МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
Высшая школа печати и медиаиндустрии имени Ивана Федорова

Ю.В. Щербина

МОДЕЛЬНО-ОРИЕНТИРОВАННОЕ ПРОЕКТИРОВАНИЕ

Лабораторные работы
Вычислительная система MATLAB для научных и инженерных расчетов

Москва 2018

Раздел 1. Расчеты с использованием MATLAB

Цели работы : Ознакомление с принципами работы системы MATLAB, матричными операциями, функциональным программированием.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ РАБОТЫ

Основным типом объектов системы MATLAB является прямоугольная матрица, элементами которой могут быть как вещественные так и комплексные числа. Скалярным переменным соответствует матрица размерности [1x1], а векторам - матрица с одним столбцом или строкой.

Матрица - это таблица данных, но особенность заключается в том, что для матриц установлены определенные правила выполнения математических операций. Далее будет показано, как в системе MATLAB реализуется выполнение матричных операций (например умножение матриц), а также выполнение специальных операций поэлементной обработки матриц как простых таблиц или массивов (Array) чисел.

Система MATLAB предоставляет различные возможности для организации расчетных работ. Во-первых система MATLAB является средой - интерпретатором команд и рабочее поле экрана (Command Window) используется для ввода данных и операторов, здесь же отображаются численные результаты, а графики выводятся в отдельные окна (Figure). При этом операторы могут содержать константы, переменные, системные функции и функции пользователя (ранее созданные и хранящиеся в рабочей директории).

В системе Matlab применяется также способ вычисления с использованием программ, написанных пользователем. Программой является внешний М-файл (текстовый файл с расширением .m), который может создаваться и модифицироваться любым текстовым редактором, но с обязательным условием экспорта в виде ASCII файла (без специальных символов редактирования и разметки документа). Выполнение программы начинается при наборе имени М-файла программы в командной строке рабочего поля системы.

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

открываются специальные окна, возможно сохранение данных в файл. Возможен ввод данных с клавиатуры.

При запуске системы открывается окно, содержащее:

- строку главного меню системы **File Edit View Graphics Debug Desktop Window Help**
- панель инструментов-кнопок (**MATLAB Toolbar**);
- панель ярлыков (**Shortcuts Toolbar**);

Основное окно включает три рабочих окна:

- окно **Current Directory/Workspace** (левое верхнее);
- окно **Command History** (левое нижнее);
- окно **Command Window** (справа).

На рис. 1 показаны окна системы, результаты работы и окно графика **Figure 1**, которое является свободно перемещаемым и на данном рисунке помещено на передний план.

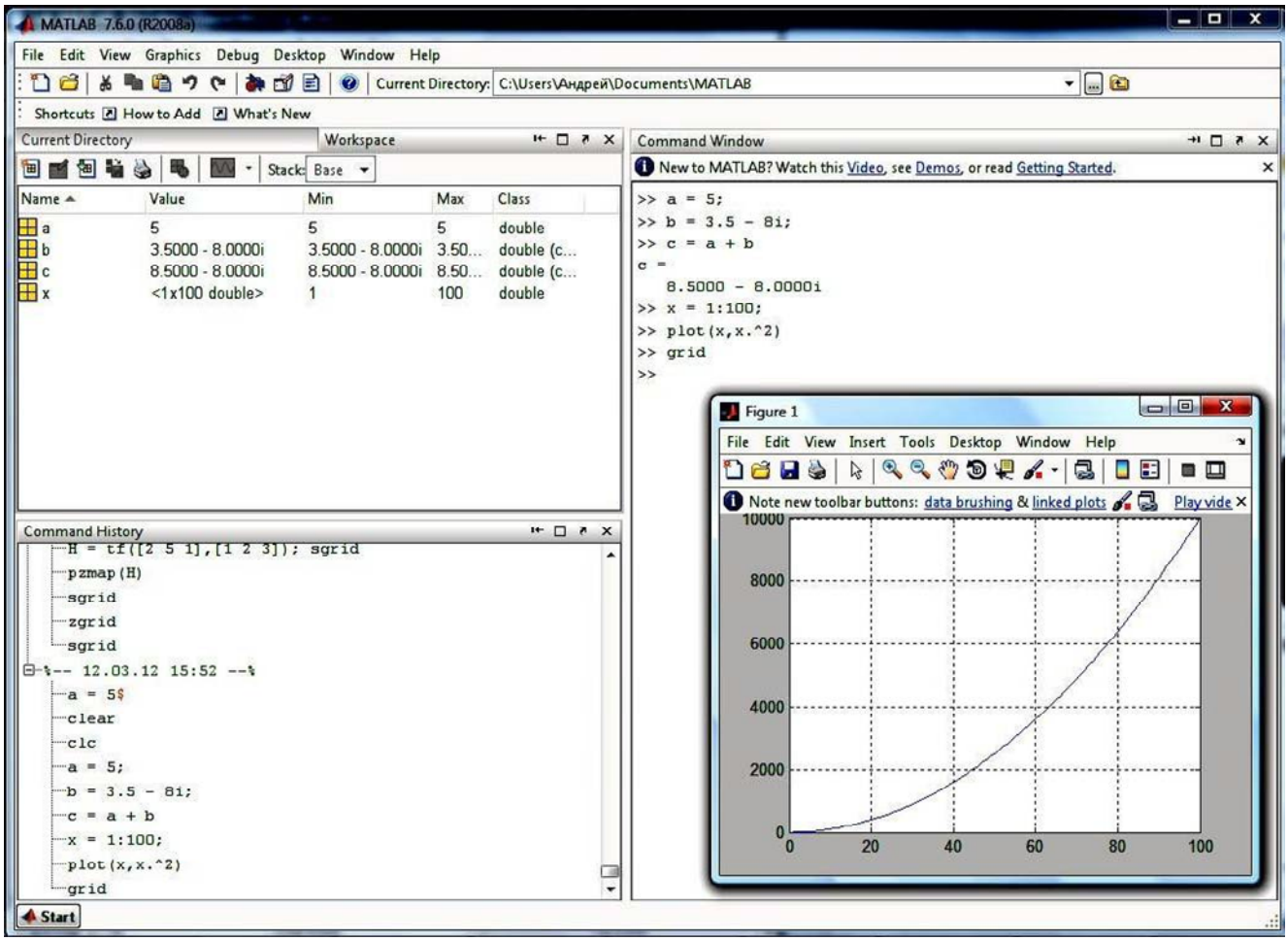


Рис.1. Рабочее окно системы Matlab

В окне **Current Directory/Workspace** на рис. 1 открыта вкладка **Workspace** в которой отображаются заполнение рабочей области (памяти) системы переменными и другими объектами. При переключении на вкладку **Current Directory** в этом окне будет отображаться перечень объектов текущей рабочей директории.

На экране можно открыть несколько окон системы, в том числе редактор, графики. По команде меню **Desktop > Desktop Layout > Default** возвращается исходное состояние и размер окон. Двойная стрелка в командном окне » указывает на начало командной строки.

Главное меню и панели инструментов показаны на рис. 2.

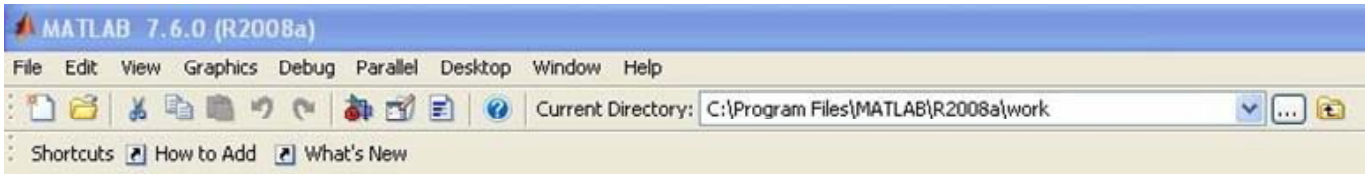


Рис. 2. Главное меню и панели инструментов

Назначение кнопок панели инструментов представлено на рис. 1.3 (в порядке следования слева направо):

- создать файл-программу;
- открыть новый файл в редакторе/отладчике;
- вырезать, копировать, вставить;
- отменить действие, вернуть действие;
- запуск Simulink (создание новой модели Simulink);
- создание графического интерфейса пользователя (GUI);
- профилирование;
- вызов справочной системы;
- демонстрационные примеры;
- параметры системы;
- текущая директория;
- просмотр путей и переход по файловой структуре

Рис. 3. Назначение кнопок панели инструментов

Математические выражения

Как и большинство систем программирования Matlab обеспечивает выполнение математических выражений, но в отличие от других языков программирования эти выражения содержат целые матрицы. В состав математических выражений входят: переменные, числа, операторы, функции.

Переменная – массив/вектор/скаляр

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

Имена переменных, констант и функций могут быть составлены из любых символов алфавита , кроме специальных и цифр, начинаются с буквы. Для идентификации переменной используются первые 31 символ имени. Система различает верхний/нижний регистр букв в именах переменных и функций (как системных, так и пользовательских, SIN(x) приводит к ошибке).

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 дейтсвительной и мнимой частями, при мнимой части проставлен символ **i** или **j** (без знака умножения);
- символ (**;**) после определения переменой подавляет вывод результата на экран, как и при записи оператора.

Примеры ввода переменных в систему.

```
» a=3
a =
     3
»
```

```
» a=3;
» a
a =
     3
»
```

```
» sqrt(-1)
ans =
      0 + 1.0000i
```
в данном случае выполнено неявное присвоение, автоматически использована системная переменная **ans** (Answer)

В том случае, если переменная уже существует, ей присваивается новое значение. При задании матрицы (массива) в памяти резервируется пространство, в соответствии с ее размером. Несмотря на выполненное выше присвоение переменной ***a*** численного значения, т.е. задании ее в виде скаляра, допустимо сразу или после каких-либо действий переопределить переменную, например, задать в виде матрицы:

```
» a = [1 2 3; 4 5 6; 7 8 9];
»
```

Или так:

```
» a = [1  2  3;
       4  5  6;
       7  8  9];
»
```

Диапазон представления чисел при вычислениях $10^{-308} – 10^{308}$, все внутренние вычисления производятся с двойной точностью. Точность вычислений с плавающей точкой характеризует специальная системная константа

eps = 2^{-52} или 2.2204e-016

Операторы

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.

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 дейтсвия и служат для оформления строки;
- символ % указывает на то, что следующие за ним символы являются комментарием, с него можно начать строку.

Признаком начала командной строки является указатель >>, выводимый системой автоматически, при готовности к приему команды.

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.

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

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
» a=3+2i;              % i без знака умножения
```

Встроенные функции

Matlab выполняет вычисление большого количества стандартных математических функций – sin, cos, exp, log, sqrt, abs и других (вызов справки с перечнем help elfun). Задание отрицательного аргумента для log, sqrt не приводит к ошибке, а автоматически вычисляется соответствующий результат в виде комплексного числа.

В системе Matlab реализованы также более сложные и специальные функции. Они сгруппированы по разделам, для их использования рекомендуется обратиться к справочной системе (help specfun, help elmat и др.)

Стандартные математические функции являются встроенными, что обеспечивает высокую скорость вычислений, однако алгоритм, описание порядка действий и вмешательство в программу недоступны.

Специальные функции реализованы в виде **m**-файлов (sinh, gamma), т.е. являются внешними. **m**-файл доступен для просмотра, анализа выполняемых операций, а также для изменений, которые необходимы для решения какой-либо конкретной задачи. **m**-файлы функций содержат в начальных строках комментарии, которые выводит справочная система при задании команды ***help funname***, где ***funname*** – имя функции.

Векторы и матрицы

```
» a = [1 2 3 4 5];      % ввод вектора
» sum(a)                % использование функции SUM
ans =
» 15
» a = [1 2; 3 4];       % сложение матриц
» b = [5 6; 7 8];
» a + b
ans =
     6   8
```

```
 »    | 10    |12              |
» c = a + b                    |% с присвоением переменной
 c =  | 6     |       8        |
      |10     |      12        |
» a(1,1)                       % задание одного элемента матрицы
ans =1
```

Элементы матрицы имеют индексы в соответствии с принятым в математике порядком:

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

```
» a = [1 2 3
       4 5 6
       7 8 9];
» a(2,3)
ans =
     6
```

```
» a
a =
     1     2
     3     4
» a'                        % транспонирование
ans =
     1     3
     2     4
```

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

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

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;
```

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). Скалярное произведение двух векторов возвращает функция **dot**, а векторное — **cross**:

```
>> s=dot(a,b)
>> c=cross(a,b)
```

Векторное произведение определено только для векторов из трех элементов.

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**).

Особенности оператора умножения (деления, возведения в степень)

MatLab поддерживает два вида вычислительных операций с векторами и матрицами: матричные и поэлементные.

Наряду с умножением по правилу матричного умножения, существует операция поэлементного умножения **.*** (точка со звездочкой). Данная операция применяется к векторам одинаковой длины 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**:

```
» a * b                 % операция с матрицами
ans =
    19    22
    43    50
» a.* b                 % операция с массивами (таблицами)
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 обязательно 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.

Оператор (:)

```
» 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))
» a = [1 2
       3 4];
» b = [5
       6];
» c = [a,b]            % добавление столбца в матрицу
c =
     1     2     5
     3     4     6
```

Программирование, файл-функция и файл-программа

Программирование в системе MatLab осуществляется путем написания M – файлов, которые могут быть либо файл-функциями, либо файл-программами. Файл-программа является текстовым файлом с расширением m (M-файлом), в котором записаны команды и операторы MatLab.

Для запуска редактора M-файлов следует нажать кнопку New M-file на панели инструментов, либо выбрать в меню File в пункте New подпункт M-file. На экране появляется окно редактора. В нем можно набрать команды, например для построения графика:

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

Для запуска программы следует выполнить команду Run из меню Debug в редакторе. Если предварительно не была выполнена команда Save, то система предложит сначала сохранить файл программы (Save and Run).

Для запуска программы можно набрать в командной строке имя M-файла (без расширения) и нажать <Enter>, то есть выполнить, как команду MatLab. При этом следует учесть, что путь к каталогу с M-файлом должен быть известен MatLab.

Когда текущий каталог установлен, то все M-файлы, находящиеся в нем, могут быть запущены из командной строки, либо из редактора M-файлов. Все переменные файл-программы после ее запуска доступны в рабочей среде, т. е. являются глобальными. Убедится в этом можно, выполнив команду **whos**. Файл-программа может использовать переменные рабочей среды. Например, если в командной строке был выполнен оператор:

```
>> a=[0.1 0.4 0.3 1.9 3.3];
```

то файл-программа, содержащая строку **bar(a)**, построит столбцевую диаграмму вектора a (разумеется, если он не был переопределен в самой файл-программе).

Файл-функции отличаются от файл-программ тем, что они могут иметь входные и выходные аргументы, а все переменные, определенные внутри файл-функции, являются локальными и не видны в рабочей среде. 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. Пример простейшей файл-функции с двумя входными и одним выходным аргументами:

```
function c=mysum(a,b)
c=a+b;
```

Практически все функции MatLab являются файл-функциями и хранятся в одноименных М-файлах. Функция **sin** допускает два варианта вызова: **sin(x)** и **y=sin(x)**, в первом случае результат записывается в **ans**, а во втором — в переменную **y**. Наша функция **mysum** ведет себя точно так же. Более того, входными аргументами **mysum** могут быть массивы одинаковых размеров или массив и число.

Разберем теперь, как создать файл-функцию с несколькими выходными аргументами. 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
```

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.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 1:

рабочая среда системы MATLAB, вычислительные операторы, задание векторов и матриц, функциональное программирование, построение графиков.

ЗАДАНИЕ НА ЛАБОРАТОРНЫЙ ПРАКТИКУМ

1. Изучить ввод данных в системе MATLAB, выполнить примеры, приведенные выше в разделе «Методические указания».
2. Выполнить операции с векторами (исходные данные взять из таблицы 1).

Для заданных векторов **a** и **b** длины **n**:

- вычислить их сумму, разность и скалярное произведение;
- образовать вектор $c = [a_1, a_2, ..., a_n, b_1, b_2, ..., b_n]$, определить его максимальный и минимальный элементы и поменять их местами;
- упорядочить вектор **c** по возрастанию и убыванию;
- переставить элементы вектора **c** в обратном порядке и записать результат в новый вектор;
- найти векторное произведение $u = [a_1, a_3, a_4]$ и $v = [b_2, b_3, b_4]$.

3. Вычислить табличное значение заданного выражения-функции ***f(x)*** для всех элементов заданной таблицы чисел — матрицы ***A*** (задать ***x = A,*** записать формулу вычисления ***f(x),***и получить результат в виде матрицы того же размера, что и исходная матрица, варианты заданий см. ниже).

4. Выполнить аппроксимацию данных полиномом и построить графики.

- задать два вектора **x** и **y** одинаковой размерности с координатами точек графика некоторой функциональной зависимости и построить график **y(x)** (функция **plot(x,y)).**
- выполнить аппроксимацию заданных точек функции по методу наименьших квадратов полиномом заданной степени (n = 2-5) (функция **polyfit**).
- выполнить вычисление значений аппроксимирующего полинома в точках, определяемых аргументом - вектором **x** и занести полученные значения в

вектор **z (**функция **polyval)**. Построить график **z(x)** и сравнить его с заданным графиком **y(x).**

- графики строить в одном окне (**plot(x,y, x,z)).** На рисунок графика нанести сетку командой **grid.**
- оценить качество приближения по разности $y(x_i)$-$z(x_i)$.

5. Анализ динамической системы с использованием функций Matlab

Задать параметры динамической системы (гиростабилизатор, прибор из курсового проекта или передаточная функция из табл.3 с отрицательной обратной связью), выполнить оценку свойств с использованием подсистемы LTI-viewer (команда **ltiview** - вызов интерактивного обозревателя свойств линейных моделей, см. с. 16).

Таблица 1 (варианты данных к п. 2 задания)

a = [0.5 3.7 6.0 -4.3 1.2 -2.7 2.4 2.2];	b = [3.6 7.0 7.0 5.4 2.6 -2.7 -6.4 0.3].
a = [-4.8 -1.3 -1.0 0.7 4.0 5.8 4.3 -8.0];	b = [-1.1 -1.9 7.1 -2.1 6.8 2.8 0.3 1.6].
a = [1.0 -3.9 -2.3 -3.3 -1.7 2.2 -0.6 1.8];	b = [2.7 -2.7 -2.2 4.4 0.4 -6.0 -3.4 -5.2].
a = [-2.4 3.3 -0.1 3.6 7.4 -2.8 0.3 2.2];	b = [6.3 0.6 4.3 -3.7 -7.0 3.7 3.7 8.0].
a = [8.4 -5.9 -6.5 -0.9 6.9 -1.7 1.7 0.8];	b = [-0.0 2.0 -1.5 7.5 -4.0 -3.0 -6.2 0.0].
a = [5.3 6.8 -7.1 6.8 -4.0 -2.3 -4.4 -0.2];	b = [7.5 -1.5 -4.9 -4.6 -2.3 -5.3 5.5 2.3].
a = [1.2 -4.1 -0.8 -0.7 -2.2 1.7 3.3 -6.1];	b = [-1.5 2.2 1.0 -4.3 -0.0 -1.8 -1.5 2.4].
a = [6.6 -5.0 -2.7 8.3 3.8 1.9 1.1 2.7];	b = [-1.0 3.2 4.2 -6.4 1.9 -6.5 -6.2 -8.1].
a = [-1.9 0.4 1.8 4.2 -3.8 -4.7 4.0 -2.1];	b = [-8.7 -4.2 -1.4 2.8 -2.2 7.8 0.0 -0.1].
a = [0.9 1.7 -3.2 -3.8 7.3 6.0 -0.2 8.6];	b = [0.6 -0.4 -6.9 -2.2 1.6 3.8 -3.2 0.4].

Таблица 2 *(варианты данных к п. 3 задания)*

1. $f(x) = x^3 - 2x^2 + \sin x - 4$; $A = \begin{bmatrix} 9.33 & -4.01 & 8.19 & 2.64 \\ 0.55 & 3.81 & 3.32 & 5.07 \end{bmatrix}$.

2. $f(x) = \dfrac{e^x - x}{e^x + x}$; $A = \begin{bmatrix} 9.32 & 0.21 & -9.89 & 3.11 \\ 0.54 & 4.99 & 5.01 & -0.03 \end{bmatrix}$.

3. $f(x) = \sqrt{1 + \sqrt{|x|^3 + 1}}$; $A = \begin{bmatrix} -1.54 & 0.49 & 3.11 & 2.99 \\ 4.05 & -5.85 & 3.72 & 0.11 \end{bmatrix}$.

4. $f(x) = e^x \sin x - e^{-x} \cos x$; $A = \begin{bmatrix} -9.04 & 3.36 & 3.09 & -2.49 \\ -4.33 & -5.09 & 9.74 & 1.65 \end{bmatrix}$.

5. $f(x) = \ln(|x|) \sin \pi x$; $A = \begin{bmatrix} 0.33 & 0.95 & 7.12 & -9.22 \\ -0.64 & 3.76 & 1.34 & -0.03 \end{bmatrix}$.

6. $f(x) = e^{x^2 + x + 1}$; $A = \begin{bmatrix} -4.53 & -2.12 & -6.54 & -3.21 \\ 3.43 & 7.43 & -0.25 & 1.64 \end{bmatrix}$.

7. $f(x) = \dfrac{\sqrt[3]{x^2 - 1}}{|x| + 3}$; $A = \begin{bmatrix} 0.23 & 3.89 & -4.23 & -7.25 \\ 5.84 & 5.13 & -0.89 & 3.55 \end{bmatrix}$.

8. $f(x) = \dfrac{1}{1 + \dfrac{1 + x}{1 - x^2}}$; $A = \begin{bmatrix} -5.84 & 9.84 & 0.23 & 1.59 \\ -9.25 & -0.25 & 1.54 & 0.43 \end{bmatrix}$.

9. $f(x) = \dfrac{x^3 + \sin x}{x^3 - \cos x} \sqrt{e^x + 1}$; $A = \begin{bmatrix} 0.64 & 6.34 & 0.32 & -4.23 \\ 1.19 & 3.23 & 1.54 & 0.43 \end{bmatrix}$.

10. $f(x) = \arcsin(\cos x^2)$; $A = \begin{bmatrix} \pi & 2.2\pi & -2\pi & 0.3\pi \\ 3\pi & -\pi & 0.1\pi & 5\pi \end{bmatrix}$.

Таблица 3 *(варианты данных к п. 5 задания)*

1.	$W(p)=\dfrac{5p+2}{p^3+3p+1}$
2.	$W(p)=\dfrac{4p^2+1}{p^3+3p^2+p+1}$
3.	$W(p)=\dfrac{p^2+2}{p^3+4p+1}$
4.	$W(p)=\dfrac{p+0.3}{p^3+3p^2+1}$
5.	$W(p)=\dfrac{p-1}{p^3+3p^2+p+1}$
6.	$W(p)=\dfrac{3p^2+p}{p^3+4p+1}$
7.	$W(p)=\dfrac{p^2-p+1}{p^3+4p^2+4p+1}$
8.	$W(p)=\dfrac{-3p^2+p}{p^2+4p+1}$
9.	$W(p)=\dfrac{p^2+p+1}{p^4+p^2+p+1}$
10.	$W(p)=\dfrac{p^2-3p-3}{p^4+p^2+p+1}$

Методические указания к п. 5 *(вызов обозревателя LTIviewer)*

Зададим LTI модель передаточной функцией, например

```
>> w=tf([1],[1 1])
```

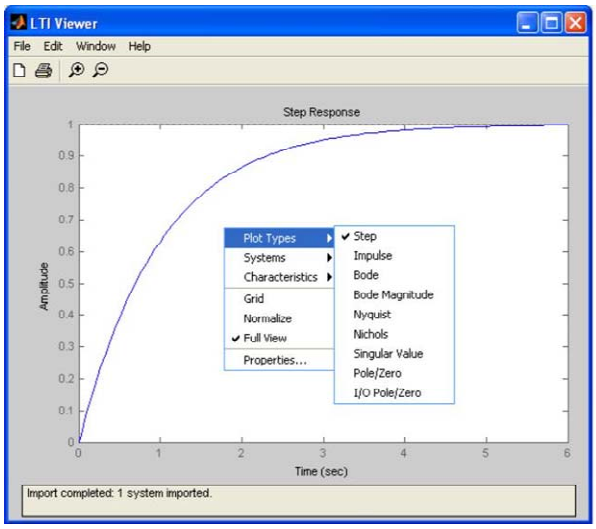
```
 Transfer function:
```

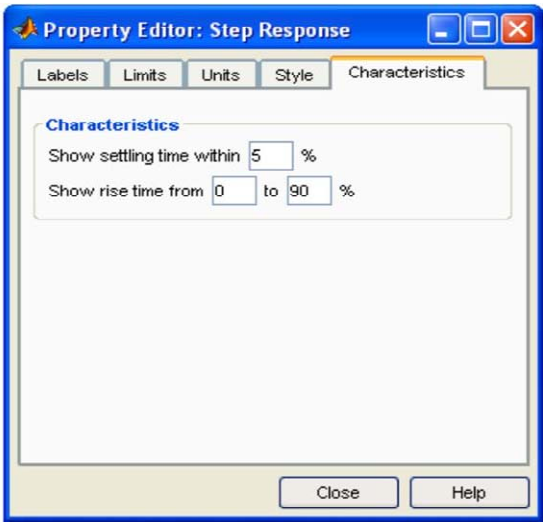
$$\frac{1}{s+1}\text{,}$$

затем вызываем LTIviewer командой **`>> ltiview`**

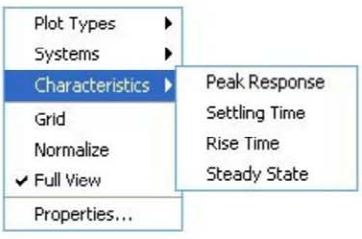
При этом появляется рабочее окно с заготовкой графика. По команде меню File – Import появляется окно, представляющее содержимое рабочей области, в данном случае там присутствуют для объекта – системы w и w1. Выбрав объект,

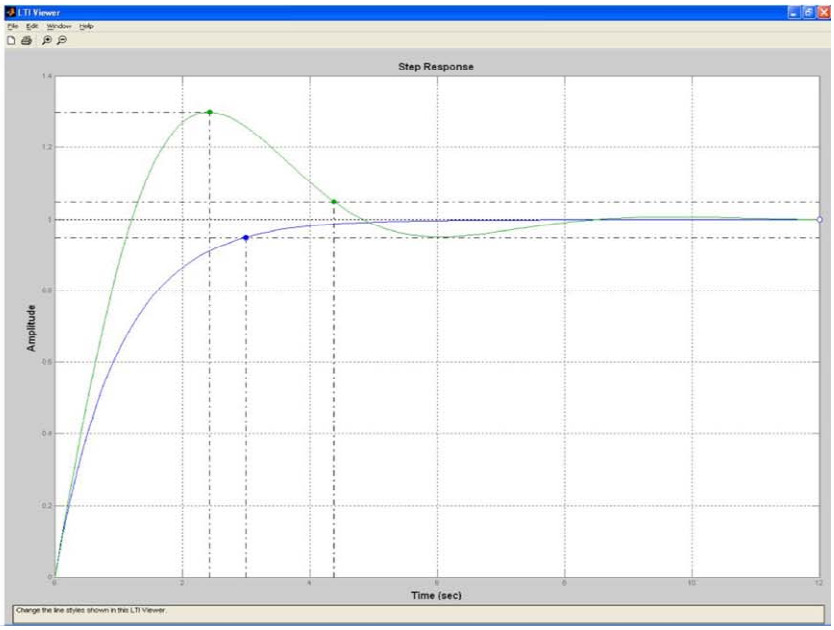
нажимаем OK и смотрим результат на графике. Правая кнопка мыши открывает выпадающее меню, позволяющее выполнить настройку графиков, в том числе выбрать требуемую характеристику системы (реакция на ступенчатый или импульсный входной сигнал, ЛАЧХ, ЛФЧХ и др.) и другие параметры обозревателя.





В LTIviewer можно загрузить несколько систем, в данном случае это w и w1. заданные характеристики переходного процесса рассчитываются автоматически. Параметры задаются в выпадающем меню **Properties**.





Результат представлен на графике.

РАЗДЕЛ 2. АНАЛИЗ ДИНАМИЧЕСКИХ ОБЪЕКТОВ С ИСПОЛЬЗОВАНИЕМ СИСТЕМЫ MATLAB И SIMULINK

Цели работы: Ознакомление с пакетом прикладных программ Control System Toolbox системы MATLAB, предназначенным для работы с линейными стационарными системами и интерактивным пакетом SIMULINK, предназначенным для моделирования нелинейных динамических систем.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ РАБОТЫ

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.

Основные классы объектов в MatLAB

В системе MatLAB определены 6 встроенных классов вычислительных объектов:

double	Числовые массивы и матрицы действительных или комплексных чисел с плавающей запятой в формате двойной точности, наиболее распространенный тип переменных в системе MatLAB, с которым оперирует большинство функций и процедур
sparse	Двумерные комплексные разреженные матрицы двойной точности. Разреженная структура применяется для компактного хранения матриц с незначительным количеством ненулевых элементов. Разреженные матрицы требуют применения специальных методов для решения задач
char	Массивы символов - переменные, являющиеся совокупностью символов, каждый символ занимает 16 битов, эту совокупность часто называют строкой.
struct	Массивы записей (структуры). Объекты класса struct состоят из нескольких составляющих, которые называются полями, каждое из которых имеет собственное имя.
cell	Массивы ячеек Переменные класса cell (ячейки) являются

	совокупностью некоторых других массивов. Массивы ячеек позволяют объединить связанные данные (возможно, разных типов и размеров) в единую структуру.
uints	Массивы 8-разрядных целых чисел без знаков, позволяет хранить целые числа от 0 до 255 в 1/8 части памяти, необходимой для чисел двойной точности. Никакие математические операции для этого класса данных не определены.

Каждому типу данных соответствуют собственные функции и операторы обработки, т.е. методы.

Производные классы MatLAB

Рассмотренные выше классы объектов построены таким образом, что на их основе могут быть созданы новые классы объектов.

В языке MatLAB отсутствует необходимость и возможность предварительного объявления типа или класса переменных, которые будут использованы. То же самое относится и к объектам любых вновь создаваемых классов.

Объекты класса создаются в виде структур (записей), т.е. относятся к потомкам (наследникам) класса **struct**. Поля структуры и операции с полями являются доступными только внутри методов данного класса.

Все М-файлы, определяющие методы для объектов данного класса, должны размещаться в специальном каталоге, который называется каталогом класса и обязательно имеет имя, состоящее из знака @ (коммерческое "эт") и имени класса, т.е. имеет вид @<имя класса>. Каталог класса должен быть подкаталогом одного из каталогов, описанных в путях доступа системы MatLAB, но не самим каталогом. Каталог класса обязательно должен содержать М-файл с именем, совпадающим с именем класса. Этот файл называют конструктором класса. Назначение такого М-файла — создавать объекты этого класса, используя данные в виде массива записей (структуры) и приписывая им метку класса.

В системе MatLAB на этой основе создан и используется встроенный класс объектов **sym**, с которым работает пакет символьных вычислений Symbolic Math Toolbox, и который позволяет выполнять вычисления с символьными переменными и матрицами. Пакет Control System Toolbox использует класс объектов LTI и три

его дочерних подкласса **tf**, **zpk**, **ss**, которые поддерживают алгоритмы анализа линейных стационарных систем автоматического

Классы пакета CONTROL

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 трем разным представлениям ЛСС:

1. TF-объект (Transfer Function — передаточная функция);
2. ZPK-объект (Zero-Pole-Gain — нули-полюсы-коэффициент передачи);
3. SS-объект (State Space — пространство состояния).

Объект LTI, как наиболее общий, содержит информацию, независящую от конкретного представления ЛСС (непрерывного или дискретного), 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. Наконец, объект класса SS определяется четверкой матриц, описывающих динамическую систему в пространстве состояния. Ниже приведены основные атрибуты этих классов, их обозначения и смысл.

Атрибуты (поля) LTI-объектов

Ниже NU, NY и NX определяют число входов (вектор U), выходов (вектор Y) и переменных состояния (вектор X) ЛСС соответственно; ОМ (SISO) — одномерная система, т.е. система с одним входом и одним выходом; ММ (MIMO) — многомерная система (с несколькими входами и выходами).

	Специфические атрибуты 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**)] и **[(s^2-5s+6)/(s^2+s)]**
Variable	Имя (тип) переменной (из перечня): возможны варианты: s, p, z, гл-1 или q. По умолчанию принимается s (для непрерывных переменных) и z (для дискретных); имя переменной влияет на отображение и создает дискретную ПФ для дискретных сигналов
	Специфические атрибуты ZPK-объектов
z	Нули: вектор-строка для ОМ-систем; для ММ-систем — массив ячеек из векторов-строк размером NY на NU
p	Полюсы: вектор-строка для ОМ-систем; для ММ-систем — массив ячеек из векторов-строк размером NY на NU
k	Коэффициенты передачи: число — для ОМ-систем, матрица NY на NU для ММ-систем
Variable	Имя (тип) переменной (из перечня) То же, что и для TF-объекта (см. выше)
	Специфические атрибуты SS-объектов (моделей пространства состояния)
a,b,c,d	A,B,C,D — матрицы, в соответствии с уравнениями в переменных состояния: x = Ax + Bu, y = Cx + Du
e	E — матрица для систем Descriptor'a (описателя). По умолчанию E = eye(size(A))
StateName	Имя переменной состояния (не обязательное). Массив ячеек NX на I из строк (используйте '' для состояний без имени). Пример: {'положение'; 'скорость'}

	Атрибуты, общие для всех LTI-моделей
Ts	Дискрет по времени (в секундах) Положительный скаляр (период дискретизации) для дискретных систем. Ts = -1 для дискретных систем с неустановленной частотой дискретизации. Ts = 0 для непрерывных систем.
Td	Задержки входов (в секундах): Вектор 1 на NU промежутков времен задержек входов. Установка Td как скаляра определяет единую задержку по всем входам. Используется только для непрерывных систем. Используйте D2D для установки задержек в дискретных системах. Td = [] для дискретных систем
InputName	Имена входов: строка для систем с одним входом. Массив ячеек NU на 1 из строк для систем с несколькими входами (используйте '' для переменных без имени). Примеры: 'момент' или {'напор1 ; 'отклонение элеронов1}'
OutputName	Имена выходов: строка для систем с одним выходом. Массив ячеек NY на 1 из строк для систем с несколькими выходами (используйте '' для переменных без имени). Пример: 'мощность' или {'скорость' ; 'угол атаки'}
Notes	Заметки: любая строка или массив ячеек из строк символов. Пример: 'Эта модель создана в январе 2000'
Userdata	Дополнительная информация или данные. Может быть любого типа MATLAB

Функции из перечня методов класса LTI:

tfdata, ssdata, zpkdata, step, impulse, rlocus, pade, series, parallel, bode, margin, nichols, nyquist, ss2ss, augstate, damp, get, issiso, Iticheck, trange, balreal, display, gram, kalman, set, tzero, dssdata, kalmd, modred, pzmap, sigma, uplus, canon, eig, inherit, Iqgreg, quickset, connect, estim, initial, Iqry, norm, reg, covar, evalfr, isct, Isim, rlocfind, ctrb, fgrid, isempty, Iti, obsv,

Перечень основных процедур пакета CONTROL, сгруппированных по функциональному назначению

	Создание LTI-моделей
ss	Создает модель пространства состояния
zpk	Создает модель нули/полюсы/коэффициенты передачи
tf	Создает модель передаточной функции
dss	Специфицирует описатель модели пространства состояния
fflt	Специфицирует цифровой фильтр

set	Установка/модификация атрибутов LTI-модели
Itiprops	Детальная справка об атрибутах LTI-моделей

Извлечение данных

ssdata	Извлечение матриц пространства состояния
zpkdata	Извлечение данных о нулях/полюсах/КП
tfdata	Извлечение числителя(-лей) и знаменателя(-лей) ПФ
dssdata	Получение информации о версии описателя SSDATA
get	Получение информации о значениях свойств LTI-модели

Получение информации об отдельных характеристиках модели

class	Информация о типе модели (ss, zpk или tf)
size	Информация о размерах матриц входа и выхода
isempty	Проверка, является ли LTI-модель пустой
isct	Проверка, является ли модель непрерывной
isdt	Проверка, является ли модель дискретной
isproper	Проверка, является ли модель правильной
issiso	Проверка, имеет ли модель один вход и один выход
isa	Проверка, является ли LTI-модель моделью заданного типа

Преобразование системы

ss	Преобразование в пространство состояния
zpk	Преобразование в нули/полюсы/КП

Работа с передаточными функциями с использованием функций пакета CONTROL Передаточная функция формируется функцией **tf**, в качестве аргументов задаются коэффициенты полиномов числителя и знаменателя в порядке убывания степени оператора **s**:

```
>> tf([3 1], [1 1 1])
   Transfer function:
    3 s + 1
  -----------
  s^2 + s + 1
```

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 данную переменную можно использовать при обращении к функциям или выполняя необходимые преобразования, используя операторы Matlab. Например, передаточная функция модели ***SYS*** , состоящей из параллельно соединенных блоков

SYS1 и ***SYS2***, рис. 1 может быть получена следующим образом:

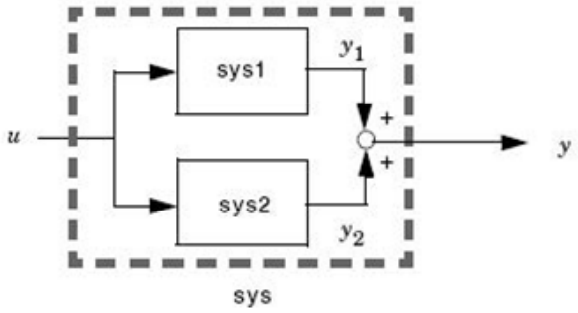


Рис. 1

В первом варианте результат получен путем операции сложения, а во втором – с использованием функции **parallel.** И в том и в другом случае получается идентичный результат, рис. 2.

```
sys1 = tf([1 1],[1 2 1])
sys2 = tf([1],[3 1])
sys = sys1 + sys2
```

или

```
sys1 = tf([1 1],[1 2 1])
 sys2 = tf([1],[3 1])
 parallel(sys1,sys2)
```

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

    4 s^2 + 6 s + 2
-----------------------
3 s^3 + 7 s^2 + 5 s + 1
```

Рис. 2

Для преобразования моделей предназначены следующие функции:

parallel	параллельное соединение
feedback	соединение с обратной связью

series	последовательное соединение
append, connect	различные соединения блоков

Выполняя преобразования систем с использованием операторов и функций необходимо внимательно следить за получаемыми результатами. Например, передаточную функцию системы с единичной отрицательной обратной связью можно получить с использованием операторов Matlab:

```
>> w1 = tf([3 1], [1 1
1]) Transfer
function:
  3 s + 1
-----------
s^2 + s + 1
>> 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
```

Однако результат не вполне понятен, т.к. увеличился порядок системы. Преобразование с помощью функции **feedback** дает иной результат:

```
>> W2 = feedback(w1,1)
Transfer function:
   3 s + 1
-------------
s^2 + 4 s + 2
```

Выполним преобразование передаточной функции **W1 = w1/(1+w1)** 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)
```

Из полученного результата видно, что в числителе и знаменателе присутствует множитель второго порядка **(s^2+s+1)**, который соответствует знаменателю передаточной функции прямой цепи и подлежит сокращению в результирующей формуле. Проверяем:

```
>> zpk(W2)
Zero/pole/gain:
   3 (s+0.3333)
--------------------
(s+3.414) (s+0.5858)
```

Результат соответствует формуле для **W1** при сокращении сомножителя **(s^2+ s+1)**.

Анализ свойств динамической системы

При анализе системы, состоящей из нескольких блоков, возникает задача получения передаточной функции для всей системы или для подсистемы, включающей в себя ряд блоков. Для выполнения преобразований целесообразно задать передаточные функции всех блоков системы, а затем выполнять расчеты для подсистем и системы в целом.

Например, для расчета параметров передаточной функции силового гиростабилизатора можно написать следующую программу.

```
% исходные данные
H = 2000; Jp = 0.8;
Jst = 125; Dst = 300; Dp = 0;
Kr = 50000;
% расчет параметров
Omega = H *
sqrt(1+Dst*Dp/H^2)/sqrt(Jst * Jp); t0
= 1/Omega;
dzeta =
((Dp/(2*H))/sqrt(Jst/Jp)+(Dst/(2*H))*sqrt(Jp/Jst))*(1/sqr
t(1+ (Dst*Dp)/H^2));
t1 = 2*t0;
% формирование передаточных функций
w1 = tf([Kr/H],[1 0]);
w2 = tf([1],[t0^2 2*t0*dzeta 1]);
w3 = tf([1], [t1 1])*tf([1], [t1 1]); %
корректирующее звено % передаточная функция
гиростабилизатора с коррекцией
W = w1 * w2 * w3;
```

В результате получена передаточная функция

```
>> W
 Transfer function:
                          25
   ------------------------------------------------------
2.5e-009 s^5 + 5.06e-007 s^4 + 0.0001262 s^3 + 0.02006 s^2 + s
```

Или, в форме **zpk:**

```
>> zpk(W)
Zero/pole/gain:
                        10000000000
                 --------------------------------
               s (s+100)^2 (s^2    + 2.4s + 4e004)
```

В Matlab предусмотрены функции извлечения данных из модели. С помощью функции **tfdata** получим коэффициенты полиномов числителя и знаменателя передаточной функции гиростабилизатора:.

```
>> [n,d] = tfdata(W, 'v')
```

```
n =
     0     0     0     0     0    25
d =
   2.5000e-009 5.0600e-007 1.2620e-004 2.0060e-002 1.0000e+000
```

С помощью функции **roots** получим значения корней полинома знаменателя

```
>>roots(d)
ans =
          0
 -1.2000e+000 +2.0000e+002i
 -1.2000e+000 -2.0000e+002i
 -1.0000e+002
-1.0000e+002
```

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

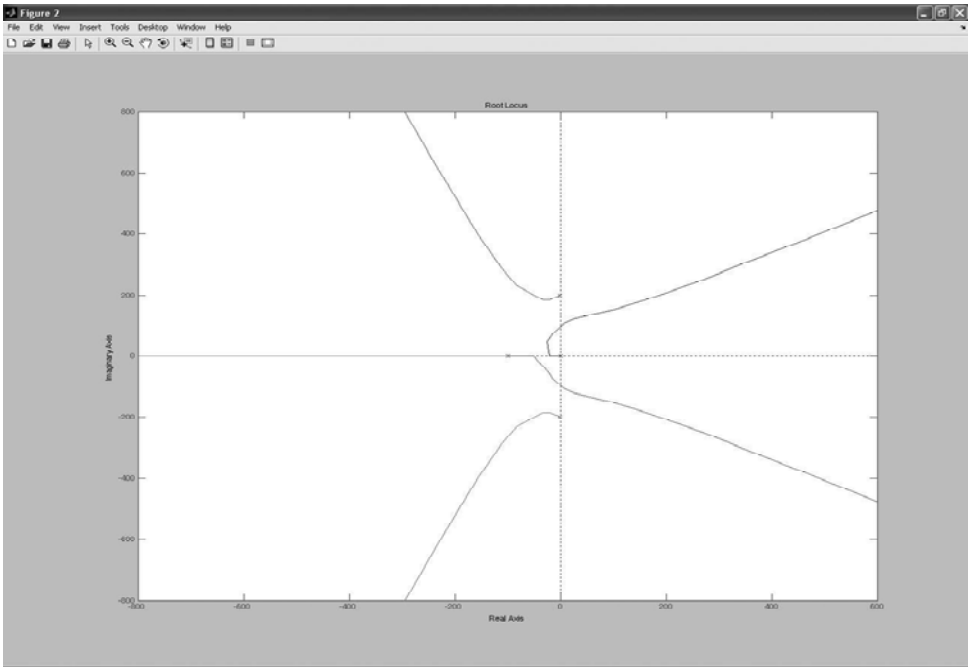
```
>>[np,dz]
=pzmap(W) np =
          0 -
 1.2000e+000
 +2.0000e+002i -
 1.2000e+000 -
 2.0000e+002i -
 1.0000e+002 -
 1.0000e+002

dz =
  Empty matrix: 0-by-1
```

Функция **damp** вычисляет показатели демпфирования и собственные частоты системы:

```
>> damp(W)
                                  Freq.
      Eigenvalue     Damping     (rad/s)
                               -
  0.00e+000         1.00e+000   0.00e+000
 -1.00e+002         1.00e+000   1.00e+002
 -1.00e+002         1.00e+000   1.00e+002
 -1.20e+000 +
 2.00e+002i         6.00e-003   2.00e+002
 -1.20e+000 -
 2.00e+002i         6.00e-003   2.00e+002
```

Функция **rlocus** строит корневой годограф.



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. Нужный блок можно найти, набрав егь название в строке поиска и нажав кнопку **Find Block** в строке инструментов (бинокль). На рис. 1 показано, как система нашла блок **Integrator**.

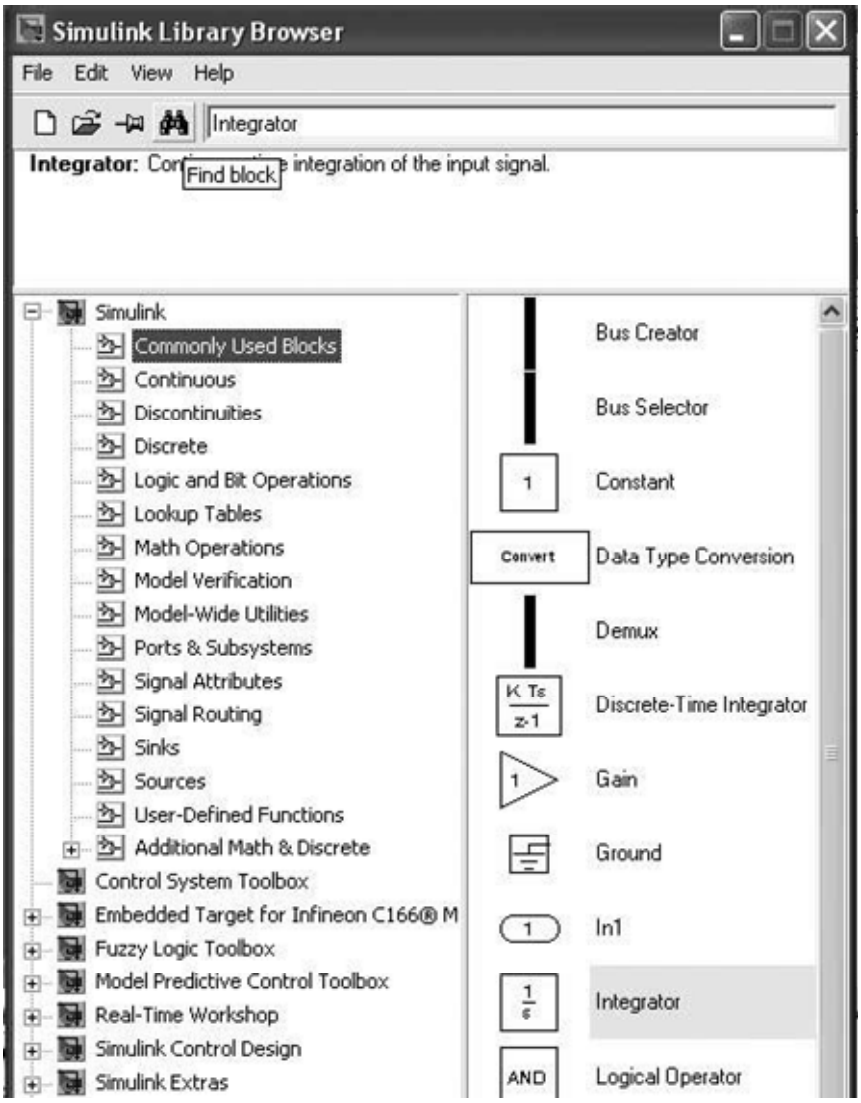


Рис. 1

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.

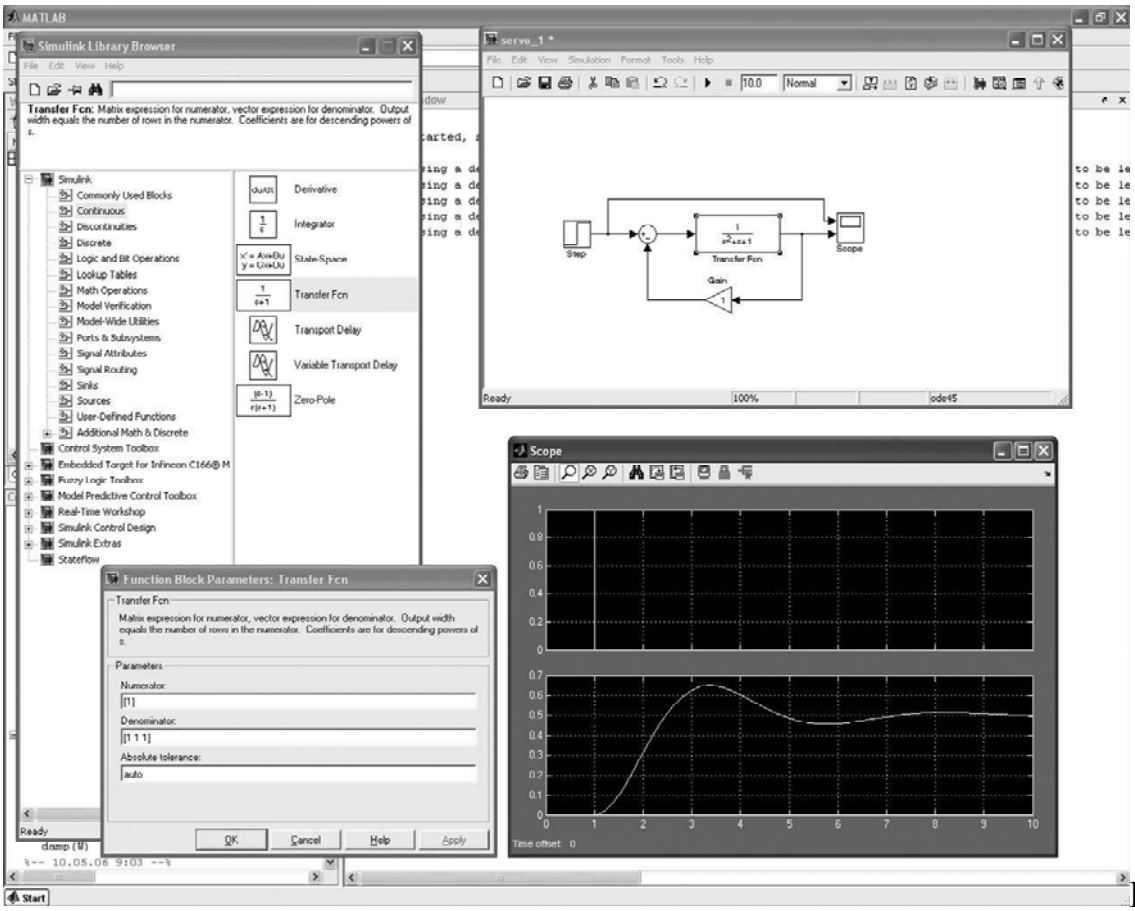


Рис. 2

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.

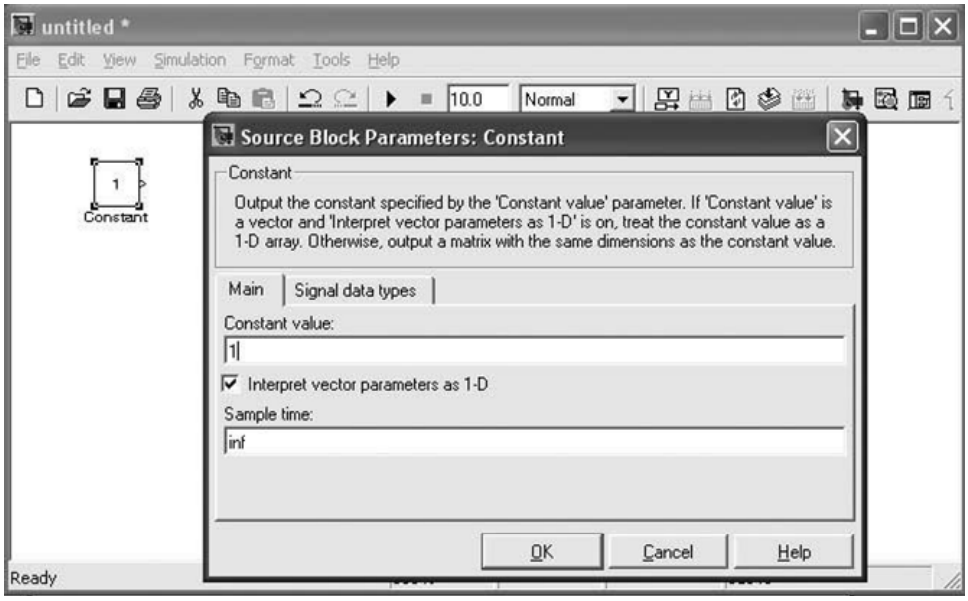


Рис. 3

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.

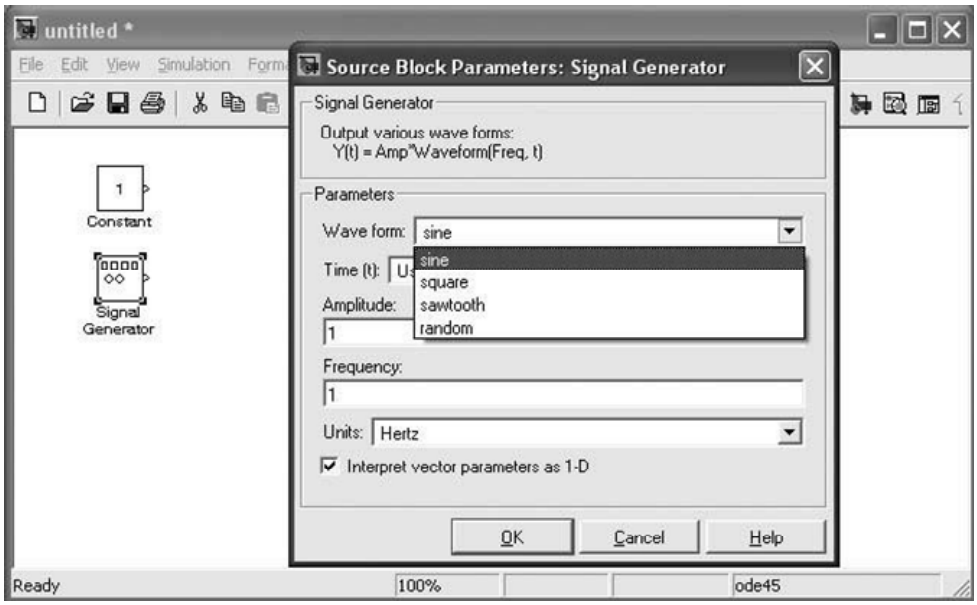


Рис. 4

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.

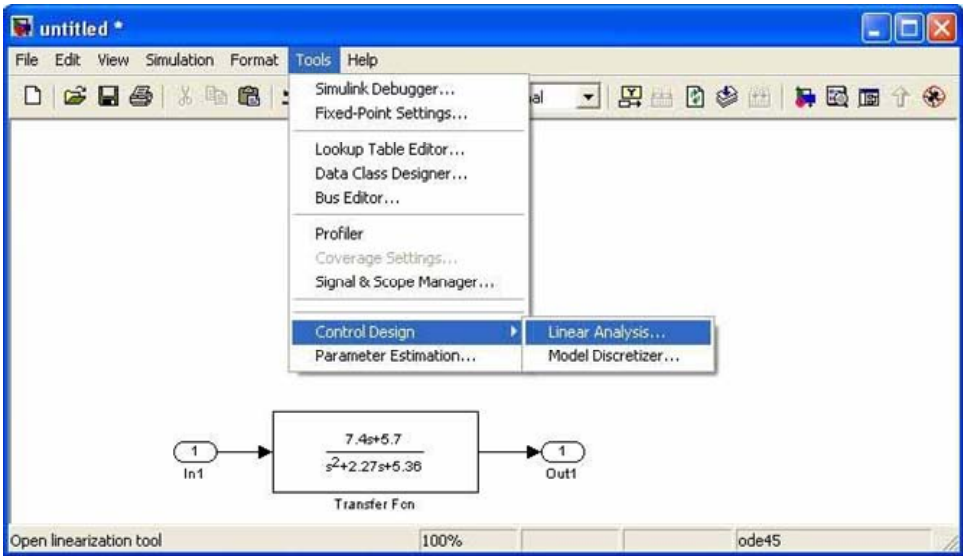


Рис. 5

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.

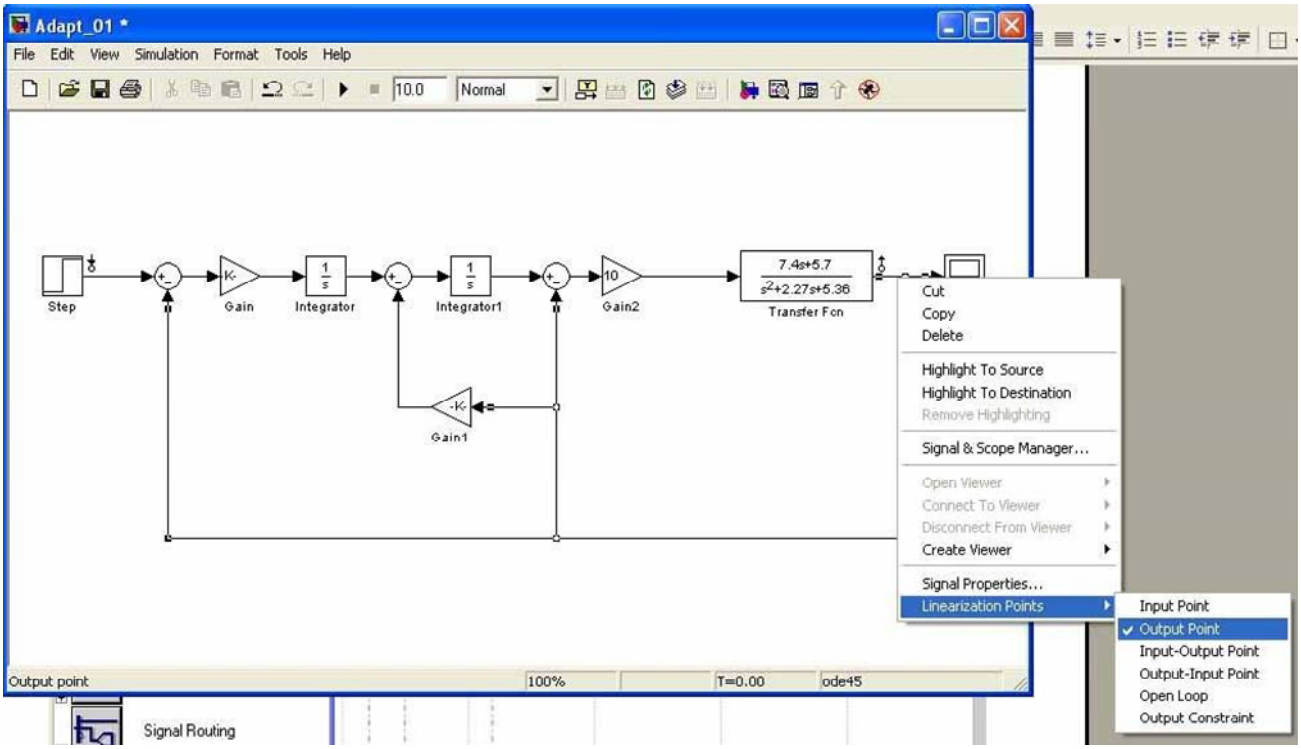


Рис. 6

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. 7 и в меню **Edit** задать команду **Create Subsystem**, рис. 8.

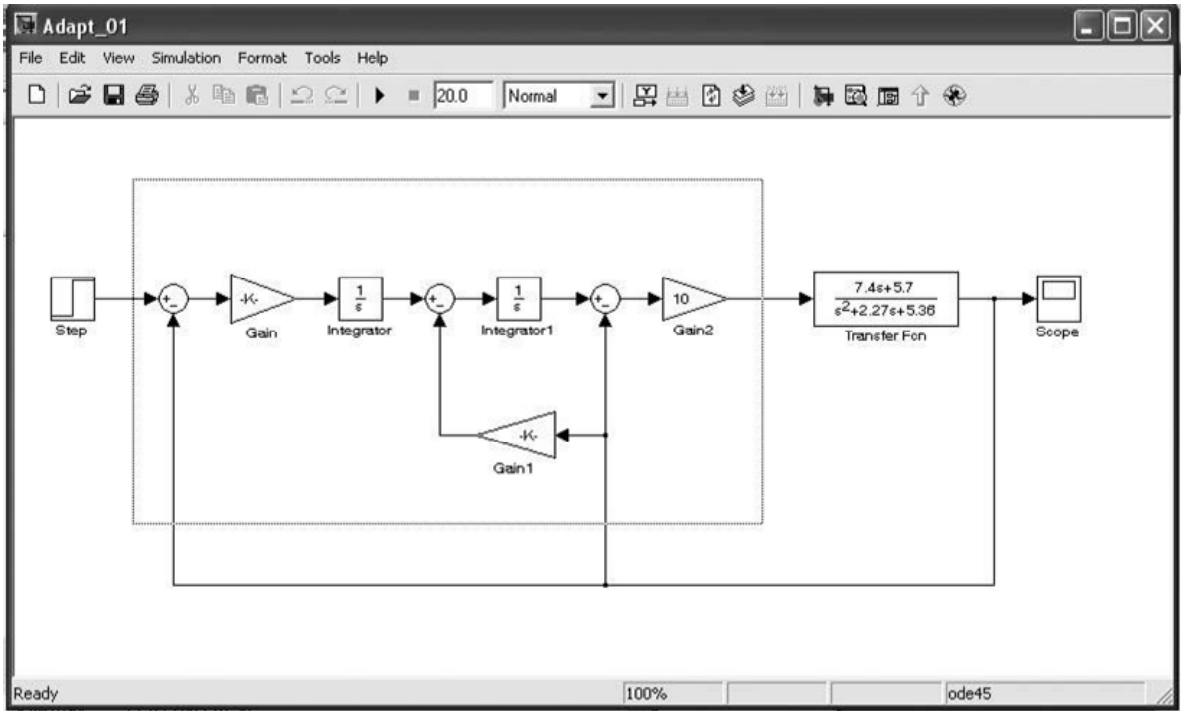


Рис. 7

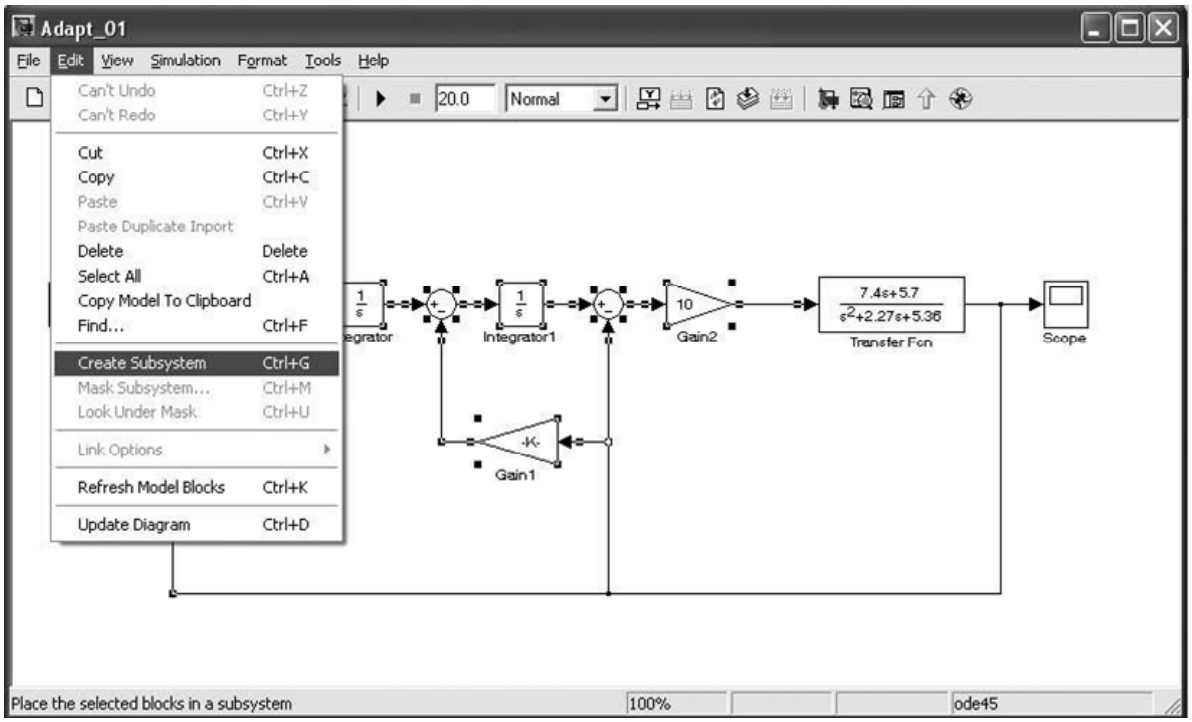


Рис. 8

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.

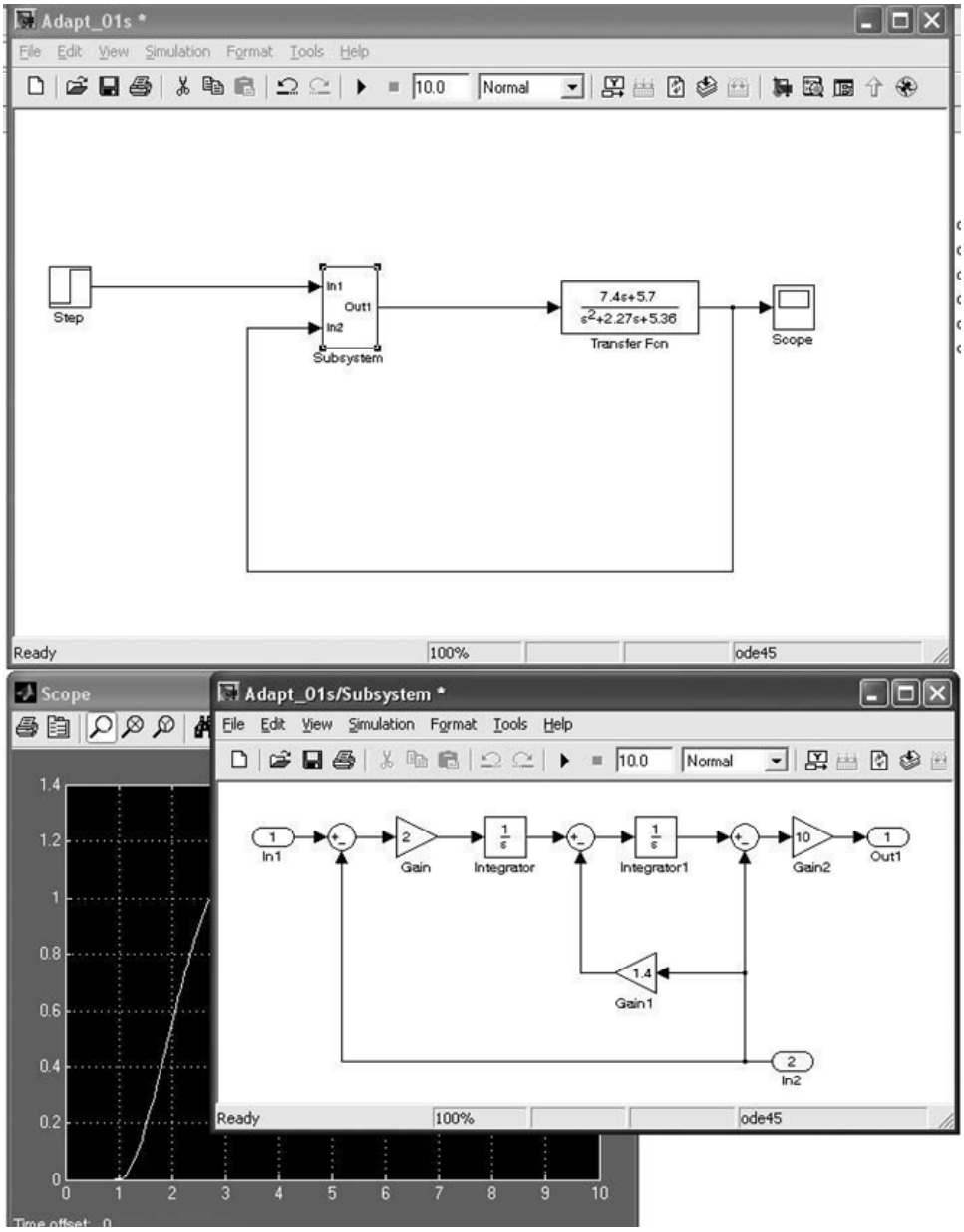


Рис. 9

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

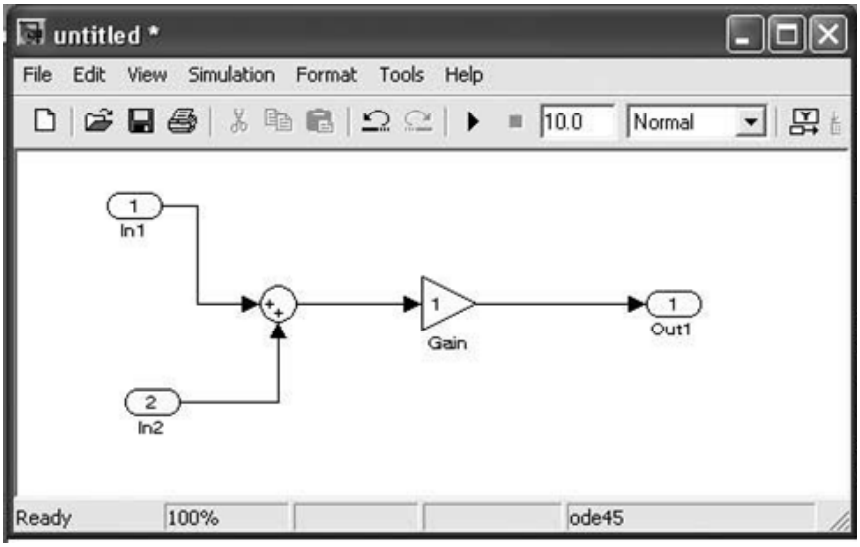


Рис. 10

Передача данных из Simulink в рабочую среду MATLAB

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.

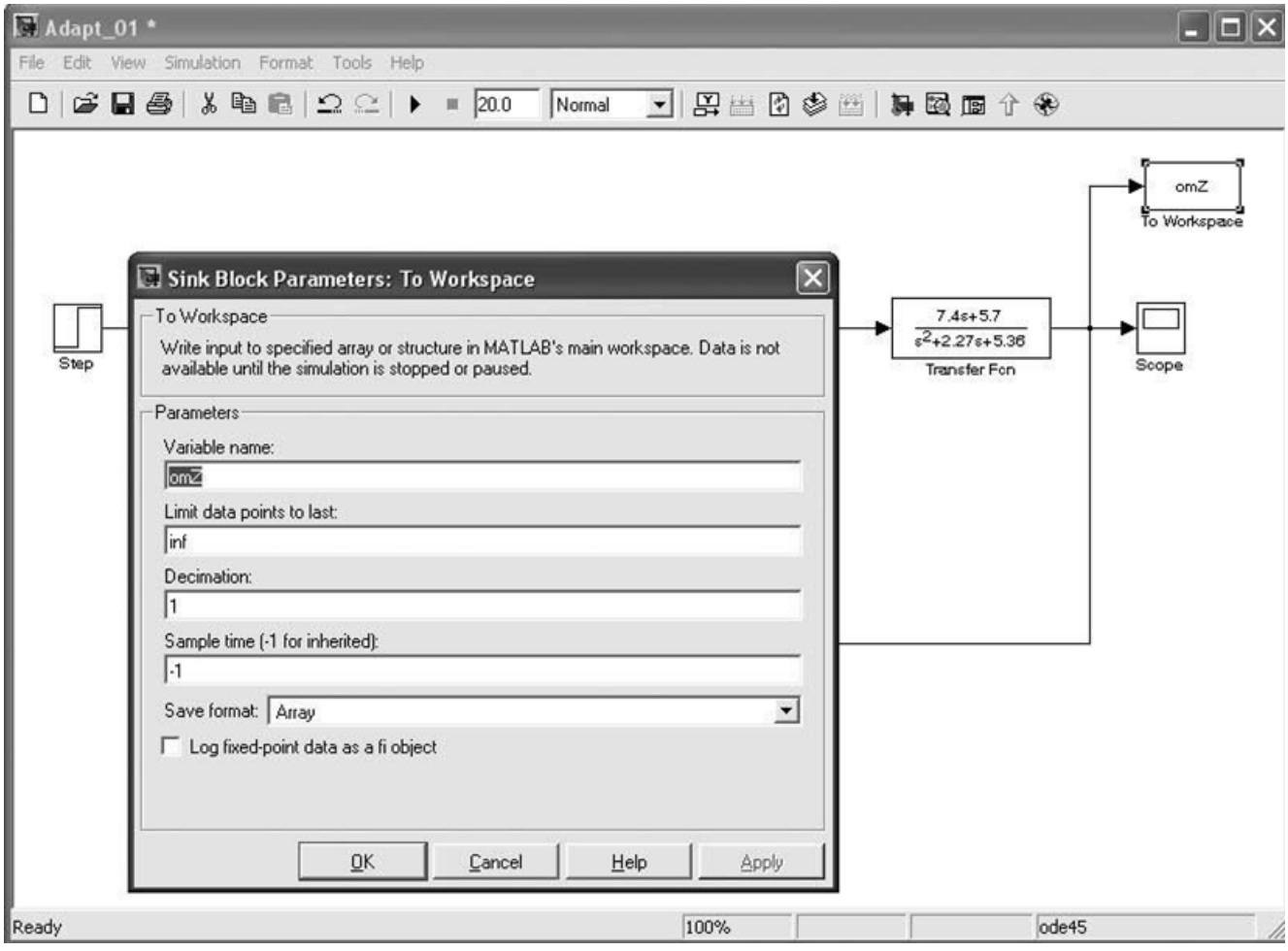


Рис. 11

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 2: анализ линейных стационарных систем в MATLAB, обработка передаточных функций, создание и анализ моделей динамических объектов в Simulink.

ЗАДАНИЕ НА ЛАБОРАТОРНЫЙ ПРАКТИКУМ

1. Изучить способы задания параметров линейных стационарных систем (LTI) в MATLAB. Задать параметры передаточных функций блоков из схемы гиростабилизатора (прибор из курсового проекта или передаточная функция из табл.3 (с. 15) с отрицательной обратной связью), сформировать передаточную функцию замкнутой и разомкнутой систем.
2. Выполнить анализ свойств гиростабилизатора: найти корни характеристического полинома передаточной функции **(roots)**, построить ЛАХ, корневой годограф, переходный процесс (использовать LTIviewer, вызов из командной строки, метод указания с. 16).
3. Выполнить анализ свойств гироскопического стабилизатора с использованием Simulink (использовать LTIviewer, вызов из Simulink). Параметры переходного процесса вывести в рабочую среду MATLAB.
4. Выполнить настройку и анализ свойств динамической системы ЛА-САУ с использованием Simulink. Сформировать подсистему из блоков эталонной модели в прямой цепи системы. Построить графики переходных процессов для заданных переменных системы.

Определить:

- время регулирования;
- перерегулирование.

Задача: выполнить настройку канала управления угловой скоростью тангажа самолета с адаптивным алгоритмом управления на основе эталонной модели в прямой цепи. Модель объекта управления – самолета, задана в виде передаточной функции

$$W(p)=\frac{b_1 p + b_0}{p^2 + a_1 p + a_0}=\frac{\omega_z(p)}{\delta_в(p)} \qquad (1)$$

где ω_z и $\delta_в$ – приращения угловой скорости тангажа и угла отклонения руля высоты;

$a0$, $a1$, $b0$, $b1$ – коэффициенты, определяемые аэродинамическими свойствами самолета. Алгоритм управления построен не основе эталонной модели, динамические свойства которой определены дифференциальным уравнением второго порядка:

$$\ddot{\omega}_z^* + g_1\dot{\omega}_z^* + g_0\omega_z^* = g_0\omega_z^0 \qquad (2)$$

Числовые значения параметров g_0 и g_1 назначаются такими, чтобы динамические параметры системы самолет–САУ соответствовали заданным требованиям к качеству переходных процессов, протекающих при поступлении на вход заданного сигнала $\omega z0 = const$.

Структурная схема системы САУ представлена на рис.12.

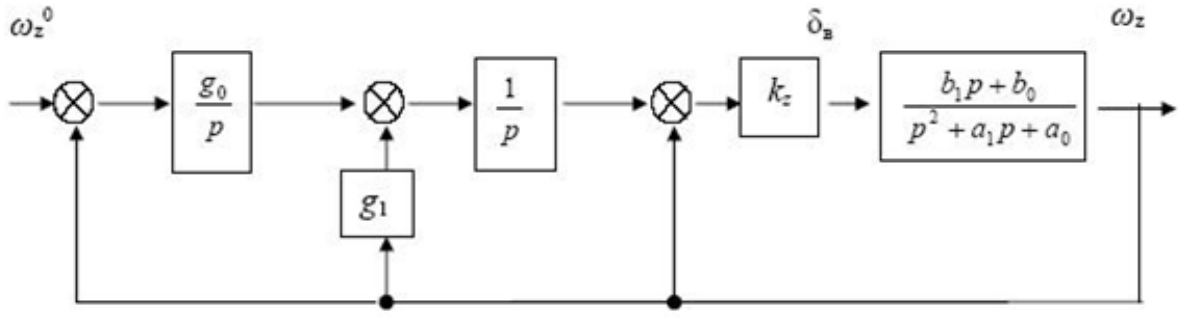


Рис. 12

Параметры $g0$ и $g1$ эталонной модели рассчитываются по заданным значениям времени регулирования tрег и коэффициента демпфирования ξ (постоянная времени $\boldsymbol{T} \approx t$рег /3, собственная частота $\omega = 1/\boldsymbol{T}$, время регулирования tрег выбирается большим, чем постоянная времени объекта регулирования, в данном случае tрег = 1.5 – 5 с, ξ = 0.5 – 1).

Величину коэффициента $\boldsymbol{k}z$ следует уточнить экспериментально. Теоретически, при $\boldsymbol{k}z \to \infty$, переходный процесс в системе ЛА-САУ в точности соответствует заданным параметрам эталонной модели. Но большие значения коэффициентов невозможно реализовать средствами аналоговой техники, а в цифровом вычислителе усложняется алгоритм контроля. Приемлемое качество регулирования в системе данного вида достигается при $\boldsymbol{k}z = 1 – 10$.

Модель системы для данного примера, сформированная в среде Matlab-Simulink, приведена на рис. 13.

Варианты задания

Номер режима полета	a_0	a_1	b_0	b_1
1.	5.36	2.27	5.7	7.4
2.	89.48	6.9	159.0	57.0
3.	121.4	4.66	96.6	42.0
4.	8.52	1.66	7.78	13.8
5.	18.11	0.65	3.4	10.0

Вариант	Номер режима полета	коэффициент демпфирования ξ	время регулирования $t_{рег}$
1.	1	0.4, 0.7, 1.0	3 с
2.	1	0.4, 0.7, 1.0	4 с
3.	1	0.4, 0.7, 1.0	6 с
4.	2	0.4, 0.7, 1.0	3 с
5.	2	0.4, 0.7, 1.0	4 с
6.	2	0.4, 0.7, 1.0	6 с
7.	3	0.4, 0.7, 1.0	3 с
8.	3	0.4, 0.7, 1.0	4 с
9.	3	0.4, 0.7, 1.0	6 с
10.	4	0.4, 0.7, 1.0	3 с
11.	4	0.4, 0.7, 1.0	4 с
12.	4	0.4, 0.7, 1.0	6 с
13.	5	0.4, 0.7, 1.0	3 с
14.	5	0.4, 0.7, 1.0	4 с
15.	5	0.4, 0.7, 1.0	6 с

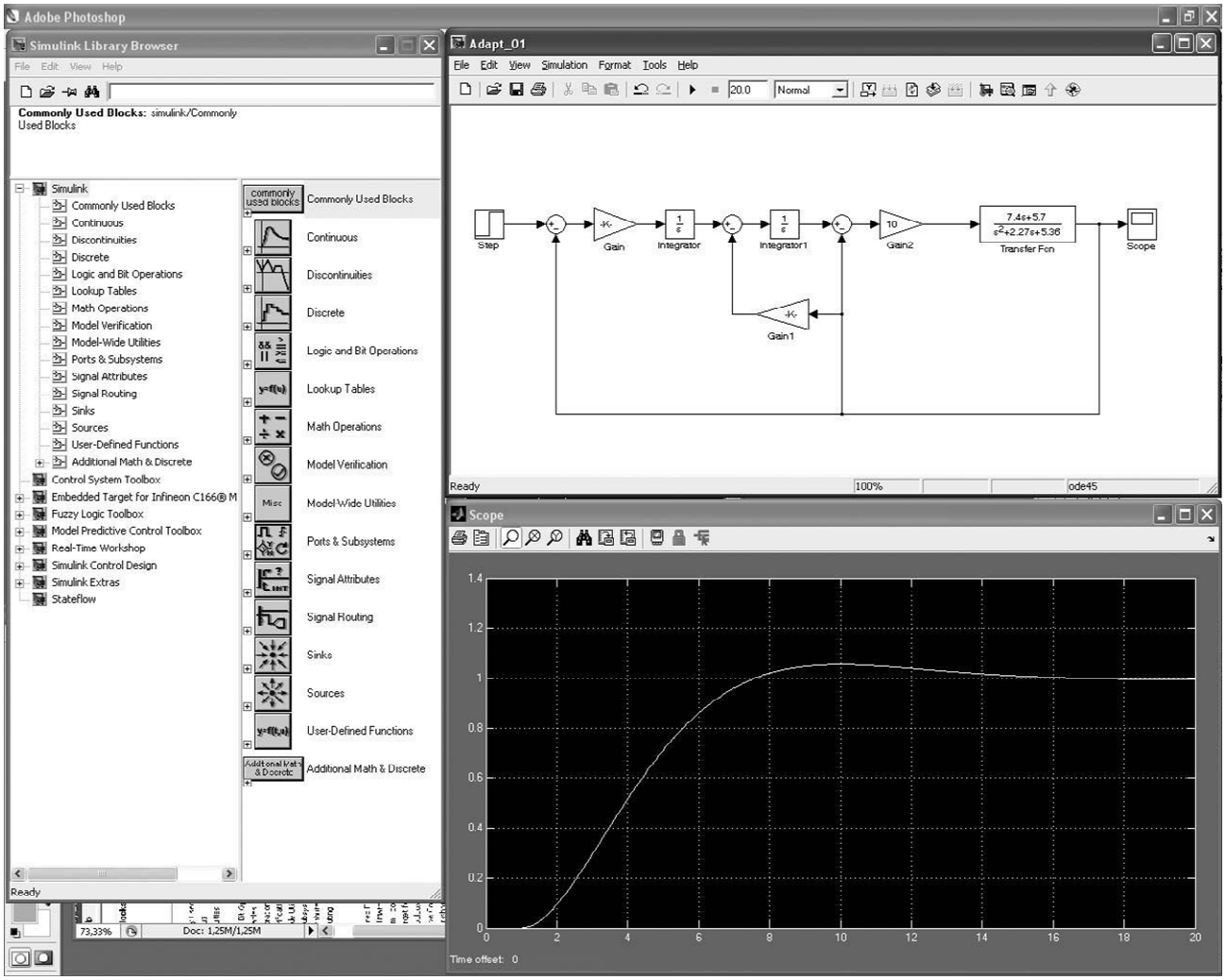


Рис. 13

Контрольные вопросы

1. Интерфейс системы Matlab, меню, рабочая директория, командная строка
2. Интерфейс системы Matlab, настройка окон, просмотр рабочей области
3. Интерфейс системы Matlab, редактор (Editor)
4. Интерфейс Simulink, меню, библиотека блоков Simulink
5. Toolbox и Blockset, назначение, состав, использование, справка
6. Editor – редактор, основные функции, трассировка программ
7. Переменные, операторы, функции – имена, правила записи, порядок использования
8. Переменные в среде Matlab, имена, присвоение значений, комплексные числа, матрицы и векторы, системные переменные и константы
9. Ввод матриц и векторов, просмотр переменных, выбор элемента матрицы, выбор строки или столбца матриц

10. Интерактивный режим работы в Matlab, командная строка, ввод и вывод данных, построение графика
11. Просмотр рабочего пространства, переменных, массивов
12. Справочная система Matlab, команды help и doc
13. LTI – модели, характеристическое уравнение, корни полинома
14. Программирование в Matlab, командное окно, программы, функции (правила оформления)
15. LTI – модели, виды, задание параметров модели
16. LTI – модели, преобразования
17. LTI – модели, анализ свойств, (в Matlab и Simulink), преобразования
18. LTI – модели, передаточная функция, задание параметров, анализ свойств (в Matlab и Simulink), преобразования
19. LTI-Viewer, анализ свойств модели (в Matlab и Simulink)
20. Формирование модели в Simulink: библиотека блоков, блоки и их параметры, входы и выходы, Subsystem, параметры моделирования динамических систем
21. Порядок подготовки системы дифференциальных уравнений для моделирования в Simulink
22. Анализ свойств динамических систем в Matlab и Simulink