Исследование ЛДС в среде MATLAB

Содержание

Основы работы в среде MATLAB

Модель ЛДС в виде передаточной функции

Передаточная функция в среде MATLAB вводится в виде отношения двух многочленов (полиномов) от комплексной переменной s. Полиномы хранятся как массивы коэффициентов, записанных по *убыванию* степеней. Например, передаточная функция

$$F(s) = \frac{2s+4}{s^3+1.5s^2+1.5s+1}$$

вводится следующим образом[1]

```
>> n = [2 4]
n =
     2     4
>> d = [1 1.5 1.5 1]
d =
    1.0000    1.5000    1.5000    1.0000
>> f = tf ( n, d )
Transfer function:
        2 s + 4
-------------------------
s^3 + 1.5 s^2 + 1.5 s + 1
```

или сразу, без предварительного построения числителя и знаменателя:

```
>> f = tf ( [2 4], [1 1.5 1.5 1] );
```

В памяти создается объект класса **tf**, описывающий передаточную функцию. Точка с запятой в конце команды подавляет вывод на экран.

По передаточной функции можно легко построить модель в форме «нули-полюса»

```
>> f_zpk = zpk(f)

Zero/pole/gain:
        2 (s+2)
----------------------
(s+1) (s^2  + 0.5s + 1)
```

Нулями называются корни числителя, полюсами – корни знаменателя. Эта функция имеет один нуль в точке $s=-2$ и три полюса в точках $s=-1$ и $s=-0{,}25\pm 0.9682i$. Паре комплексных полюсов соответствует квадратный трехчлен.

Модель ЛДС в пространстве состояний

Модель в пространстве состояний связана с записью дифференциальных уравнений в стандартной форме Коши (в виде системы уравнений первого порядка):

[1] Черным цветом обозначается ввод пользователя, синим – ответ среды MATLAB.

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

Здесь x – вектор переменных состояния размера $n \times 1$, u – вектор входных сигналов (вектор управления) размера $m \times 1$ и y – вектор выходных сигналов размера $p \times 1$. Кроме того, A, B, C и D – постоянные матрицы. Согласно правилам матричных вычислений, матрица A должна быть квадратной размера $n \times n$, матрица B имеет размер $n \times m$, матрица C – $p \times n$ и матрица D – $p \times m$. Для систем с одним входом и одним выходом[2] матрица D – скалярная величина.

Для преобразования передаточной функции в модель в пространстве состояний используется команда

```
>> f_ss = ss ( f )
a =
             x1         x2         x3
   x1      -1.5   -0.1875  -0.03125
   x2         8          0          0
   x3         0          4          0
b =
        u1
   x1  0.5
   x2    0
   x3    0
c =
         x1     x2     x3
   y1     0    0.5   0.25
d =
       u1
   y1   0
```

Это означает, что матрицы модели имеют вид

$$A = \begin{bmatrix} -1.5 & -0.1875 & -0.03125 \\ 8 & 0 & 0 \\ 0 & 4 & 0 \end{bmatrix},\ B = \begin{bmatrix} 0.5 \\ 0 \\ 0 \end{bmatrix},\ C = \begin{bmatrix} 0 & 0.5 & 0.25 \end{bmatrix},\ D = 0.$$

Модель в пространстве состояний можно построить не для всех передаточных функций, а только для *правильных*, у которых степень числителя не выше, чем степень знаменателя. Например, передаточная функция

$$W(s) = \frac{2s^2 + 3s + 1}{s + 5}$$

– неправильная, она не может быть преобразована в модель в пространстве состояний.

Используют также понятие *строго правильной функции*, у которой степень числителя *меньше*, чем степень знаменателя. Если построить модель в пространстве состояний для такой функции, матрица D будет равна нулю, то есть, прямая передача с входа на выход отсутствует (при скачкообразном изменении входа сигнал на выходе будет непрерывным).

[2] В зарубежной литературе для одномерных систем используется сокращение SISO = *Single Input Single Output*.

Действие	Команды MATLAB
1. Очистите рабочее пространство MATLAB (память).	`clear all`
2. Очистите окно MATLAB**.**	`clc`
3. Посмотрите краткую справку по команде **`tf`.**	`help tf`

4. Введите передаточную функцию $F(s)=\frac{n_2 s^2+n_1 s+n_0}{s^3+d_2 s^2+d_1 s+d_0}$ как объект **`tf`**.	`n = [n2 n1 n0]`
	`d = [1 d2 d1 d0]`
	`f = tf ( n, d )`

5. Проверьте, как извлечь из этого объекта числитель и знаменатель передаточной функции.	`[n1,d1] = tfdata ( f, 'v' )`
6. Найдите нули и полюса передаточной функции.	`z = zero ( f )` `p = pole ( f )`
7. Найдите коэффициент усиления звена в установившемся режиме.	`k = dcgain ( f )`
8. Определите полосу пропускания системы (наименьшую частоту, на которой АЧХ становится меньше, чем -3 дБ).	`b = bandwidth ( f )`
9. Постройте модель системы в пространстве состояния.	`f_ss = ss ( f )`
10. Сделайте так, чтобы коэффициент прямой передачи звена (D) был равен 1.	`f_ss.d = 1`

Работа с LTIViewer

Здесь и далее: `ПКМ` – правая кнопка мыши, `ЛКМ` – левая кнопка мыши.

Действие	Команды MATLAB
1. Запустите модуль **LTIViewer.**	`ltiview`
2. Загрузите модели `f` и `f_ss`.	LTI Viewer `File – Import`
3. Отключите/подключите системы.	LTI Viewer `ПКМ – Systems`
4. Постройте импульсную характеристику (весовую функцию).	LTI Viewer `ПКМ – Plot Types – Impulse`
5. Постройте переходные характеристики систем.	LTI Viewer `ПКМ – Plot Types – Step`

6. Сделайте, чтобы на графике для каждой функции были отмечены: • максимум • время переходного процесса[3] • время нарастания (от 10% до 90% установившегося значения) • установившееся значение	LTI Viewer ПКМ – Characteristics: • Peak Response • Settling Time • Rise Time • Steady State
7. Щелкая мышью по меткам-кружкам, выведите на экран рамки с численными значениями этих параметров и расположите их так, чтобы все числа были видны.	
8. Экспортируйте построенный график в отдельное окно.	LTI Viewer File – Print to Figure
9. Скопируйте график в буфер обмена в формате векторного метафайла.	print -dmeta

Частотная характеристика

Чтобы построить частотные характеристики в MATLAB, надо сначала создать массив частот в нужном диапазоне. Для этого можно использовать функции **linspace** (равномерное распределение точек по линейной шкале) и **logspace** (равномерное распределение точек по логарифмической шкале). Команда

```
>> w = linspace (0, 10, 100);
```

строит массив из 100 точек с равномерным шагом в интервале от 0 до 10, а команда

```
>> w = logspace (-1, 2, 100);
```

– массив из 100 точек с равномерным шагом по логарифмической шкале в интервале от 10^{-1} до 10^{2}.

Частотная характеристика на сетке **w** для линейной модели **f** (заданной как передаточная функция, модель в пространстве состояний или в форме «нули-полюса») вычисляется с помощью функции **freqresp**:

```
>> r = freqresp(f, w);
```

Функция **freqresp** возвращает трехмерный массив. Это связано с тем, что она применима и для многомерных моделей (с несколькими входами и выходами), передаточная функция которых представляет собой матрицу. Первые два индекса обозначают строку и столбец в этой матрице, а третий – номер точки частотной характеристики. Для системы с одним входом и одним выходом удобно преобразовать трехмерный массив в одномерный командой

```
>> r = r(:);
```

Для вывода графика АЧХ на экран можно использовать команды MATLAB

[3] По умолчанию в MATLAB время переходного процесса определяется для 2%-ного отклонения от установившегося значения.

```
>> plot ( w, abs(r) );
>> semilogx ( w, abs(r) );
>> loglog ( w, abs(r) );
```

В первом случае масштаб обеих осей координат – линейный, во втором случае используется логарифмический масштаб по оси абсцисс (частот), в последнем – логарифмический масштаб по обеим осям. Для вычисления фазы (в градусах) используется команда

```
>> phi = angle(r)*180/pi;
```

после чего можно строить ФЧХ, например:

```
>> semilogx ( w, phi );
```

Действие	Команды MATLAB
1. Создайте массив частот для построения частотной характеристики[4] (100 точек в интервале от 10^{-1} до 10^{2} с равномерным распределением на логарифмической шкале).	`w = logspace(-1, 2, 100);`
2. Рассчитайте частотную характеристику исходной системы [5]…	`r = freqresp ( f, w );` `r = r(:);`
3. … и постройте ее на осях с логарифмическим масштабом по оси абсцисс.	`semilogx ( w, abs(r) )`
4. Постройте сигнал, имитирующий прямоугольные импульсы единичной амплитуды с периодом 4 секунды (всего 5 импульсов).	`[u,t] = gensig('square',4);`
5. Выполните моделирование и постройте на графике сигнал выхода системы `f` при данном входе.	`lsim (f, u, t)`

Полюса и нули

Для нахождения полюсов передаточной функции **`f`** можно использовать функцию

```
>> p = pole ( f )
```

Вызов функции

```
>> [w0,zeta,p] = damp ( f )
```

позволяет найти не только полюса **`p`**, но также соответствующие им собственные частоты **`w0`** и коэффициенты демпфирования **`zeta`** в виде массивов.

[4] Точка с запятой в конце команды подавляет вывод на экран результата выполнения. Это удобно при работе с большими массивами.

[5] Частотная характеристика возвращается в виде трехмерного массива, в котором каждый элемент имеет 3 индекса: строка, столбец (для многомерных моделей) и номер точки частотной характеристики. Для системы с одним входом и одним выходом команда `r = r(:);` преобразует эти данные в в обычный одномерный массив.

Нули передаточной функции **f** вычисляются как

```
>> z = zero ( f );
```

Устойчивость системы не зависит от расположения нулей, но они существенно влияют на переходные процессы. Команда

```
>> pzmap ( f );
```

строит карту расположения нулей (они обозначаются кружками) и полюсов (крестики) системы на комплексной плоскости.

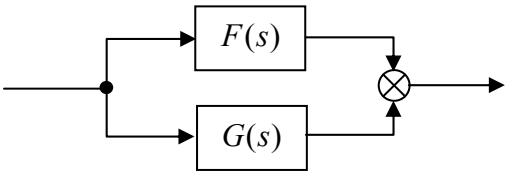
Действие	Команды MATLAB
1. Постройте на графике расположение нулей и полюсов системы.	`pzmap ( f )`
2. Определите коэффициенты демпфирования и собственные частоты для всех элементарных звеньев (первого и второго порядка).	`[wc,ksi,p] = damp ( f )`

Модели соединений систем

Для построения моделей соединений систем в MATLAB используются знаки арифметических действий. Эти операции перегружены, то есть, переопределены специальным образом для объектов классов **tf**, **ss** и **zpk**. Введем исходные модели, с которыми будем выполнять все операции:

```
>> f = tf(1, [1 1]);
>> g = tf(1, [2 1]);
```

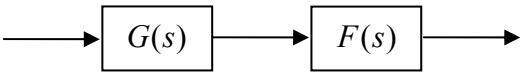
- параллельное соединение



$$W(s) = F(s) + G(s)$$

```
>> w = f + g
Transfer function:
    3 s + 2
---------------
2 s^2 + 3 s + 1
```

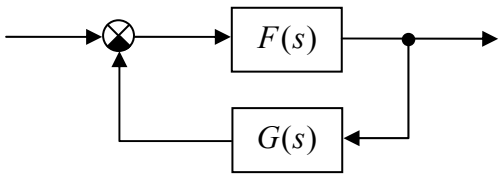
- последовательное соединение



$$W(s) = F(s)G(s)$$

```
>> w = f * g
Transfer function:
       1
---------------
2 s^2 + 3 s + 1
```

- контур с отрицательной обратной связью



$$W(s) = \frac{F(s)}{1 + G(s)F(s)}$$

```
>> w = feedback(f, g)
Transfer function:
     2 s + 1
---------------
2 s^2 + 3 s + 2
```

Можно вычислить эту передаточную функцию и так:

```
>> w = f / (1 + g*f)
Transfer function:
     2 s^2 + 3 s + 1
-----------------------
2 s^3 + 5 s^2 + 5 s + 2
```

Этот результат может показаться неожиданным. Дело в том, что обе передаточных функции имеют первый порядок, то есть, описываются дифференциальным уравнением (ДУ) первого порядка. Поэтому вся система должны описываться второго порядка, а мы получили третий. Чтобы разобраться в этом, преобразуем модель к форме «нули-полюса»:

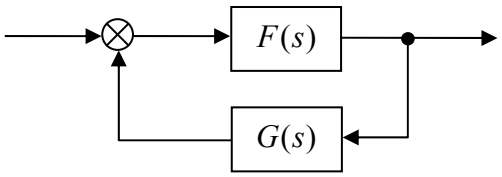
```
>> w_zpk = zpk( w )
Zero/pole/gain:
      (s+1) (s+0.5)
-----------------------
(s+1) (s^2  + 1.5s + 1)
```

Видно, что числитель и знаменатель передаточной функции содержат общий множитель **s+1**, который можно сократить, и остается система второго порядка. Для этого надо построить *минимальную реализацию*, сократив общие множители:

```
>> w = minreal ( w )
Transfer function:
     s + 0.5
--------------
s^2 + 1.5 s + 1
```

Эта передаточная функция совпадает с той, что выдает функция **feedback**.

- контур с положительной обратной связью



$$W(s) = \frac{F(s)}{1 - G(s)F(s)}$$

```
>> w = feedback(f, -g)
```

или

```
>> w = feedback(f, g, 1)
```

или

```
>> w = minreal ( f/(1 - g*f))
```

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

Действие	Команды MATLAB
1. Введите передаточную функцию модели судна $W_1(s)=\frac{K}{s(T_1 s+1)}$ как объект **tf**.	`W1 = tf ( K, [T1 1 0] )`
2. Введите передаточную функцию интегрирующего звена $W_2(s)=\frac{1}{T_2 s}$.	`W2 = tf ( 1, [T2 0] )`
3. Постройте передаточную функцию, замкнув интегратор единичной отрицательной обратной связью.	`W2z = feedback ( W2, 1 )`
4. Постройте передаточную функцию последовательного соединения объекта с приводом.	`W = W1 * W2z`
5. Постройте переходную характеристику для полученной модели.	`step ( W )`
6. Постройте ЛАФЧХ разомкнутой системы[6].	`bode ( W )`
7. Отметьте точки, определяющие пересечение ЛАЧХ с прямой 0 дБ и пересечение ЛФЧХ с прямой -180^0.	Figure No. 1 ПКМ – Characteristics – Stability (Minimum Crossing)
8. Определите запасы устойчивости по амплитуде (*Gain margin*) и фазе (*Phase margin*).	Figure No. 1 ЛКМ на метках-кружках
9. Найдите максимальный коэффициент усиления разомкнутой системы.	Figure No. 1 ПКМ – Characteristics – Peak Response

Работа с SISOTool

Действие	Команды MATLAB
1. Постройте передаточную функцию $W_3(s)=\frac{1}{T_3 s+1}$.	`H = tf ( 1, [T3 1] )`
2. Импортируйте передаточную функцию W как модель объекта (*Plant*) и W_3 как модель датчика (*Sensor*).	SISO Design Tool File - Import

[6] В зарубежной литературе ЛАФЧХ называют диаграммой Боде.

3. Отключите изображение корневого годографа так, чтобы в окне осталась только ЛАФЧХ.	SISO Design Tool `View – Root Locus` (отключить)

4. Для того, чтобы сразу видеть изменения переходных процессов, запустите **LTIViewer**[7] из верхнего меню окна **SISOTool**. Расположите два окна рядом, чтобы они не перекрывали друг друга.	SISO Design Tool
	`Analysis –`
	`Response to Step Command`

5. Оставьте только график переходного процесса на выходе, отключив вывод сигнала управления.	LTI Viewer
	`ПКМ – Systems –`
	`Closed loop r to u`

6. Определите перерегулирование σ и время переходного процесса t_p [8].	LTI Viewer `ПКМ – Characteristics –` • `Peak Response` • `Settling Time`
7. Постройте передаточную функцию замкнутой системы.	`Phi = W / (1 + W*W3)`
8. Постройте минимальную реализацию передаточной функции замкнутой системы.	`Phi = minreal(Phi)`

Моделирование в SIMULINK

Создание моделей в SIMULINK

Пакет SIMULINK предназначен для моделирования систем. Вся модель строится из блоков, имеющих входы и выходы. Существует библиотека стандартных блоков, кроме того, можно создавать свои собственные блоки любой сложности. Существует две группы специальных устройств – источники сигналов (**Sources**) и устройства вывода (**Sinks**).

Блоки имеют названия. Для того, чтобы изменить название, надо щелкнуть по нему ЛКМ и отредактировать текст[9].

Каждый блок имеет свои настраиваемые свойства. Для их изменения надо дважды щелкнуть на блоке и изменить нужные значения в диалоговом окне.

Для того, чтобы повернуть блок на 90 градусов, надо выделить его и нажать клавиши **Ctrl+R**. Комбинация **Ctrl+I** позволяет выполнить зеркальное отражение входов и выходов.

[7] LTI = Linear Time-Invariant, линейная стационарная система.
[8] По умолчанию в Matlab время переходного процесса определяется для 2%-ного отклонения от установившегося значения.
[9] В некоторых версиях MATLAB существуют проблемы с русской буквой «Я», поэтому рекомендуется использовать английские названия блоков.

Верхнее меню **Format** предназначено для изменения оформления выделенного блока. Также для этой цели можно использовать контекстное меню **ПКМ – Format**. Для выделенного блока можно изменить цвет текста и линий (**Foreground color**), цвет фона (**Background color**), вывести тень (**Show drop shadow**), переместить название на другую сторону (**Flip name**).

Для выделения одного блока или соединительной линии надо щелкнуть ЛКМ по нужному элементу. Для того, чтобы выделить несколько блоков, надо «обвести» их при нажатой ЛКМ. Клавиша **Delete** удаляют выделенную часть. Чтобы скопировать блок (или выделенную часть), надо перетащить его при нажатой *правой* кнопке мыши (ПКМ).

Блоки соединяются линиями связи, по которым распространяются сигналы. Для того, чтобы соединить блоки, надо щелкнуть ЛКМ по источнику сигнала и затем, при нажатой клавише **Ctrl**, по блоку-приемнику. Можно также протянуть мышкой линию связи между нужными выходом и входом.

Чтобы подать один сигнал на два блока (сделать «развилку»), надо сначала создать одну линию обычным способом. Чтобы провести вторую линию, следует нажать *правую* кнопку мыши на линии в точке развилки и протащить линию ко второму блоку.

Модель можно скопировать в буфер обмена в виде растрового рисунка. Для этого в окне модели надо выбрать в верхнем меню пункт **Edit – Copy model to clipboard**. Предварительно лучше уменьшить размеры окна до минимальных, чтобы не было белых полей.

Для того, чтобы запустить моделирование, надо щелкнуть ЛКМ по кнопке  на панели инструментов. Эта же кнопка позволяет остановить моделирование при необходимости.

Параметры моделирования (метод интегрирования, обработка ошибок) устанавливаются с помощью окна **Simulation – Parameters**. Самые важные параметры – это время моделирования (**Stop time**) и метод численного интегрирования уравнений (**Solver options**).

Основные источники сигналов (Sources)

Constant – сигнал постоянной величины.

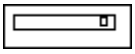
Step – ступенчатый сигнал, меняется время скачка (**Step Time**), начальное (**Initial Value**) и конечное значение (**Final Value**).

Ramp – линейно возрастающий сигнал с заданным наклоном (**Slope**). Можно задать также время начала изменения сигнала (**Start Time**) и начальное значение (**Initial Value**).

Pulse Generator – генератор прямоугольных импульсов, задаются амплитуда (**Amplitude)**, период (**Period**), ширина (**Pulse Width**, в процентах от периода), фаза (**Phase Delay**).

Sine Wave – синусоидальный сигнал, задается амплитуда (**Amplitude**), частота (**Frequency**), фаза (**Phase**) и среднее значение (**Bias**).

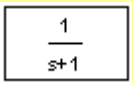
Основные устройства вывода (Sinks)



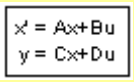
Display – цифровой дисплей, показывает изменение входного сигнала в цифровом виде.

Scope – осциллограф, показывает изменение сигнала в виде графика, позволяет передавать данные в рабочую область MATLAB для последующей обработки и оформления.

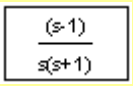
Линейные системы (Continuous)



Transfer Fcn – передаточная функция, в параметрах задаются числитель (**Numerator**) и знаменатель (**Denominator**) в виде полиномов.



State Space – модель в пространстве состояний, в параметрах задается четверка матриц, определяющих модель, и начальные условия для вектора состояния (**Initial conditions**).



Zero-Pole – модель в форме «нули-полюса», в параметрах задаются массивы нулей (**Zeros**), полюсов (**Poles**), а также коэффициент усиления (**Gain**).

Integrator – интегратор с возможностью установки начальных условий (**Initial condition**), а также пределов насыщения (**Lower saturation limit** и **Upper saturation limit**). Когда сигнал выхода выходит за границы, определяемые этими пределами, интегрирование прекращается.

Другие часто используемые блоки

Math Operations

Gain – усилитель, задается коэффициент усиления (**Gain**).

Sum – сумматор, используется для сложения и вычитания входов. Параметр **List of signs** задает количество входов, их знаки («**+**» для сложения и «**–**» для вычитания). Промежутки между входами (обозначаются знаком **|**).

Trigonometric Function – тригонометрическая функция.

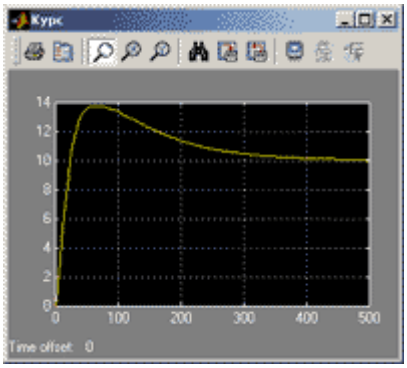
Signal Routing

Manual Switch – ручной переключатель, позволяет двойным щелчком переключать выход на один из двух входных сигналов.

Mux – мультиплексор, объединяет несколько сигналов в один «жгут» (векторный сигнал), в параметрах задается число входов (**Number of Inputs**).

Demux – демультиплексор, позволяет «разбить» векторный сигнал на несколько скалярных, в параметрах задается число выходов (**Number of Outputs**).

Блок *Scope*



В окне блока **Scope** изображается график изменения входного сигнала. Если вход соединен с выходом мультиплексора, сразу строится несколько графиков (по размерности входного «жгута»).

По умолчанию на оси ординат используется диапазон от -5 до 5. Если этот вариант не подходит, выбрать масштаб автоматически (так, чтобы весь график был виден) можно с помощью кнопки  . Соседняя кнопка  сохраняет эти настройки для следующих запусков.

Кнопка  открывает окно настроек, причем наиболее важные данные содержатся на вкладке **Data history**. Если не сбросить флажок **Limit data points**, в памяти будет сохраняться только заданное число точек графика, то есть, при большом времени моделирования начало графика будет потеряно.

Отметив на этой же странице флажок **Save data to workspace** можно сразу передать результаты моделирования в рабочую область MATLAB для того, чтобы их можно было дальше обрабатывать, выводить на графики и сохранять в файле. Поле **Variable name** задает имя переменной в рабочей области, в которой сохраняются данные. В простейшем случае выбирается формат **Array** (в списке **Format**). Это означает, что данные будут сохраняться в массиве из нескольких столбцов (первый столбец – время, второй – первый сигнал, третий – второй сигнал и т.д., по порядку входов мультиплексора).

Оформление графиков

Для создания нового окна для рисунка в MATLAB используется команда

```
>> figure(1);
```

Вместо единицы можно ставить любой номер рисунка. Если рисунок с таким номером уже есть, он становится активным и выводится на первый план. Если такого рисунка нет, он создается и становится активным.

В MATLAB есть возможность строить несколько графиков на одном рисунке. Иначе говоря, рисунок можно разбить на «клетки», в каждой из которых строится отдельный график. Для этого надо сделать активным нужный рисунок и применить команду

```
>> subplot(2, 1, 1);
```

Первое число в команде **`subplot`** показывает количество «строк» в такой матрице, второе – количество столбцов, третье – какой по счету график сделать активным (считая по строкам, справа налево и сверху вниз). Все дальнейшие команды (**`plot`**, **`title`**, **`xlabel`**, **`ylabel`**, **`legend`** и др.) относятся к этому «подграфику».

В командах можно передавать в качестве аргументов не целые массивы, а их части. Например, по команде

```
>> plot(x(1:20), y(11:30));
```

строится график, на котором по оси абсцисс откладываются значения элементов массива **x** с номерами от 1 до 20, а по оси ординат – соответствующие им значения из массива **y** с номерами от 11 до 30.

Двоеточие означает «все строки» или «все столбцы». Например, по команде

```
>> plot(x(:,1), x(:,2));
```

строится зависимость между первым и вторым столбцами массива **x** (здесь двоеточие вместо первого индекса обозначает «все строки»).

С помощью команды **plot** (а также и других подобных – **semilogx**, **semilogy**, **loglog**) можно строить несколько линий на одном графике. Для этого среди аргументов перечисляются пары массивов:

```
>> plot(x, y, v, z);
```

Первая линия будет показывать зависимость **y** от **x**, а вторая – зависимость **z** от **v**. массивы в каждой паре должны быть одинаковой длины. При желании можно указать цвета для каждой линии, Например,

```
>> plot(x, y, 'b', v, z, 'g');
```

Первая линия (зависимость **y** от **x**) будет синей, вторая (зависимость **z** от **v**) – зеленой. Можно использовать следующие цвета

b	синий (**blue**)
g	зеленый (**green**)
r	красный (**red**)
c	голубой (**cyan**)
m	фиолетовый (**magenta**)
y	желтый (**yellow**)
k	черный (**black**)

По умолчанию первая линия – синяя, вторая – зеленая и т.д. в порядке перечисления цветов в списке. Дополнительно можно указать тип линии

-	сплошная
:	точечная
-.	штрих-пунктирная
--	штриховая

Например,

```
>> plot(x, y, 'b:', v, z,'g--');
```

Первая линия – точечная синего цвета, вторая – штриховая зеленого цвета. По умолчанию все линии сплошные.

Для оформления графика также используются команды

title заголовок графика

xlabel название оси абсцисс

ylabel название оси ординат

У всех этих команд обязателен один аргумент – текст в апострофах.

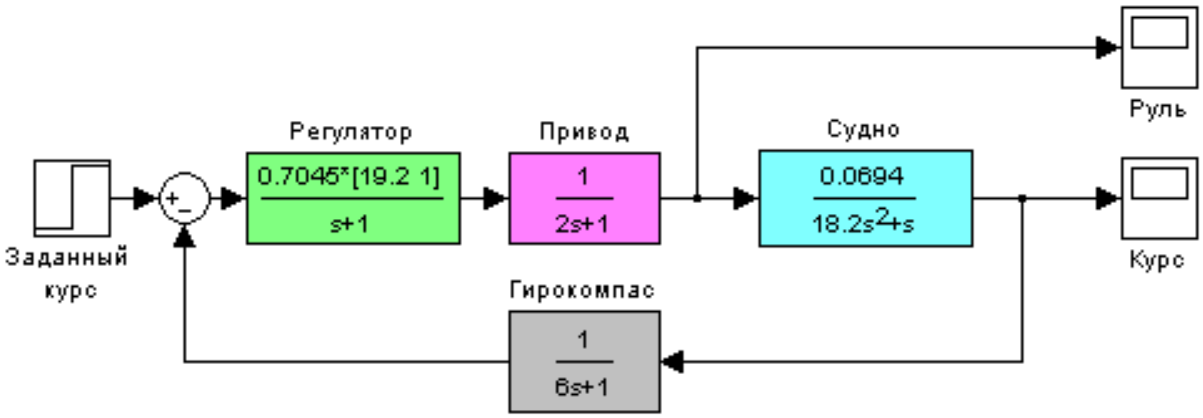
Команда **legend** служит для вывода легенды графика. Легенда нужна, если на графике есть несколько линий и надо показать, что обозначает каждая из них.

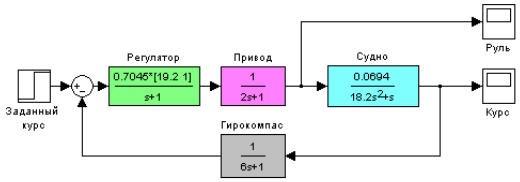
Параметрами команды **`legend`** являются символьные строки, их должно быть столько, сколько построено линий.

В надписях можно использовать некоторые команды системы TeX[10]. Например, греческие буквы записываются в виде «**`\alpha`**», «**`\beta`**» и т.д. Верхний индекс (степень) обозначается знаком «**`^`**», Например, a^2 запишется как «**`a^2`**». Для обозначения индекса используют нижнее подчеркивание, например, a_{22} кодируется как «**`a_{22}`**».

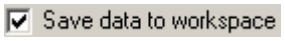
Пример. Система управления судном по курсу.

Схема системы управления судном по курсу.



Действие	Команды MATLAB
1. Создайте модель ЛДС управления судном по курсу.	
2. Установите время моделирования 100 секунд.	`Simulation – Simulation parameters` `100 в поле Stop time`
3. Выполните моделирование.	`ЛКМ по кнопке` ▶
4. Посмотрите результаты моделирования, открыв окна для блоков **Курс** и **Руль**.	`Двойной щелчок по блоку`
5. Настройте масштаб по осям в окнах обоих блоков.	`ЛКМ по кнопке` 🔭 `– установить оптимальный масштаб`
6. Сохраните настройки.	Курс `ЛКМ по кнопке`

[10] TeX – лучшая в мире система подготовки текстов с математическими формулами, разработанная математиком и программистом Дональдом Кнутом (автором многотомного издания «Искусство программирования для ЭВМ»). Чаще всего используются так называемые макропакеты (надстройки), расширяющие возможности ядра TeX, например LaTeX или AMSTeX.

7. Сделайте так, чтобы результаты моделирования передавались с обоих блоков **Scope** в рабочую область MATLAB в виде матриц, в которых первый столбец – время, а второй – сигнал (курс или угол поворота руля).	 ЛКМ по кнопке  вкладка Data history   Variable name: phi (Курс) или delta (Руль) Format: Array
8. Выполните моделирование еще раз.	ЛКМ по кнопке 
9. Перейдите в командное окно MATLAB и создайте новое окно для графика. В одном окне будут построены две кривых на разных осях.	`figure(1);`
10. Разбейте окно на 2 части по вертикали и сделайте активным первый график. Первое число в команде **`subplot`** означает количество ячеек с графиками по вертикали, второе – по горизонтали, третье – номер ячейки, которую надо сделать активной[11].	`subplot(2, 1, 1);`
11. Постройте график изменения курса. В команде **`plot`** сначала указывают массив абсцисс, затем – массив ординат. Двоеточие означает, что используются все строки.	`plot(phi(:,1),phi(:,2));`
12. Введите заголовок графика.	`title('Курс');`
13. Введите названия осей координат. Внутри апострофов для ввода греческих букв разрешается использовать команды LaTeX, Например, «`\phi`» означает греческую букву ϕ, а «`\delta`» – букву δ.	`xlabel('Время, сек');` `ylabel('\phi, градусы');`

14. Аналогично постройте во второй ячейке график изменения угла поворота руля, используя данные из массива **`delta`**, полученного в результате моделирования.	`subplot(2, 1, 2);`
	`plot(delta(:,1),delta(:,2));`
	`title('Угол поворота руля');`
	`xlabel('Время, сек');`
	`ylabel('\delta, градусы');`

Подсистемы

Если на схеме много блоков, она становится громоздкой и работать с такой схемой неудобно. Чтобы не перегружать схемы, можно объединять блоки в *подсистемы*. Проще всего выделить нужные блоки мышкой («обвести» при нажатой ЛКМ) и нажать клавиши **Ctrl+G** (или выбрать пункт меню **Edit – Create subsystem**). На основной схеме

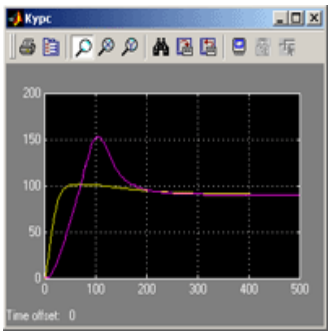
[11] При вводе этой и следующих команд окно с графиком не появляется на экране. Чтобы увидеть изменения, надо вручную сделать его активным, щелкнув мышью на соответствующей кнопке в панели задач.

подсистема изображается как блок типа **Subsystem** (из группы **Ports and Subsystems**). Этот блок можно добавить и вручную, перетащив из окна *Library Browser*.

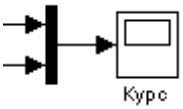
С помощью двойного щелчка ЛКМ можно «открыть» подсистему. Входы обозначаются блоками **In**, а выходы – блоками **Out** (также из группы **Ports and Subsystems**). Можно добавлять новые входы и выходы, удалять ненужные, менять названия, работая с ними так же, как с остальными блоками.

Внутри подсистем можно создавать вложенные подсистемы. В подсистеме не может быть блоков с одинаковыми названиями, однако в разных подсистемах – могут быть.

Сохранить модель можно из окна любой подсистемы, но закрытие окна подсистемы не приводит к закрытию модели.



Блок *Scope* (несколько сигналов)



К входу осциллографа (блока **Scope**) можно подключать несколько сигналов одновременно. Для этого их надо объединить в один векторный сигнал («жгут») с помощью блока **Mux** (мультиплексор) из группы **Signal Routing**.

Если используется два входных сигнала, первый изображается желтой линией, второй – фиолетовой. При передаче данных в рабочую область MATLAB в формате **Array**, массив будет содержать три столбца – время, первый сигнал и второй сигнал. Если сигналов больше, соответственно увеличивается количество столбцов массива.

Программирование в MATLAB

Скрипты

При работе в MATLAB часто для получения нужного результата надо ввести последовательно несколько команд. Если выяснится, что в какой-то команде была сделана ошибка или нужно изменить исходные данные, все команды придется вводить снова. Чтобы не набирать их вручную, можно записать всю последовательность команд на диск в виде текстового файла (M-файла с расширением `.m`), а затем выполнять его, вызывая по имени. Такой файл называется *скриптом.*

Скрипт – это программа, которая представляет собой список команд на языке системы MATLAB. Скрипты можно создавать и редактировать в любом простейшем текстовом редакторе (например, в *Блокноте*), однако удобнее всего использовать встроенный редактор MATLAB, в котором есть подсветка синтаксиса (команды, символьные строки, комментарии и другие элементы программы выделяются разными цветами).

В M-файле перечисляются последовательно все необходимые команды. Точка с запятой в конце команды подавляет вывод результата на экран. Можно располагать в одной строке несколько команд, разделяя их запятой (если нужен вывод результата на

экран) или точкой с запятой. Если надо перенести длинную команду на следующую строчку, в конце строки ставится троеточие.

Комментарием считается все, что расположено справа от знака % до конца строки. Его можно ставить в любом месте строки, например, справа от команды MATLAB.

Для вызова скрипта надо набрать его имя в командном окне MATLAB и нажать клавишу **Enter**. Скрипт должен находиться в рабочей папке (имя которой показано в окне **Current directory** в верхней части командного окна) или в одной из папок, входящих в *путь для поиска*. В путь для поиска включены папки, в которых находятся М-файлы для стандартных функций системы MATLAB и дополнительных пакетов (*toolbox*). Если надо, чтобы скрипт запускался из любой папки, надо включить папку, где он находится, в путь для поиска (команда **File – Set Path** верхнего меню).

Запустить скрипт можно непосредственно из окна редактора MATLAB, нажав на клавишу **F5**. Можно выполнить не весь скрипт, а только некоторые строки – их нужно выделить и нажать клавишу **F9**. Можно расположить два окна (редактор и командное окно MATLAB) рядом так, чтобы они не перекрывали друг друга. Тогда при выполнении скрипта (или отдельных команд) сразу будет виден результат.

Если в командах скрипта есть ошибки (или они возникли при выполнении), соответствующие сообщения выводятся в командное окно Matlab.

Форматирование графика

По умолчанию команда **plot** рисует на активном графике новую кривую, стирая старые линии. Для того, чтобы существующие кривые сохранились, перед вызовом **plot** надо выполнить команду

```
>> hold on
```

Обратная ей команда

```
>> hold off
```

восстанавливает стандартный режим.

Каждый объект на графике (оси координат, надписи, линии и т.п.) представляют собой *объекты*, имеющие свойства. Для того, чтобы получить значение свойства, используют команду **get**, а для изменение свойства – команду **set**. Сокращение **gca** обозначает текущие (активные) оси координат (*get current axes*). Команда

```
>> get(gca)
```

выводит на экран все свойства осей и их значения. Для управления размером шрифта (он измеряется в пунктах) используется свойство **Font Size**:

```
>> get(gca, 'FontSize')   % определить размер шрифта
ans =
     10
>> set(gca, 'FontSize', 16)      % изменить размер шрифта
```

Для того, чтобы настроить свойства отдельной линии, надо сначала получить ее *хэндл* (*handle* – ручка, рукоятка, указатель). Так называется уникальный числовой код объекта, через который к этому объекту можно обращаться. Команда **gca** в самом деле возвращает хэндл текущих координатных осей. Команда

```
>> h = get(gca, 'Children')
```

записывает в переменную **h** массив хэндлов – указателей на объекты линии. Для каждой линии можно установить толщину (в пунктах, по умолчанию – 0,5 пункта) .

```
>> set(h(1),'LineWidth',1.5)
>> set(h(2),'LineWidth',1.5)
```

Аналогично можно управлять и другими свойствами.

Передача данных в модель SIMULINK

Если параметры блоков модели часто изменяются, удобнее сделать так, чтобы их можно было менять прямо в командном окне MATLAB или даже из скрипта. Для этого при задании параметров блоков надо использовать не числовые значения, а имена переменных. При запуске моделирования SIMULINK будет искать переменные с такими именами в рабочей области MATLAB и подставлять их значения. Например, в параметрах блока **Transfer Fcn** можно задать

Numerator: **n**
Denominator: **d**

Тогда для того, чтобы звено соответствовало передаточной функции $\frac{s+2}{2s^2+3s+1}$ в командном окне MATLAB надо задать значения для этих массивов

```
>> n = [1 2]
n =
     1     2
>> d = [2 3 1]
d =
     2     3     1
```

При любом изменении этих массивов в рабочей области меняются и свойства соответствующего блока в модели. Можно использовать и более сложные выражения, например, в поле *Numerator* можно ввести **Kc*[Ts 1 0]**. Это означает, что числитель имеет вид $K_C(T_s s^2 + s)$, при старте моделирования из рабочей области MATLAB будут загружены значения двух переменные с именами **Kc** и **Ts**.

Функции в MATLAB

Все M-файлы, с которыми работает система MATLAB, делятся на две категории: *скрипты* и *функции*. Скрипт – это просто последовательность команд, в которой используются переменные из основного рабочего пространства MATLAB. Функция – это подпрограмма, которая принимает аргументы (параметры) и возвращает результаты. В отличие от функций в большинстве языков программирования, функция MATLAB может возвращать несколько результатов (а не один). Функция отличается от скрипта тем, что имеет заголовок, который начинается словом **function**. Например, заголовок

```
function [a,b,c,d] = qq ( x, y, z )
```

определяет функцию с именем **qq**, которая принимает три параметра (**x**, **y** и **z**) и возвращает 4 результата (**a**, **b**, **c** и **d**). В отличие от большинства современных языков, типы переменных (целая, вещественная, символьная, массив и т.д.) не определяются заранее, каждая из них содержать любые допустимые в MATLAB данные.

Функция записывается в M-файл (с расширением **.m**), имя которого должно совпадать с именем функции. Например, функция **qq** должна быть записана в файл **qq.m**.

Современные версии MATLAB вообще не обращают внимание на имя функции (в заголовке), важно только имя файла.

Функция имеет своё пространство переменных и не может напрямую обращаться к переменным основного рабочего пространства. Внутри функции доступны аргументы, кроме того, можно вводить и использовать новые переменные. Для того, чтобы вернуть нужные значения, надо записать их в переменные **a**, **b**, **c** и **d**. Оператор **return** используется для досрочного возврата из функции (до этого во все выходные переменные должны быть записаны нужные значения).

Функции могут вызываться по имени из командной строки MATLAB, из скрипта или из другой функции. Например, для вызова рассмотренной выше функции **qq** надо набрать команду вида

```
[w,e,r,t] = qq ( x1, 4*x2, 3 )
```

В данном случае при работе функции вместо **x** используется значение переменной **x1**, вместо **y** – значение выражения **4*x2**, а вместо **z** – число 3. Результаты функции записываются в переменные **w**, **e**, **r** и **t**.

При вызове функции количество входных и выходных переменных может быть меньше (но не больше!), чем в заголовке функции. Внутри функции число входных параметров хранится в специальной переменной **nargin**, а число выходных величин – в переменной **nargout**.

Некоторые стандартные функции MATLAB

Система MATLAB включает большой набор стандартных функций. Большинство этих функций также оформлены в виде М-файлов, их адрес можно узнать с помощью команды **which** (см. работу № 1). В этой работе используются три функции:

- **abs** – вычисление модуля числа или модуля каждого элемента массива
- **max** – вычисление максимального значения в массиве (есть также функция min для вычисления минимального значения)
- **find** – поиск элементов массива, соответствующих заданному условию.

Работа с функциями abs и max достаточно очевидна, поэтому рассмотрим только функцию **find**. Она возвращает индексы (номера) элементов массива, которые удовлетворяют заданному условию. Например, если A – массив, а b – число, по команда

```
>> ind = find ( A > b )
```

в переменную **ind** записывается массив номеров элементов массива A, которые больше b. В условии можно использовать знаки **<**, **>**, **<=**, **>=**, **= =** (равно), **~=** (не равно). Сложные условия строятся с помощью операций ~ (НЕ), **&** (И) и **|** (ИЛИ) так же, как и в других языках программирования.