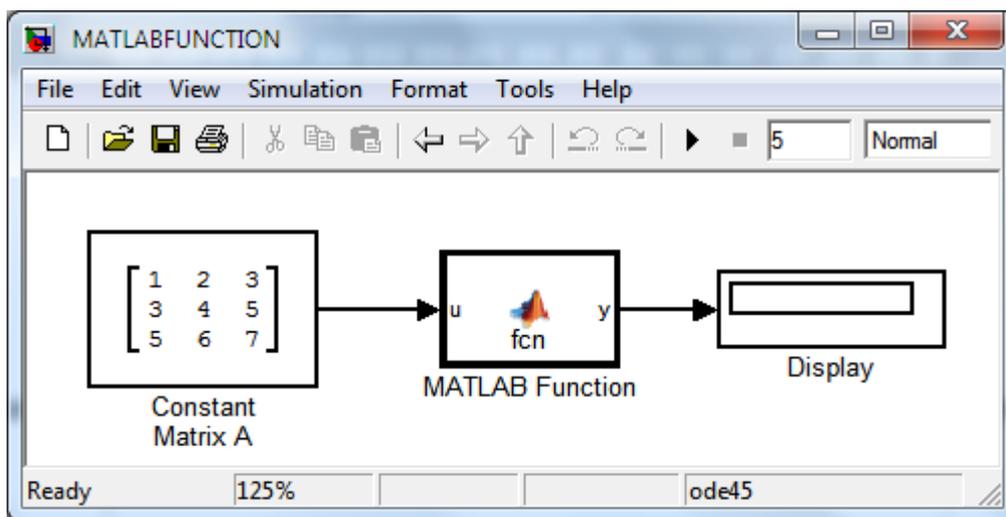


Рис. 6.37. Вид модели, вычисляющей собственные значения входной матрицы **A**

При формировании входной матрицы использовался блок Constant из библиотеки Sources. В поле **Constant value** окна параметров блока Constant задана матрица:

```
[1 2 3;3 4 5;5 6 7]
```

Для вывода на экран числовых значений использовался блок Display библиотеки Sinks.

## ЛИТЕРАТУРА

1. Достаточно общая теория управления. Постановочные материалы учебного курса факультета прикладной математики – процессов управления Санкт-Петербургского государственного университета (1997 – 2003 гг.). – СПб., 2003. – 419 с.
2. Миронов С.В. Метасистемный подход в управлении: Монография. / С.В. Миронов, А.М. Пищухин – Оренбург: ГОУ ОГУ, 2004.– 338 с.
3. Артамонов Д.В. Основы теории линейных систем автоматического управления: Учебн. пособие. / Д.В. Артамонов, А. Д. Семёнов. – Пенза: Изд-во Пенз. гос. ун-та, 2003.–135 с.
4. Гурко А.Г. Теория автоматического управления: Учебно-методическое пособие / А.Г. Гурко, И. Ф. Ерёменко и др. – Х., ХНАДУ, 2009. – 216 с.
5. Филипс Ч. Системы управления с обратной связью / Ч. Филипс, Р. Харбор. Пер. с англ. Б.И. Копылова. – М.: Лаборатория Базовых Знаний, 2001 – 616 с.:ил.
6. Дэбни Дж. Simulink 4. Секреты мастерства / Дж. Дэбни. – М.: Лаборатория знаний, 2003. – 403 с.
7. Гурко О.Г. Аналіз та синтез систем автоматичного управління у MATLAB: Навчальний посібник / О.Г. Гурко, І.Ф. Єрмоєнко. – Харків, ХНАДУ, 2012. – 303 с.
8. Бороденко В.А. Исследование систем управления в среде MATLAB: Монография. / В.А. Бороденко. – Павлодар : Кереку, 2011. – 318 с.
9. Suspension: System Modeling: Control Tutorials for MATLAB&Simulink. [Электронный ресурс] – Режим доступа: <http://ctms.engin.umich.edu/CTMS/index.php?example=Suspension&section=SystemModeling>
10. Hyundai. Открываем для себя завод Hyundai под Петербургом и седанчик Solaris. [Электронный ресурс] / Алексей Смирнов – 22 сентября 2010. – Режим доступа: [http://www.drive.ru/hyundai/news/2010/09/22/po\\_polnoy\\_programme.html](http://www.drive.ru/hyundai/news/2010/09/22/po_polnoy_programme.html).
11. Методы классической и современной теории автоматического управления: Учебник в 5-и тт.; 2-е изд., перераб. и доп. Т.1: Математические модели, динамические характеристики и

- анализ систем автоматического управления / Под ред. К.А. Пупкова, Н.Д. Егупова. – М.: Издательство МГТУ им. Н.Э. Баумана, 2004. – 656 с., ил.
12. Методы классической и современной теории автоматического управления: Учебник в 5-и тт.; 2-е изд., перераб. и доп. Т.2: Статистическая динамика и идентификация систем автоматического управления / Под ред. К.А. Пупкова, Н.Д. Егупова. – М.: Издательство МГТУ им. Н.Э. Баумана, 2004. – 640 с., ил.
  13. Техническая кибернетика. Теория автоматического регулирования. Книга 2. Анализ и синтез линейных непрерывных и дискретных систем автоматического регулирования. Под. Ред. В.В. Солодовникова. М.: Машиностроение, 1967. 682 с.
  14. Бесекерский В.А Теория систем автоматического управления/ В.А.Бесекерский, Е.П. Попов/ изд. 4-е, переб. и доп. – СПб, Изд-во «Профессия», 2003. – 752 с.
  15. Денисенко Виктор. ПИД-регуляторы: принципы построения и модификации Статья в СТА. В записную книжку инженера, 2006, №4, С. 66-74; 2007, №1, С.78-88.
  16. Derek Atherton. An introduction to Nonlinearity in Control Systems. Ventus Publishing, 2011, 176 pages.

Навчальне видання

БОГОМОЛОВ Віктор Олександрович  
ГУРКО Олександр Геннадійович  
КЛИМЕНКО Валерій Іванович  
ЛЕОНТЬЄВ Дмитро Миколайович  
КРАСЮК Олександр Миколайович

## МОДЕЛЮВАННЯ СИСТЕМ КЕРУВАННЯ В SIMULINK

*Навчальний посібник*

(російською мовою)

Відповідальний за випуск *Клименко В. І.*

Комп'ютерна верстка *Устименко М. О.*

Дизайн обкладинки *Нерівня Д.Ю.*

Редактор *Кузьміна Л.В.*

Підписано до друку 12.03.2018 р. Формат 60x84 1/16. Папір офсетний.

Гарнітура Times New Roman Суг. Віддруковано на ризографі.

Умовн. друк. арк. 12,79. Обл.-вид арк. 11,9.

Замовлення № 79/18. Тираж 50 прим. Ціна договірна.

### **Видавництво**

**Харківський національний автомобільно-дорожній університет  
Видавництво ХНАДУ, 61002, Харків-МСП, вул. Петровського, 25.  
Тел./факс: (057) 700-38-64; 707-37-03, e-mail: rio@khadi.kharkov.ua**

Свідоцтво Державного комітету інформаційної політики, телебачення та радіомовлення України про внесення суб'єкта видавничої справи до Державного реєстру видавців, виготівників і розповсюджувачів видавничої продукції, серія № ДК № 897 від 17.04.2002 р.