«Московский государственный технический
университет
имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Основы работы с вычислительной системой Matlab и пакетом визуального моделирования Simulink

Кафедра ИУ-2

ПРИБОРЫ И СИСТЕМЫ ОРИЕНТАЦИИ, СТАБИЛИЗАЦИИ И НАВИГАЦИИ

Малахов Андрей Анатольевич

2017

Кто, когда, и для чего? Клив Моулер (Cleve Moler) разработал MATLAB как язык программирования в конце 1970-х годов когда он был деканом факультета компьютерных наук в Университете Нью-Мексико.

Джон Литтл (*John N. (Jack) Little*) и Стив Бангерт (*Steve Bangert*)

Совместными усилиями они создали MATLAB на C и основали в 1984 компанию The MathWorks Inc (http://www.mathworks.com/).

В начале 80-х гг. Джон Литл разработал версии системы MATLAB для компьютеров класса IBM PC, VAX и Macintosh. В дальнейшем были созданы версии для рабочих станций Sun, компьютеров с операционной системой UNIX и многих других типов больших и малых ЭВМ.

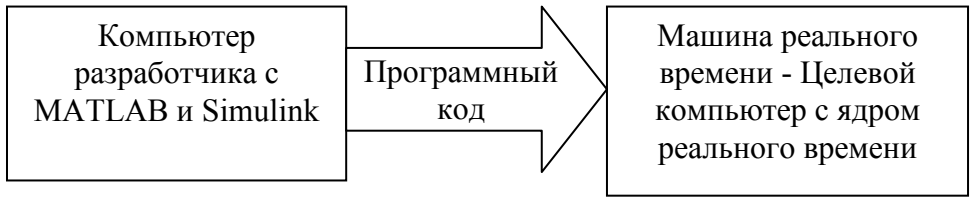
Первоначально MATLAB предназначался ***для проектирования систем управления*** (основная специальность Джона Литтла), но быстро завоевал популярность во многих других научных и инженерных областях. Он также широко использовался и в образовании, в частности, для преподавания линейной алгебры и численных методов.

Содержание курса	
Лекция 1. Введение.	Инсталляция системы, системные требования, основные компоненты рабочего пространства. Обзор библиотек Toolbox, Blockset. Главное меню, панель инструментов, управление окнами. Рабочая директория. Работа в командном окне, запись чисел и операторов, операции с векторами и матрицами. Функциональное программирование в Matlab. Функции и подпрограммы-функции, входные и выходные параметры, глобальные и локальные переменные. Оформление файлов функций. Editor: Редактор – отладчик. Запуск и трассировка программ. Вывод результатов. Графики. Сохранение данных в файл.
Лекция 2. Работа с пакетом Control. Линейные стационарные системы и их модели	Базовые классы объектов Matlab. Работа с передаточными функциями tf, моделями ss и zpk. Функции для работы с полиномами и характеристическими уравнениями. Анализ свойств динамических объектов. LTI-viewer.
Лекция 3. Система Simulink.	Библиотека Simulink. Сборка модели из блоков, задание параметров блоков.
Лекция 4. Библиотека пакетов и расширений Simulink.	Моделирование физических систем.

Лабораторный практикум	
Лабораторная работа № 1. Основы работы с Matlab	Рабочие окна системы, Help, Demo. Настройка основных параметров – меню Preferences Переменные, матрицы, векторы. Операторы Matlab Интерактивные вычисления в командной строке, построение графиков
Лабораторная работа № 2. Программирование в Matlab	Editor, отладка (трассировка) программ. Функциональное программирование
Лабораторная работа № 3. LTI-системы	Работа с передаточными функциями. Полиномы и характеристические уравнения. Анализ динамических систем, LTI-viewer
Лабораторная работа № 4. Основы работы с Simulink	Подготовка и настройка модели. Настройка решателя. Моделирование динамических систем (гиростабилизатор, самолет) Пакеты расширений – BlockSet. Моделирование физических систем (модель двигателя постоянного тока).

Литература	Полезные ссылки
1. Дьяконов В. П. MATLAB 7.*/R2006/R2007: Самоучитель. – М.: ДМК Пресс, 2008. – 768 с.: ил. 2. Дьяконов В. П. Matlab и Simulink для радиоинженеров. – М.: ДМК Пресс, 2011. – 976 с.: ил. 3. Черных И.В. Моделирование электротехнических устройств в Matlab, SimPowerSystems и Simulink. – СПб.: Питер, 2008. – 290.: ил. 4. Дьяконов В. П. MATLAB и SIMULINK для радиоинженеров. – М.: ДМК ПРЕСС МиМ. 2011. 976 с. ил.2 5. Черных И. В. Моделирование электротехнических устройств в MATLAB, SimPowerSystems и Simulink. – М.:ДМК Пресс; СПб.: Питер, 2008 г. – 288 с. ил.	1. http://www.mathworks.com – сайт компании The MathWorks 2. http://www.mathtools.net/ – научный портал, поддерживаемый компанией MathWorks 3. http://www.exponenta.ru – образовательный математический сайт 4. http://sl-matlab.ru/ – центр компетенций MathWorks 5. http://softline.ru/ – учебный центр Softline

MATLAB	Программная среда для технических расчетов: – de facto индустриальный стандарт, высокоуровневый язык программирования для разработки алгоритмов – Численные расчеты – Анализ данных и визуализация – Пакеты инструментов для обработки сигналов, обработки изображений, статистических расчетов, оптимизации, символьной математики – Основа всех продуктов MathWorks
SIMULINK	Визуальная среда для моделирования, симуляции, разработки динамических и встраиваемых систем: – Линейные, нелинейные, дискретные, непрерывные, гибридные и многоскоростные системы – Решения для систем управления, обработки сигналов, систем связи и других областях с применением системного инжиниринга – Основа для Модельно-Ориентированного Проектирования

Simulink Real-Time – ***От симуляции на десктопе к реальному времени*** **Машины реального времени Speedgoat** ***для Simulink Real-Time*****:** **1 Специализированный компьютер (Intel Core i7 2.5 GHz CPU, SSD до1 TB)** **2 I/O модули** **3 Драйверы, кабели и разъемы для подключения**	**Создание приложений реального времени из моделей Simulink и загрузка на выделенный компьютер за 3 автоматических шага:** **1 Автоматическая Генерация кода** **2 Сборка кода (Компилятор)** **3 Загрузка на целевой компьютер** 

Пакеты расширений системы Matlab

MathWorks MATLAB - **подраздел СИСТЕМЫ УПРАВЛЕНИЯ**
MathWorks Control System Toolbox Научные расчеты

Программное обеспечение MathWorks Control System Toolbox – пакет расширения MATLAB для анализа, проектирования и разработки систем автоматического управления. Control System Toolbox включает в себя всевозможные функции и графические приложения для работы с динамическими объектами и линейными замкнутыми системами управления...

MathWorks Simulink Control Design Научные расчеты

Программное обеспечение Simulink Control Design является комплектом расширений Simulink для линеаризации комплексных нелинейных объектов. Функции и графические инструменты Simulink Control Design позволяют проводить анализ линеаризованной модели в частотной области, настраивать параметры регулятора и синтезировать систему управления...

MathWorks Model Predictive Control Toolbox Научные расчеты

Программное обеспечение MathWorks Model Predictive Control Toolbox – это пакет расширения MATLAB для исследования и проектирования алгоритмов управления с предсказанием динамики. MathWorks Model Predictive Control Toolbox позволяет создавать системы адаптивного управления для сложных систем с одним или несколькими входами (выходами) и различными ограничениями...

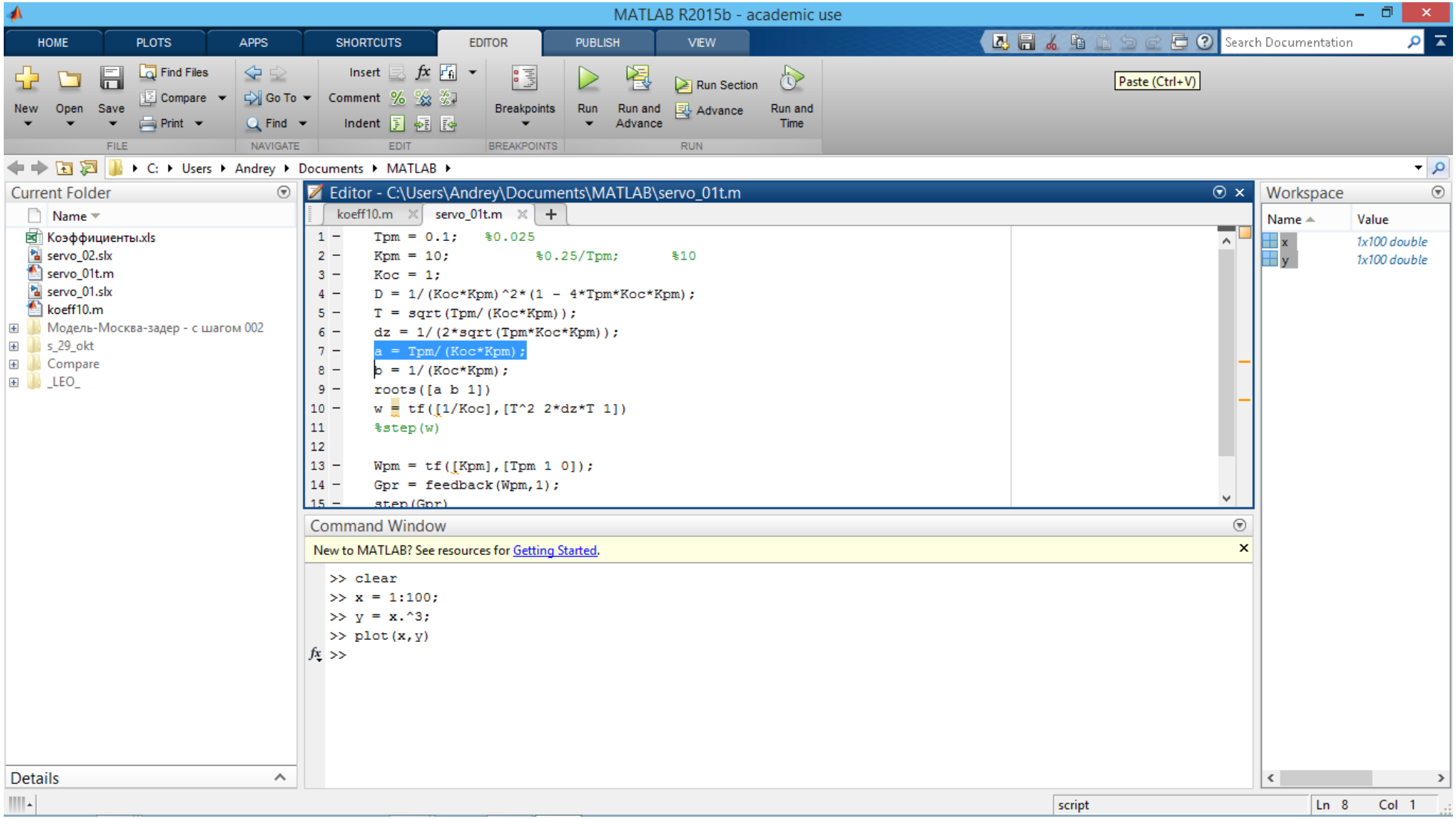
MathWorks Robust Control Toolbox Научные расчеты

Программное обеспечение MathWorks Robust Control Toolbox – это пакет расширения MATLAB для разработки систем управления объектами с неопределенностями и нелинейностями различного типа. MathWorks Robust Control Toolbox позволяет проектировать и настраивать системы управления с учетом чувствительности к неопределенным параметрам, возмущениям и ошибкам модели...

MathWorks Model-Based Calibration Toolbox Машиностроение

Программное обеспечение MathWorks Model-Based Calibration Toolbox – это пакет расширения MATLAB, предназначенный для калибровки моделей сложных систем и механизмов. Пакет MathWorks Model-Based Calibration Toolbox опирается на высокотехнологичные вычислительные средства MATLAB и широкие возможности имитационного моделирования Simulink...

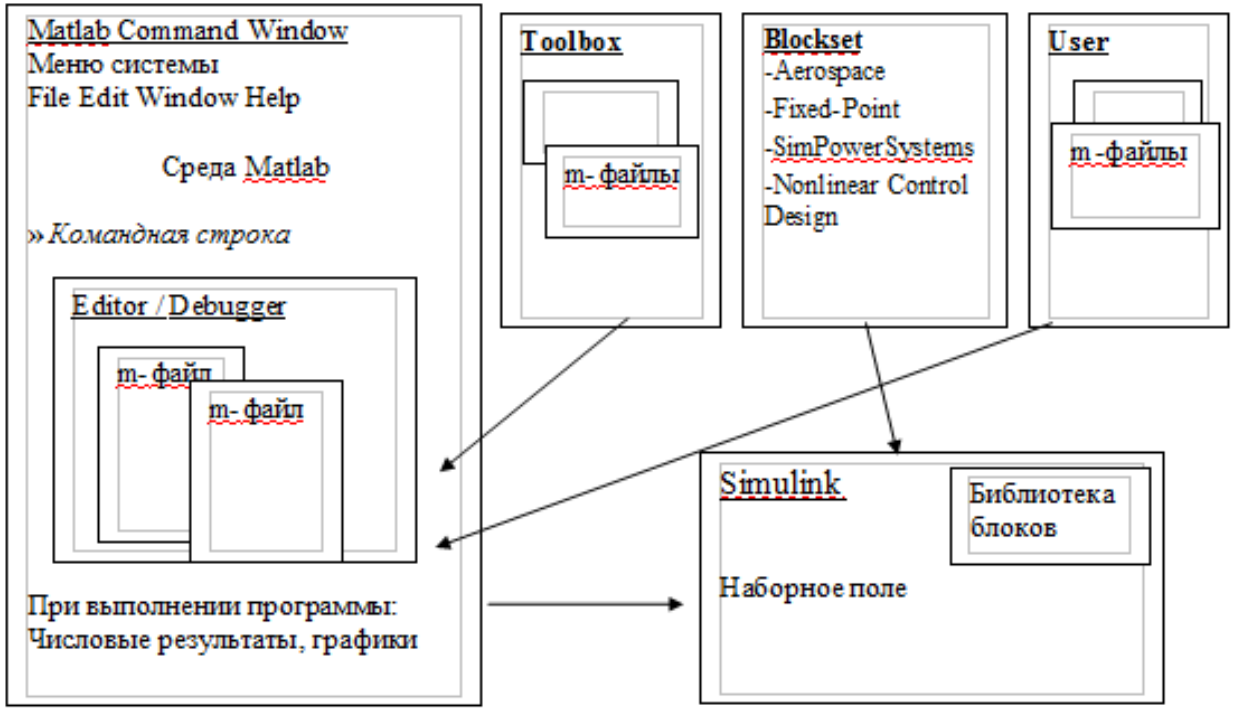
Интерфейс системы Matlab При запуске системы открывается рабочий стол (**desktop**) системы:



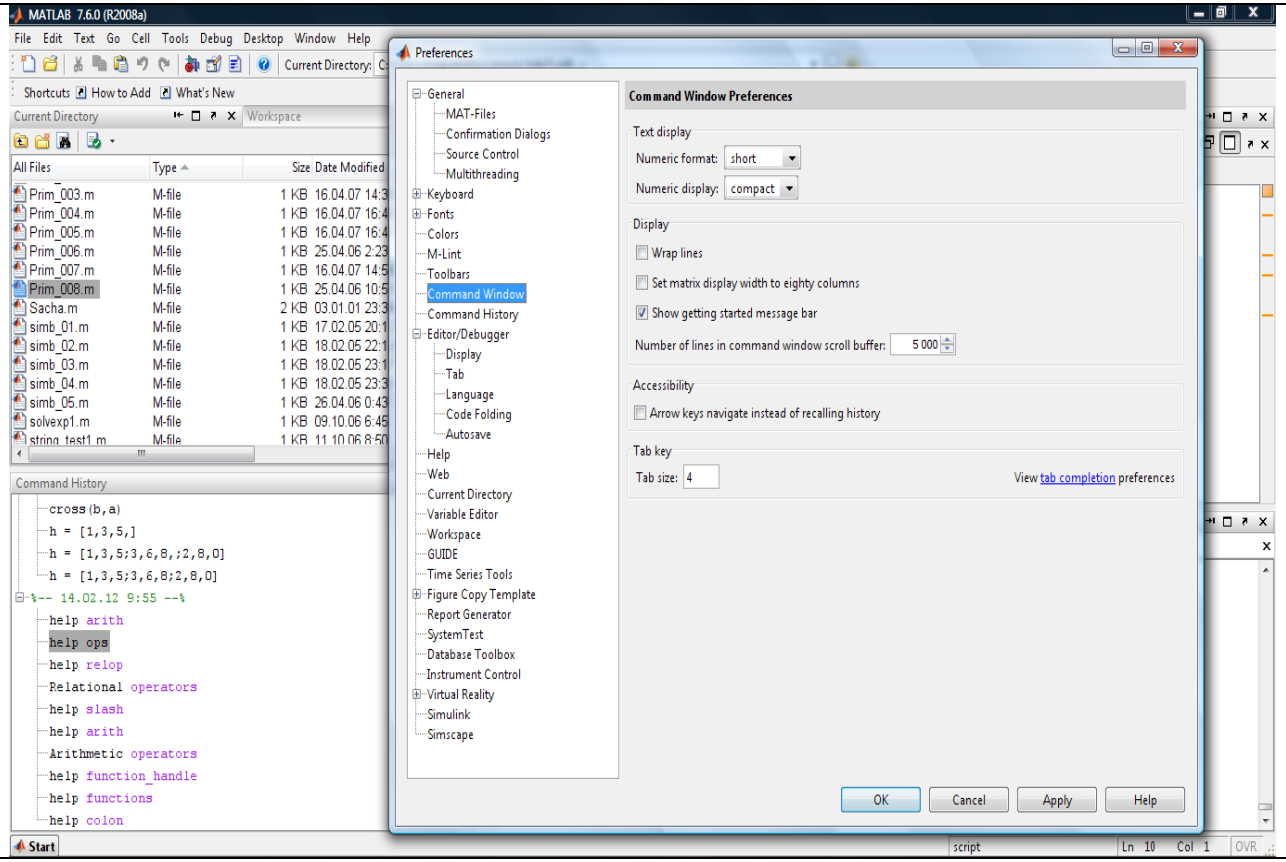
Интерфейс рабочего стола содержит: - меню, расположенное в верхней части окна, выделенное серым цветом заливки; содержит вкладки **HOME**, **PLOTS**, **APPS**; начинать работу следует на вкладке **HOME;**	Набор окон и их расположение можно настроить, используя кнопку меню Layout и перемещая мышкой окна и их границы.

- адресную строку – указатель рабочего каталога, позволяющую выбрать рабочую директорию; - окно в центре – **Command Window** в котором появляется двойная стрелка (>>) указывающая на начало командной строки; - окно слева – **Current Folder** в котором отображаются файлы текущего каталога и определенные данные описания их свойств; - окно справа – **Workspace**, представляющие все глобальные переменные, использованные в командах или при выполнении программ; - окно справа внизу - **Command History,** в котором повторены все команды, выполненные в командной строке.	Для создания программы (m – файла) необходимо использовать кнопку New в верхней левой части меню. При этом будет открыто окно редактора – Editor, которое можно встроить в блок окон, щелкнув по кнопке **Dock**, затем мышкой переместить его в удобное место, например, над **Command Window**. Далее, в процессе работы могут появиться другие окна, например графики (Figure) или интерфейс приложений (например, при вызове LTI viewer). Вызов библиотеки Simulink осуществляется нажатием кнопки меню Simulink Library.

Схема рабочего пространства



Настройка основных параметров – меню Preferences



Основы программирования в MATLAB

Matlab – интерпретатор и система программирования

A. Работа в командном окне.

Примеры 1: Запись чисел и операторов (Демонстрация примера Prim_001.m)

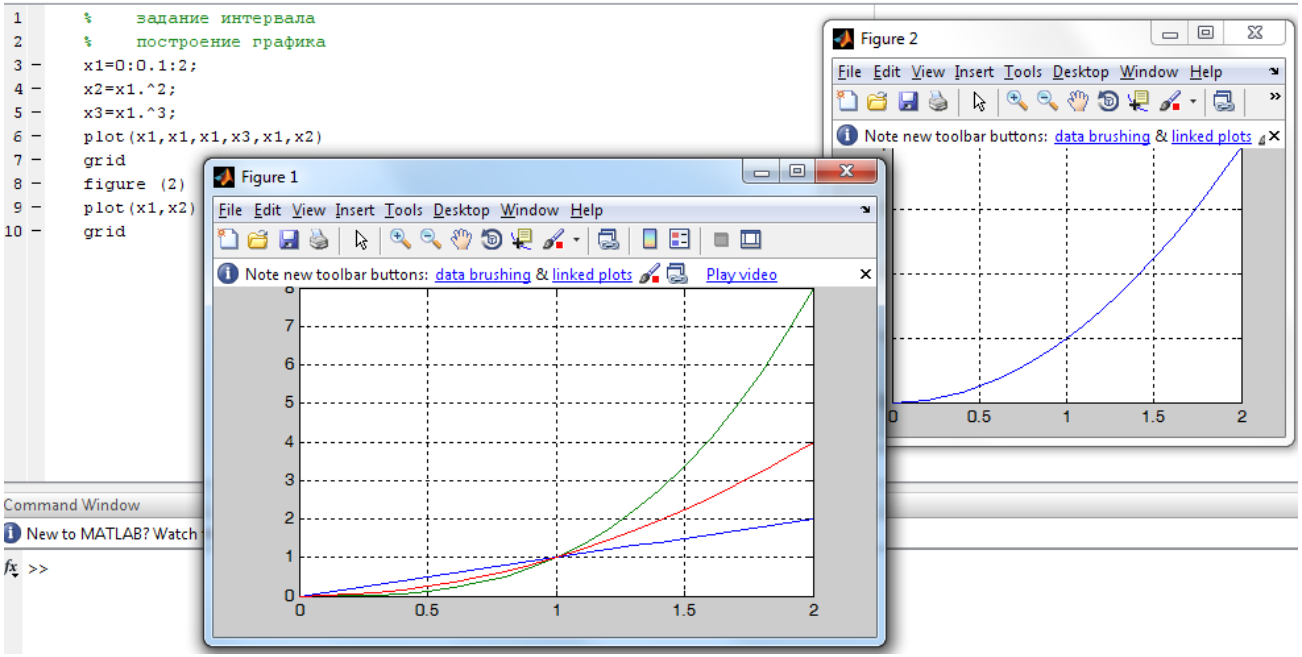
B. Программирование в Matlab. Функциональное программирование в Matlab

Простые переменные, матрицы и векторы — правила записи, встроенные константы, комплексные числа

Форматы чисел, установка в меню и командой (оператор) format, вывод результатов в рабочую область, сохранение в файл

Операторы: арифметические, особенности умножения, деление (/ **и** \), двоеточие (:), транспонирование

Формирование векторов и матриц, операции с векторами и матрицами.



C. Справочная система Help и демонстрационные примеры Demo.

D. Примеры

Пример – расчет переходного процесса, в примере рассматриваем:

- переход от операторных уравнений движения (передаточных функций) к уравнениям во временной области (в нормализованном виде)
- оформление цикла
- формирование матрицы – таблицы выходных параметров
- оформление программы интегрирования в виде подпрограммы-функции с входными и выходными параметрами

Математические выражения

В состав математических выражений входят: переменные, числа, операторы, функции.

Переменная – это массив (матрица), вектор или скаляр.

Matlab не требует какого-либо описания типа переменной или размерности массива.

Имена переменных, констант и функций могут быть составлены из любых символов **латинского алфавита**, кроме специальных и цифр, **начинаются с буквы**.

Для идентификации переменной используются первые 31 символ имени.

Системные константы

pi = 3.14159265358979

i = мнимая единица, то же **j**

Inf = бесконечность, результат деления на 0

NaN = неопределенное значение (Not a Number), 0/0, inf-inf.

eps = 2^{-52} или 2.2204e-016 – характеризует точность вычислений с плавающей точкой

Ввод и вывод данных

В языке MATLAB нет явных операторов ввода-вывода данных.

Результаты работы системы при выполнении команд или М- файла программы выводятся на рабочее поле экрана и на графики, для которых открываются специальные окна.

Справка – Help – Операторы Matlab

>> help ops

>> help **ops** – Operators and special characters.

See also: **arith** – Arithmetic operators

relop – Relational operators

slash – Matrix division

Операторы Matlab

Язык Matlab это язык операторов. Операторы задаются по одному в командной строке для исполнения в интерактивном режиме или в виде списка в **m**- файле или в **script**- файле. Фактически **m**- файл является программой, которая интерпретируется и выполняется системой Matlab. После выполнения **m**- файла в операционной среде системы, до завершения сеанса, остаются все значения глобальных переменных и они доступны для выполнения любых действий путем задания операторов в командной строке.

Для построения выражений используются арифметические операторы и знаки, указывающие на порядок действий (**»help ops**).

Признаком начала командной строки является указатель >>, выводимый системой автоматически, при готовности к приему команды. В тексте программ этот символ не нужен!

Две формы записи операторов:

- с явным присвоением: ***переменная = выражение;***

- с неявным присвоением: ***выражение;***

Оператор содержит:

- имена переменных и числовые константы;
- имена функций;
- специальные символы указывающие на выполняемые действия + - * / ^ ' и на порядок действий ();
- символ (**;**) указывающий на завершение строки матрицы и на подавление вывода результата на экран;
- разделители операторов при записи более одного оператора в строке (,);
- символы продолжения строки для записи более 256 символов – точки, не менее двух;
- пробелы, в любых местах, они не влияют на выполняемые действия и служат для оформления строки;
- символ % указывает на то, что следующие за ним символы являются комментарием, с него можно начать строку.

Синтаксис операторов

В MATLAB используются все буквы латинского алфавита от A до Z, цифры от 0 до 9.

Большие и малые буквы – различаются системой

Специальные символы применимы ограниченно, по сравнению с офисными системами!

Для построения выражений используются арифметические операторы и знаки, указывающие на порядок действий

Операторы Арифметические: + - * /

Логические

Логическое И	&; and (and (a, b))
Логическое ИЛИ	| ; or (or (a, b))
Логическое НЕ	~ ; not (not (a, b))
Исключающее ИЛИ	xor (xor (a, b))
Верно, если все элементы вектора равны нулю	any (any (a))
Верно, если все элементы вектора не равны нулю	all (all (a))

Отношения

Логическое И	&; and (and (a, b))
Логическое ИЛИ	| ; or (or (a, b))
Логическое НЕ	~ ; not (not (a, b))
Исключающее ИЛИ	xor (xor (a, b))
Верно, если все элементы вектора равны нулю	any (any (a))
Верно, если все элементы вектора не равны нулю	all (all (a))

Синтаксис операторов - arithmetic operators

Для арифметических операторов есть эквивалентные функции:

Binary addition	A+B	plus(A,B)
Unary plus	+A	uplus(A)
Binary subtraction	A-B	minus(A,B)
Unary minus	-A	uminus(A)
Matrix multiplication	A*B	mtimes(A,B)
Arraywise multiplication	A.*B	times(A,B)
Matrix right division	A/B	mrdivide(A,B)
Arraywise right division	A./B	rdivide(A,B)
Matrix left division	A\B	mldivide(A,B)
Arraywise left division	A.\B	ldivide(A,B)
Matrix power	A^B	mpower(A,B)
Arraywise power	A.^B	power(A,B)
Complex transpose	A'	ctranspose(A)
Matrix transpose	A.'	transpose(A)

Note For some toolboxes, the arithmetic operators are overloaded – операторы выполняют расширенные функции, определенные в рамках конкретных пакетов расширений (see the documentation for the toolbox).

Форматы чисел

Диапазон представления чисел при вычислениях 10^{-308} – 10^{308}, все внутренние вычисления производятся с двойной точностью.

Запись действительных чисел выполняется:

- в десятичной форме, знак плюс и точка у целых чисел не обязательны.

- в показательной форме по основанию десять; показатель степени отделен от мантиссы символом e или E, пробел не допускается;
- явного определения целых чисел нет;

- комплексное число представлено действительной и мнимой частями, при мнимой части проставлен символ i или j (без знака умножения);

>> help format

FORMAT Set output format.

FORMAT SHORT Scaled fixed point format with 5 digits.

FORMAT LONG Scaled fixed point format with 15 digits for double and 7 digits for single.

FORMAT SHORT E Floating point format with 5 digits.

FORMAT LONG E Floating point format with 15 digits for double and 7 digits for single.

FORMAT SHORT G Best of fixed or floating point format with 5 digits.

FORMAT LONG G Best of fixed or floating point format with 15 digits for double and 7 digits for single.

FORMAT SHORT ENG Engineering format that has at least 5 digits and a power that is a multiple of three

FORMAT LONG ENG Engineering format that has exactly 16 significant digits and a power that is a multiple of three.

Запись операторов в командном окне Matlab

При записи оператора с неявным присвоением, результат вычислений присваивается автоматически внутренней переменной **ans** (answer), может быть вызван и использован по этому имени и сохраняется до выполнения следующего оператора с неявным присвоением.

Примеры записи переменных, скаляров и матриц, выполнение простых операторов:

» a = 3; b = -5; % символ ; подавляет эхо-вывод

» c = (a + b)/2

c =

-1

» pi % встроенная константа

ans =

3.1416

» i % комплексные числа, i - встроенная константа

ans =

0 + 1.0000i

Matlab – Элементарные функции

Вызов справки

help elfun

$\|x\|$ – модуль	abs(x)
e^x – экспонента	exp(x)
$\|x\|$ – модуль	abs(x)
e^x – экспонента	exp(x)
– натуральный логарифм	log(x)
– логарифм по основанию 2	log2(x)
– десятичный логарифм	log10(x)
2^x – 2 в степени x	pow(x)
– квадратный корень	sqrt(x)

Все встроенные элементарные функции должны записываться в программах *малыми буквами!*

Специальные функции сгруппированы по разделам

help **elmat** – Elementary matrices and matrix manipulation (Zeros, Ones, Size)

help **specfun** – Specialized math functions (Factorial, Coordinate transforms)

Векторы и матрицы

Ввод матриц

» a = [1 2 3; 4 5 6; 7 8 9];

Или так:

```
» a = [1  2  3;
        4  5  6;
        7  8  9];
» a = [1 2; 3 4];          % сложение матриц
» b = [5 6; 7 8];
» c = a + b       % с присвоением переменной
c =
    6     8
   10    12
 » a + b          % с неявным присвоением
ans =

    6     8
   10    12
» a(1,1)          % задание одного элемента матрицы
ans =
    1
» a = [1   2   3
        4   5   6
        7   8   9];
» a(2,3)
ans =
    6
» a
» a'      % транспонирование
```

```
ans =
   1   4   7
   2   5   8
   3   6   9
```

Формирование векторов и матриц- символы (,)(***comma***) и (;)(***semicolon***)

Prim_002.m

```
%    присоединение столбца
a = [1 2
        3 4]
B = [5
        6]
s1=[a,b]
%    или строки к матрице
c=[5 6]
s2=[a;c]
```

Сложение и вычитание вектора-строки и вектора-столбца или векторов разных размеров приводит к ошибке. Операция * предназначена для умножения векторов по правилу матричного умножения. Поскольку MatLab различает вектора-строки и вектора-столбцы, то допустимо либо умножение вектора-строки на такой же по длине вектор-столбец (скалярное произведение), либо умножение вектора-столбца на вектор-строку (внешнее произведение, в результате которого получается прямоугольная матрица).

Функции обработки векторов

Функции	Назначение
s=sum(a)	Сумма всех элементов вектора **a**
p=prod(a)	Произведение всех элементов вектора **a**
m=max(a)	Нахождение максимального значения среди элементов вектора **a**
[m,k]=max(a)	Второй выходной аргумент **k** содержит номер максимального элемента в векторе **a**
m=min(a)	Нахождение минимального значения среди элементов вектора **a**

[m,k]=min(a)	Второй выходной аргумент **k** содержит номер минимального элемента в векторе **a**
m=mean(a)	Вычисление среднего арифметического элементов вектора **a**
a1=sort(a)	Упорядочение элементов вектора по возрастанию
[a1,ind]=sort(a)	Второй выходной аргумент **ind** является вектором из целых чисел от 1 до **length(a)**, который соответствует проделанным перестановкам
L=length(a)	Нахождение длины вектора
x1 – fliplr(x)	Переворот вектора (матрицы) слева направо
y1 = rot90(y)	Поворот матрицы на 90 градусов против часовой стрелки
s=dot(a,b)	Скалярное произведение двух векторов
c=cross(a,b)	Векторное произведение определено только для векторов из трех элементов.

Вызвав справку по приведенным в таблице функциям (**help funname**) можно найти много других функций, предназначенных для преобразования векторов и матриц.

Для операции **транспонирования** зарезервирован апостроф **'**. Если вектор содержит комплексные числа, то операция ' приводит к комплексно-сопряженному вектору. При вычислении скалярного и векторного произведений функциями **cross** и **dot** не обязательно следить за тем, чтобы оба вектора были либо столбцами, либо строками. Результат получается верный, например, при обращении **c=cross(a,b')**, только **c** становится вектором-строкой.

Для обработки матриц также существуют специальные функции, например, функции для создания стандартных матриц: **zeros**, **eye**, **ones**, **rand**, **diag** (см. **help matlab\elmat**).

Оператор (:) (Colon)

```
» a = [1 2 3 4
       5 6 7 8];
» sum(a(:,1))% знак : => операция со столбцом
ans =
     6
» a(2,:)
ans =
     5     6     7     8
»
```

```
» 1:6                % интервал значений целых чисел
ans =
     1     2     3     4     5     6
» a = 1:5       % то же с присвоением
a =
     1     2     3     4     5

» a = 1.1:5.5   % с десятичными знаками,
                % шаг единица
a =
    1.1000    2.1000    3.1000    4.1000    5.1000
»
» a = 0.1:0.1:0.5   % дробный шаг
a =
    0.1000    0.2000    0.3000    0.4000    0.5000
```

Индексация двоеточием позволяет выделить идущие подряд элементы в новый вектор. Начальный и конечный номера указываются в круглых скобках через двоеточие, например:

```
» z =[ 0.2 -3.8 7.9 4.5 7.2 -8.1 3.4];
» znew = z(3:6)
znew =
7.9000 4.5000 7.2000 -8.1000
```

Вызов функции **prod** с заданием интервала с помощью двоеточия вычисляет произведение элементов вектора **z** со второго по шестой:

```
» p=prod(z(2:6))
```

Указание номеров элементов вектора можно использовать и при вводе векторов, последовательно добавляя новые элементы (не обязательно в порядке возрастания их номеров). Команды:

```
>> h=10;  h(2)=20;  h(4)=40;
```

приводят к образованию вектора:

```
>> h
h =
10 20 0 40
```

Для ввода первого элемента **h** не обязательно указывать его индекс, т.к. при выполнении оператора **h=1** создается вектор (массив размера один на один). Следующие операторы присваивания приводят к автоматическому увеличению длины вектора **h**, а пропущенные элементы (в нашем случае **h(3)**) получают значение ноль.

Индексация вектором служит для выделения элементов с заданными индексами в новый вектор. Индексный вектор должен содержать номера требуемых элементов, например:

```
» z=[0.2 -3.8 7.9 4.5 7.2 -8.1 3.4];
>> ind=[3 5 7];
>> znew=z(ind)
znew =
7.9000 7.2000 3.4000
```

Вектора-столбцы с одинаковым числом элементов можно складывать и вычитать друг из друга при помощи знаков "+" и "–". Такое действие верно и для векторов-строк:

```
>> c=a+b;
>> w=u-v;
```

Особенности оператора умножения (деления, возведения в степень)

MatLab поддерживает два вида вычислительных операций с векторами и матрицами: матричные, выполняемые по правилам линейной алгебры и табличные, выполняемые поэлементно. Знак-точка (.) отличает табличные операции от матричных. Так, наряду с умножением по правилу матричного умножения, существует операция поэлементного умножения .* (точка со звездочкой). Данная операция применяется к векторам одинаковой длины и приводит к вектору той же длины, что исходные, элементы которого равны произведениям соответствующих элементов исходных векторов. Аналогично может быть выполнена операция с матрицами одинаковой размерности, в этом случае матрицы - операнды обрабатывается системой как таблицы, выполняется перемножение соответствующих элементов таблиц, результатом будет матрица такой же размерности. Например, для введенных ранее матриц a и b:

операция с матрицами	**» a * b %** **ans =** **19 22** **43 50**
операция с массивами (таблицами	**» a.* b %)** **ans =** **5 12** **21 32**

Аналогичным образом выполняется поэлементное деление ./ (точка с косой чертой). Кроме того, операция .\ (точка с обратной косой чертой) осуществляет обратное поэлементное деление, то есть выражения a./b и b.\a эквивалентны. Возведение элементов вектора a в степени, равные соответствующим элементам b, производится с использованием .^. Для транспонирования векторов-строк или векторов-столбцов предназначено

сочетание .' (точка с апострофом). Операции ' и .' для вещественных векторов приводят к одинаковым результатам. Не требуется применять поэлементные операции при умножении вектора на число и числа на вектор, делении вектора на число, сложении и вычитании вектора и числа. При выполнении, например, операции a*2, результат представляет собой вектор того же размера, что и a, с удвоенными элементами.

Справочная система (HELP)

Для обращения к справочной системе необходимо в командном окне MATLAB набрать команду

» help

При этом будет представлен перечень разделов (HELP topics) справочной системы. Ниже приведены разделы, на которые следует обратить внимание в первую очередь.

» help matlab\ops - выводит перечень операторов и специальных символов, используемых в системе.

» help arith - об арифметических операторах,

» help punct - об использовании специальных символов в командах,

» help colon - о применении специального символа **:** (двоеточие), который управляет выполнением ряда важных операций с матрицами.

» help matlab\lang - описание языка системы для работы в режиме интерпретации команд и программирования (написания М- файлов).

» help matlab\elmat- простые матрицы и базовые операции с матрицами.

» help matlab\elfun - элементарные, базовые функции системы, в том числе тригонометрические, экспоненциальные, обработки комплексных чисел и т.д.

» help matlab\matfun - функции линейной алгебры и матричного анализа.

» help matlab\polyfun - функции работы с полиномами и интерполяции.

» help matlab\plotxy - построение графиков по двум координатным осям.

Программирование в Matlab Операторы и функции

Аппроксимация данных полиномом

Prim_aprox1.m

```
 x=1:11
y=[3 8 9 4 3 9 10 4 2 1 0]
k=polyfit(x,y,8)
z=polyval(k,x)
plot(x,y,x,z,x,y-z)
grid
x1 = 1:.1:11;
z1 = polyval(k,x1);
plot(x,y,x1,z1,x,y-z)
grid
```

Формирование вектора выходных значений на примере интегрирования дифференциального уравнения

Исходные данные: передаточная функция

Алгоритм: метод Эйлера

Prim_001.m

```
     T=3; k = 2; h = T/10;
     y=0; py=0; x0=1;
     yout=[0];
hold on
grid
for i =1:100,
   y1=y + h*(-1/T*y + k/T*x0);
   yout=[yout;y1];
   y=y1;
   plot(i,y,'*r')   %    точка на графике
end  % for i
hold off
     figure(2)
     yout
     length(yout)
     n=1:length(yout);
     plot(n,yout)
     grid
```

Программы Matlab – скрипты, подпрограммы – функции создаются в виде m-файлов, которые по существу являются простыми текстовыми файлами, которым задано расширение .m и содержат последовательность команд-операторов Matlab.

Программирование, редактирование и отладка скриптов (программ) и функций (подпрограмм) осуществляется в окне редактора/отладчика.

Editor - редактор М – файлов, реализует интерфейс редактирования и отладки (***debugging***).

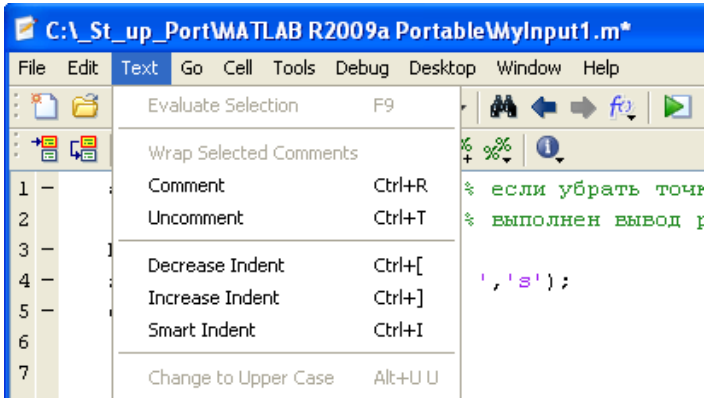
Для создания нового m-файла необходимо выполнить команду «New Script» (New > M-file) из меню File, по которой будет вызван редактор с открытым окном для записи скрипта или функции. В редакторе могут быть одновременно открыты несколько m-файлов.

Запустить программу можно только после сохранения созданного или измененного файла. Запуск и сохранение можно выполнять одним нажатием кнопки RUN, в том случае, если был создан новый файл, будет открыто диалоговое окно сохранения, в котором необходимо указать имя файла и путь к рабочему каталогу. В случае запуска программы после внесения изменений файл сохраняется автоматически.

В редакторе можно набрать команды, например для построения графика:

```
x=[-1:0.01:1];
y=exp(x);
plot(x,y)
grid on
title('Экспоненциальная функция')
```

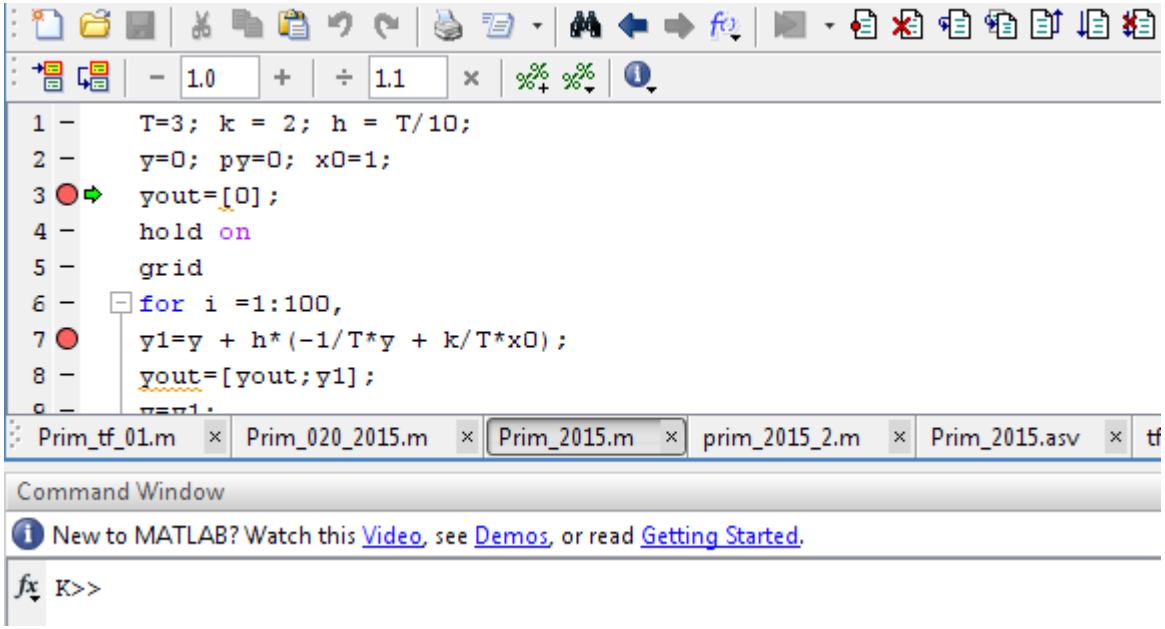
Меню Text позволяет управлять оформлением и форматированием строк программы



Редактор выполняет синтаксический контроль программного кода по мере ввода текста и осуществляет следующие цветовые выделения:

- ключевые слова языка программирования — синий цвет;
- операторы, константы и переменные — черный цвет;
- комментарии после знака % — зеленый цвет;
- символьные переменные (в апострофах) — зеленый цвет;
- синтаксические ошибки — красный цвет.

Отладка программы осуществляется через интерфейс редактора путем задания точек останова (breakpoints) щелчком левой кнопки мыши по столбцу справа от номера строки.

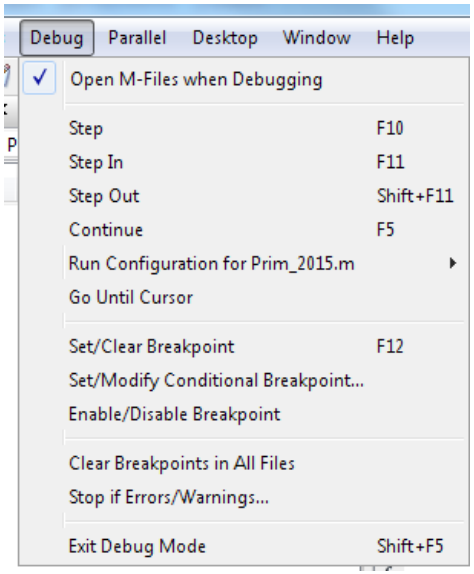


Чтобы начать отладку вам необходимо запустить программу на выполнение, в точке останова появляется зеленая стрелка-указатель начала следующего оператора, а в Command Window появляется символ **K>>**, указывающий на переход управления на клавиатуру, что позволяет выполнить команду просмотра переменных или выполнение вспомогательного расчета с использованием переменных из рабочей области.

При останове можно наблюдать изменение переменных программы в окне Workspace.

Текст ошибок, возникающих во время выполнения функции будет выводиться в командное окно.

Дальнейшее выполнение программы можно выполнить по шагам командой Step или командой Continue до следующей точки останова.



Отладчик позволяет проверять ход вычислений внутри функций (подпрограмм) по команде Step In, при этом текст функции будет открыт в новом окне редактора, а в окне Workspace появятся и будут доступны для просмотра все локальные переменные функции. Выход из функции с выполнением всех операторов функции осуществляется по команде Step Out.

Файл-функция и файл-программа

Файл-функции отличаются от файл-программы тем, что они могут иметь входные и выходные аргументы, которые являются глобальными переменными, а все переменные, определенные внутри файл-функции, являются локальными и не видны в рабочей среде.

Пример простейшей файл-функции с двумя входными и одним выходным аргументами.

М-файл, содержащий файл-функцию, должен начинаться с заголовка, после него записываются операторы MatLab.

Заголовок состоит из слова **function**, списка выходных аргументов, имени файл-функции и списка входных аргументов. Аргументы в списках разделяются запятой.

Специальный оператор завершения текста файл-программы не требуется!

В тексте файл-функции должны быть записаны операторы вычисления выходных переменных**.**

Входными аргументами **mysum** могут быть массивы одинаковых размеров или массив и число.

Список выходных аргументов в заголовке файл-функции заключается в квадратные скобки, сами аргументы отделяются запятой.

В качестве примера ниже приведена файл-функция **quadeq**, которая по заданным коэффициентам квадратного уравнения находит его корни:

```
function [x1,x2]=quadeq(a,b,c)
D=b^2-4*a*c;
x1=(-b+sqrt(D))/(2*a);
x2=(-b-sqrt(D))/(2*a);
```

При вызове **quadeq** из командной строки используйте квадратные скобки для указания переменных, в которые будут занесены значения корней. В конце строки не ставим точку с запятой и система выдает ответ на экран в командное окно.

```
>> [r1,r2]= quadeq(1,3,2)
r1 =
-1
r2 =
-2
```

Организация циклов - *Repeat statements*

for, if, while, switch, break, continue, end, colon

Операторы отношений – ***relation operation (rop)***

```
 ==,  <,  >,  <=,  >=,  ~=
```

```
for R = 1:N
            for C = 1:N
                A(R,C) = 1/(R+C-1);
            end
 end
```

Возможности ввода-вывода данных в Matlab

Функция input выводит на экран запрос и ждет ответа пользователя с клавиатуры: Функция присвоит назначенной переменной введённое с клавиатуры значение.	`a = input('Enter a = ');` `b = a^2;`
В данном примере, явно указав выполнение табличной операции, можно ввести вектор или матрицу.	`a = input('Enter a = ');` `b = a.^2;`

Функция input может также возвращать не числовое, а строковое выражение, для этого необходимо добавить признак текстовой переменной 's' к списку параметров функции	`a = input('Enter a = ');`
	`b = a.^2;`
	`ad = input('Adress:','s')`

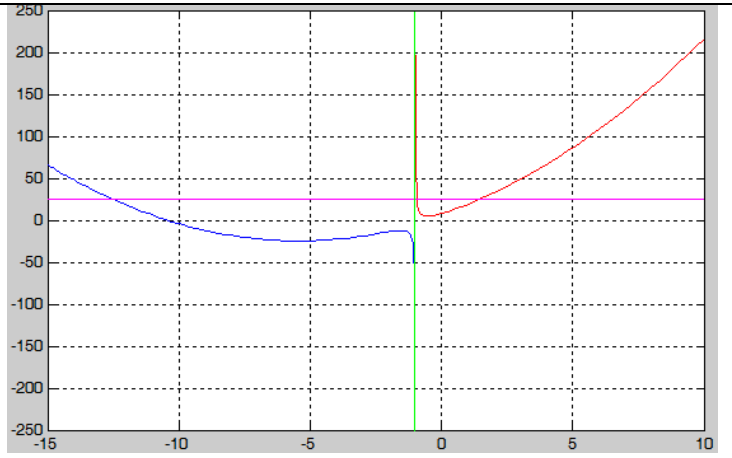
Если с клавиатуры будет введено арифметическое выражение, в том числе с использованием уже существующих в рабочей области переменных, функция вычисляет его и возвращает соответствующее значение.	`a = input('Enter a = ');`
	`b = a.^2;`
	`soop = input('оператор d = ','s')`
	`d = eval(soop)`

Построение графиков

Prim13_1.m	

`% график табличных данных функции`
`% аргумент задан на двух интервалах`
`clf reset`
`x2 = -0.99:0.01:5;`
`x1 = -10:0.01:-1.01;`
`x = [x1, x2];`
`y = (x.^3 + 12 *x.^2 + 17 *x + 8)./(x + 1);`
`plot(x,y); grid`	

Prim13_1_1.m	$y = \frac{x^3+12x^2+17x+8}{x+1}$

`% график аналитической функции`
`figure(1)`
`clf reset`
`fplot('(x.^3 + 12 *x.^2 + 17 *x + 8)./(x + 1)',[-15 -1.01])`
`hold on`
`fplot('(x.^3 + 12 *x.^2 + 17 *x + 8)./(x + 1)',[-0.99 10], 'r')`
`fplot('25', [-15 10],'m-')`
`plot([-1 -1],[-250 250],'g-')`
`grid on`
	

Полезные команды и функции, разделы справочной информации

>> clear - очистка **Workspace**
>> clear var – очистка переменной **var**
>> clear globals – очистка глобальных переменных
>> clc – очистка командного окна
>> home – возврат курсора в ВЛУ
>> clf reset – очистка окна графика

Просмотр рабочего пространства в процессе решения задач

```
 >> who
Your variables are:
 T     ans   h     i     k     n     py    x0    y     y1    yout
>> whos
Name        Size                    Bytes  Class
T           1x1                         8  double array
ans         1x1                         8  double array
```

```
 >> help filename
```

>> type filename – вывод строк комментариев до первой исполняемой команды

>> lookfor keyword - выполнить поиск М-функции по ключевому слову; при этом анализируется первая строка комментария

```
>> help global
global Define global variable.
    global X Y Z defines X, Y, and Z as global in scope.
isglobal True for global variables.
isglobal is obsolete and will be discontinued in a future version of MATLAB.
```

Пакет Control, LTI – объекты

Классы вычислительных объектов в MATLAB

Классом в MatLAB принято называть определенную форму представления вычислительных объектов в памяти компьютера в совокупности с правилами (процедурами) их преобразования.
Класс определяет тип переменной, а правила — операции и функции, которые могут быть применены к этому типу. В свою очередь, тип определяет объем памяти, которая отводится под запись переменой в память и структуру размещения данных в этом объеме.
Операции и функции, которые могут быть применены к определенному типу переменных, образуют методы этого класса.

Пакет прикладных программ (ППП) Control System Toolbox (сокращенно —CONTROL) сосредоточен в подкаталоге CONTROL каталога TOOLBOX системы MatLAB.
Основными вычислительными объектами этого ППП являются:
- родительский объект (класс) LTI (Linear Time-Invariant System — линейные, инвариантные во времени системы); в русскоязычной литературе за этими системами закрепилось название линейные стационарные системы (ЛСС).
- дочерние объекты (классы), т.е. подклассы класса LTI, соответствующие разным представлениям ЛСС:

Model Type	**Description**
tf	Transfer function model in polynomial form
zpk	Transfer function model in zero-pole-gain (factorized) form
ss	State-space model
frd	Frequency response data model
pid	Parallel-form PID controller (Proportional-Integral-Derivative)

Объект LTI, как наиболее общий, содержит информацию, не зависящую от конкретного представления ЛСС, а также от имен входов и выходов.
Дочерние объекты определяются конкретной формой представления ЛСС, т.е. зависят от модели представления.
Объект класса TF характеризуется векторами коэффициентов числителя и знаменателя рациональной передаточной функции.
Объект класса ZPK характеризуется векторами, содержащими значения нулей, полюсов передаточной функции системы и коэффициента передачи системы.
Объект класса SS определяется четверкой матриц, описывающих динамическую систему в пространстве состояния.
Объект класса FRD создается на основании отклика системы на заданном частотном спектре и ориентирован на применении при проведении измерительных экспериментов.
Виды LTI-систем:
SISO (Single In Single Out) — одномерная (ОМ) система , т.е. система с одним входом и одним выходом.

MIMO (Multiple Input Multiple Output) — многомерная (ММ) система с несколькими входами и выходами.
TF-объект (Transfer Function — передаточная функция);
ZPK-объект (Zero-Pole-Gain — нули-полюсы-коэффициент передачи);
SS-объект (State Space — пространство состояния).

Функции создания LTI – модели и преобразования из одной формы представления в другую

tf	Создание и преобразование в передаточные функции
zpk	Создание и преобразование в нули/полюсы/коэффициент усиления
ss	Создание и преобразование в пространство состояния

TF creates a continuous-time transfer function SYS with numerator(s) NUM and denominator(s) DEN.
SYS = TF(NUM,DEN)
The output SYS is a TF object
ZPK creates zero-pole-gain models or convert to zero-pole-gain format.
SYS = ZPK(Z,P,K) - continuous-time zero-pole-gain (ZPK) model SYS with zeros Z, poles P, and gains K.
The output SYS is a ZPK object
SS creates state-space model or converts model to state space.
SYS = SS(A,B,C,D) - creates a SS object SYS representing the continuous-time state-space model:

$$\dot{x} = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

You can set D=0 to mean the zero matrix of appropriate dimensions

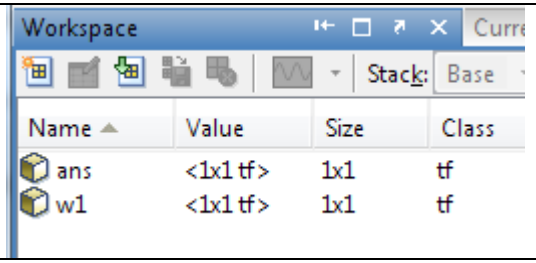
	Специфические атрибуты (поля) передаточных функций (TF-объектов)
num	Числитель - вектор-строка для ОМ-систем; для ММ-систем — массив ячеек из векторов-строк размером NY на NU (например, {[1 0] 1 ; 3 [1 2 3]})

den	Знаменатель - вектор-строка для ОМ-систем; для ММ-систем — массив ячеек из векторов-строк размером NY на NU. Например: **tf({-5 ; [1 -5 6]} , {[1 -1] ; [1 1 0]})** определяет систему с одним входом и двумя выходами **[-5 /(s-1)]** **[(s^2-5s+6)/(s^2+s)]**
Variable	Имя (тип) переменной Возможны варианты: s, p, z По умолчанию принимается s (для непрерывных переменных) и z (для дискретных). Имя переменной влияет на отображение и создает дискретную ПФ для дискретных сигналов

Аналогичные описания атрибутов заданы для всех форм представления ЛСС

Работа с передаточными функциями с использованием пакета CONTROL

Передаточная функция формируется функцией **tf**, в качестве аргументов задаются коэффициенты полиномов числителя и знаменателя в порядке убывания степени оператора **s**.
Целесообразно выполнить присвоение модели, представляемой данной передаточной функцией некоторой переменной:
	`>> w1 = tf([3 1], [1 1 1])`
	
	`Transfer function:`
	`  3 s + 1`
	`-----------`
	`s^2 + s + 1`

Теперь переменную **w1** можно использовать при обращении к функциям или выполняя необходимые преобразования, используя операторы Matlab.	

Для преобразования моделей предназначены специальные функции:

parallel	параллельное соединение
feedback	соединение с обратной связью
series	последовательное соединение
append, connect	различные соединения блоков

Замкнутая система с обратной связью: Предполагается отрицательная обратная связь.	`SYS = FEEDBACK(SYS1,SYS2)` `u --->O---->[ SYS1 ]----+---> y` `		Y = SYS * u` ` +-----[SYS2]<---+`
Положительную обратную связь включает дополнительный параметр:	`SYS = FEEDBACK(SYS1,SYS2,+1)`

Проверка соединения замкнутой системы с единичной отрицательной обратной связью

Вычисление по формуле:

```
>> W1 = w1/(1+w1)
Transfer function:
   3 s^3 + 4 s^2 + 4 s + 1
-----------------------------
s^4 + 5 s^3 + 7 s^2 + 6 s + 2
```

Вычисление с использованием функции **feedback**

```
>> W2 = feedback(w1,1)

Transfer function:
   3 s + 1
-------------
s^2 + 4 s + 2
```

Результаты не совпали?

Проверка - Факторизация – преобразование в **zpk** форму
 	
`>> zpk(W1)`
`Zero/pole/gain:`
`  3 (s+0.3333) (s^2  + s + 1)`
`---------------------------------`
`(s+3.414) (s+0.5858) (s^2  + s + 1)`
	`>> zpk(W2)`
	
	`Zero/pole/gain:`
	`  3 (s+0.3333)`
	`--------------------`
	`(s+3.414) (s+0.5858)`

Результат соответствует расчету **W2** с использованием функции **feedback** при сокращении сомножителя `(s^2  + s + 1)`	`Аналогичный результат может быть получен с использованием функции` **mineral** `mineral(W1)`

Функции извлечения данных из модели

tfdata	Извлечение параметров передаточных функций
zpkdata	Извлечение нулей/полюсов/коэффициентов усиления
ssdata	Извлечение матриц пространства состояния
get	Получение свойств LTI- модели

TFDATA Quick access to transfer function data [NUM,DEN] = TFDATA(SYS)

Если выполнить обращение к функции без указания параметров, будет получена информация только о структуре данных модели:

```
>> tfdata(W2)
ans =
    [1x3 double]
```

Для получения коэффициентов полиномов (в векторной форме), следует задать параметр **'v'**

```
>>  tfdata(W2,'v')
ans =
     0     3     1
```

Для получения данных числителя и знаменателя необходимо указать в качестве выходных параметров два вектора

```
>> [n,d] = tfdata(W2,'v')
n =
     0     3     1
d =
     1     4     2
```

Проверять результаты вычислений и преобразований моделей можно по корням числителя и знаменателя передаточных функций с использованием функции **roots**

Функции анализа свойств модели

```
>> step(W1)
>> impulse(W1)
>> bode(W1)
>> margin(W1)
>> rlocus
```

```
>> pole(W2)
ans =
   -3.4142
   -0.5858
```

```
>> zero(W2)
ans =
   -0.3333
```

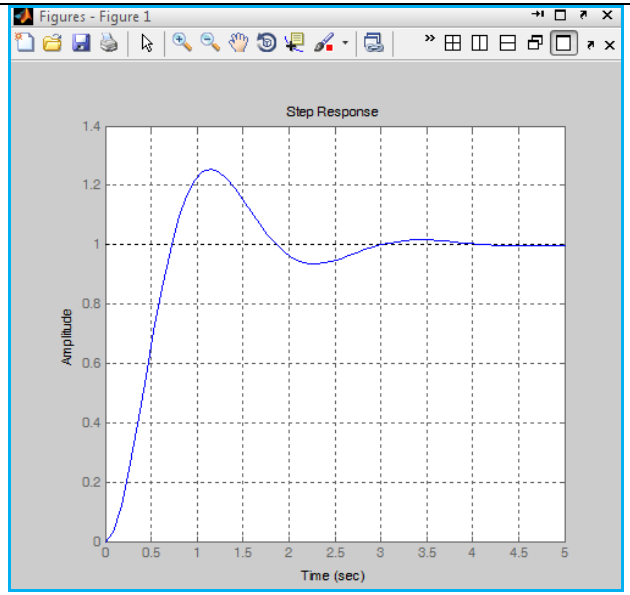
```
>> damp(W1)

        Eigenvalue            Damping     Freq. (rad/s)

 -5.86e-001                  1.00e+000      5.86e-001
 -5.00e-001 + 8.66e-001i     5.00e-001      1.00e+000
 -5.00e-001 - 8.66e-001i     5.00e-001      1.00e+000
 -3.41e+000                  1.00e+000      3.41e+000
```

```
>> W1
 Transfer function:
        9
---------------
s^2 + 2.4 s + 9
```

Характерные точки графика

```
>> s1 = stepinfo(W1)
 s1 =
         RiseTime: 0.4929
     SettlingTime: 2.8036
      SettlingMin: 0.9356
      SettlingMax: 1.2538
        Overshoot: 25.3794
       Undershoot: 0
             Peak: 1.2538
         PeakTime: 1.1480
```

s1 – новый объект – структура - *МАССИВ ЗАПИСЕЙ*



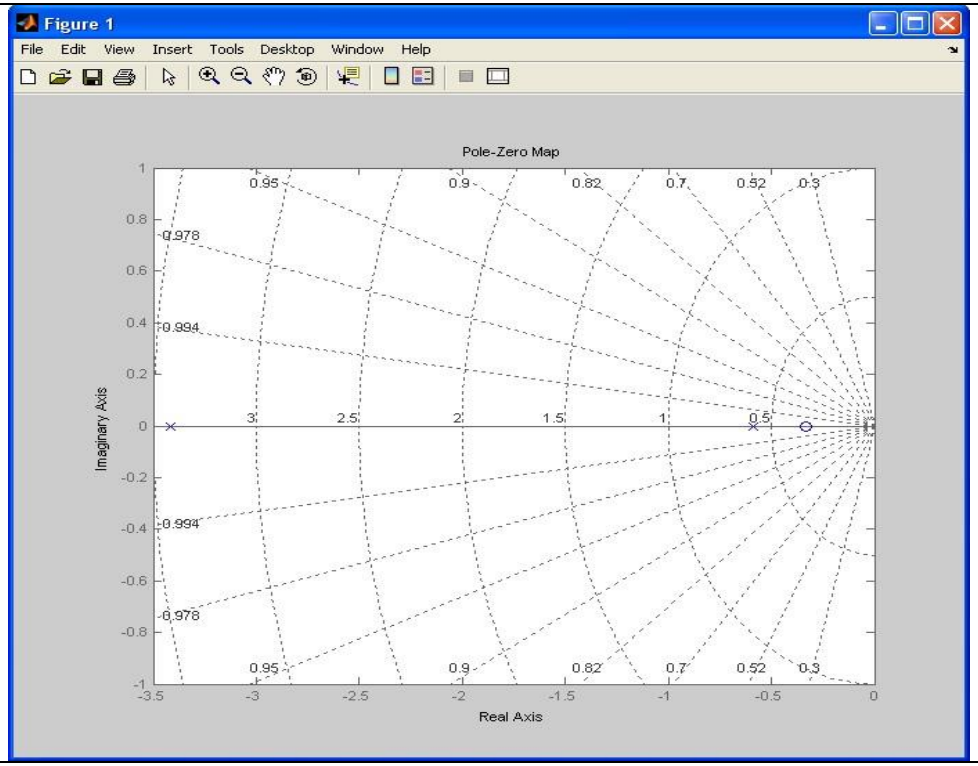
Отклик во времени – численные значения по шагам интегрирования

```
>> [y t]=step(W1);
>> [t y]
```

```
ans =
         0         0
    0.1809    0.3950
    0.3618    0.5915
    0.5427    0.6826
    0.7236    0.7183
    0.9045    0.7255
    1.0854    0.7185
    . . . . . . . . .
```

Нули и полюсы на графике

```
>> pzmap(W2)
>> grid
```



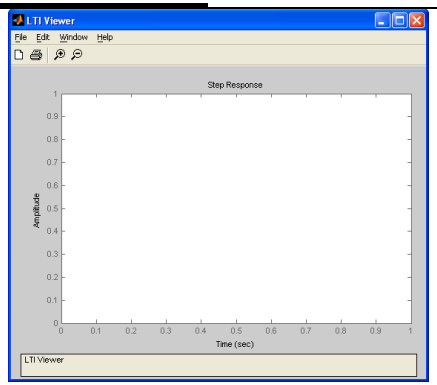
С помощью функции **pzmap** получим значения полюсов и нулей системы:

```
>> [np,dz] =pzmap(W)
np =
             0
 -1.2000e+000 +2.0000e+002i
 -1.2000e+000 -2.0000e+002i
 -1.0000e+002
 -1.0000e+002
dz =
   Empty matrix: 0-by-1
```

LTIviewer – интерактивный обозреватель свойств линейных моделей

Зададим две LTI модели передаточными функциями

```
>> w=tf([1],[1 1])
 Transfer function:
  1
-----
s + 1
и
>> w1=tf([1 1],[1 1 1])
 Transfer function:
   s + 1
-----------
s^2 + s + 1
```



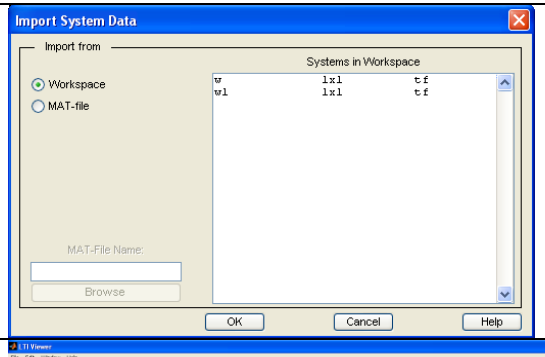
затем вызываем LTIviewer командой

```
>> ltiview
```

По команде меню FILE – IMPORT появляется окно, представляющее содержимое рабочей области, в данном случае там присутствуют для объекта – системы w и w1

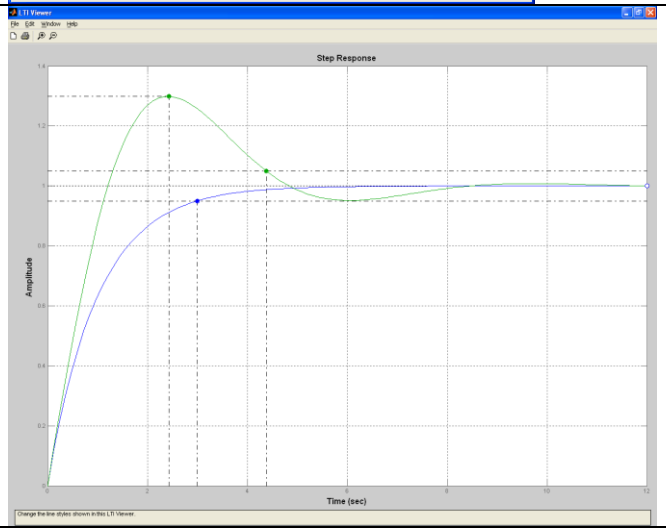
Можно сразу задать идентификаторы созданных LTI – систем во входных параметрах

>> ltiview(w,w1)



Правая кнопка мыши открывает выпадающее меню, позволяющее выполнить настройку графиков, в том числе выбрать требуемую характеристику системы (реакция на ступенчатый или импульсный входной сигнал, ЛАФЧХ и др.) и другие параметры обозревателя.

Параметры задаются в выпадающем меню **Properties**



Представление LTI-системы в пространстве состояний. SS модели в Matlab.

Модель «Вход – Выход» в пространстве состояний формируется на основе системы дифференциальных уравнений первого порядка, разрешенных относительно производных, которую принято называть нормальной формой Коши.

Для создания модели динамической системы в пространстве состояния предназначена функция ***ss*** из пакета ***Control System Toolbox***

Входные параметры ***A,B,C,D***

Размерность матриц:

x(t) – *вектор состояния* – столбец размерности ***n***, который включает в себя переменные объекта, однозначно определяющие его состояние;

A – квадратная *матрица параметров системы* размерности ***[n * n];***

B – *матрица управления* размерности ***[n * m]***, в которой ***i***-й столбец (***1*** ≤ ***i*** ≤ ***m***) содержит параметры, характеризующие воздействие ***i***-го входного сигнала на соответствующий ***j***-й параметр состояния (***1*** ≤ ***j*** ≤ ***n***);

Соотношение размерности матриц показано н арисунке =>

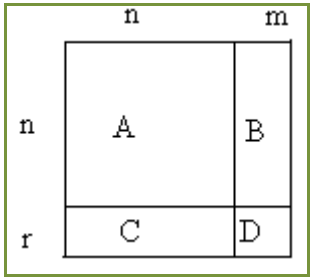
C – матрица выхода размерности ***[n * r]***, (***k*** ≤ ***n***):

D – матрица прямой связи.

n – количество переменных состояния,

m – число входных сигналов (сигналы управления),

r – число выходных сигналов



ПРИМЕР. Уравнения объекта в форме Коши (короткопериодическое продольное движение ЛА)

$$\begin{cases} \frac{d\omega_z}{dt} = (-c_1 - c_5)\omega_z + (c_4c_5 - c_2)\alpha - c_3\delta_в \\ \frac{d\alpha}{dt} = \omega_z - c_4\alpha \end{cases}$$

Система задана двумя уравнениями, имеет два выходных параметра: угловая скорость тангажа ω_z и угол атаки α.

Для работы с функциями Matlab представим дифференциальные уравнения (1.21) в матричной форме:

Программа – скрипт:

```
%% ss system
A = [(-c1 -c5)  (c4*c5 -c2)
          1           -c4];
B = [c3
      0 ];    c = [1 0
                   0 1];     d = 0;
Model_ss = ss(A,B,c,d,'statename',{'Om_z' 'alfa'},...
    'inputname',{'elevator'},...
     'outputname',{'Om_z' 'alfa'});
stepplot(Model_ss)
Model_ss
```

$$\begin{cases} \dot{X} = AX + BU \\ Y = CX + DU \end{cases}$$

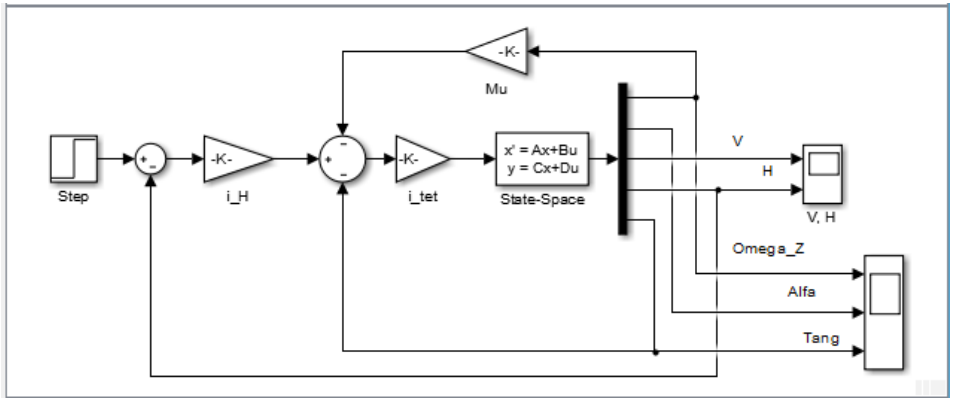
X – вектор состояния: $X = \begin{bmatrix} \omega_z \\ \alpha \end{bmatrix}$

A – матрица коэффициентов: $A = \begin{bmatrix} (-c_1 - c_5) & (c_4c_5 - c_2) \\ 1 & -c_4 \end{bmatrix}$

U – вектор входа системы: $U = \begin{bmatrix} \delta_в \\ 0 \end{bmatrix}$

B – матрица управления: $B = \begin{bmatrix} -c_3 \\ 0 \end{bmatrix}$

C – матрица выхода: $C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

	D – матрица, характеризующая связь входного сигнала с выходным: $D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ Y – вектор выхода системы. ω_z и α (см. матрицу выхода C)
Создание SS-модели в Simulink Для построения модели используем полные уравнения продольного длиннопериодического движения самолета. Модель объекта в пространстве состояний создается на основе блока ***State-Space***, которые находится во вкладке ***Continuous*** библиотеки ***Simulink***. Входом блока является сигнал управления, который в данном примере представляет собой величину отклонения руля высоты и является скалярной переменной. Выход блока является вектором, элементы которого представляют пять параметров по порядку следования заданных уравнений состояния: угловая скорость тангажа, угол атаки, высота, скорость и угол тангажа. Для того, чтобы преобразовать векторный сигнал на выходе модели в скалярные значения следует использовать блок ***Demux***, которые находится во вкладке ***Signal Routing*** библиотеки ***Simulink***.	 От линий связи, по которым проходят значения выходных параметров, можно взять сигналы для контура управления. На схеме сформирован контур демпфирования по параметру ω_z, а также контур автопилота стабилизации тангажа и высоты полета.

Simulink – среда визуального моделирования

Начало работы –создание файла модели и вызов библиотеки Simulink –	modelname.xls
1. Выборка блоков в разделах библиотеки и расстановка блоков на схеме модели	используются различные разделы библиотеки и наборы Blockset
2. Соединение блоков, модификация вида и ориентации	набор простой модели осуществляется на одном наборном поле для сложной модели формируем основную модель и подсистемы (Subsystem)
3. Задание параметров блоков	входы-выходы могут зависеть от параметров, например, число входов сумматора, мультиплексора параметры блоков можно задать глобальными переменными в рабочей области Matlab параметры блоков можно задать вычислительными операторами

4. Установить параметры расчета модели	параметры блоков переменные, заданные или рассчитанные программно данные из сохраненного файла datname.mat
5. Настроить Решатель	время процесса метод решения и параметры шага, точности
6. Организовать вывод результатов	в графические окна, в рабочую среду Matlab, в файл
7. Проверка результатов, выполнение расчетов с изменением параметров	

Для вызова библиотеки **Simulink** необходимо задать команду **simulink** в командной строке MATLAB или нажать кнопку ***Simulink*** на панели инструментов MATLAB.

Окно библиотеки содержит позиции меню **File Edit View Help** и панель инструментов

Для дальнейшей работы есть возможность создать новую модель или открыть файл модели, созданный ранее.

Нужный блок можно найти, набрав его название в строке поиска и нажав кнопку **Find Block** в строке инструментов (бинокль). На рисунке показано, как система нашла блок **Integrator.**

Для того, чтобы создать новую модель, следует нажать кнопку **Create a new model** на панели инструментов библиотеки или выполнив соответствующую команду из меню **File**. При этом открывается новое окно – поле сборки модели.

Двойной щелчок по блоку открывает окно установки параметров

Для создания надписи нужно указать мышью место надписи и дважды щелкнуть левой клавишей мыши, появится прямоугольная рамка с курсором ввода.

Аналогичным образом можно изменить и подписи к блоками моделей.

Меню Format содержит команды измененная ориентации блока, цвет, тень, шрифт и др.

Меню View – команда Zoom-In увеличивает масштаб чертежа, толщину линий, увеличивается четкость схемы при выводе на печать .

Основы работы с системой Simulink

SIMULINK является программной системой интерактивного моделирования и позволяет использовать вычислительные возможности MATLAB для решения широкого круга задач моделирования и анализа систем.
Для вызова подсистемы **Simulink** необходимо задать команду **simulink** в командной строке MATLAB или нажать кнопку ***Simulink*** на панели инструментов MATLAB. При этом открывается окно **Simulink Library Brouser** – библиотека Simulink рис. 1. Окно библиотеки содержит позиции меню **File Edit View Help** и панель инструментов
Для дальнейшей работы есть возможность создать новую модель или открыть файл модели, созданный ранее, выполнив команду из меню или нажав кнопку панели инструментов.

Модель состоит из блоков и связей, или линий передачи сигналов. Блоки сгруппированы по назначению, группы отображаются в виде иерархической структуры в окне библиотеки.
Для того, чтобы создать новую модель, следует нажать кнопку **Create a new model** на панели инструментов библиотеки или выполнив соответствующую команду из меню **File**. При этом открывается новое окно – поле сборки модели. Если выполняется команда открытия ранее созданного файла, открывается окно с моделью, собранной ранее, рис. 2.

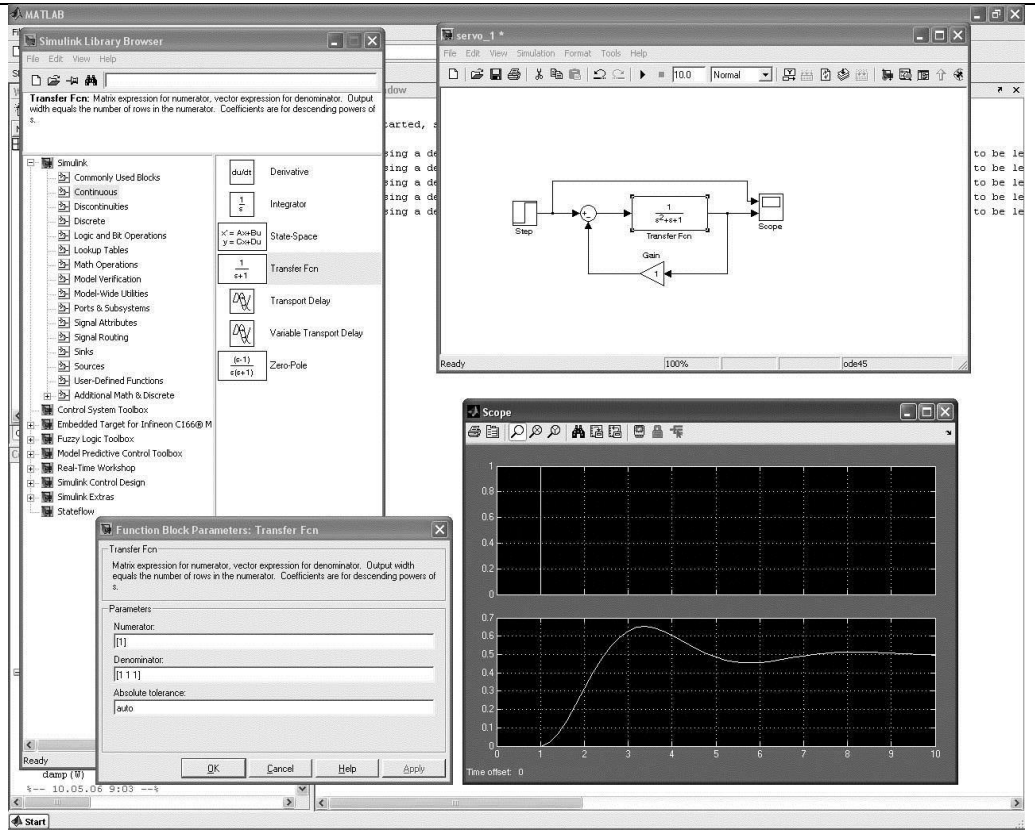


Рис. 2

Для "сборки" схемы модели блок из окна библиотеки необходимо перетащить в окно (поле сборки) новой модели. Двойной щелчок по блоку открывает окно установки параметров. На рис. 3 показан блок модели – источник постоянного сигнала с единичным значением.
Более сложные блоки требуют задания большего числа параметров, например, для генератора необходимо ввести форму колебаний, амплитуду и частоту, причем частоту можно задать в единицах Герц или рад/с рис. 4.

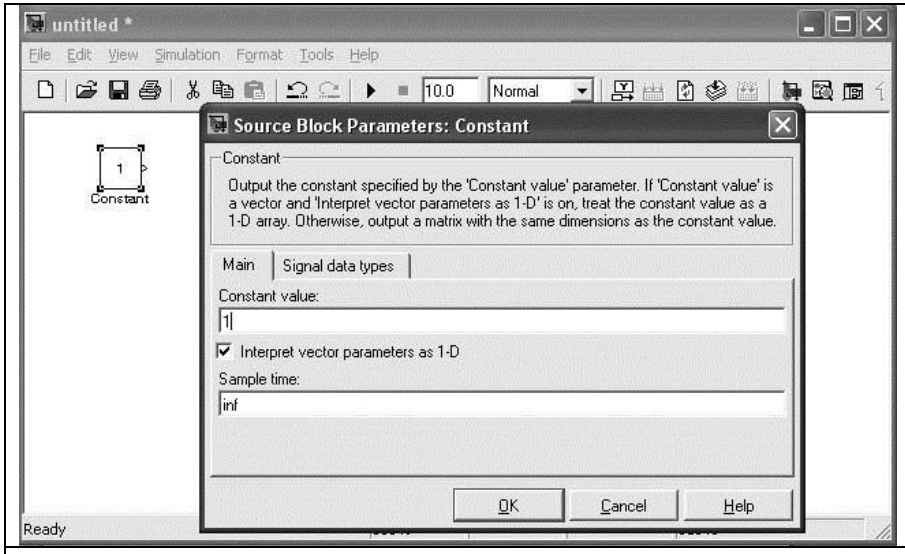


Рис. 3

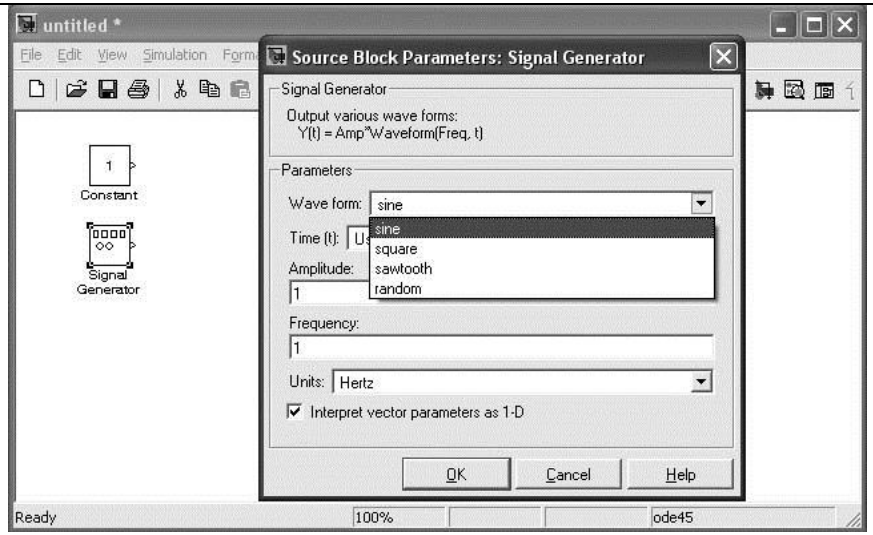


Рис. 4

Анализ свойств динамической системы

Для анализа свойств динамической системы, созданной в Simulink можно применить LTI-viewer. Для этого необходимо ввести в схему модели блоки **In** и **Out** на вход и выход, соответственно, рис. 5.

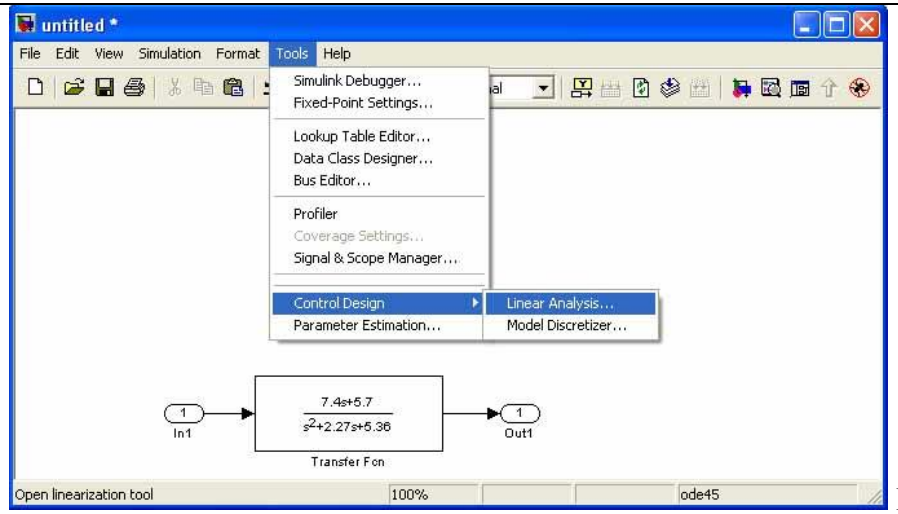


Рис. 5

.

Для обозначения входа и выхода можно также указать на точку схемы (линию связи блоков) мышью и нажав правую кнопку. При этом выпадает меню, в котором следует выбрать позицию **Linearisation Points** и далее **Input** или **Output** рис. 6. Затем, через меню **Tools-Control Design-Linear Analysis** запускаем LTI-viewer

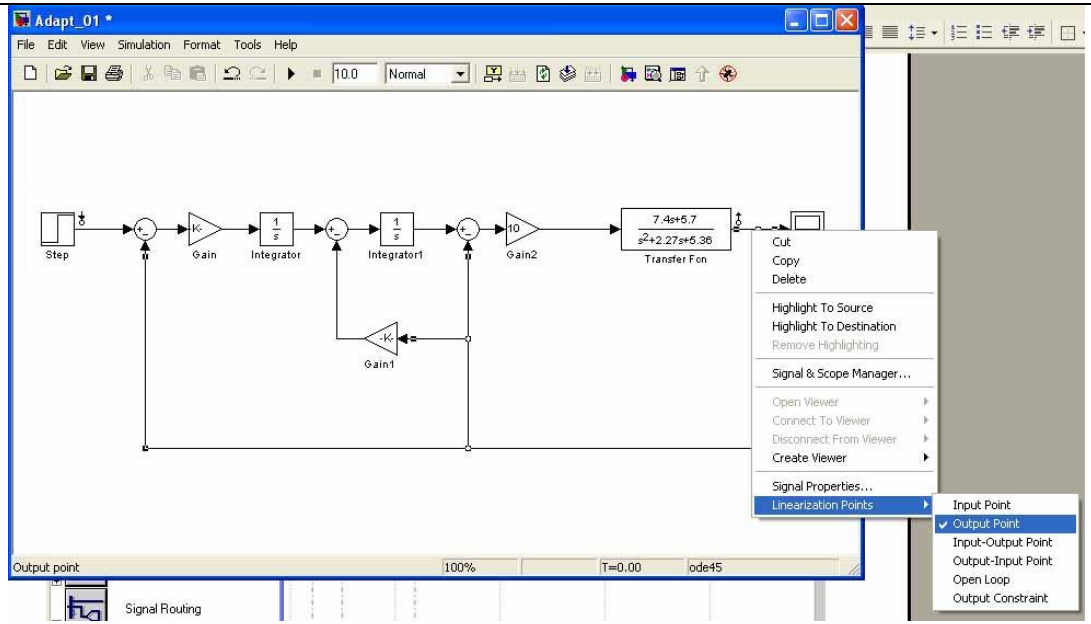


Рис. 6

Создание подсистемы

При создании больших и сложных моделей можно собирать схему по частям, из подсистем. Если схема модели полностью готова, можно обвести с помощью мышки группу блоков рис. 7и в меню **Edit** задать команду **Create Subsystem**, рис. 8.

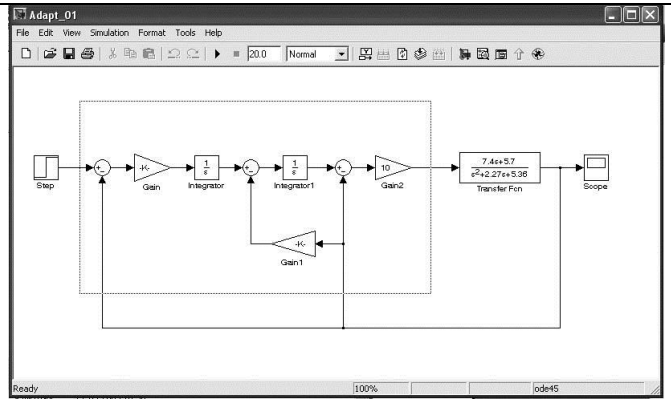
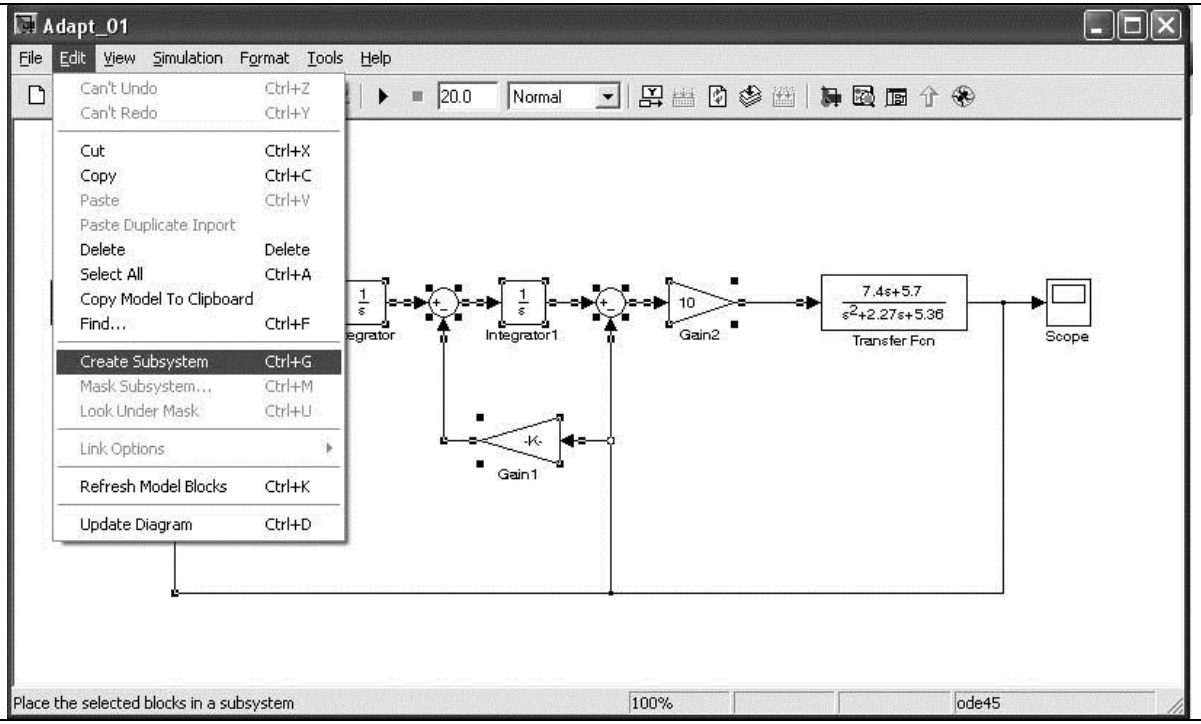


Рис. 7

Рис. 8


Adapt_01
File Edit View Simulation Format Tools Help
Can't Undo Ctrl+Z
Can't Redo Ctrl+Y
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Paste Duplicate Inport
Delete Delete
Select All Ctrl+A
Copy Model To Clipboard
Find... Ctrl+F
Create Subsystem Ctrl+G
Mask Subsystem... Ctrl+M
Look Under Mask Ctrl+U
Link Options
Refresh Model Blocks Ctrl+K
Update Diagram Ctrl+D
20.0
Normal
Integrator1
Gain2
7.4s+5.7
s2+2.27s+5.36
Transfer Fcn
Scope
-K-
Gain1
Place the selected blocks in a subsystem
100%
ode45

В результате этого выделенная группа блоков объединяется в один блок (Subsystem), который хранится в том же файле и раскрывается при щелчке левой кнопкой мышки по блоку на схеме всей модели, рис. 9.

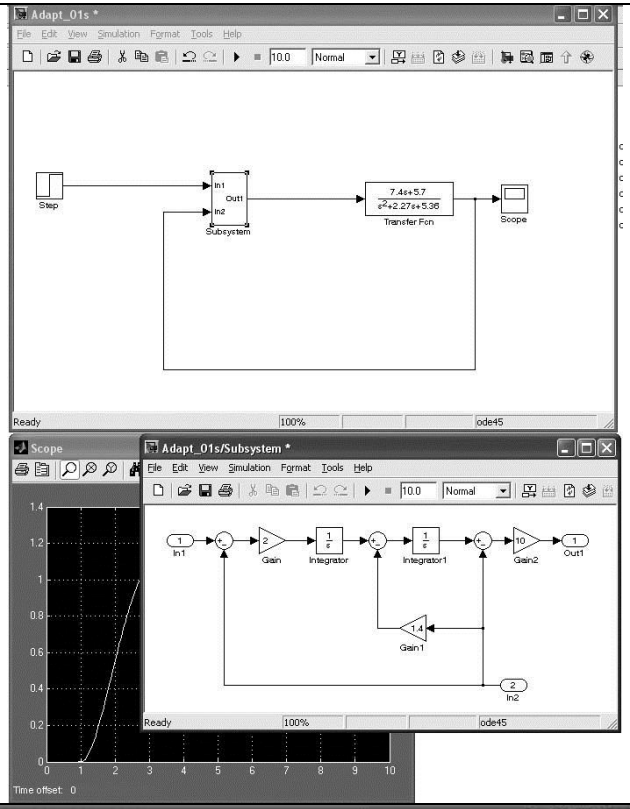


Рис. 9

Из группы блоков можно сразу набирать подсистему, для этого следует поместить на наборное поле необходимое количество блоков **In** и **Out**, рис. 10.

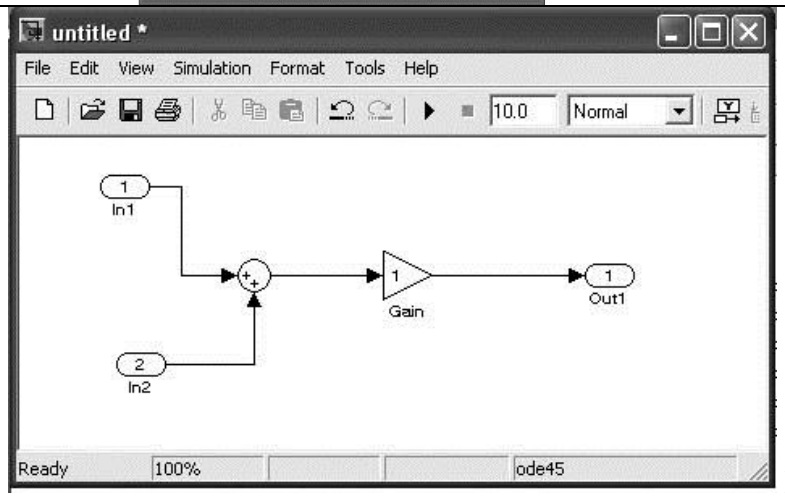


Рис. 10

Передача данных из Simulink в рабочую среду MATLAB– блок To Workspace

Результаты расчетов, выполняемые в **Simulink** не отображаются в рабочем пространстве MATLAB. Дл того, чтобы присвоить значения параметров, например, данные переходного процесса, какой-либо переменой MATLAB, необходимо ввести в схему блок **To Workspace**, в окне его параметров задать имя переменой и формат вывода – **Array,** рис. 11. Теперь, при выполнении моделирования, в рабочей среде MATLAB появится переменная с заданным именем, также появится переменная **tout**, с дискретами времени.

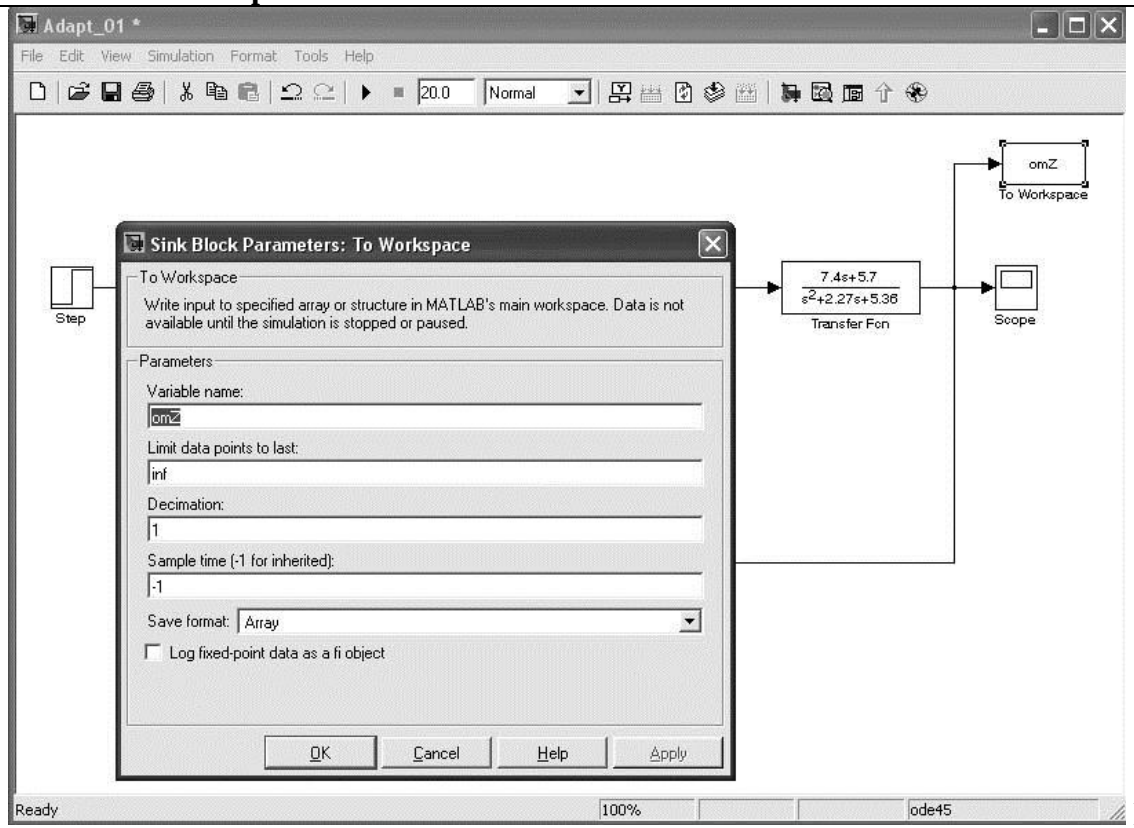
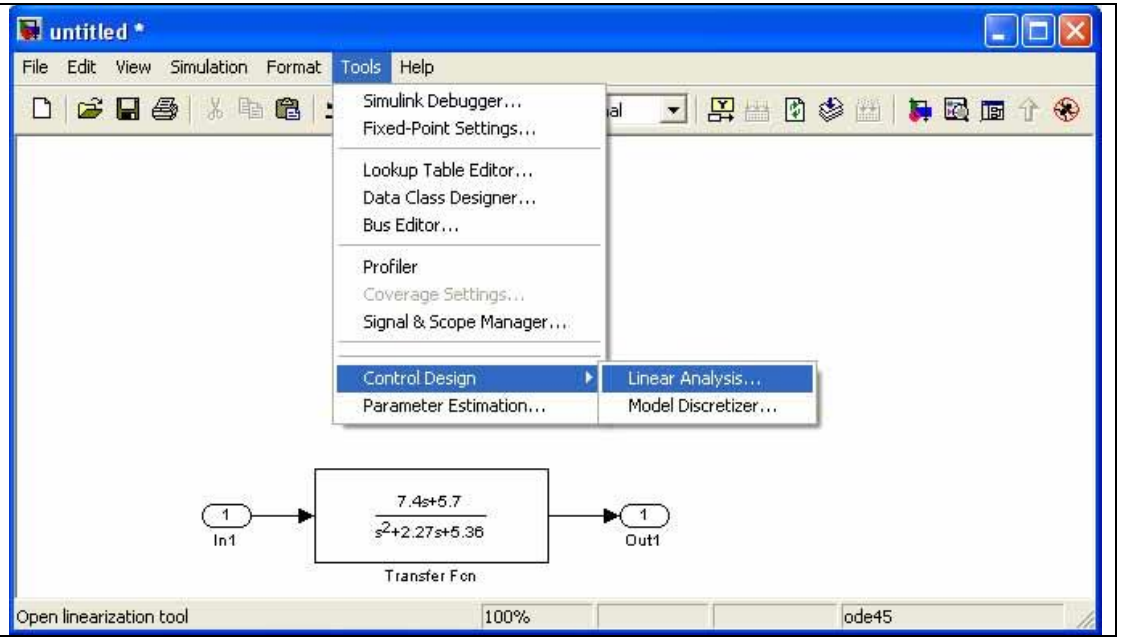


Рис. 11

Для анализа свойств динамической системы, созданной в Simulink можно применить LTI-viewer. Для этого необходимо ввести в схему модели блоки **In** и **Out** на вход и выход, соответственно:



Настройка параметров моделирования – решатель

В закладке решателя Solver имеется три группы параметров:

• **Simulation time** — интервал моделирования указывается в виде начального (Start time — обычно нулевое значение) и конечного (Stop time) значений времени;

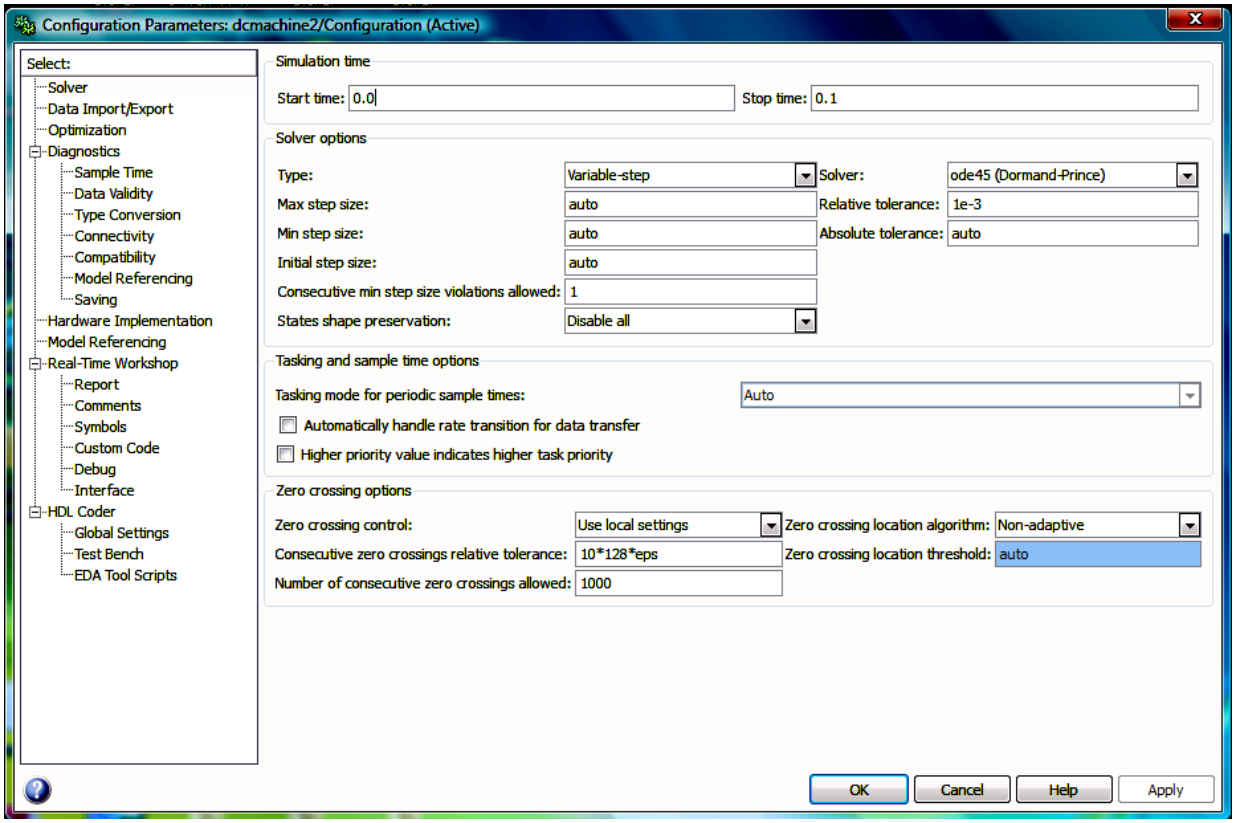
• **Solver options** — параметры решателя, определяемые методом интегрирования (Type) с фиксированным (Fixed-step) или с переменным (Variable-step) шагом;

• **Output options** — параметры вывода.

По умолчанию используется наиболее универсальный решатель типа ode45 («нежёсткий»).

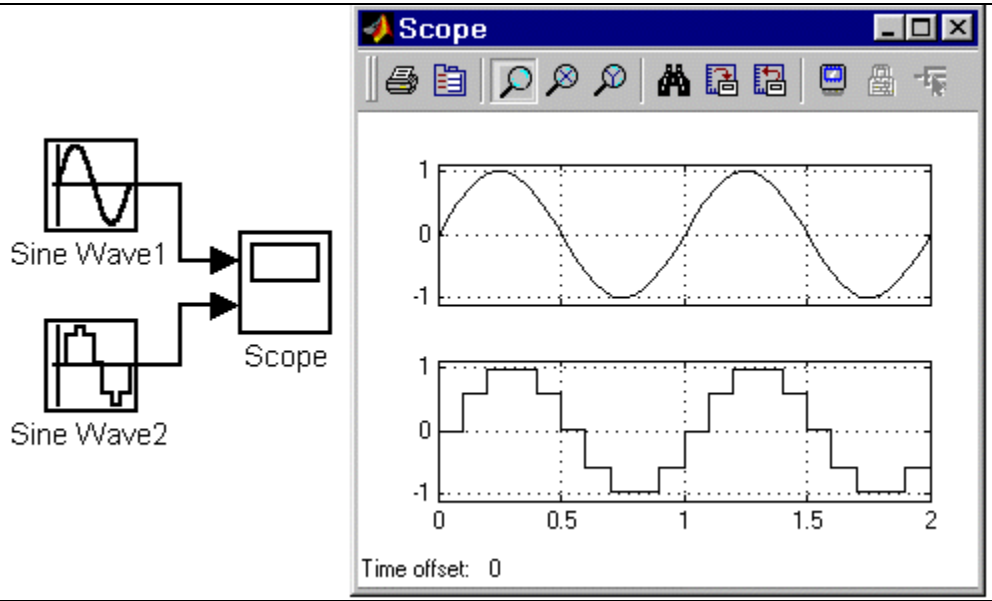
При решении жесткой задачи (разброс корней более двух порядков) при интегрировании возможен расходящийся переходный процесс.

В этом случае нужно использовать жёсткие решатели, помеченные буквой s (stiff). Тут наиболее точным является решатель ode15s.


Configuration Parameters: dcmachine2/Configuration (Active)
Select:
Solver
Data Import/Export
Optimization
Diagnostics
Sample Time
Data Validity
Type Conversion
Connectivity
Compatibility
Model Referencing
Saving
Hardware Implementation
Model Referencing
Real-Time Workshop
Report
Comments
Symbols
Custom Code
Debug
Interface
HDL Coder
Global Settings
Test Bench
EDA Tool Scripts
Simulation time
Start time: 0.0
Stop time: 0.1
Solver options
Type: Variable-step
Solver: ode45 (Dormand-Prince)
Max step size: auto
Relative tolerance: 1e-3
Min step size: auto
Absolute tolerance: auto
Initial step size: auto
Consecutive min step size violations allowed: 1
States shape preservation: Disable all
Tasking and sample time options
Tasking mode for periodic sample times: Auto
Automatically handle rate transition for data transfer
Higher priority value indicates higher task priority
Zero crossing options
Zero crossing control: Use local settings
Zero crossing location algorithm: Non-adaptive
Consecutive zero crossings relative tolerance: 10*128*eps
Zero crossing location threshold: auto
Number of consecutive zero crossings allowed: 1000
OK
Cancel
Help
Apply

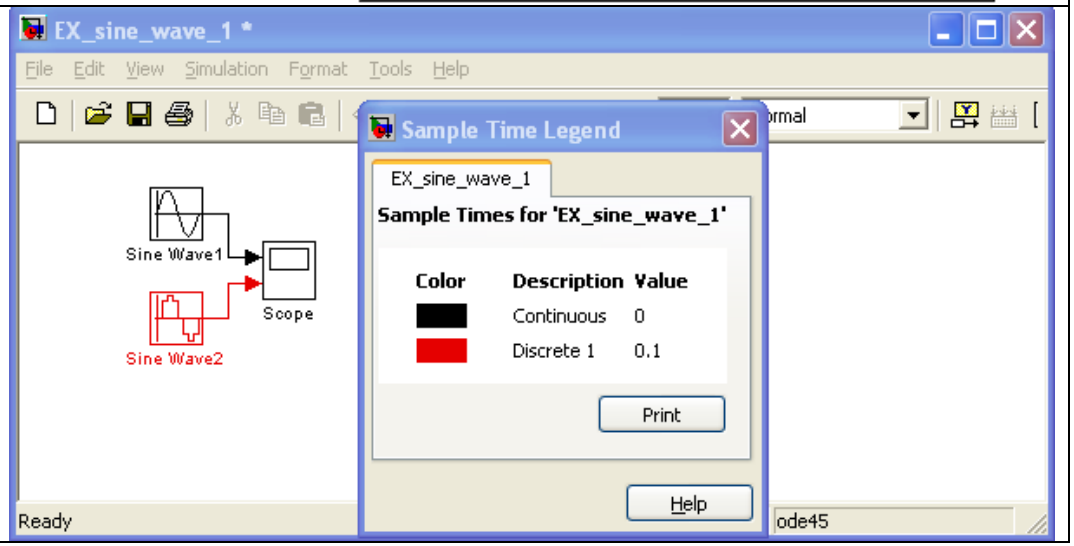
Вывод данных в Simulink с заданием интервала модельного времени

Sample time – Шаг модельного времени. Используется для согласования работы источника и других компонентов модели во времени.
Параметр может принимать следующие значения:
0 (по умолчанию) – Используется при моделировании непрерывных систем.
0 (положительное значение) – Задается при моделировании дискретных систем. В этом случае шаг модельного времени можно интерпретировать как шаг квантования по времени выходного сигнала.
-1 – Шаг модельного времени устанавливается таким же, как и в предшествующем блоке, т.е. блоке, откуда приходит сигнал в данный блок.
Этот параметр может задаваться для большинства блоков библиотеки **Simulink**.



Sample time = 0 для блока **Sine Wave 1**и **Sample time** = 0.1 для блока **Sine Wave 2**

Managing the Sample Time Legend
В окне модели – меню Format – Sample Time Display



Simulink – пакеты расширений. Simscape – физическое моделирование

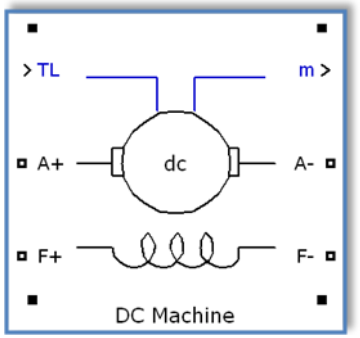
Blockset – пакет расширений для Simulink

Simscape™ предоставляет окружение для моделирования и симуляции физических систем, содержащих компоненты из различных инженерных сфер деятельности: механических, электрических, гидравлических и других.
При помощи Simscape вы создаете модель системы так же, как если бы вы собирали физическую систему. Simscape использует подход, называемый "физическая сеть", также известный как каузальное моделирование/
Для построения модели: компоненты (блоки), относящиеся к физическим элементам, таким, как насосы, двигатели и операционные усилители, соединяются линиями, представляющими физические соединения, по которым передается энергия.
Этот подход позволяет описывать физическую структуру системы, более чем математические выражения, лежащие в основе этой системы.
Из модели, которая близка по виду к чертежу, Simscape автоматически выводит дифференциальные уравнения, характеризующие поведение системы. Эти уравнения совмещаются с остальной моделью Simulink и решаются напрямую.

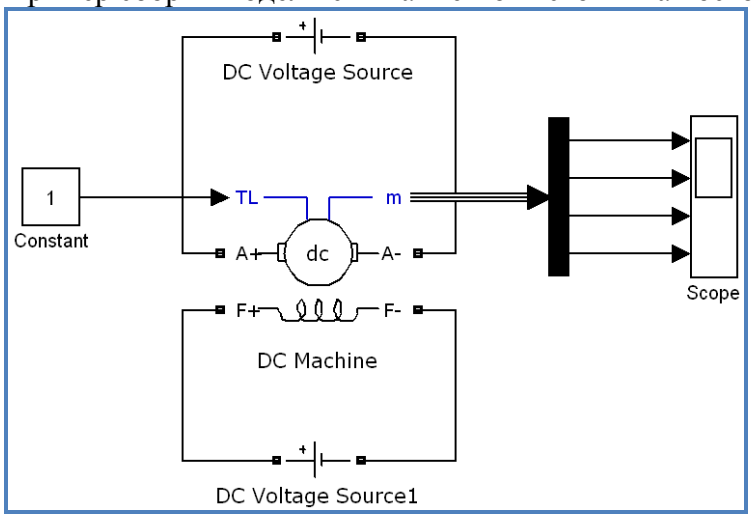
Physical Domain	**Across Variable**	**Through Variable**
Electrical	Voltage	Current
Hydraulic	Pressure	Flow rate
Magnetic	Magnetomotive force (mmf)	Flux
Mechanical rotational	Angular velocity	Torque
Mechanical translational	Translational velocity	Force
Pneumatic	Pressure and temperature	Mass flow rate and heat Flow
Thermal	Temperature	Heat flow
Thermal liquid	Pressure and temperature	Mass flow rate and thermal flux

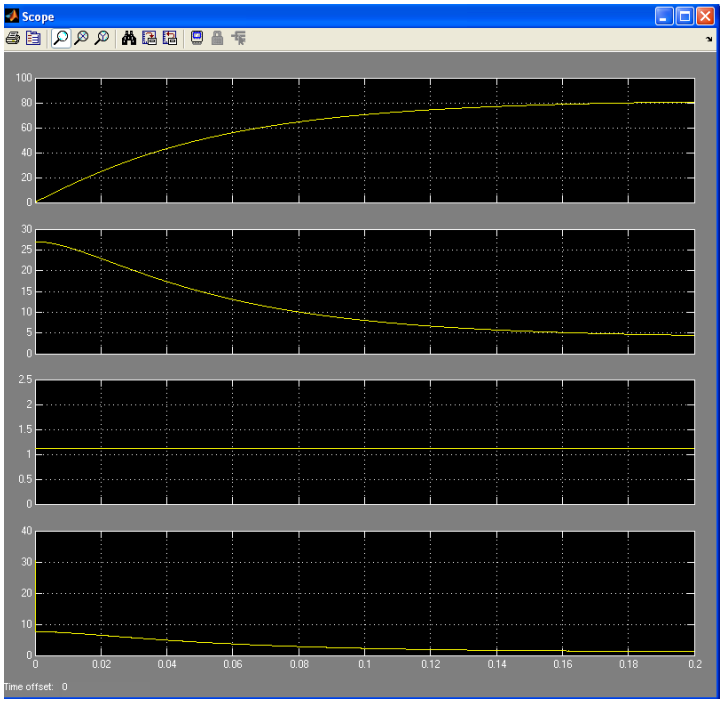
SimPowerSystems (SPS) -модели имеют следующие особенности и отличия от блоков Simulink

Входы и выходы SPS-блоков являются эквивалентами электрических контактов и не показывают направление передачи сигнала.
Электрический ток может через вход или выход блока протекать в двух направлениях - внутрь блока и наружу.
Соединения между блоками = электрические провода, по которым ток может протекать также в двух направлениях.
Линии связи – провода, могут быть соединены между собой, но для выполнения такого соединения должны использоваться специальные блоки - Connectors (соединители)
Simulink-блоки и SimPowerSystems-блоки не могут быть непосредственно соединены друг с другом. Сигнал от S-блока можно передать к SPS-блоку через управляемые источники тока или напряжения, а наоборот - с помощью измерителей тока или напряжения.

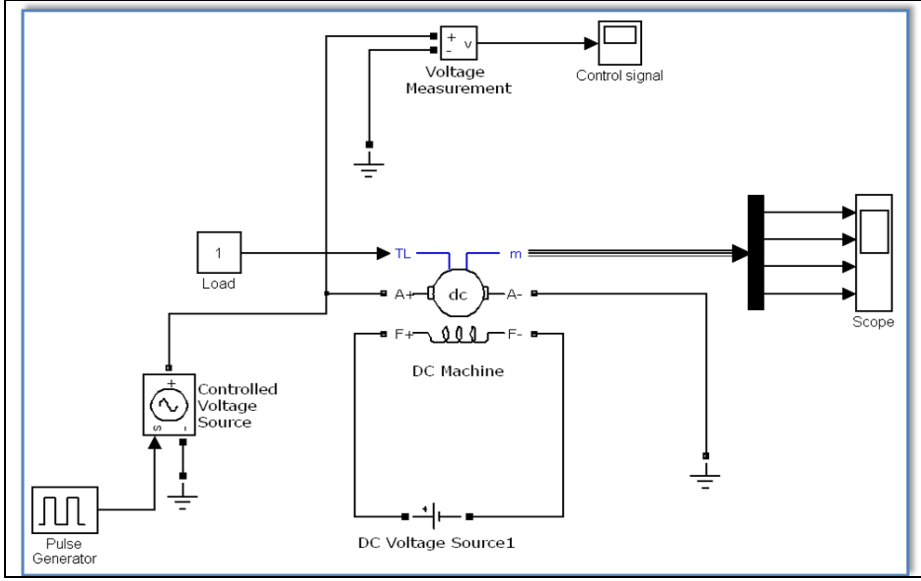
DC Machine – Машина постоянного тока – ДПТ 	У данного блока шесть выводов – выходных портов модели: • F+,F- - положительный и отрицательный выводы обмотки возбуждения; • A+,A- - положительный и отрицательный выводы цепи якоря; • TL – на этот вход подается момент сопротивления; • m – вектор выходных параметров, содержащий компоненты: - угловая скорость ротора (wr); - ток якоря (Ia); - ток обмотки возбуждения (If); - момент электромеханический (Te).
Параметры блока DC Machine • Armature resistance and inductance – активное сопротивление (Ом) и индуктивность(Гн) якоря; • Field resistance and inductance – активное сопротивление (Ом) и индуктивность (Гн) обмотки возбуждения; • Field- armature mutual inductance Laf – взаимная индуктивность (Гн) якоря и обмотки возбуждения; • Total inertia J – суммарный момент инерции (кг*м2); • Viscous friction coefficient Bm – коэффициент вязкого трения (Н*м*с); • Initial speed – начальная скорость (rad/s).	

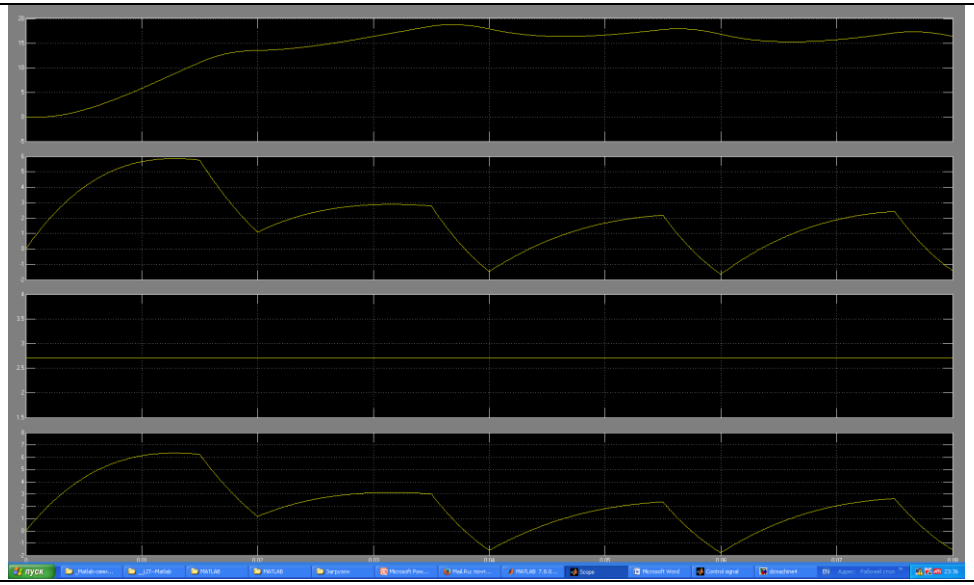
Пример сборки модели с питанием от источника постоянного тока





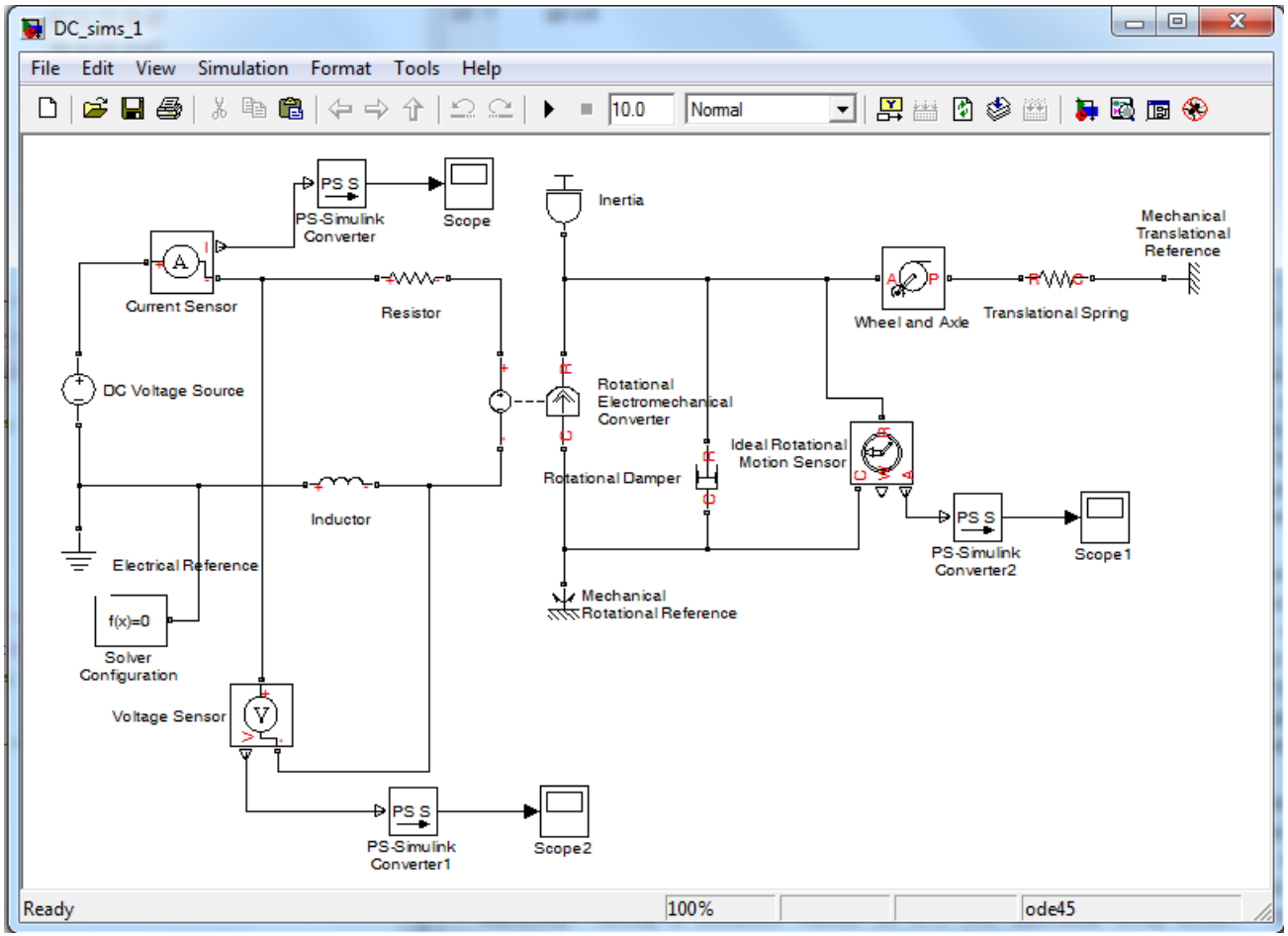
Моделирование DC Machine с импульсным питанием

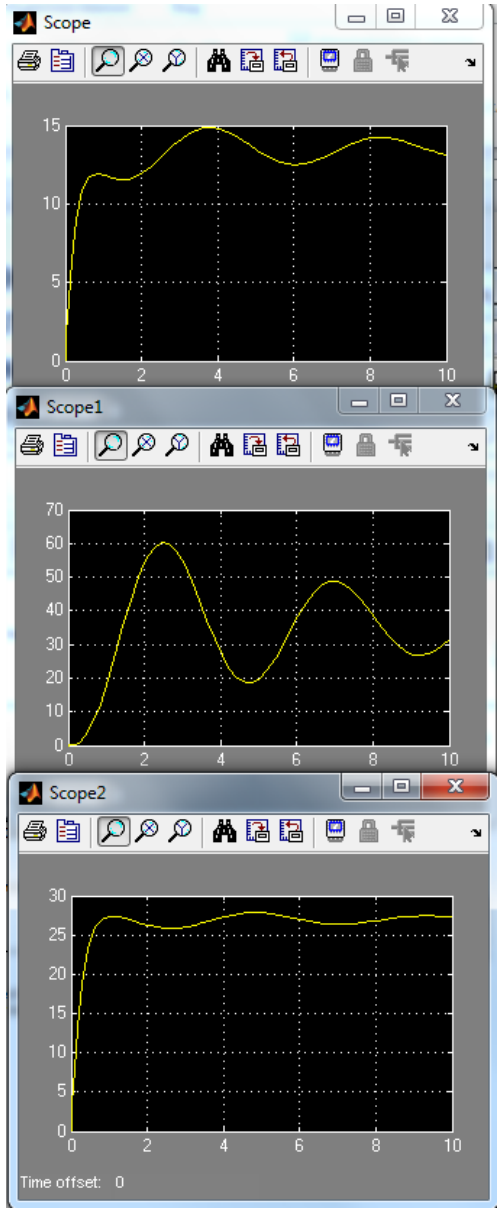




Моделирование двигателя постоянного тока с использованием электрических и механических компонентов физического моделирования

Сопряжение электрических и механических компонентов модели



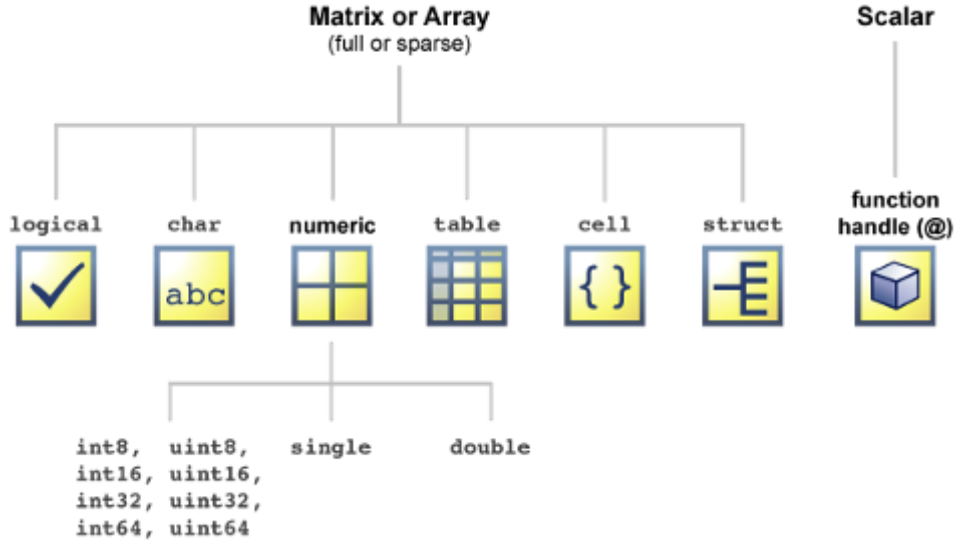


ПРИЛОЖЕНИЕ

Fundamental MATLAB Classes

There are 16 fundamental classes in MATLAB. Each of these classes is in the form of a matrix or array. With the exception of function handles, this matrix or array is a minimum of 0-by-0 in size and can grow to an n-dimensional array of any size. A function handle is always scalar (1-by-1).
All of the fundamental MATLAB classes are shown in the diagram =>

All numeric types support basic array operations, such as subscripting, reshaping, and mathematical operations.



Anonymous Functions

You can create handles to anonymous functions. An anonymous function is a one-line expression-based MATLAB function that does not require a program file. Construct a handle to an anonymous function by defining the body of the function, anonymous_function, and a comma-separated list of input arguments to the anonymous function, arglist. The syntax is:

h = @(arglist)anonymous_function

For example, create a handle, sqr, to an anonymous function that computes the square of a number, and call the anonymous function using its handle.

```
sqr = @(n) n.^2;
x = sqr(3)
x =
    9
```

Интерактивные приложения MATLAB

http://www.mathworks.com/discovery/matlab-apps.html

MATLAB® apps are interactive applications written to perform technical

Интерактивные приложения MATLAB предоставляют возможность

computing tasks. Apps are included in many MATLAB products. The Apps tab of the MATLAB Toolstrip shows you the apps that you currently have installed. There are several ways to get more apps: from MATLAB File Exchange, through additional MATLAB products, and by building your own. You can create apps and share them with members of the MATLAB community.	решать различные технические задачи в удобном и наглядном интерфейсе. Доступ к интерактивным приложениям осуществляется через вкладку Apps
Control System Design and Analysis APP AVAILABLE IN	
 **Control System Designer** Design single-input, single-output (SISO) controllers Control System Toolbox	 **MPC Designer** Design and simulate model predictive controllers Model Predictive Control Toolbox
 **Control System Tuner** Tune fixed-structure control systems Robust Control Toolbox	 **Neuro-Fuzzy Designer** Design, train, and test Sugeno-type fuzzy inference systems Fuzzy Logic Toolbox
 **Fuzzy Logic Designer** Design and test fuzzy inference systems Fuzzy Logic Toolbox	 **PID Tuner** Tune PID controllers Control System Toolbox
 **Linear System Analyzer** Analyze time and frequency responses of linear time-invariant (LTI) systems Control System Toolbox	 **System Identification** Identify models of dynamic systems from measured data

Simulink Real-Time – ***От симуляции на десктопе к реальному времени***

Машины реального времени Speedgoat *для Simulink Real-Time*:

1 Специализированный компьютер (Intel Core i7 2.5 GHz CPU, SSD до1 TB)
2 I/O модули
3 Драйверы, кабели и разъемы для подключения

Создание приложений реального времени из моделей Simulink и загрузка на выделенный компьютер за 3 автоматических шага:

1 Автоматическая Генерация кода
2 Сборка кода (Компилятор)
3 Загрузка на целевой компьютер