РиС на ЭВМ. Модуль MATLAB

История создания

Matlab был разработан как язык программирования в конце 1970-х годов Кливом Моулером (когда он был деканом факультета компьютерных наук в Университете Нью-Мексико). Целью разработки служила задача дать студентам факультета возможность использования программных библиотек Linpack и EISPACK без необходимости изучения Фортрана. Вскоре новый язык распространился среди других университетов и был с большим интересом встречен учёными, работающими в области прикладной математики. До сих пор в Интернете можно найти версию 1982 года, написанную на Фортране, распространяемую с открытым исходным кодом.

Джон Литтл (John N. (Jack) Little), Стив Бангерт (Steve Bangert) и Клив Моулер совместными усилиями 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.

Компоненты системы

MATLAB:

Программная среда для технических расчетов:

- 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 для всех продуктов MathWorks.

SIMULINK:

Визуальная среда для моделирования, симуляции, разработки динамических и встраиваемых систем:

- линейные, нелинейные, дискретные, непрерывные, гибридные и многоскоростные системы;
- расширения для систем управления, обработки сигналов, систем связи и других областях с применением системного инжиниринга;
- основа для Модельно-Ориентированного Проектирования.

Пакеты расширений системы Matlab:

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.

Simulink Control Design – комплект расширений **Simulink** для линеаризации комплексных нелинейных объектов. Унции и графические инструменты Simulink Control Design позволяют проводить анализ линеаризованной модели в частной области, настраивать параметры регулятора и синтезировать систему управления.

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

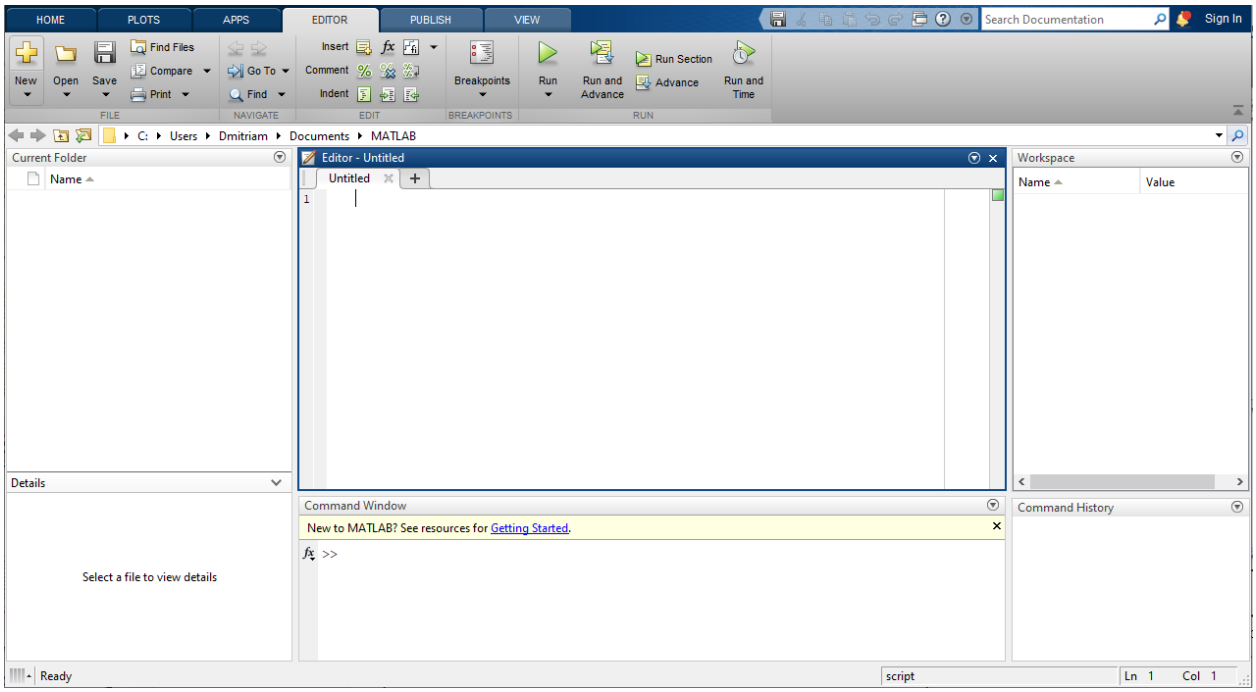
Robust Control Toolbox – это пакет расширения **Matlab** для разработки системы управления объектами с неопределенностями и нелинейностями различного типа. Robust Control Toolbox позволяет проектировать и настраивать системы управления с учетом чувствительности к неопределенным параметрам, возмущениям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

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

Полезные ссылки

Ссылка	Описание
http://www.mathworks.com	Сайт компании The MathWorks
http://www.mathtools.net	Научный портал, поддерживаемый компанией MathWorks
http://www.exponenta.ru	Образовательный математический сайт
http://sl-matlab.ru	Центр компетенций MathWorks
http://softline.ru	Учебный центр Softline

Интерфейс системы Matlab

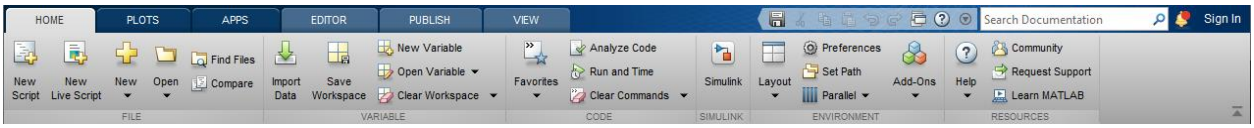


Интерфейс рабочего стола содержит:

- меню, расположенное в верхней части окна, выделенное серым цветом заливки; содержит вкладки **HOME**, **PLOTS**, **APPS**; начинать работу следует на вкладке **HOME**;
- 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.

Вкладка HOME



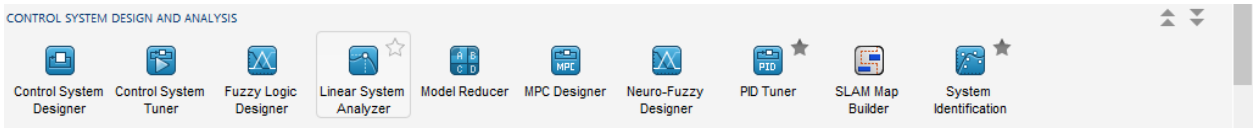
- **New Script** – создание нового файла (.m);
- **New Live Script** – создание интерактивного нового скриптового файла;
- **Open** – открытие файлов проекта;
- **Compare** – сравнение двух файлов по содержанию;
- **Import Data** – импорт различных данных в Matlab, в том числе и таблиц.
- **Save Workspace** – сохранение глобальных переменных в отдельный файл Matlab;
- **New Variable** – создание новых переменных;
- **Open Variable** – открытие ранее сохраненных переменных;
- **Clear Workspace** – удаление созданных переменных в рабочей сессии;
- Подраздел **CODE** – различные операции с кодом (**Analyze Code** - оптимизация, отладка; **Run and Time** – выполнение кода и измерение времени выполнения; **Clear Commands** - очистка области выполнения команд);
- **Simulink** – запуск модуля Simulink;
- Подраздел **ENVIRONMENT** – настройки программы, настройки расположения окон программы, редактирование рабочих директорий, параллельные вычисления, подключение дополнительных расширений;
- Подраздел **RESOURCES** – Help, полезные материалы Matlab, Community, поддержка.

Вкладка PLOTS

Работа с графиками.

Вкладка APPS

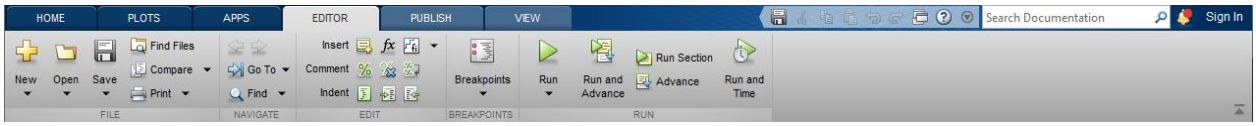
На этой вкладке показаны приложения и модули **Matlab**, которые установлены в системе. Важным элементом для наших проектов является **Control System Design and Analysis**.



Приложение **Linear System Analyzer** – позволяет анализировать временные и частотные характеристики систем LTI. (**LTI** – линейные стационарные системы, которые описываются линейными дифференциальными уравнениями). Это приложение позволяет:

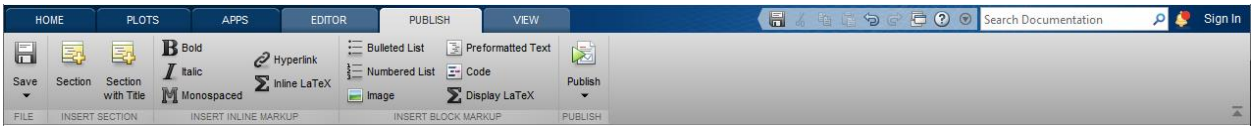
- Просматривать и сравнивать графики отклика систем SISO и MIMO или нескольких линейных моделей одновременно;
- Генерировать графики отклика системы на входные воздействия;
- Создавать графики частотных характеристик системы;
- Просматривать основные характеристики отклика системы на входные воздействия.

Вкладка EDITOR



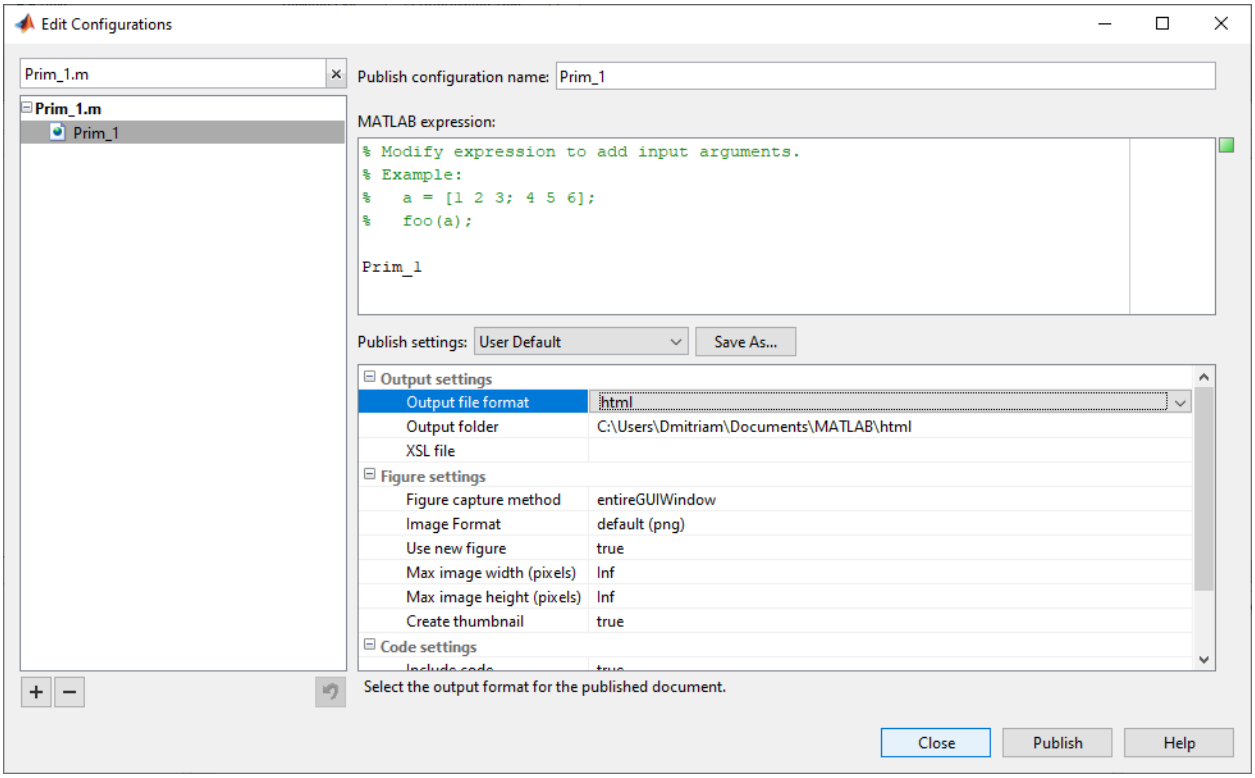
Появляется при создании или открытии файла программы (скрипта) Matlab. Скрипт – текстовый файл с расширением «.m». В данной вкладке доступны различные инструменты для работы с текстовыми файлами Matlab (создание, открытие, сохранение, сравнение файлов; добавление секций, встроенных функций, комментариев в код программы; отладка и различные режимы исполнения сценариев).

Вкладка PUBLISH

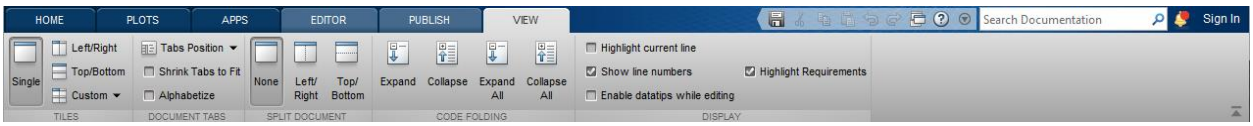


Позволяет делать вывод программы со всеми результатами в читаемый и редактируемый файл. Результаты выводятся по секциям. Позволяет создавать отчет в форматах – html, xml, doc, ppt, pdf.

Настройка формирования отчета по написанной программе вызывается нажатием кнопки «Publish» → «Edit Configurations».



Вкладка VIEW



Настройка внешнего вида рабочего пространства.

Основы программирования MATLAB

Matlab – интерпретатор и система программирования. Позволяет вводить операторы по одному в командную строку в интерактивном режиме или запускать m-файлы, в которых операторы заданы в виде списка.

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, цифр, знака подчеркивания, но должно начинаться 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;

eps = 2^{-52} или $2.2204e^{-16}$ – характеризует точность вычислений с плавающей точкой.

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.

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.

Операторы:

1) Арифметические операторы: + - * /
2) Логические операторы:

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))

3) Операторы отношения:

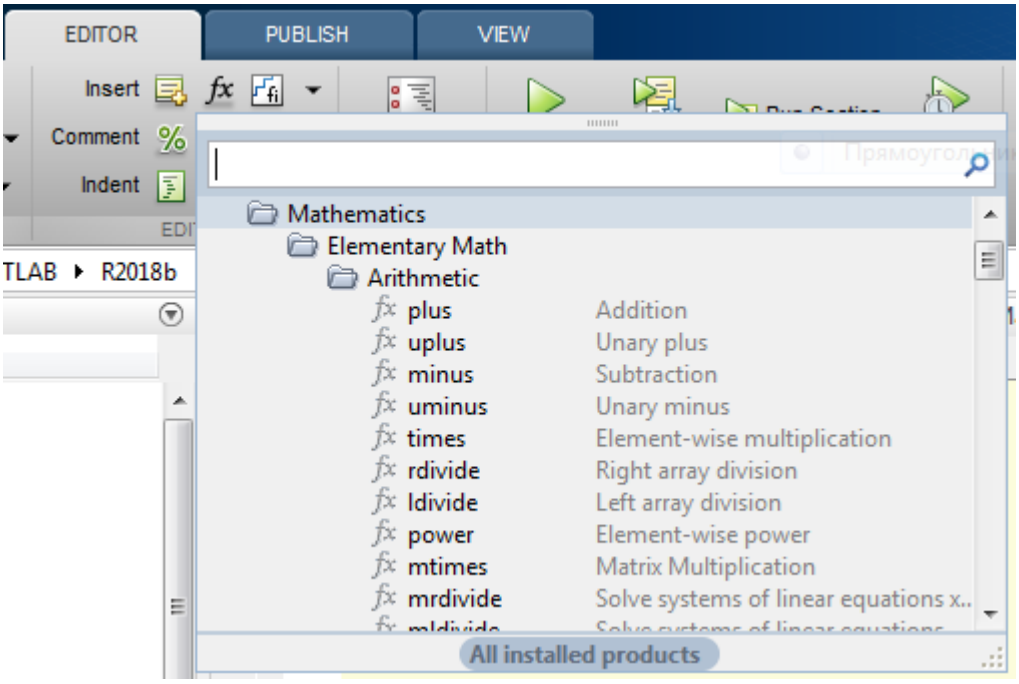
Равно	==; eq (eq(a, b))
Не равно	~=; ne (ne(a, b))
Больше	>; gt (gt(a, b))
Больше или равно	>=; ge (ge(a, b))
Меньше	<; lt (lt(a, b))
Меньше или равно	<=; le (le(a, b))

Для арифметических операторов есть эквивалентные функции:

Сложение	+	plus (plus(A,B))
Унарный плюс	+	uplus (uplus(A))
Вычитание	-	minus (minus(A,B))
Унарный минус	-	uminus (uminus(A))
Поэлементное умножение	.*	times (times(A,B))
Умножение матриц	*	mtimes (mtimes(A,B))
Поэлементное правое деление	./	rdivide (rdivide(A,B))
Матричное правое деление	/	mrdivide (mrdivide(A,B))
Поэлементное левое деление	.\	ldivide (ldivide(A,B))
Матричное левое деление	\	mldivide (mldivide(A,B))
Поэлементная степень	.^	power (power(A,B))
Матричная степень	^	mpower (mpower(A,B))
Транспонирование	.’	transpose (transpose(A))
Комплексное сопряженное транспонирование	’	ctranspose (ctranspose(A))

Дополнительную информацию по каждому оператору можно посмотреть с помощью команды «help ops»

Любой оператор можно добавить в текст программы нажатием кнопки «fx» во вкладке EDITOR. Или в командной строке также кнопкой «fx».



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

Основные форматы чисел можно посмотреть с помощью ввода команды «help format»:

```
>> 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 SHORTE   Floating point format with 5 digits.
format LONGE    Floating point format with 15 digits for double and
                7 digits for single.
format SHORTG   Best of fixed or floating point format with 5
                digits.
format LONGG    Best of fixed or floating point format with 15
                digits for double and 7 digits for single.
format SHORTENG Engineering format that has at least 5 digits
                and a power that is a multiple of three
format LONGENG  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 = 2; b = 5;              % символ ; подавляет эхо-вывод
>> c = (a + b) * 2
c =
   14
>> p                          % встроенная константа
ans =
   3.1416
>> I                          % комплексные числа, i - встроенная константа
ans =
  0.0000 + 1.0000i
```

Элементарные функции Matlab

Matlab имеет встроенные элементарные математические функции. Информацию о них можно узнать с помощью команды «**help elfun**».

Модуль	abs(x)
Экспонента	exp(x)
Натуральный логарифм	log(x)
Логарифм по основанию 2	log2(x)
Десятичный логарифм	log10(x)
2 в степени x	pow(x)
Квадратный корень	sqrt(x)

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

Так же есть специальные функции, сгруппированные 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 может осуществляться двумя способами:

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

Операции с матрицами:

Сложение матриц

```
>> a = [1 2; 3 4];
>> 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
```

Задание одного элемента матрицы

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

Транспонирование

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

Присоединение столбца

```
>> a = [1 2; 3 4]; b = [5; 6];
>> s1 = [a, b]
s1 =
    1    2    5
    3    4    6
```

Присоединение строки

```
>> a = [1 2; 3 4]; c = [5 6];
>> s2 = [a;c]
s2 =
    1    2
    3    4
    5    6
```

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 входной 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**).

Примеры использования оператора (**:**):

Использование оператора (:)

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

```
>> a(2,:)                         % операция со строкой
ans =
   5    6    7    8
```

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

```
>> 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))
p =
  7.8784e+03
```

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
```

Примечание: 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:

```
 >> a = [1; 2; 3; 4];
>> b = [5; 6; 7; 8];
>> c = a+b
c =
    6
    8
   10
   12
>> d = b-a
d =
    4
    4
    4
    4
```

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 = [1 2; 3 4];
>> b = [5 6; 7 8];
>> 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.

Полезные команды и функции

Полезные команды и функции

>> **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

>> **who** – просмотр переменных рабочего пространства в процессе решения задач

Работа с пакетом Control

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 переменной 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:

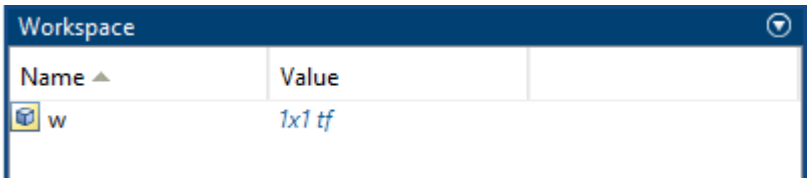
tf	Создание и преобразование в передаточные функции
zpk	Создание и преобразование в нули/полюсы/коэффициент усиления
ss	Создание и преобразование в пространство состояния

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:

```
>> w = tf([3 1], [1 1 1])
w =
   3 s + 1
 -----------
 s^2 + s + 1

Continuous-time transfer function.
```

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

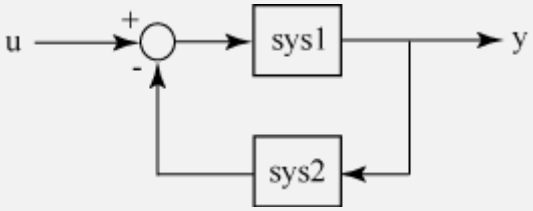


Для преобразования моделей предназначены специальные функции:

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

Замкнутая система с обратной связью

```
>> SYS = feedback(SYS1, SYS2);          % с отрицательной обратной связью
>> SYS = feedback(SYS1, SYS2, +1);      % с положительной обратной связью
```



Преобразование в ZPK форму

```
>> W1 = tf([3 1], [1 1 1])        % формирование передаточной функции W1
W1 =
  3 s + 1
 -----------
 s^2 + s + 1
>> W = feedback(W1,1)             % формирование замкнутой системы с обратной связью
W =
   3 s + 1
 -------------
 s^2 + 4 s + 2
>> Wz = zpk(W)                    % преобразование передаточной функции в zpk форму
Wz =
    3 (s+0.3333)
 --------------------
 (s+3.414) (s+0.5858)
```

С помощью данного преобразования можно представить передаточные функции в факторизованной форме. С помощью данного преобразования можно сразу увидеть корни числителя и знаменателя передаточной функции, при этом полином второго порядка указывает наличие комплексно-сопряженных корней.

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

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

```
>> tfdata(W)
ans =
  1×1 cell array
    {1×3 double}
```

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

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

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

```
>> [n,d] = tfdata(W, 'v')
n =                          % коэффициенты числителя ПФ
    0    3    1
d =                          % коэффициенты знаменателя ПФ
    1    4    2
```

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

```
>> roots(n)                  % корни числителя ПФ
ans =
   -0.3333
```

```
>> roots(d)                % корни знаменателя ПФ
ans =
  -3.4142
  -0.5858
```

С помощью функций **zero** и **pole** можно найти нули и полюса передаточной функции (нули передаточной функции — это корни числителя, полюсы — корни знаменателя):

```
>> zero(W)                 % нахождение нулей
ans =
  -0.3333

>> pole(W)                 % нахождение полюсов
ans =
  -3.4142
  -0.5858
```

Функция **damp** отображает коэффициент демпфирования, собственную частоту и постоянную времени полюсов линейной модели:

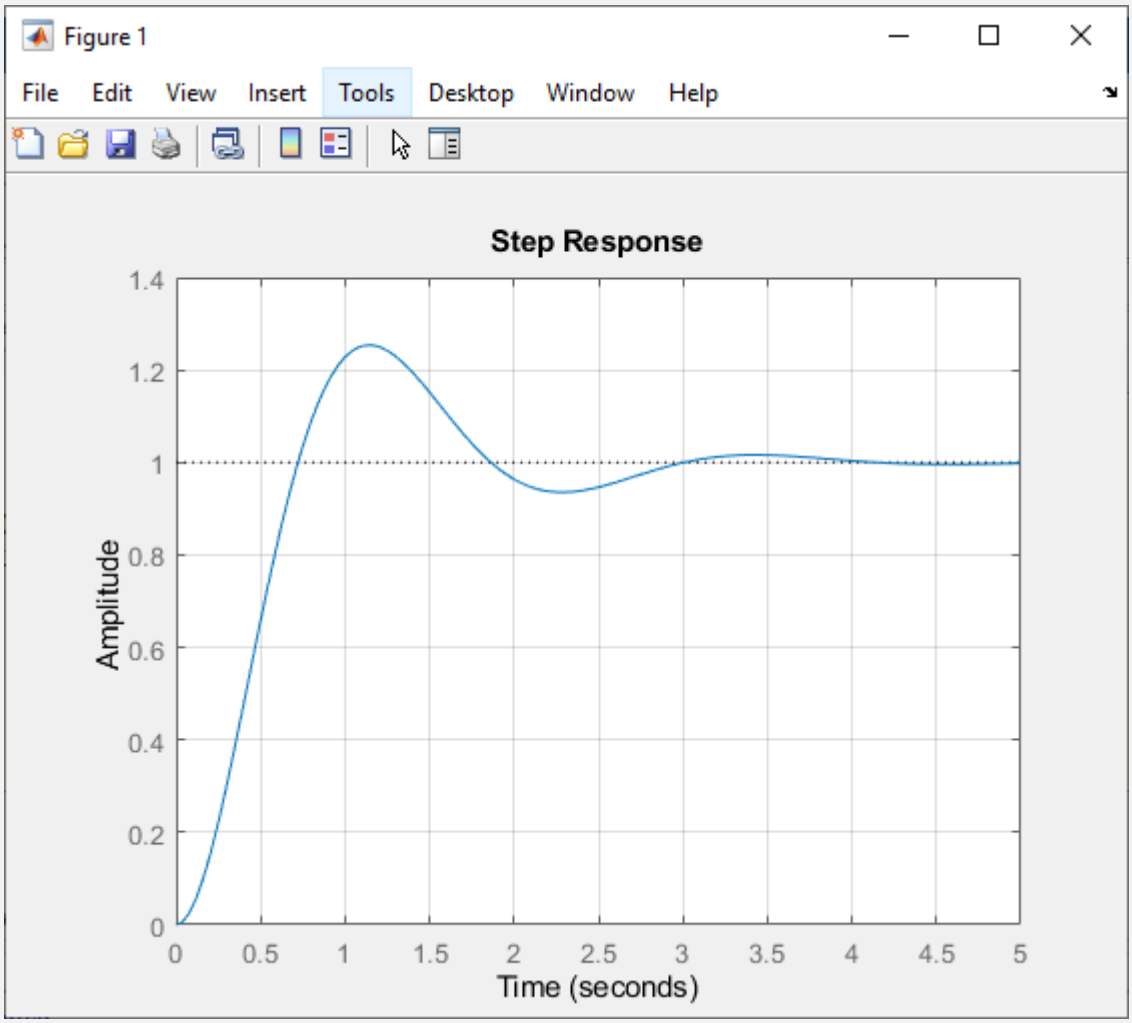
```
>> damp(W)

  Pole              Damping         Frequency       Time Constant
                                    (rad/seconds)   (seconds)

 -5.86e-01         1.00e+00        5.86e-01        1.71e+00
 -3.41e+00         1.00e+00        3.41e+00        2.93e-01
```

Функции анализа свойств модели:

step	Реакция на единичную функцию (переходная функция)
impulse	Реакция на дельта-функцию (функция веса)
bode	Логарифмические амплитудно-частотные и фазочастотные характеристики (диаграмма Боде)
margin	Отображает диаграмму Боде на экране и указывает на графике запасы по амплитуде и фазе
rlocus	График корневого годографа динамической системы
pzmap	Строит нули и полюсы на графике
nyquist	Годограф Найквиста

Построение графиков и анализ свойств системы

```
>> SYS = tf([9],[1 2.4 9])
SYS =
        9
  ---------------
  s^2 + 2.4 s + 9

>> step(SYS)                    % график переходного процесса
>> grid
```

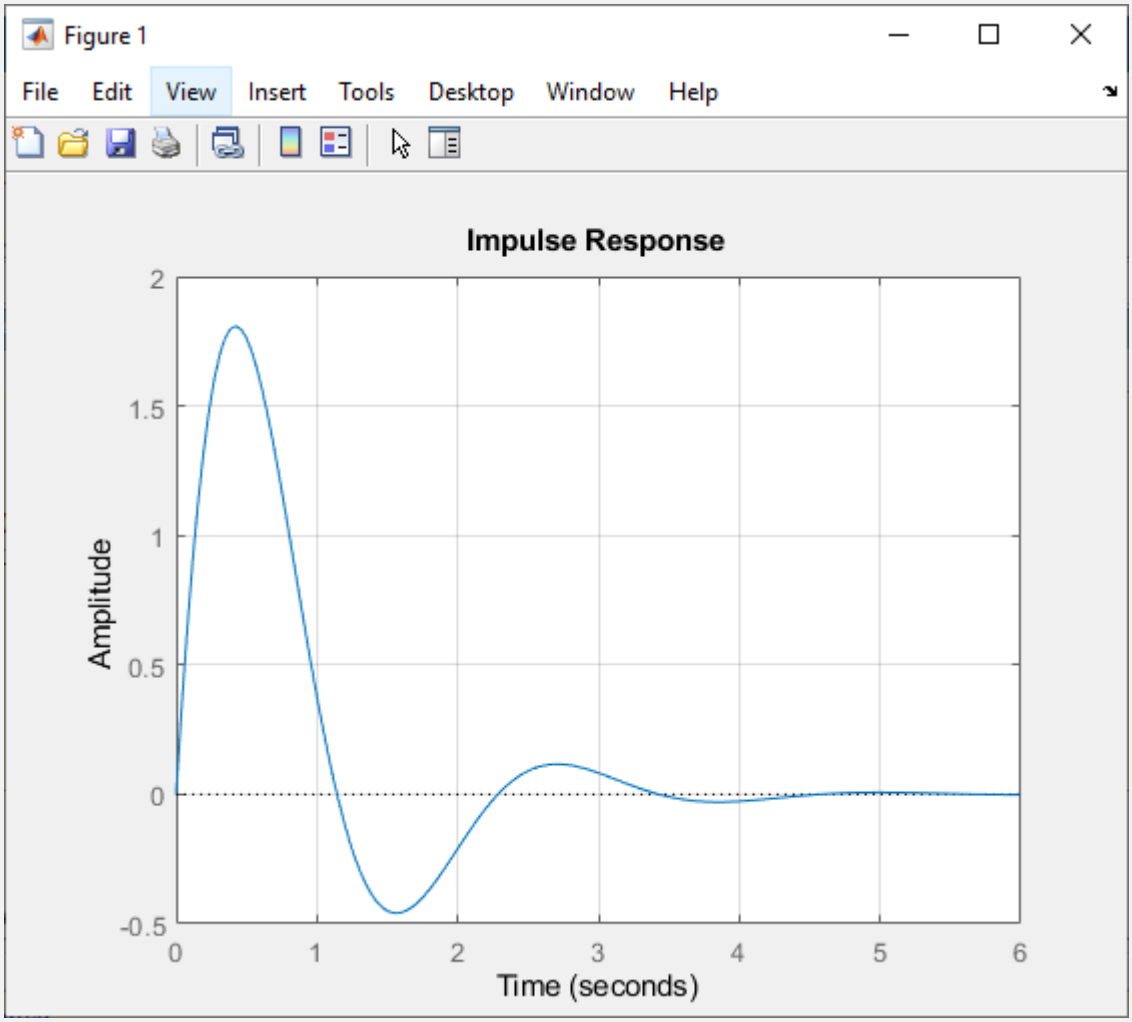



```
>> stepinfo(SYS)                % характерные точки графика
ans =
       RiseTime: 0.4884
   SettlingTime: 2.8031
    SettlingMin: 0.9065
    SettlingMax: 1.2537
      Overshoot: 25.3741
     Undershoot: 0
           Peak: 1.2537
       PeakTime: 1.1513
```

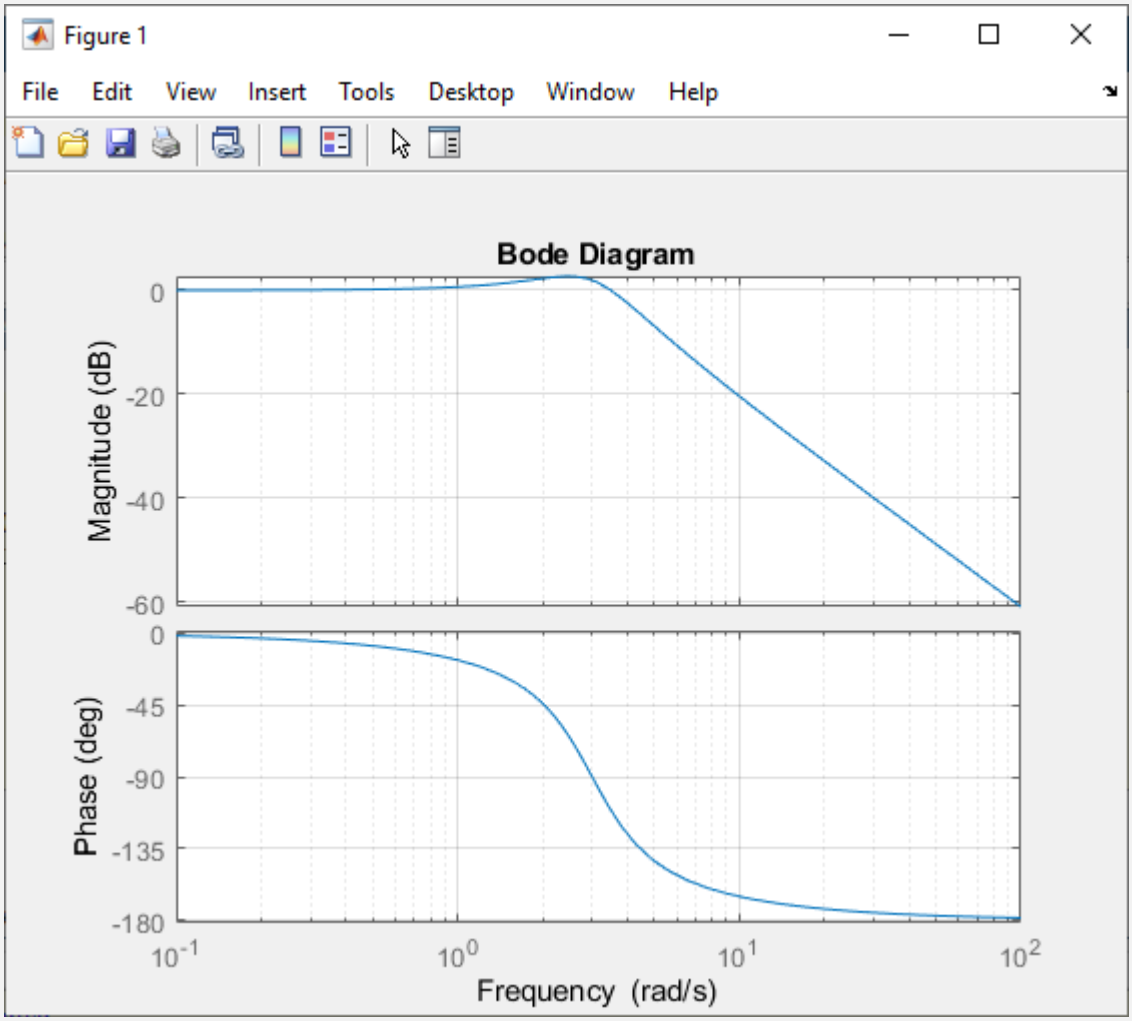
Отклик во времени – численные значения по шагам интегрирования – для вывода таблицы необходимо задать матрицу выходных параметров **y** и **t**:

```
>> [y t] = step(SYS);
>> [y t]
ans =
      0      0
 0.0064 0.0384
 0.0248 0.0768
 0.0540 0.1151
 0.0925 0.1535
 0.1391 0.1919
 . . . . . .
```

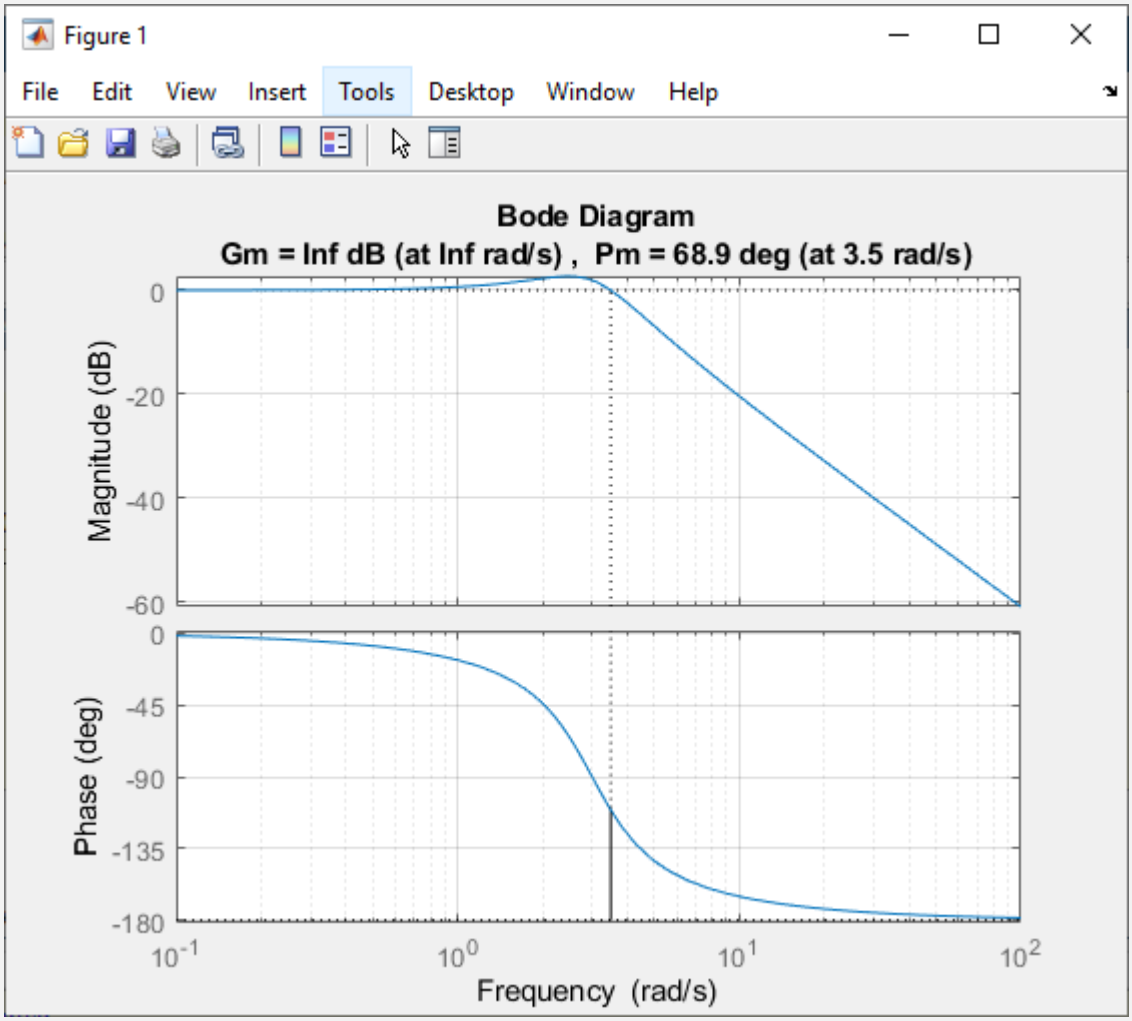
```
>> impulse(SYS)
>> grid
```



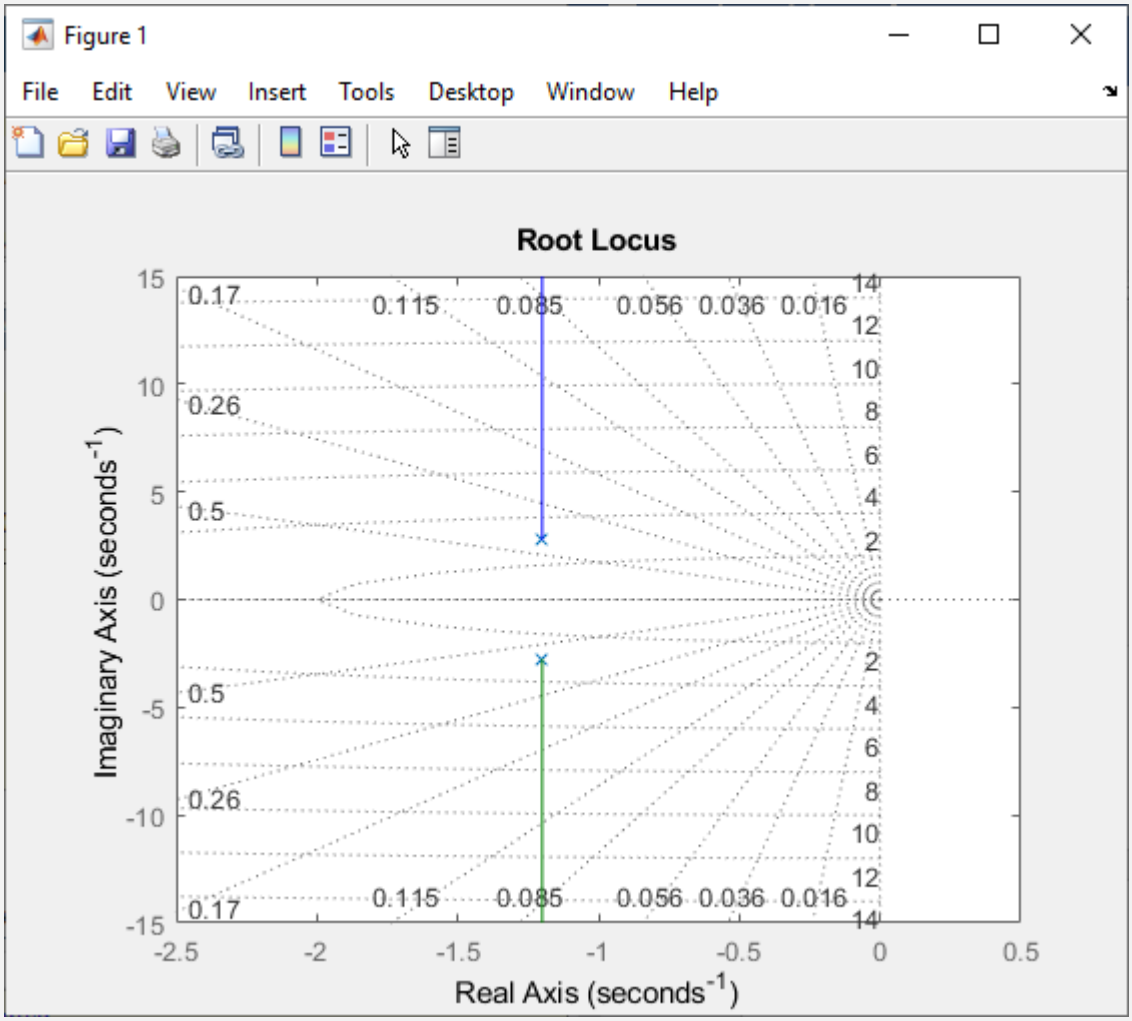
```
>> bode(SYS)
>> grid
```



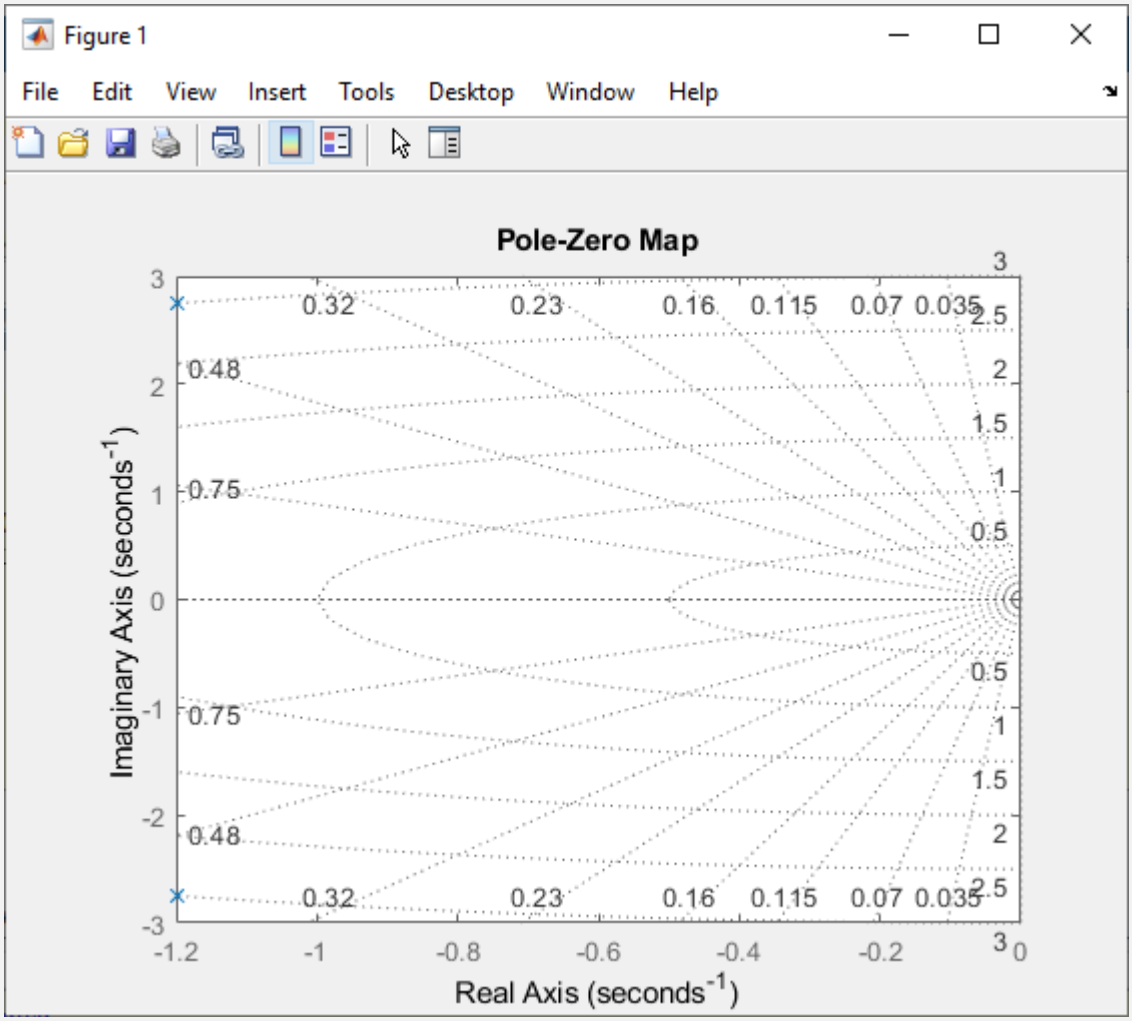
>> margin(SYS)
>> grid



```
>> rlocus(SYS)
>> grid
```



>> pzmap(SYS)
>> grid

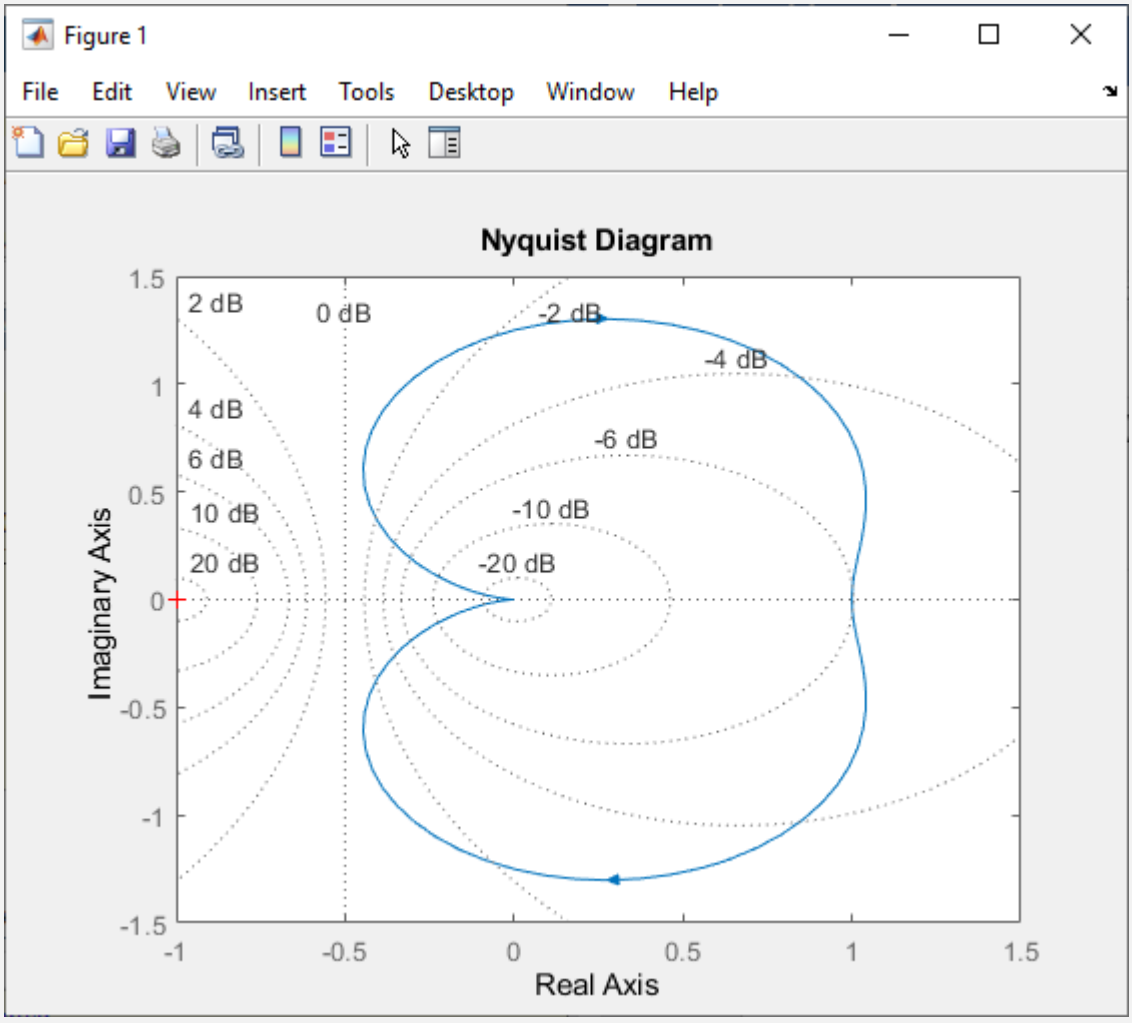


Получение численных значений полюсов и нулей:

>> [np,dz] =pzmap(SYS)
np =
-1.2000 + 2.7495i
-1.2000 - 2.7495i

dz =
0×1 empty double column vector

```
>> nyquist(SYS)
>> grid
```



Работа с LTIviewer

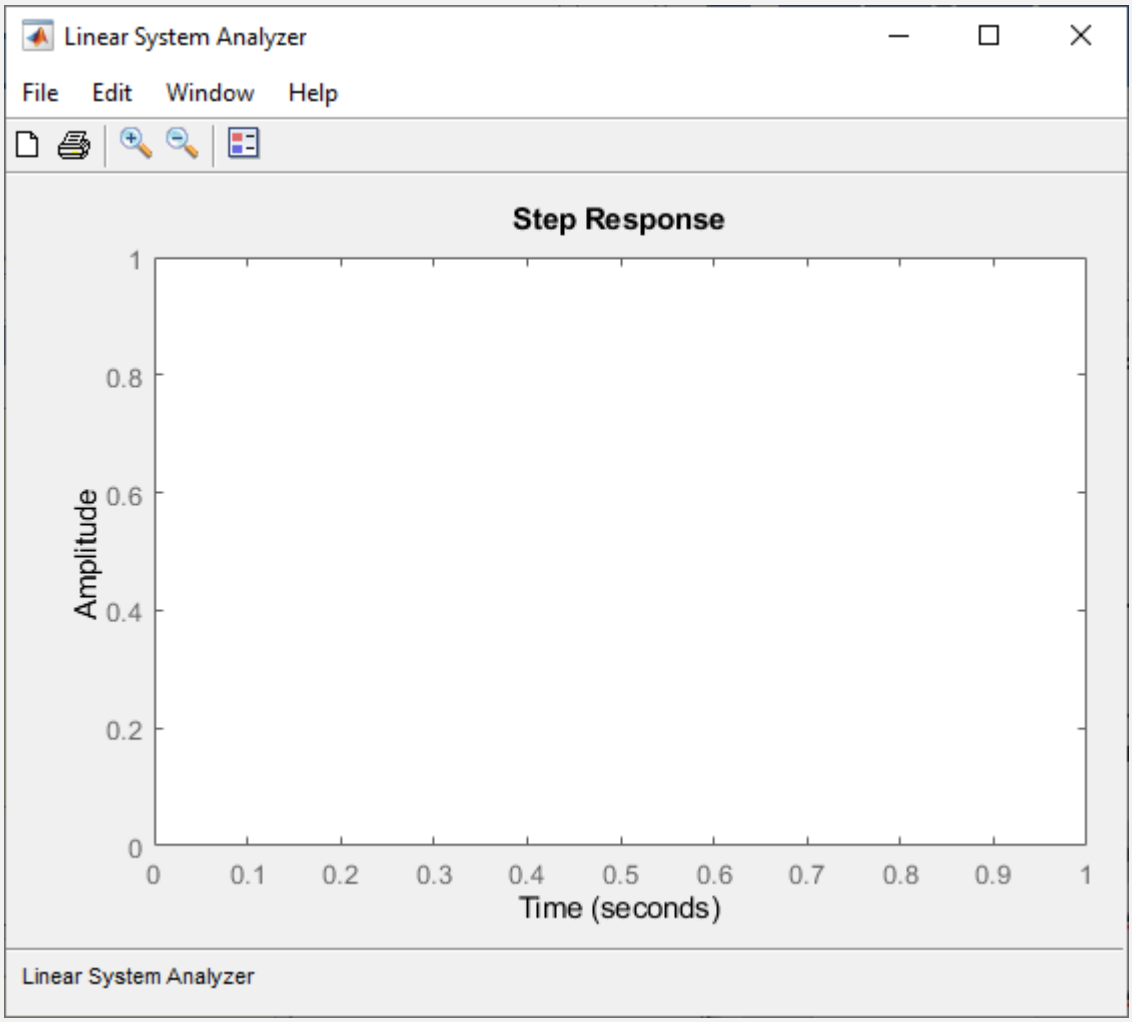
LTIviewer – интерактивный обозреватель свойств линейных моделей интерактивное приложение **LinearSystemAnalyser**.

Зададим две LTI модели передаточными функциями

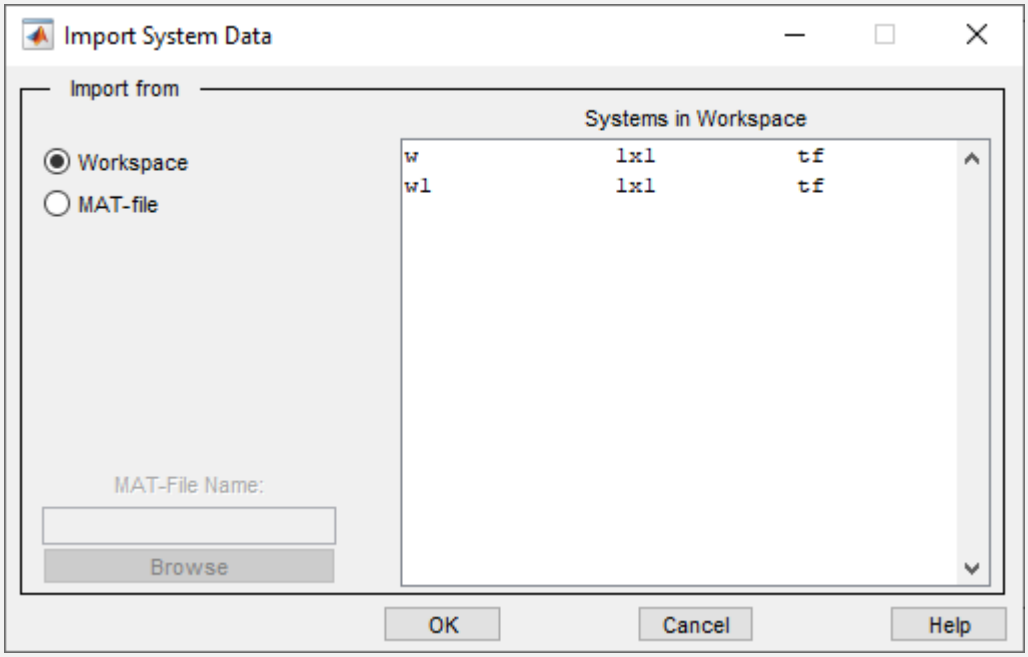
```
>> w=tf([1],[1 1])
w =
   1
 -----
 s + 1

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

Вызываем LTIviewer командой

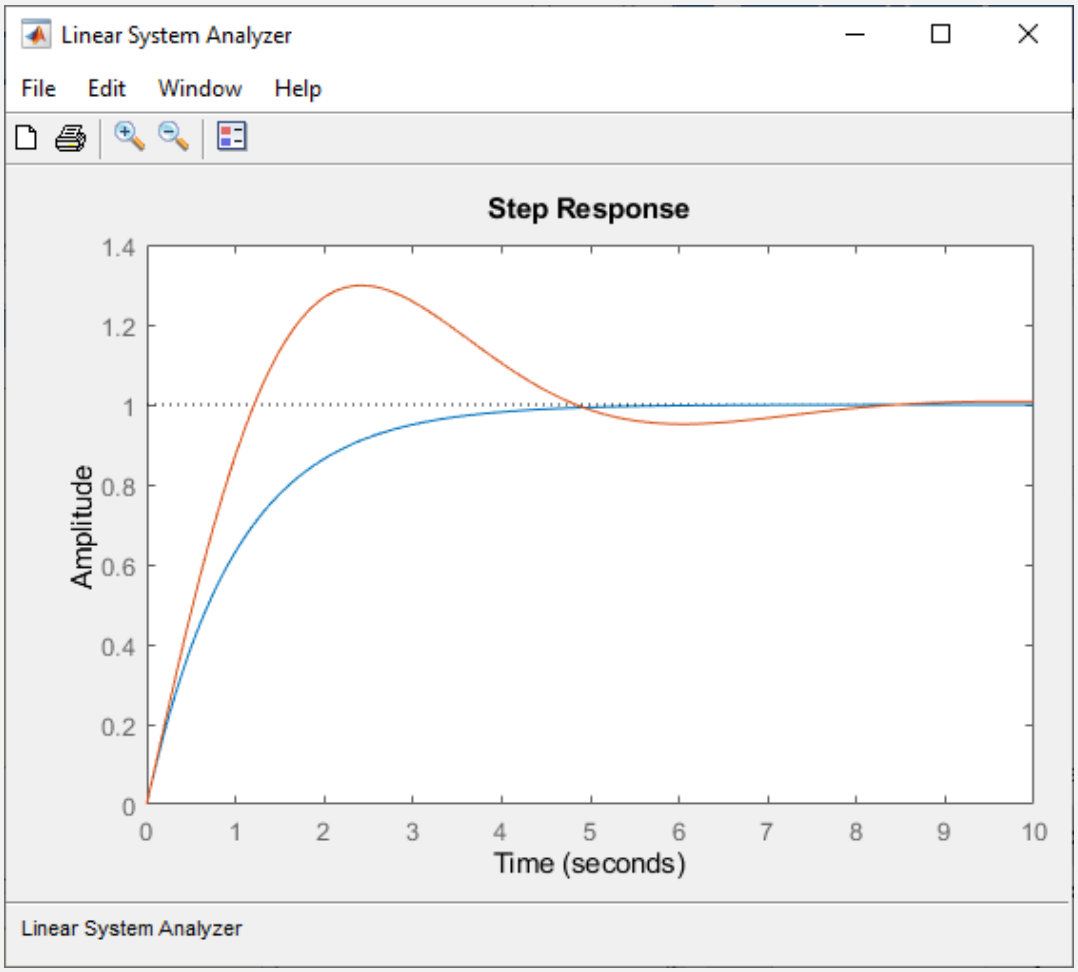
```
>> ltiview
```



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

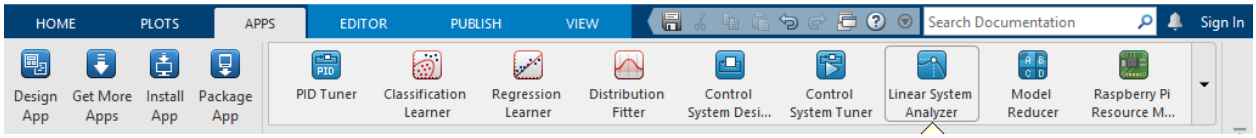


Можно сразу задать идентификаторы созданных LTI – систем во входных параметрах:
>> ltiview(w,w1)



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**.

Другой вариант вызова приложения – через меню вкладки **APPS** (**Applications**)



Изначально это меню находится в свернутом состоянии и представлено в виде одной полосы, в которой расположены пиктограммы наиболее часто используемых приложений – **Favorites** (**Избранное**). Если в используемой версии **MATLAB** приложение **Linear System Analyser** не отмечено как **Favorite** и пиктограммы нет в верхней полосе меню, то следует развернуть полный список загруженных приложений.

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.

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

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

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**);

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

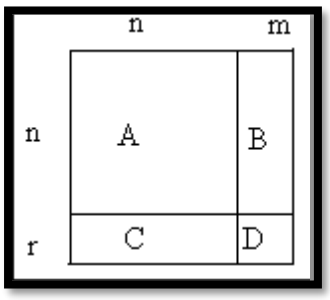
D – матрица прямой связи;

n – количество переменных состояния;

m – число входных сигналов (сигналы управления);

r – число выходных сигналов.

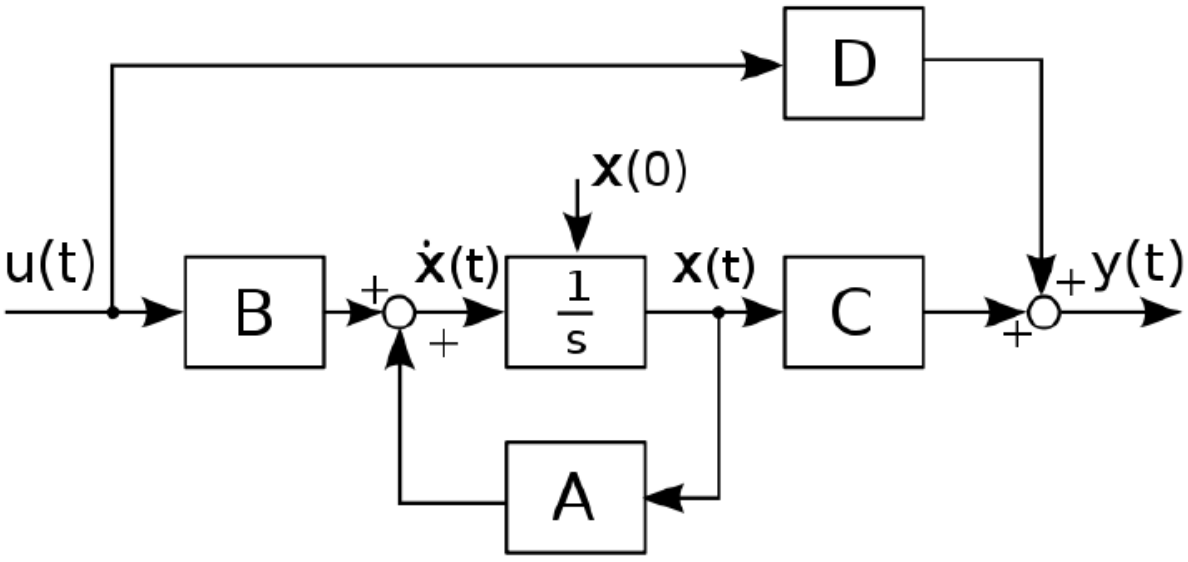
Соотношение размерности матриц показано на рисунке:



Матрица **D** необходима в том случае, когда степень числителя передаточной функции равна степени знаменателя.

В физически реализуемых системах порядок числителя передаточной функции не может превышать порядка её знаменателя (правильная и строго правильная передаточная функция) и в случае, когда порядок числителя меньше порядка знаменателя (строго правильная передаточная функция) матрица **D** является нулевой, это означает, что в системе нет явной прямой связи от входа к выходу.

Модель, соответствующая неправильной передаточной функции (у которой степень числителя выше степени знаменателя), не может быть представлена в стандартном пространстве состояний.



Вычислительная схема модели динамической системы в пространстве состояний, блок интегрирования на схеме означает покомпонентное интегрирование вектора $\dot{\boldsymbol{X}}(\boldsymbol{t})$.

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

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

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

Система задана двумя уравнениями, имеет два выходных параметра: угловая скорость тангажа $\boldsymbol{\omega_z}$ и угол атаки $\boldsymbol{\alpha}$.

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

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

X – вектор состояния	$X = \begin{bmatrix} \omega_z \\ \alpha \end{bmatrix}$
A – матрица коэффициентов	$A = \begin{bmatrix} (-c_1 - c_5) & (c_4 c_5 - c_2) \\ 1 & -c_4 \end{bmatrix}$
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)

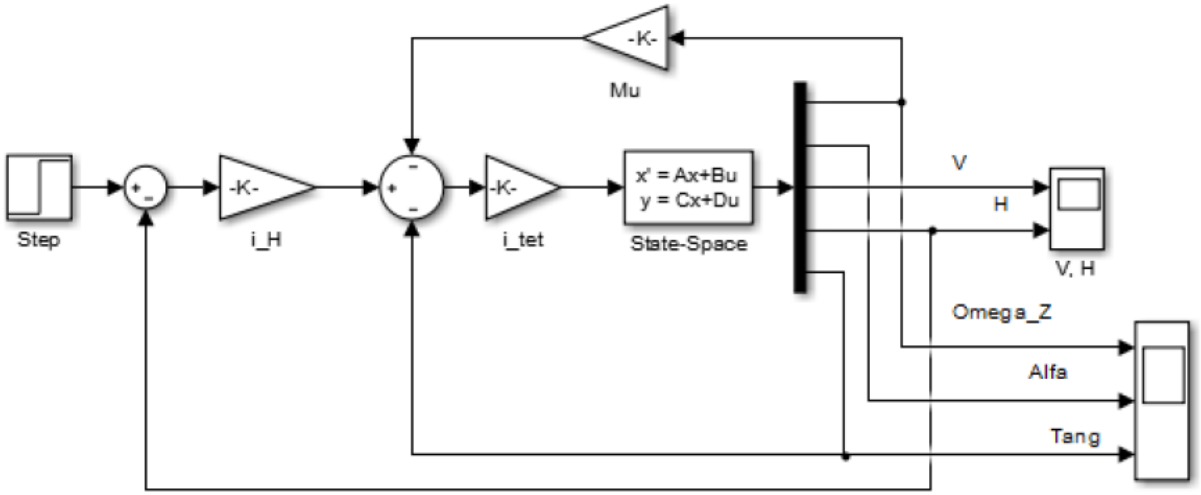
Программа – скрипт:

```
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
```

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**, который 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.

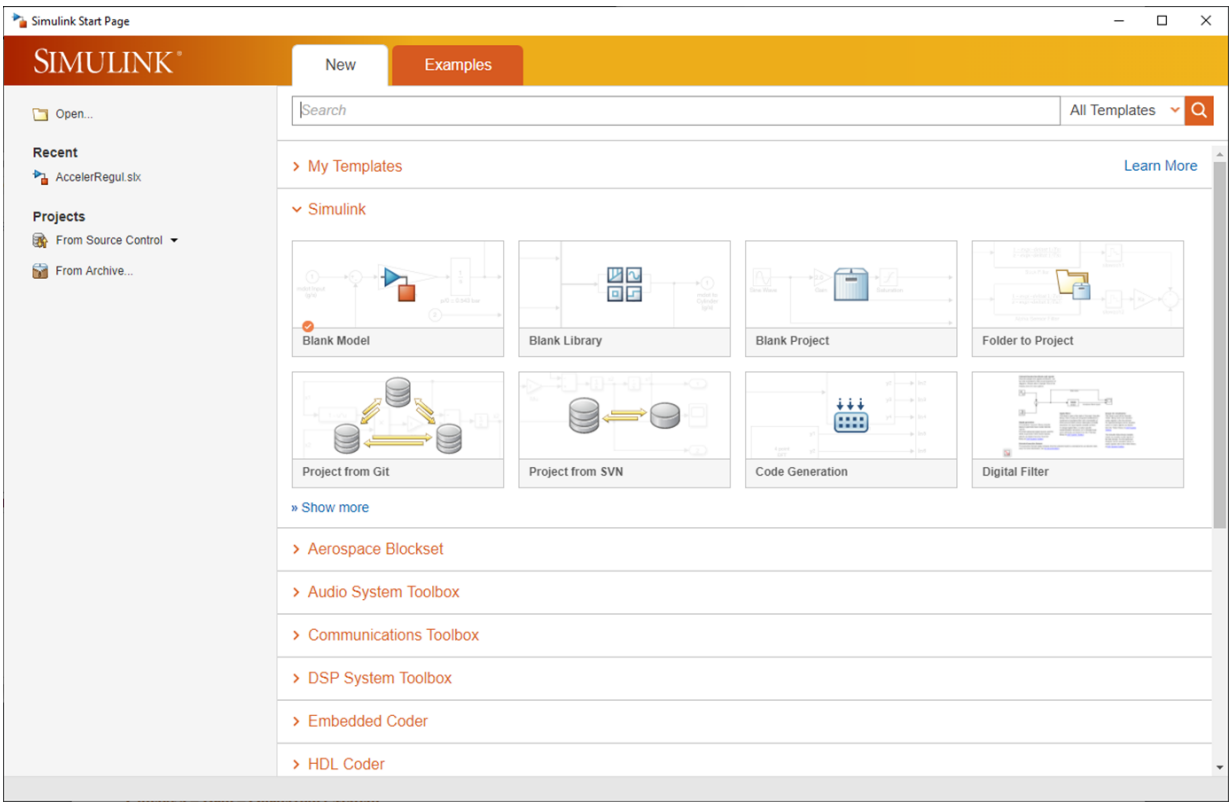
SIMULINK

Simulink – среда визуального моделирования.

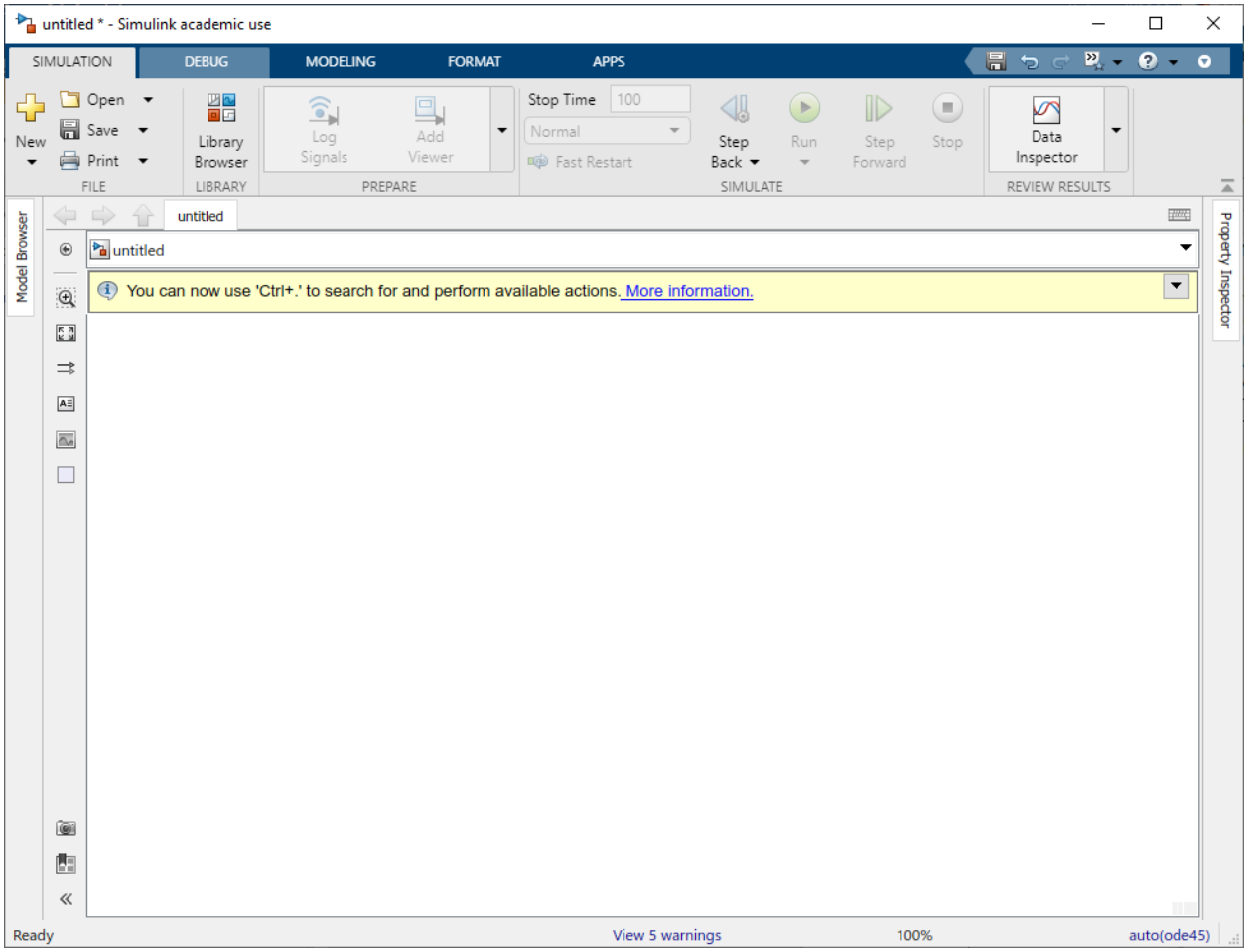
Интерфейс Simulink

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

Вкладка **New** стартового окна:



После создания нового проекта открывается основное окно **Simulink**.



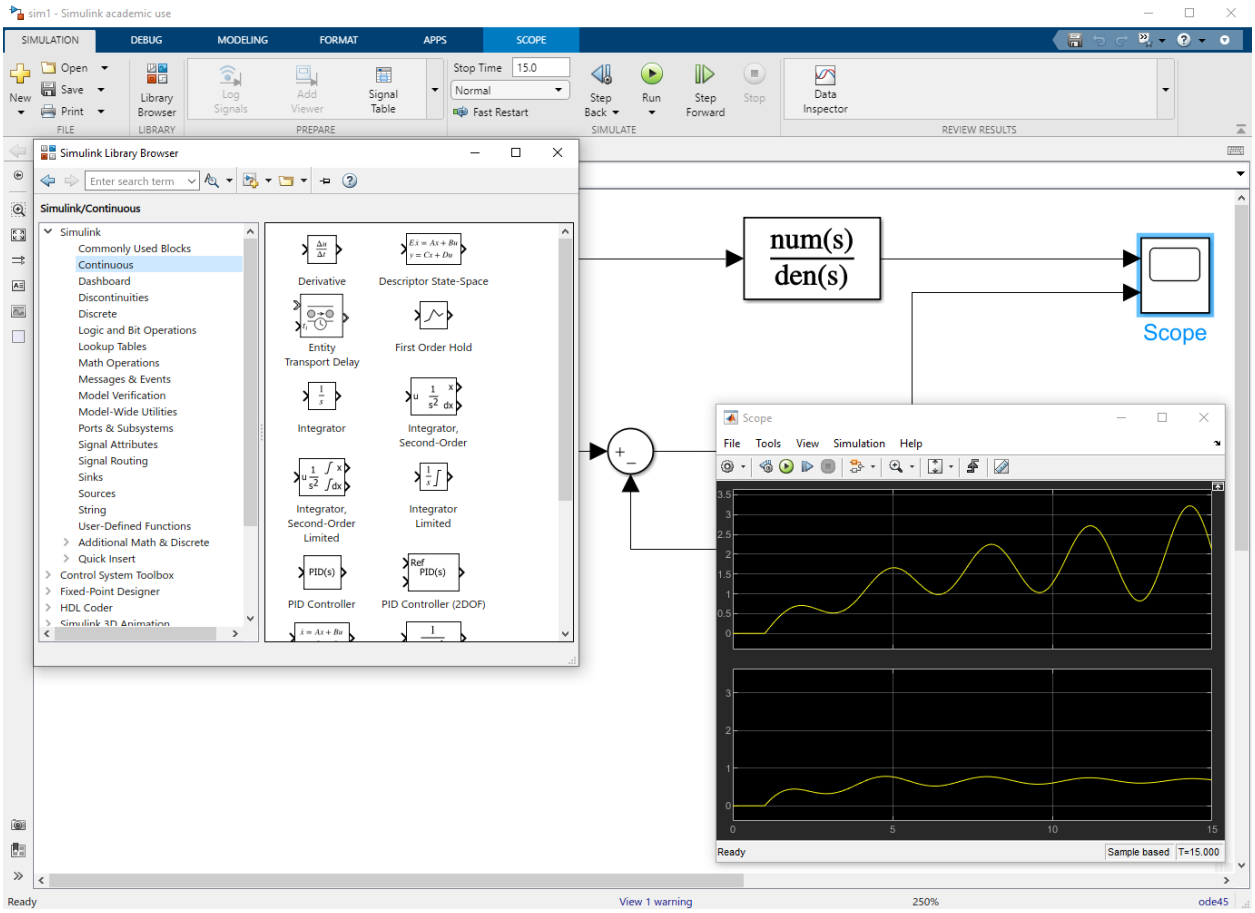
Алгоритм работы

1. Создание файла модели и вызов библиотеки **Simulink**;
2. Выборка блоков в разделах библиотеки и расстановка блоков на схеме модели (используются различные разделы библиотеки и наборы **Blockset**);
3. Соединение блоков, модификация вида и ориентации (набор простой модели осуществляется на одном наборном поле для сложной модели формируем основную модель и подсистемы (**Subsystem**));
4. 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);
5. Установить параметры расчета модели (параметры блоков; переменные, заданные или рассчитанные программно; данные из сохраненного файла datname.mat);

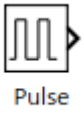
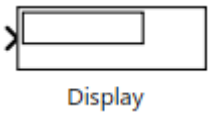
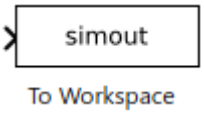
6. Настроить Решатель (время процесса, метод решения и параметры шага, точности);
7. Организовать вывод результатов (в графические окна, в рабочую среду **Matlab**, в файл);
8. Проверка результатов, выполнение расчетов с изменением параметров.

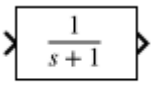
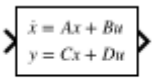
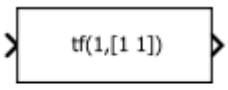
Основы работы с системой Simulink

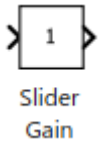
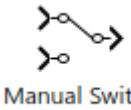
SIMULINK является программной системой интерактивного моделирования и позволяет использовать вычислительные возможности **MATLAB** для решения широкого круга задач моделирования и анализа систем.



Модель состоит из блоков и связей, или линий передачи сигналов. Блоки сгруппированы по назначению, группы отображаются в виде иерархической структуры в окне библиотеки.

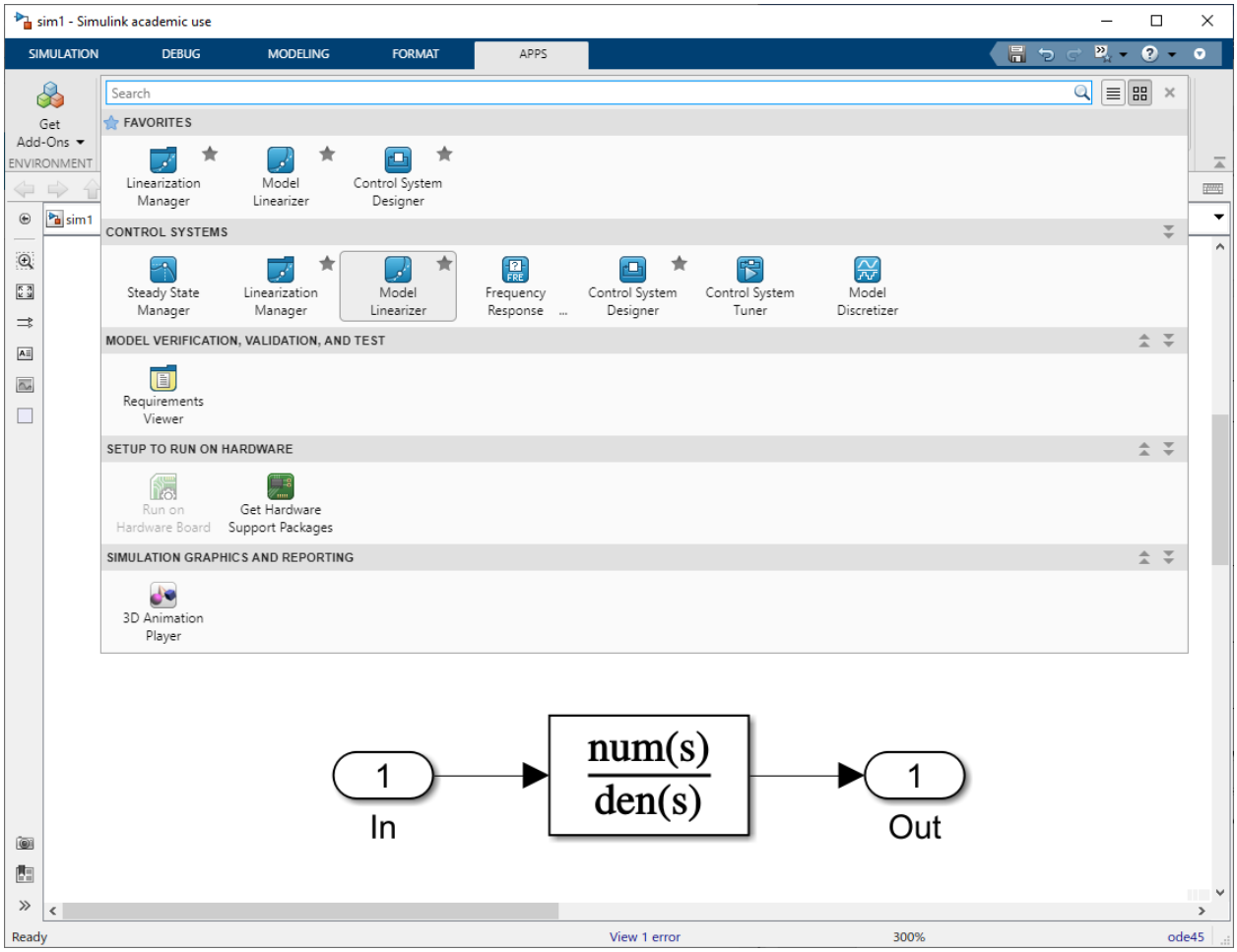
Основные блоки SIMULINK		
Входные величины – источники сигналов (Sources)		
	Step	Регулируемый по высоте скачок («**Final Value**») и определяемое начало времени скачка («**Step Time**»)
	Pulse Generator	Генератор прямоугольных импульсов (меандр), максимальное значение и частота которых может устанавливаться
	Constant	Устанавливаемая константа, не изменяющаяся во время моделирования
	Signal Generator	Генератор для задания различных входных сигналов в виде функций таких, например, как функция синуса, пилообразная или меандр с устанавливаемыми значениями амплитуды и частоты
	Clock	Генерирует на каждом шаге сигнал, значение которого равно текущему времени моделирования
	Ground	Позволяет избегать выдачи ошибочных сообщений, если неиспользуемые входы не объединены с другими блоками
Приемники выходных величин (Sinks)		
	Scope	Графическая выдача величин на монитор, похожий на осциллограф
	Display	Числовая выдача величин
	To Workspace	Выдаваемые величины могут быть записаны в выбранные переменные, которые можно обрабатывать, используя рабочую поверхность **MATLAB**, но эти переменные нельзя сохранять. Для считывания переменных можно использовать блок «**From Workspase**» библиотеки «**Sources**»

 To File	To File	Выходная величина может быть сохранена в виде вектора-строки в **.mat** файле
 Terminator	Terminator	Похож на «**Ground**»-блок, применяется для исключения ошибочных сообщений, если выход блока не соединен с другим блоком
Непрерывные функциональные блоки (Continuous)		
 Derivative	Derivative	Дифференцирующее звено (дифференциатор), но без параметров, поэтому коэффициент усиления должен быть дополнительно предусмотрен с помощью отдельного блока
 Integrator	Integrator	Интегрирующее звено (интегратор), у которого постоянная времени тоже должна быть дополнительно предусмотрена. Можно определить начальные значения выходной величины («**Initial condition**»), а также её ограничения
 Transfer Fcn	Transfer Function	Передаточная функция, числитель которой («**Numerator**») и знаменатель («**Denumerator**») могут быть заданы в виде нисходящей последовательности коэффициентов как векторы-строки. Возможно также использование уже заданных на рабочей поверхности **MATLAB** переменных для определения числителя и знаменателя
 State-Space	State-Space	Модель «Вход – Выход» в пространстве состояний формируется на основе системы дифференциальных уравнений первого порядка, разрешенных относительно производных, которую принято называть нормальной формой Коши
 PID Controller	PID Controller	Блок ПИД-регулятора
Control System Toolbox		
 LTI System	LTI Systems	Универсально применяемый блок для задания передаточных функций или систем. Вместо передаточной функции в известной форме, когда числитель и знаменатель задаются как векторы-строки коэффициентов, можно задавать имена передаточных функций, определённых в рабочем пространстве
Математические операции (Math Operations)		

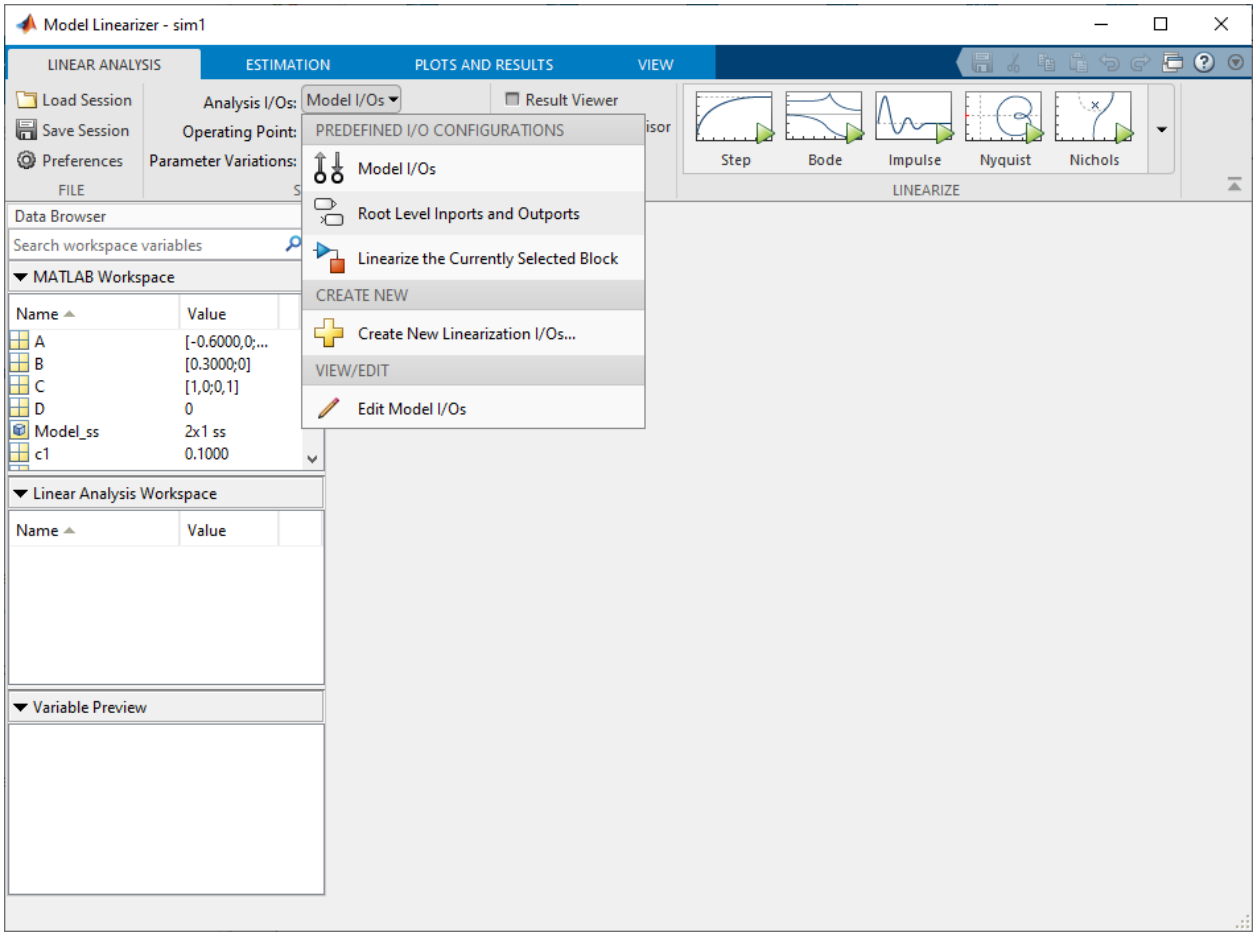
	Gain	Коэффициент усиления (множитель), применяется также как пропорциональный регулятор. Может применяться для поэлементного умножения, если в качестве сомножителя указывается вектор. Возможно применение в матричных операциях умножения
	Slider Gain	Коэффициент усиления или множитель, который можно менять во время моделирования. В отличие от «**Gain**»-блока возможно задание только скалярных величин. Часто применяют для того, чтобы увеличивая коэффициент усиления привести контур регулирования к границе устойчивости
	Sum	Сложение или вычитание различных сигналов, необходимых для реализации обратных связей в контурах регулирования или при параллельном включении различных блоков
Маршрутизация сигналов (Signal Routing)		
	Manual Switch	Переключатель между двумя возможными состояниями с помощью двойного щелчка левой кнопкой мыши. Может применяться, например, для подключения или отключения тех или иных регуляторов. Неподключенные входы должны быть заземлены (блок «Ground»)
	Multiport Switch	Пропускает на выход в зависимости от значения управляющего сигнала (самый верхний вход слева) тот или иной входной сигнал. Номер пропускающего порта при этом равен значению управляющего сигнала, значение же управляющего сигнала может округляться до целого. Нумерация входов сверху вниз. Число входов можно задавать
	Mux	Объединяет несколько сигналов в вектор
	Demux	Расщепляет сигнал-вектор на отдельные составляющие-скаляры или векторы меньшей размерности, которые поступают как общий сигнал

Анализ свойств динамической системы

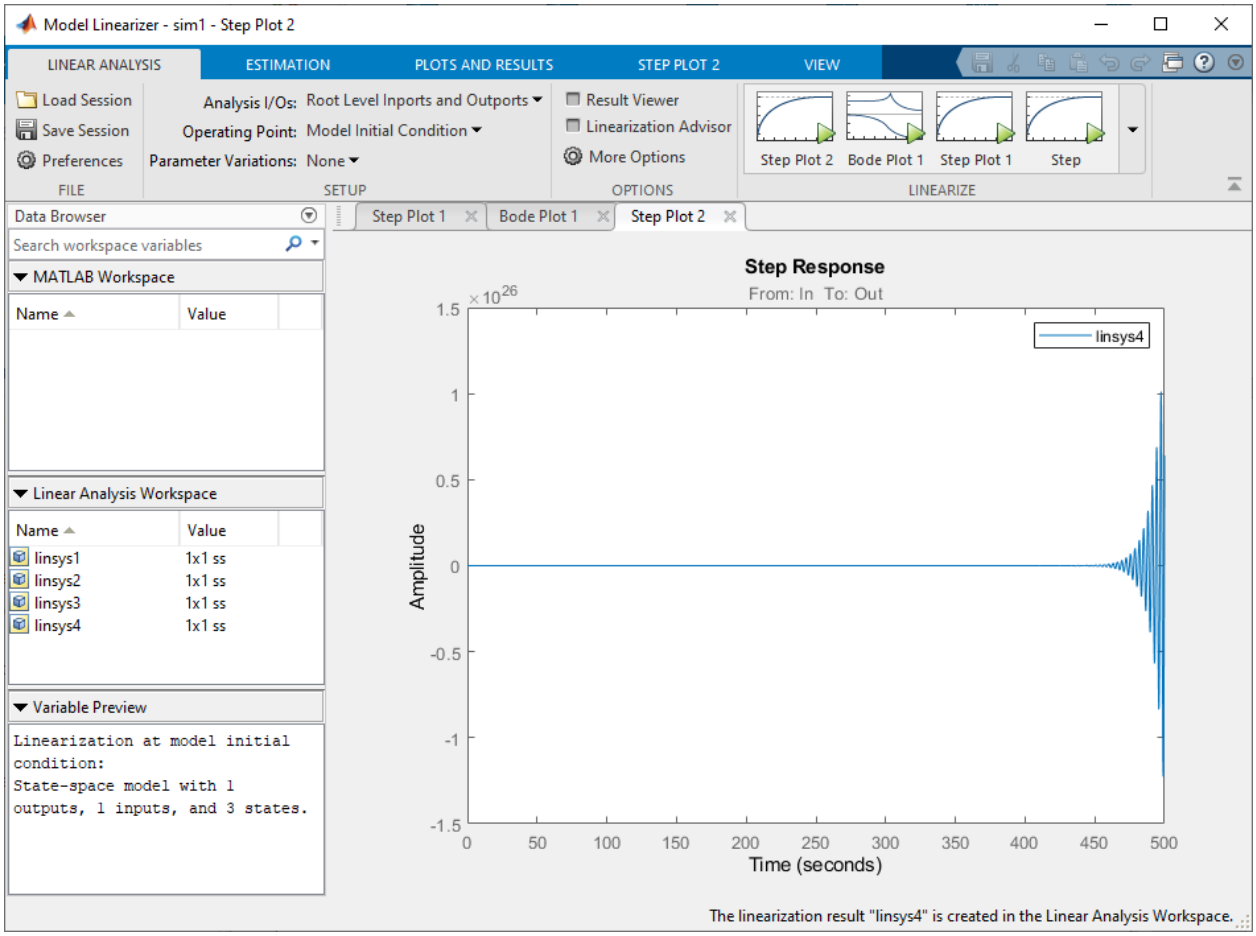
Для анализа свойств динамической системы, созданной в **Simulink** можно применить **LTI-viewer**. Для этого необходимо ввести в схему модели блоки **In** и **Out** на вход и выход, соответственно:



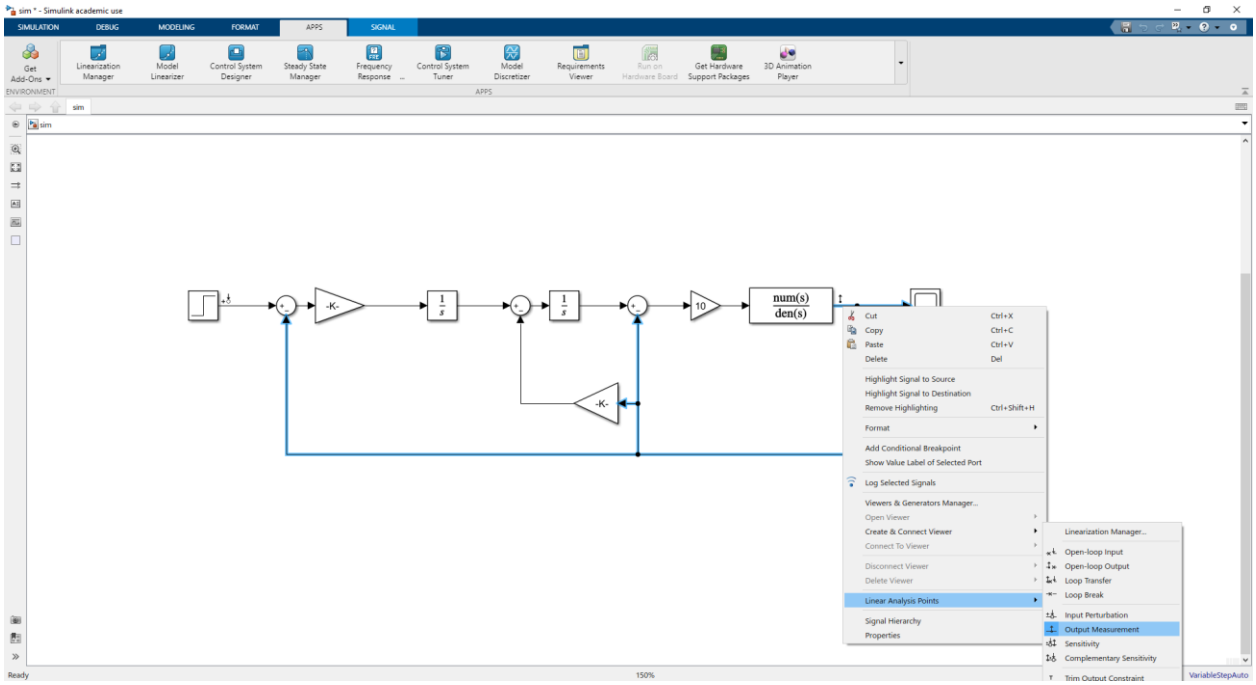
В открывшемся окне нужно задать точки входа и выхода:



Далее можно проанализировать систему и построить различные графики:

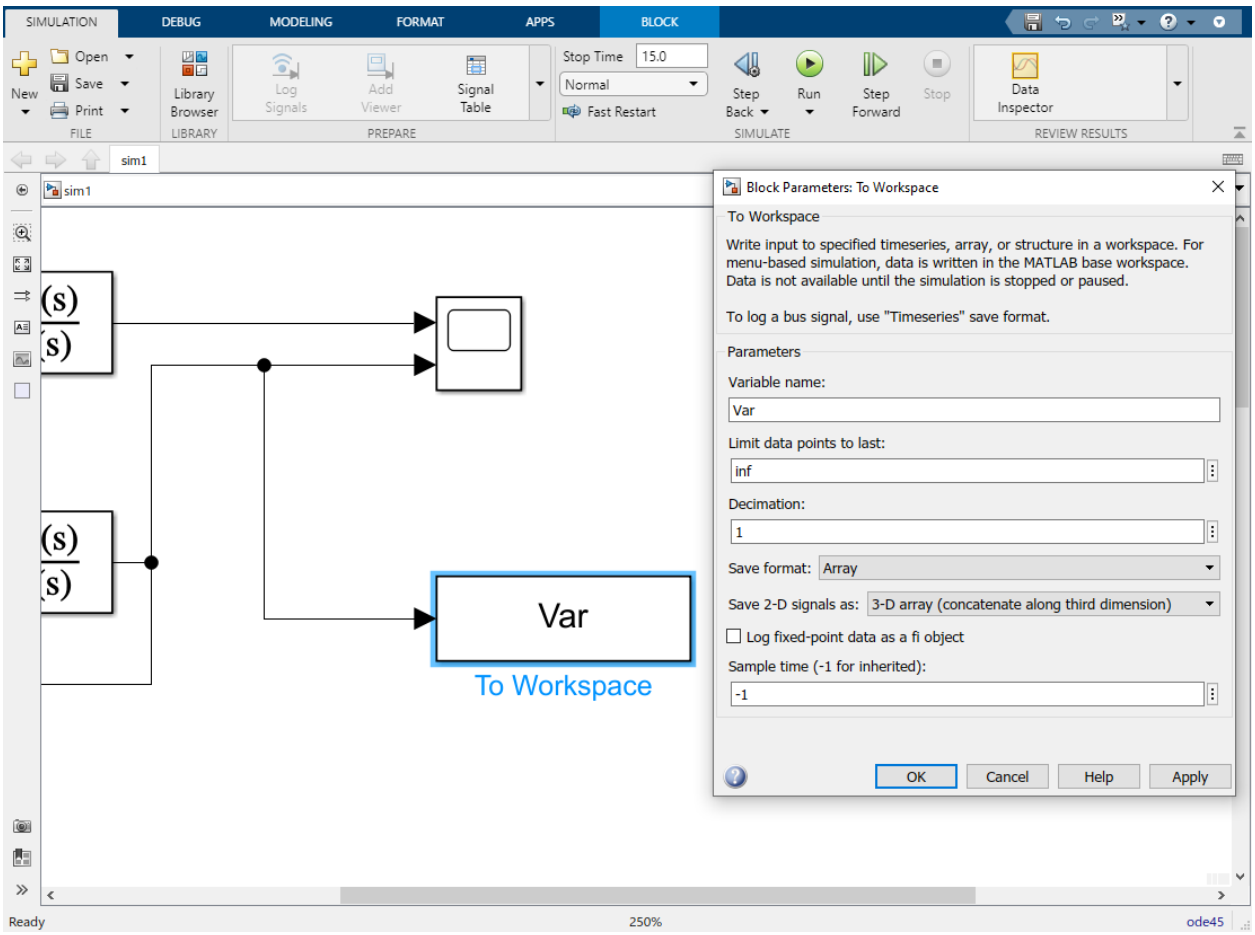


Для обозначения входа и выхода можно также указать на точку схемы (линию связи блоков) мышью и нажав правую кнопку. При этом выпадает меню, в котором следует выбрать позицию **Linearisation Points** и далее **Input** или **Output:**



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

Результаты расчетов, выполняемые в **Simulink,** не отображаются в рабочем пространстве MATLAB. Для 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.**



Теперь, при выполнении моделирования, в рабочей среде **MATLAB** появится переменная с заданным именем, также появится переменная **tout**, с дискретами времени.